


RESEARCH ARTICLE

Open Access



Realization of CAD-integrated shell simulation based on isogeometric B-Rep analysis

T. Teschemacher^{1*} , A. M. Bauer¹, T. Oberbichler¹, M. Breitenberger¹, R. Rossi², R. Wüchner¹ and K.-U. Bletzinger¹

*Correspondence:
tobias.teschemacher@tum.de
¹Lehrstuhl für Statik, Technische
Universität München, Arcisstr. 21,
80333 Munich, Germany
Full list of author information is
available at the end of the article

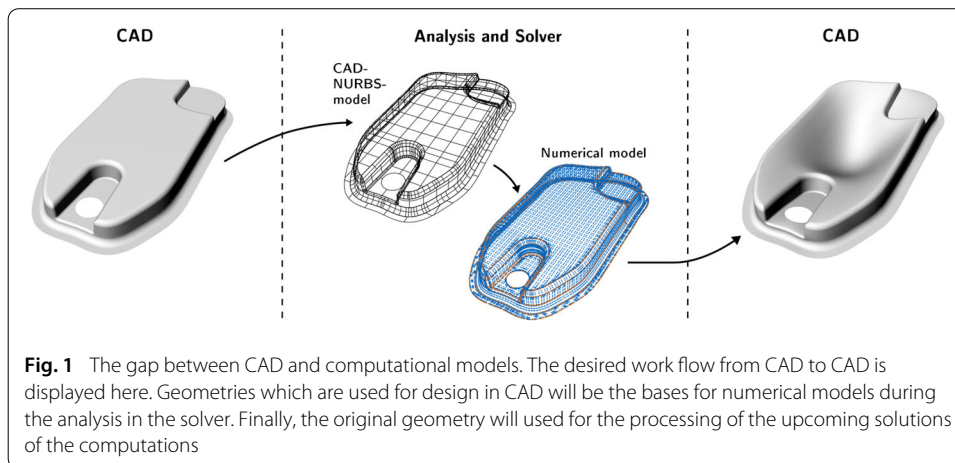
Abstract

An entire design-through-analysis workflow solution for isogeometric B-Rep analysis (IBRA), including both the interface to existing CADs and the analysis procedure, is presented. Possible approaches are elaborated for the full scope of structural analysis solvers ranging from low to high isogeometric simulation fidelity. This is based on a systematic investigation of solver designs suitable for IBRA. A theoretically ideal IBRA solver has all CAD capabilities and information accessible at any point, however, realistic scenarios typically do not allow this level of information. Even a classical FE solver can be included in the CAD-integrated workflow, which is achieved by a newly proposed *meshless* approach. This simple solution eases the implementation of the solver backend. The interface to the CAD is modularized by defining a database, which provides IO capabilities on the base of a standardized data exchange format. Such database is designed to store not only geometrical quantities but also all the numerical information needed to realize the computations. This feature allows its use also in codes which do not provide full isogeometric geometrical handling capabilities. The rough geometry information for computation is enhanced with the boundary topology information which implies trimming and coupling of NURBS-based entities. This direct use of multi-patch trimmed CAD geometries follows the principle of embedding objects into a background parametrization. Consequently, redefinition and meshing of geometry is avoided. Several examples from illustrative cases to industrial problems are provided to demonstrate the application of the proposed approach and to explain in detail the proposed exchange formats.

Keywords: Isogeometric B-Rep analysis, IBRA, IGA, Exchange format, Computer-aided design, CAD, CAD-integrated analysis, Design-through-analysis workflow

Introduction

The possibility of bridging the gap between CAD and computational models drove over the last years the development of “Isogeometric” approaches (as shown in Fig. 1). Such techniques, which employ directly the NURBS discretization in the computational process, proved very successful in addressing a variety of problems thanks to the excellent mathematical properties of the NURBS basis.



Unfortunately, the direct use of general CAD models in the computational process turned out to be very demanding and is to date not yet fulfilled. The key to the outstanding difficulties can be found in the pervasive use of “trimming”, a technology by which some parts of the domain can be excluded from the model by prescribing their shape within the parametric discretization of a regular quadrilateral NURBS. Moreover, the usage of trimming requires the description of the topology (e.g. connectivity of patches) of CAD models, usually given by a boundary representation (B-Rep). Thus NURBS-based B-Rep models are the standard model description within CAD systems for practical engineering problems.

While the idea of NURBS-based B-Rep models and the modelling with them is conceptually intuitive and is very mature within CADs, including such capability within a computational model is far from trivial, since the introduction of trimming lines breaks the continuity of the shape functions employed in the calculation [1]. A number of different research lines, oriented to the solution of such problem were presented over the years. For example, the use of T-Splines [2,3] allows sidestepping the difficulty by providing a way to mesh complex surfaces without having to use the trimming technology. Even though such technique has been partially successful, it relies on a user-driven mesh cleaning step, and hence does not constitute a viable solution in the challenge of using unmodified CAD data.

More recently, the introduction of the isogeometric B-Rep analysis (IBRA) technology [1,4] provided a novel approach to address the challenge. The idea leveraged by IBRA is to keep using all control points included in the model, considering however that only a portion of the domain, the one enclosed by trimming lines, is actually considered in the computational structural analysis, more specifically in the integration process. Such an approach employs the fundamental idea of “Embedded techniques” in which objects are enclosed within a non-matching computational domain enabling the application of boundary conditions at arbitrary positions within the computational domain.

In the IBRA approach, as in the original CAD discretization, each NURBS patch is completely independent of the neighbors, whilst being part of the overall topology. This makes it possible to identify the active and inactive portions of each patch working directly in the parametric domain. The method is then completed by reconstructing the desired continuity by constraining the solution to match the continuity requirements along the trimming boundary. The imposition of such constraint is typical to embedded/unfitted techniques

and can be performed in different ways, for example by a penalty approach [4] but also by employing Lagrange multipliers [5,6] or Nitsche-type methods [5]. In a broad sense, CutFEM [7] and finite cell [8,9] approaches can be considered as variations of such idea.

In the practice, achieving a convenient implementation of such model poses important challenges, since it does not fit well with the traditional finite element workflow. The purpose of the current paper is therefore twofold: firstly the bandwidth of possible realizations of the IBRA technology with suitable solver designs is described, secondly an integrated approach to the whole computational pipeline, i.e. from CAD to computation, is defined for the different levels of isogeometric fidelity in the structural solvers.

A theoretically ideal IBRA solver has all CAD capabilities and information accessible at any point, which is not achievable in realistic scenarios. Therefore, variants of optimally CAD-integrated solvers need to be elaborated with distinct CAD-related functionality which results in different type and amount of data at the interface between CAD and structural analysis. As the other extreme, even a classical FE solver can be included in the CAD-integrated workflow, which is achieved by a new *meshless* approach. This facilitates significantly the implementation of the IBRA approach in any solver. To this end, the key observation is that the implementation of IBRA (or of any FEM-type calculation) on the level of assembly only relies on the knowledge of shape functions, shape function derivatives and integration weights at the integration points. Once such information is available, each integration point can be treated as an independent “element” connecting the “cloud” of control points whose shape functions are non-zero at the integration point position.

The advantage is that the integration points do not need to be located according to a regular tensor-product based structure, thus naturally fitting the need of covering an irregularly trimmed domain homogeneously.

Following this idea, the paper addresses in detail how multiple patches as well as trimming lines and coupling information can be conveniently treated in the framework of the proposed approach. This is leveraging the idea that the support (read as “cloud” or relevant control points) of integration points located at the domain interfaces (trimming lines or patch boundaries) can naturally span multiple domains. This approach thus allows decoupling the calculations between a geometrical kernel, in charge of generating suitable integration points, identifying the relevant clouds of control points and computing the shape functions, and a computational kernel completely agnostic to such geometric operations.

Our claim is that this naturally defines a computational pipeline from CAD to calculation (and eventually back to CAD), which can be decomposed in modules, each largely independent of the others. A group may then decide to address the complete pipeline or to focus on some of the rings of the chain, be it in the geometrical decomposition or the computational back end, the same way as it is normally done in the FEM community (meshing and computation) but without losing the advantage of preserving the exact geometry and the advantages of the NURBS basis through the entire pipeline.

In order to fulfil this vision, an exchange mechanism is needed, since currently existing formats (e.g. IGES [10,11] and STEP [12]) are not designed with such a purpose in mind. The paper is thus completed by the description of a mechanism for data exchange to and from CAD systems. Such mechanism is designed to allow different levels of integration of the CAD capabilities (resulting from the identified distinct levels of isogeometric fidelity in the respective solver), allowing to either directly manage the CAD import/export problem or to rely on the availability of a preprocessing library able to make available the

data needed for the mentioned *meshless* approach, thus dumping to disk and reloading when needed the control point cloud as well as all the information needed to perform calculations.

From a formal point of view, the structure of the paper is as follows:

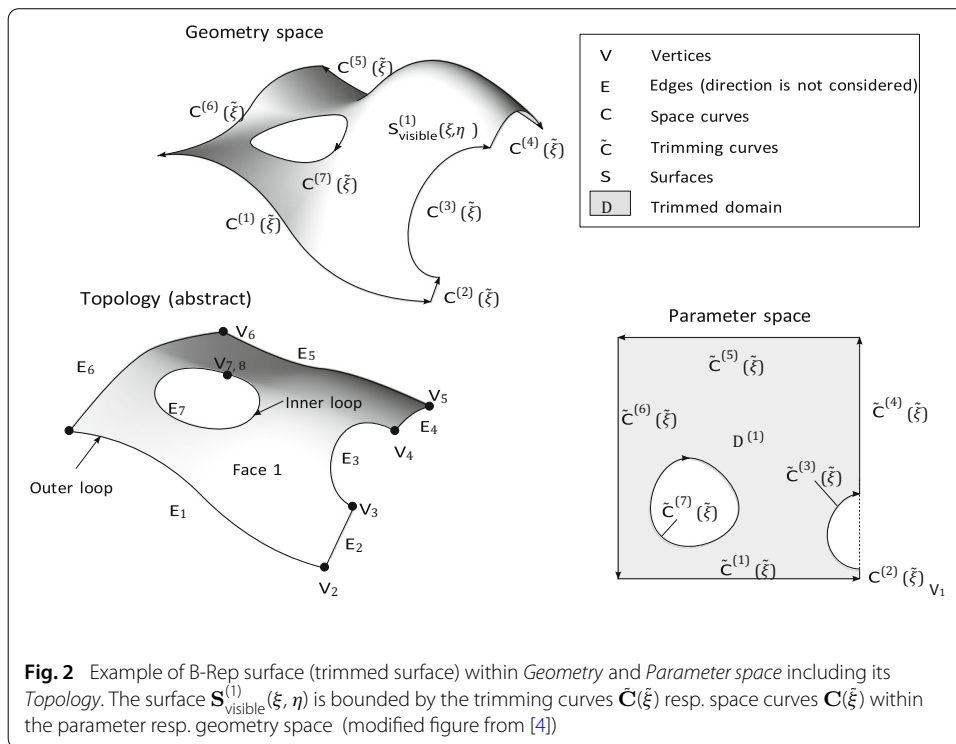
- “Isogeometric B-Rep analysis (IBRA)” section summarizes briefly the main aspects and components as well as the required notation for IBRA.
- “Solver design” section investigates systematically the possible solver designs suitable for IBRA.
- “Design-through-analysis workflow” section identifies the required CAD–CAE-coupling data and defines data interfaces for the IBRA design-through-analysis workflow.
- “IBRA exchange format” section elaborates exchange formats for the necessary data interfaces which eventually enable the IBRA workflow for complex geometry models.
 - “Data interface—Geometry” section explains the geometrical description of surface models including their topologies.
 - “Data interface—Integration domains” section describes the corresponding integration domains.
 - In “Data interface—Integration points” section a possible data exchange on level of integration points i.e. the *meshless* integration points is shown.
- “Simulation of real CAD models” section demonstrates with some advanced structural analysis problems based on real-world CAD models that the presented workflow is working successfully.
- “Conclusion” section summarizes the document and gives an outlook to further research.
- Appendix A provides some basic and precisely documented examples for a better understanding of the proposed format for the geometries (corresponding to “ID systems” and “Data interface—Geometry” sections).
- Appendix B contains some well-documented examples for a better understanding of the proposed format for the integration domains (corresponding to “Integration domains within IBRA” and “Data interface—Integration domains” sections).

Isogeometric B-Rep analysis (IBRA)

Isogeometric B-Rep analysis [4] can be seen as an extension of the *isogeometric analysis* (IGA). IBRA uses in addition to the basis functions from CAD, the *Boundary Representation* (B-Rep, see also “Boundary representation (B-Rep)” section) description for approximating solution fields. Thus, it allows to analyze thin-walled structures directly based on the CAD model.

NURBS-based B-Rep models

Most CAD systems in mechanical engineering use NURBS-based B-Rep models since they are well suited for modelling complex shapes like car bodies, airplanes, and other products.



Boundary representation (B-Rep)

In geometric modelling, B-Rep is a method for representing shapes using boundaries. The boundary representation of an object consist of two parts:

- Geometry (shape), which defines the spatial position, curvatures, etc.
- Topology, which allows to make links between geometrical entities.

The three main topology entities are

- Faces.
- Edges.
- Vertices.

Thus, a solid for example is defined by a set of enclosing surfaces, named *faces*. Those faces are bounded by *edges* (E) lying on the surface. These edges are geometrically represented by curves. Finally, the curves are bounded by points named vertices (V). The set of curves that are enclosing the surfaces are called trimming loop. One distinguishes between inner (holes) and outer loops. Inner loops are defined clockwise and outer loops are defined counter-clockwise. An example of a B-Rep surface, i.e. a trimmed surface is shown in Fig. 2.

NURBS basis functions for curves and surfaces

In this section some basics about NURBS and the used notation are summarized. A detailed description of NURBS is given in [13]. Since NURBS are a generalization of B-Splines the latter are explained first.

B-Spline basis functions $N_{i,p}$ depend on the knot vector Ξ , which is defined by a set of non-descending parameters, and a polynomial degree p . The basis functions can be evaluated by the *Cox-de Boor* [14,15] recursion formula.

A geometry could be expressed by a linear combination of n shape functions with their respective control points \mathbf{P}_i . The formula for a B-Spline curve $\mathbf{C}(\xi)$ is given by

$$\mathbf{C}(\xi) = \sum_{i=1}^n N_{i,p}(\xi) \cdot \mathbf{P}_i \tag{1}$$

In contrast to that, NURBS basis functions have an additional weight w_i for every control point. The weight controls the influence of a control point \mathbf{P}_i respectively of the corresponding shape function $N_{i,p}$ on the final geometry. The NURBS becomes a B-Spline if all weights are equal. Otherwise, it leads to rational basis functions that allow the exact representation of any conic section properly (e.g. circles) which makes NURBS popular in computer-aided design.

Considering a weight for each control point leads to the formula for NURBS curves in Eq. (2) with the corresponding basis functions $R_{i,p}$.

$$\mathbf{C}(\xi) = \sum_{i=1}^n \frac{N_{i,p}(\xi) \cdot w_i}{\sum_{j=1}^n N_{j,p}(\xi) \cdot w_j} \mathbf{P}_i = \sum_{i=1}^n R_{i,p}(\xi) \cdot \mathbf{P}_i \tag{2}$$

NURBS surfaces are defined by a tensor product of NURBS basis functions with the two parametric dimensions ξ and η . The corresponding geometry description for NURBS surfaces is given by

$$\mathbf{S}(\xi, \eta) = \sum_{i=1}^n \sum_{j=1}^m \frac{N_{i,p}(\xi) \cdot M_{j,q}(\eta) \cdot w_{ij} \cdot \mathbf{P}_{ij}}{\sum_{k=1}^n \sum_{l=1}^m N_{k,p}(\xi) \cdot M_{l,q}(\eta) \cdot w_{kl}} = \sum_{i=1}^n \sum_{j=1}^m R_{ij,pq}(\xi, \eta) \mathbf{P}_{ij} \tag{3}$$

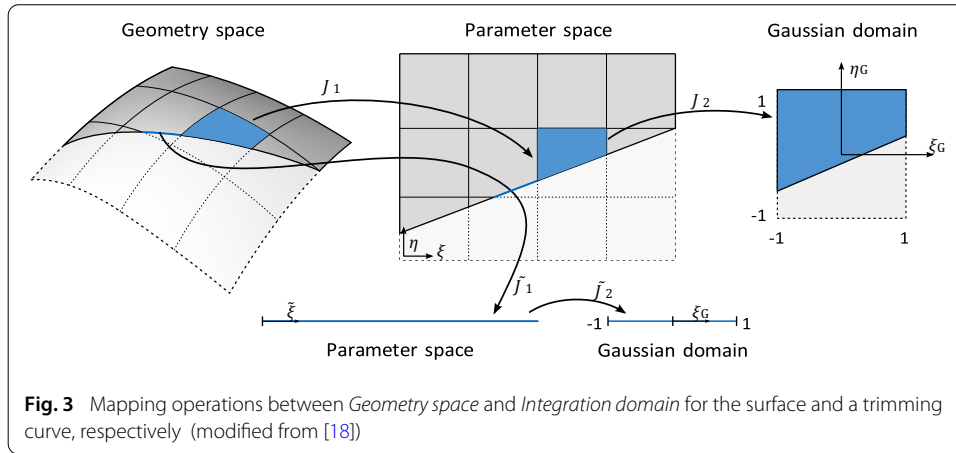
with p and q being the polynomial degrees and $N_{i,p}(\xi)$ and $M_{j,q}(\eta)$ the corresponding independent shape functions.

Trimmed NURBS surfaces

A trimmed NURBS surface is described by a NURBS surface and a set of M properly ordered boundary (trimming) curves $\tilde{\mathbf{C}}_k(\tilde{\xi})$ with $k = 1, \dots, M$ lying within the parameter space of the surface (see also [16]). Thus, a trimmed surface is a partially visible surface, defined by the *trimmed domain* which is described by the trimming curves. In general, trimming curves can be of any form, however, when dealing with NURBS entities, it is desirable to represent these with NURBS, too. The curves $\tilde{\mathbf{C}}_k(\tilde{\xi})$ are joined properly to form outer and inner loops. The outer loops are oriented counter-clockwise, whereas the inner loops are oriented clockwise (see also Fig. 2). Since for geometric modeling an explicit description of the boundary within the geometry space is needed, the trimming curves $\tilde{\mathbf{C}}_k(\tilde{\xi})$ are mapped onto the surface as an explicit space curve $\mathbf{C}_k(\xi)$ (see also [17]).

B-Rep edges

Edges are the second topological entity in a B-Rep model. They describe the boundaries of the surfaces and contain furthermore topological relationships (cf. topology in Fig. 2). An edge is described within CAD systems by one space curve $\mathbf{C}(\xi)$ bounded by two *vertices*



given in spatial coordinates and links to the corresponding trimming curves $\tilde{C}(\tilde{\xi})$ of the adjacent *faces*. This information can be transferred to IBRA for coupling and boundary conditions.

The trimming curves are analogously to space curves described as NURBS curves.

$$\tilde{C}(\tilde{\xi}) = \sum_{i=1}^n R_{i,p}(\tilde{\xi}) \tilde{P}_i \tag{4}$$

Note that, all entities with specifier $\tilde{\bullet}$ refer to a parameter space of a surface. Consequently, the coordinates of the control points \tilde{P}_i are given with respect to ξ and η of each NURBS surface. $\tilde{\xi}$ denotes the curve parameter along the trimming curves.

Integration domains within IBRA

Isogeometric B-Rep analysis requires the numerical integration of trimmed domains and their boundaries resp. *edges*. The latter are needed because a strong enforcement of boundary conditions is in general not possible and thus, they require a weak imposition, which eventually leads to the evaluation of an integral.

Figure 3 summarizes the different necessary mapping operations for surfaces and edges.

Numerical integration of surfaces

The area $|A|$ of a trimmed surface element is defined within the parametric coordinates $\xi \in [\xi_s, \xi_e]$ and $\eta \in [\eta_s, \eta_e]$. The corresponding control points of the curve segment are mapped into the *Gaussian domain* \mathcal{G} by shifting, scaling and rotating. This curve is then used for constructing an auxiliary surface \hat{S} in the *Gaussian domain*, which in return can be integrated as a conventional untrimmed NURBS surface. More details can be found in [1].

The corresponding formula is given by

$$|A| = \int_A dA = \int_{\xi_s}^{\xi_e} \int_{\eta_s}^{\eta_e} J_1 d\xi d\eta = \int_{\mathcal{G}} J_1 J_2 d\mathcal{G} \tag{5}$$

with \mathcal{G} being the *Gaussian domain*. In Eq. (5) the Jacobian J_1 represents the mapping from *Geometry to Parameter space* (see Fig. 3). This mapping can be derived by using the base vectors \mathbf{g}_1 and \mathbf{g}_2 (shown in Fig. 4) as follows

$$J_1 = \|\mathbf{g}_1 \times \mathbf{g}_2\|_2 \tag{6}$$

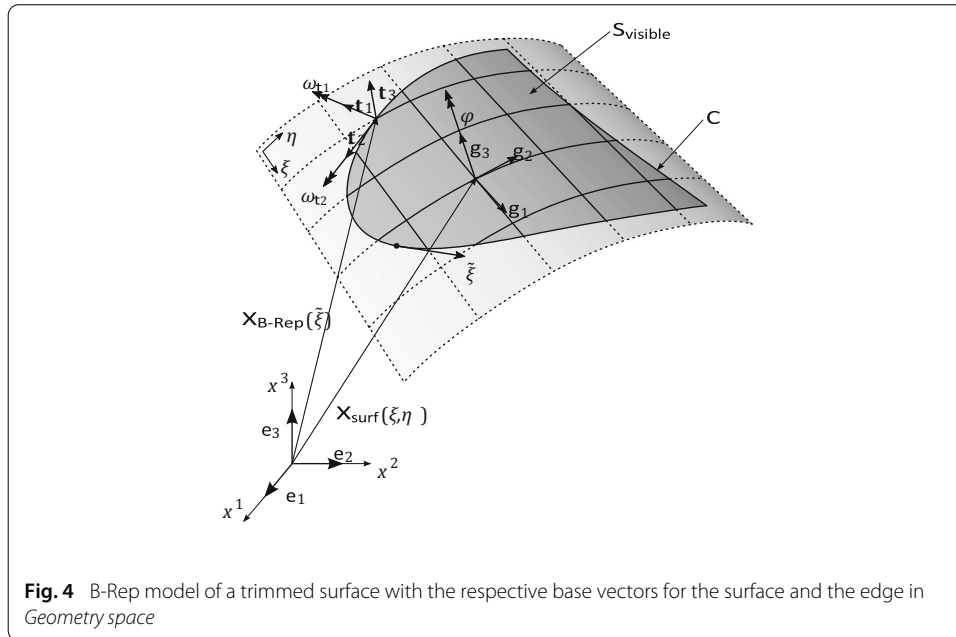


Fig. 4 B-Rep model of a trimmed surface with the respective base vectors for the surface and the edge in Geometry space

The mapping from *Parameter space* to the *Gaussian domain* \mathcal{G} (with $\xi_{\mathcal{G}} \in [-1, 1] \times \eta_{\mathcal{G}} \in [-1, 1]$) is defined as the Jacobian J_2

$$J_2 = \frac{\partial \xi}{\partial \xi_{\mathcal{G}}} \frac{\partial \eta}{\partial \eta_{\mathcal{G}}}, \tag{7}$$

where ξ and η are the parameters in *Parameter space* and $\xi_{\mathcal{G}}$ and $\eta_{\mathcal{G}}$ the corresponding parameters which describe the *Gaussian domain*.

The mapping J_2 is deformation independent and can thus be included in the so-called *weighting factor* \tilde{w}_l (see also [1]) of a quadrature point l which is given by

$$\tilde{w}_l = J_2 w_l \tag{8}$$

with w_l being the Gaussian quadrature weight used for integrating the *Gaussian domain*.

With the knowledge of the *weighting factor* \tilde{w}_l the area of a surface can be easily computed as follows

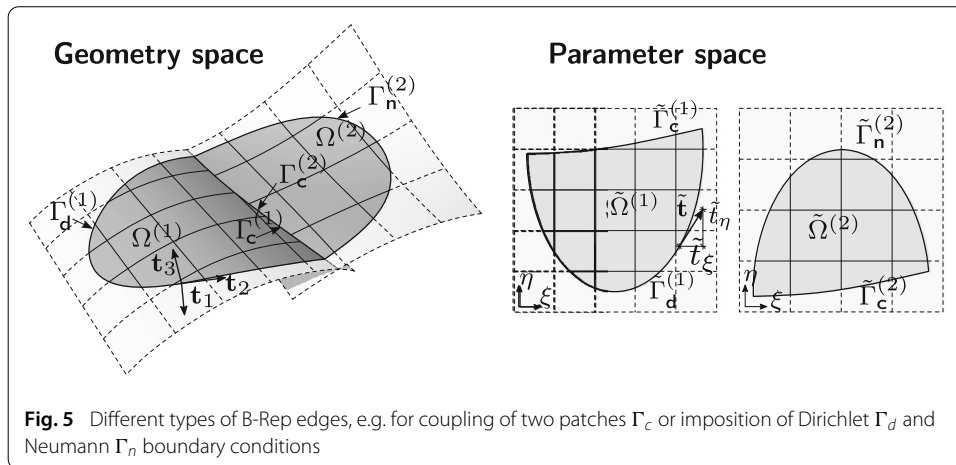
$$|A| \approx \sum_{l=1}^{n_{qp}} J_1 \tilde{w}_l. \tag{9}$$

Numerical integration of edges

A B-Rep edge element is a segment of a NURBS curve, defined within a trimming curve of an underlying surface. The B-Rep edge element is used for distinct purposes e.g. patch coupling or imposition of Neumann and Dirichlet boundary conditions. Examples of B-Rep edges are given in Fig. 5. The length of a B-Rep edge $|\Gamma_e|$ can be computed as follows

$$|\Gamma_e| = \int_{\Gamma_e} d\Gamma_e = \int_{\tilde{\xi}} \tilde{J}_1 d\tilde{\xi} = \int_{\mathcal{G}} \tilde{J}_1 \tilde{J}_2 d\mathcal{G}, \tag{10}$$

where \tilde{J}_1 describes the mapping from *Geometry* to *Parameter space* of the trimming curve with its parameter $\tilde{\xi}$. \tilde{J}_2 represents the mapping from *Parameter space* to the *Gaussian domain* \mathcal{G} (see Fig. 3).



\tilde{J}_1 can be evaluated as follows

$$\tilde{J}_1 = \|(\mathbf{g}_1 \cdot \tilde{t}_\xi + \mathbf{g}_2 \cdot \tilde{t}_\eta)\|_2, \tag{11}$$

with the two base vectors \mathbf{g}_1 and \mathbf{g}_2 of the surface and the components of the trimming curve tangent given by $\tilde{t}_\xi = \frac{\partial \xi}{\partial \tilde{\xi}}$ and $\tilde{t}_\eta = \frac{\partial \eta}{\partial \tilde{\xi}}$ as shown in Fig. 5. The second mapping parameter \tilde{J}_2 is defined as

$$\tilde{J}_2 = \frac{\partial \tilde{\xi}}{\partial \xi_G}. \tag{12}$$

This \tilde{J}_2 mapping is deformation independent and can thus be included analogously to the surface integration in the so called *weighting factor* \tilde{w}_l (see also [1]) of a quadrature point l given by

$$\tilde{w}_l = \tilde{J}_2(\tilde{\xi}) \cdot w_l \tag{13}$$

with w_l being the Gaussian quadrature weight used for integrating the *Gaussian domain*.

As \tilde{J}_2 the *weighting factor* \tilde{w}_l is deformation independent. Thus, it can be precomputed. The length of a B-Rep edge can be easily computed with \tilde{w}_l as follows

$$|\Gamma_e| \approx \sum_{l=1}^{n_{qp}} \tilde{J}_1 \tilde{w}_l. \tag{14}$$

Numerical Integration Procedure

The *Numerical Integration Procedure* is one of the important and challenging parts of the IBRA workflow. During this process a proper *Integration domain* is defined and created. The procedure is split to the *Integration Domain* of surfaces (see “Surface integration procedure” section) and of edges (see “Coupling edge integration procedure” section). The necessary tasks, difficulties and possible ways of the procedure are described in this section.

Surface integration procedure

IBRA evaluates element functions over trimmed NURBS surfaces by calculating integration points inside the trimmed domain. Figure 6 shows different approaches to define the required integration points:

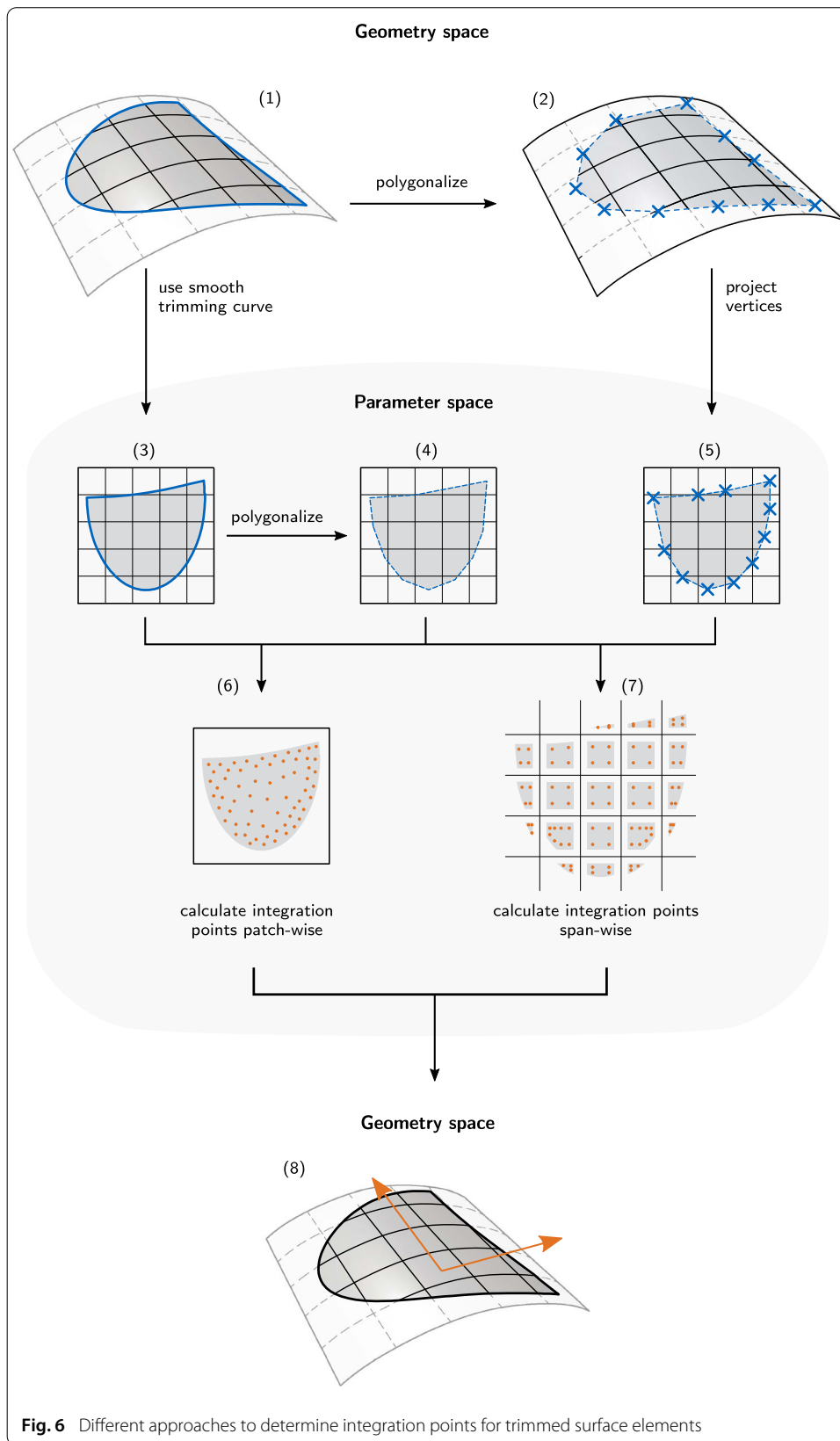
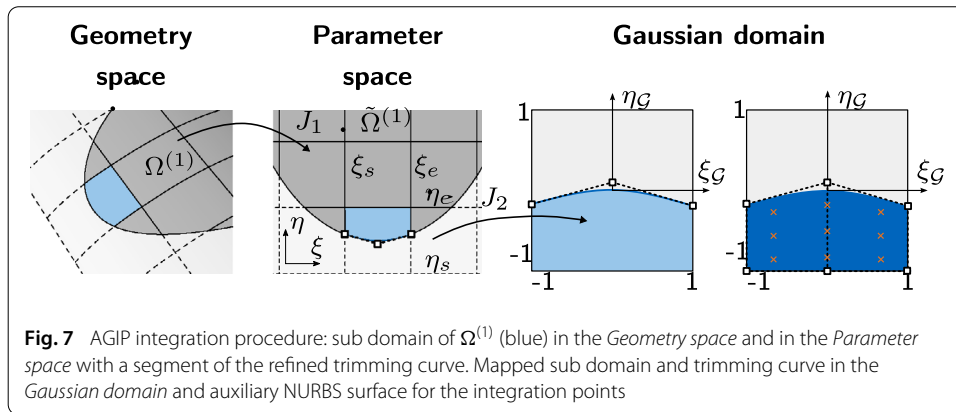


Fig. 6 Different approaches to determine integration points for trimmed surface elements



- The first step is to decide if the boundaries are approximated for a simpler computation of the integration domains. The approximation can be realized by transforming the exact boundaries in geometry space (1) to a polygon (2) by mapping the vertices into the parameter space (5) or by a polygonalization of the NURBS trimming curves (3) in the parameter space (4). Doing the approximation in geometry space allows to define the tolerances for the polygonalization (max. edge length, angle deviation, etc.) in real units while using the parameter space leads to simpler 2D operations. Using the smooth boundary curve (1 and 3) leads to an exact representation of the trimming domain but also to complex geometric operations.
- The next step is the subdivision into integration domains for the computation of the integration points. Here, one can generally divide the methods in the two categories of patch-wise (6) and span-wise integration (7). Patch-wise integration has the advantage of less integration points whereas patch-wise segmentation has to deal with less trimming scenarios.
- The necessary entities in the geometry space (8) for the element formulation can finally be derived at the defined integration points.

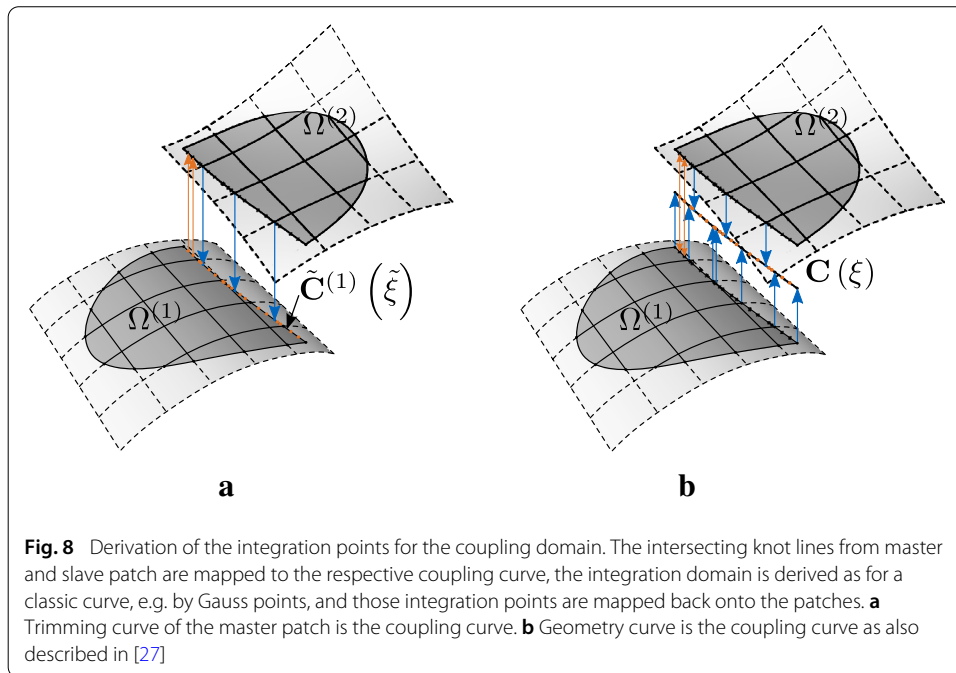
Further literature on trimming and the achievable quality of the solutions by employing alternative approaches to the presented AGIP method by [1] in “Numerical integration of surfaces” section, can be found in [19–26] (Fig. 7).

Coupling edge integration procedure

The discretization has to be evaluated over the trimming curves of both patches to evaluate the coupling edge information. Either the geometry or the parameter curve can be used for the discretization of the *Integration Domain* at the edge (cf. Fig. 8). As a consequence, different mapping operations of knot lines and integration points become necessary.

In the following the two different projection techniques will be explained briefly (see Fig. 8). One will take the parameter curve as reference, one is describing the discretization on the geometry curve.

- The discretization is made on the parameter curve $\tilde{C}^{(1)}(\tilde{\xi})$ of patch (1) (Fig. 8a). The procedure of this strategy can be as follows:
 - Find all intersections of the parameter curve of patch (2) with the knots of the underlying patch.



- Project the geometric positions of all intersections to the parameter curve $\tilde{C}^{(1)}(\tilde{\xi})$ and find all intersections of the parameter curve with the knots of patch (1).
 - All intersections provide space for curves in *Gaussian domain*. It should be considered the highest polynomial order of both patches.
 - Take integration points on patch (1) additionally project the integration points to the *Parameter space* of patch (2) to have the position on both patches.
- b. A curve $C(\xi)$ in *Geometry space* which represents both parameter curves is used for discretization of the coupling curve (Fig. 8b). The procedure using the geometry curve can look as follows:
- Obtain geometry curve fitting to both parameter curves. The curve will be already given as described in the format from “Data interface—Geometry” section.
 - Find all intersections of both parameter curves with the knots of their underlying patches, respectively.
 - Project all obtained intersections to the *Parameter space* of the geometry curve $C(\xi)$.
 - In all of the upcoming spaces between the intersections are introduced *Gaussian domains*. The highest polynomial order of both patches is to be used here.
 - Project the geometric position of all integration points to parameter space of both patches as the position on both patches is needed.

In this context the use of the parameter curve is preferred. Generally by choosing the right parameters the solution of both strategies converge. In case of the use of the parameter curve there is no necessity to compute a geometry curve. In the last step the points have to be projected to parameter space. This is, depending on the complexity of the surface, very costly as it is a non-linear operation. Also convergence is not guaranteed. By using

the parameter curve only the projection to patch (2) is needed. Whilst using the geometry curve the integration points have to be projected to both patches.

Analysis-related enhancement of geometrical data

In the following the analysis-related enhancements of geometrical data is given for surface- and B-Rep edge element formulations.

Surface element formulations are used to represent the shape and solution of the physical problem. Examples for element formulations of isogeometric surface elements for membranes can be found in [28,29] and for shells in [30–38].

B-Rep element formulations can be used for imposing different types of boundary conditions on arbitrary locations within the geometry model. In principle, the following types of boundary conditions can be enforced in a weak sense:

- Neumann boundary conditions.
- Dirichlet boundary conditions.
- Mechanically motivated, e.g. cables (Philipp et al. [29]) or beams (Bauer et al. [39]).
- Patch coupling conditions to connect distinct patches with arbitrary parameterizations.

In general, these boundary conditions have to be fulfilled along the whole edge. Note that knot lines have to be taken into consideration for accurate integration results [1].

There exist several general methods for implying Dirichlet boundary conditions. Two of them, namely

- Penalty approach,
- Lagrange multiplier method,

will be briefly outlined within the context of IBRA for enforcing continuity (coupling) between patches and imposing prescribed displacements or rotations in the following sections.

This section describes the basic principles of line-wise imposed boundary conditions and can easily be transferred to other approaches as e.g. in [5,6,19,24,40–46].

All approaches have the integration along the edge in common, as proposed in “Numerical integration of edges” section.

The use of Nitsche’s technique in the imposition of interpatch-continuity also represents a common and well explored alternative [5,44,47–52]. While such option is clearly superior to the use of simpler approaches, it is less efficient in implementation and computational costs, in the sense that it requires modifying the variational form as well as evaluating boundary integrals at the boundaries of interest. Thus, in the current work it is focused on the alternatives, the penalty method and the Lagrange multiplier method.

Continuity between patches

The following section explains a weak G^0 and G^1 coupling of two trimmed patches on a B-Rep edge (see also $\tilde{\Gamma}_e^{(1)}$ and $\tilde{\Gamma}_e^{(2)}$ in Fig. 5). Two different approaches are explained briefly by using the trimming curve of the master patch as integration domain. This explains the required data for simulations integrated in CAD and thus supports the elaboration of the exchange formats in “Design-through-analysis workflow” section.

Penalty approach

Considering the virtual work term $\delta W_{B-Rep}^{penalty}$ along a B-Rep edge a G^1 continuity can be enforced along it as follows

$$\delta W_{B-Rep}^{penalty} = \delta W^{disp} + \delta W^{rot} . \tag{15}$$

Equation (15) contains two expressions one for coupling the displacements δW^{disp} , i.e. G^0 , and one for coupling the rotations δW^{rot} , i.e. G^1 , along an edge. They are given by

$$\delta W^{disp} = -\alpha_{disp} \int_{\Gamma_e^{(1)}} \left(\mathbf{u}^{(1)} - \mathbf{u}^{(2)} \right) \cdot \left(\delta \mathbf{u}^{(1)} - \delta \mathbf{u}^{(2)} \right) d\Gamma_e^{(1)} \tag{16}$$

$$\delta W^{rot} = -\alpha_{rot} \int_{\Gamma_e^{(1)}} \left(\omega_{T_2}^{(1)} - \omega_{T_2}^{(2)} \right) \cdot \left(\delta \omega_{T_2}^{(1)} - \delta \omega_{T_2}^{(2)} \right) d\Gamma_e^{(1)} \tag{17}$$

with

$$\delta \omega_{T_2}^{(i)} = \frac{\partial \omega_{T_2}^{(i)}}{\partial \mathbf{u}^{(i)}} \cdot \delta \mathbf{u}^{(i)} . \tag{18}$$

Here $\mathbf{u}^{(1)}$ resp. $\omega_{T_2}^{(1)}$ represent the displacement resp. rotation around the tangent of the boundary of the master patch. The index two is used for the corresponding quantities on the slave side. As the penalty factor can differ between displacements and rotations, two distinct factors are introduced, α_{disp} and α_{rot} . The right choice of the penalty factor is very important as a bad choice of the penalty factor can lead to numerical problems (see also [4]). The additional virtual work $\delta W_{B-Rep}^{penalty}$ is used to account for the coupling conditions in a weak sense. In case of matching discretizations this vanishes and the coupling is inherently satisfied.

The discrete form of Eq. (16) is given for example by

$$\delta W^{disp} \approx \alpha_{disp} \cdot \sum_k^{n_{qp}} \tilde{w}_k \tilde{J}_1^k \left(\sum_i^{n_{cp}^{(1)}} R_i^{(1)}(\xi_k^{(1)}, \eta_k^{(1)}) \cdot \mathbf{u}_i^{(1)} - \sum_j^{n_{cp}^{(2)}} R_j^{(2)}(\xi_k^{(2)}, \eta_k^{(2)}) \cdot \mathbf{u}_j^{(2)} \right) \tag{19}$$

with \tilde{w}_k and \tilde{J}_1^k being the weighting factor and the Jacobian for the boundary $\Gamma^{(1)}$ as given in Eq. (14). Exemplary integration points are shown in Fig. 8.

The stiffness of the coupled system is given as follows

$$\mathbf{K} = \begin{bmatrix} \mathbf{K}^{(1)} + \mathbf{K}_p^{(1)} & \mathbf{C}_p^{(1,2)} \\ \mathbf{C}_p^{(2,1)} & \mathbf{K}^{(2)} + \mathbf{K}_p^{(2)} \end{bmatrix} \tag{20}$$

with $\mathbf{K}^{(i)}$ being the stiffness of the patches. $\mathbf{K}_p^{(i)}$ and $\mathbf{C}_p^{(i,j)}$ are the additional penalty stiffness and the penalty coupling cross terms, respectively. From it the coupling terms for the penalty method can be extracted and Eq. (19) can be written in matrix vector notation as

$$\mathbf{K}_p = \alpha \cdot \begin{bmatrix} \mathbf{K}_p^{(1)} & \mathbf{C}_p^{(1,2)} \\ \mathbf{C}_p^{(2,1)} & \mathbf{K}_p^{(2)} \end{bmatrix} = \alpha \int_{\Gamma_e^{(1)}} \mathbf{H}^T \cdot \mathbf{H} d\Gamma_e^{(1)} \tag{21}$$

$$\mathbf{K} \cdot \mathbf{u} = \mathbf{f} \tag{22}$$

with \mathbf{K} resp. \mathbf{f} being the corresponding coupling matrix (here without considering the numerical integration) resp. force vector. The displacement vector is represented by \mathbf{u} .

$$\mathbf{u} = \left[u_{x,1} \ u_{y,1} \ u_{z,1} \ \cdots \ u_{x,n} \ u_{y,n} \ u_{z,n} \right]^T \tag{23}$$

The matrix \mathbf{H} for continuity on displacements is defined as follows

$$\mathbf{H} = \begin{bmatrix} R_1 & 0 & 0 & \cdots & R_n & 0 & 0 \\ 0 & R_1 & 0 & \cdots & 0 & R_n & 0 \\ 0 & 0 & R_1 & \cdots & 0 & 0 & R_n \end{bmatrix} \tag{24}$$

with $n = n_{cp}^{(1)} + n_{cp}^{(2)}$ the number of all control points and R_{cp} being as follows

$$R_1^{(1)} \cdots R_{n_{cp}^{(1)}}^{(1)}, R_1^{(2)} \cdots R_{n_{cp}^{(2)}}^{(2)}$$

with $n_{cp}^{(i)}$ being the number of control points and $R_{cp}^{(i)}$ the NURBS of each patch respectively.

The penalty method is easy to implement and does not need extra degrees of freedom. The stiffness matrix is positive definite which results that the coupled system has a unique solution, except if the conditioning of the system is bad due to a too high penalty factor. The method is called variationally inconsistent as it is not possible to recover from the weak form back to the strong form. Thus, convergence curve levels off.

Due to high local entries in the stiffness matrix problems occur especially for explicit dynamics.

The penalty factor has to be chosen by the user and can not be generalized which leads to higher work load and pre knowledge during the simulation.

Lagrange multiplier method

To avoid the a priori estimation of the penalty factor the Lagrange method uses for the penalty factor a function λ (see also [5]) with its own degrees of freedom (dofs) along the edge which is independent from the displacements. The control points of patch (1) are used in this work and with it the discrete description along the coupling edge.

A virtual work term $\delta W_{B-Rep}^{Lagrange}$ is used to enforce weak G^0 and G^1 continuity along the edge, analogously to the penalty approach for the Lagrange multiplier method

$$\delta W_{B-Rep}^{Lagrange} = \delta W^{disp} + \delta W^{rot} . \tag{25}$$

The terms of virtual work for displacement and rotation coupling are derived for the Lagrange multiplier field as follows

$$\delta W^{disp} = \int_{\Gamma_e^{(1)}} \delta \lambda^{(1)} \left(u^{(1)} - u^{(2)} \right) d\Gamma_e^{(1)} + \int_{\Gamma_e^{(1)}} \lambda^{(1)} \left(\delta u^{(1)} - \delta u^{(2)} \right) d\Gamma_e^{(1)} \tag{26}$$

$$\delta W^{rot} = \int_{\Gamma_e^{(1)}} \delta \lambda^{(1)} \left(\omega_{T_2}^{(1)} - \omega_{T_2}^{(2)} \right) d\Gamma_e^{(1)} + \int_{\Gamma_e^{(1)}} \lambda^{(1)} \left(\delta \omega_{T_2}^{(1)} - \delta \omega_{T_2}^{(2)} \right) d\Gamma_e^{(1)} \tag{27}$$

To solve problems using the Lagrange multiplier methods Eq. (25) can be expressed in following equation system

$$\begin{bmatrix} \mathbf{K}^{(1)} & \mathbf{0} & \mathbf{\Lambda}^T \\ \mathbf{0} & \mathbf{K}^{(2)} & \mathbf{0} \\ \mathbf{\Lambda} & \mathbf{0} & \mathbf{0} \end{bmatrix} \cdot \begin{bmatrix} \mathbf{u}^{(1)} \\ \mathbf{u}^{(2)} \\ \boldsymbol{\lambda} \end{bmatrix} = \begin{bmatrix} \mathbf{f}^{(1)} \\ \mathbf{f}^{(2)} \\ \mathbf{0} \end{bmatrix} . \tag{28}$$

The equation can be solved with $\mathbf{\Lambda}$ being obtained as follows

$$\mathbf{\Lambda} = \int_{\Gamma_e^{(1)}} \mathbf{H}_\lambda \cdot \mathbf{H} \, d\Gamma_e^{(1)} \tag{29}$$

with \mathbf{H} from Eq. (24) and \mathbf{H} from Eq. (30).

$$\mathbf{H}_\lambda = \begin{bmatrix} R_{\lambda,1} & 0 & 0 & \cdots & R_{\lambda,n} & 0 & 0 \\ 0 & R_{\lambda,1} & 0 & \cdots & 0 & R_{\lambda,n} & 0 \\ 0 & 0 & R_{\lambda,1} & \cdots & 0 & 0 & R_{\lambda,n} \end{bmatrix} \tag{30}$$

with $n = n_{cp}^{(1)} + n_{cp}^{(2)}$ the number of all control points and $R_{\lambda,cp}$ being as follows

$$R_{\lambda,1}^{(1)} \cdots R_{\lambda,n_{cp}^{(1)}}^{(1)}, 0 \cdots 0$$

with $n_{cp}^{(i)}$ being the number of control points and $R_{cp}^{(i)}$ the NURBS of each patch respectively. The NURBS of the second patch are neglected as the Lagrange multiplier field is only applied respectively to the degrees of freedom of patch (1).

The Lagrange multiplier method is variationally consistent. The method is also fairly easy to implement but it needs additional degrees of freedom which increases the computational costs significantly.

As the approach is a saddle point formulation, there are zeros in the diagonal within the discrete equation system. Therefore, direct or GMRES for iterative, solvers have to be used. In order for a unique solution to be guaranteed, an LBB-type(Ladyzenskaja–Babuška–Brezzi) condition (see [53]) for the discrete problem has to be satisfied which for a general formulation is not straight forward. However, for simpler problems it can be shown that special choices of the Lagrange multipliers discretizations fulfil an LBB-condition (see [54]).

The method does not allow to use direct solvers as there can occur zero-diagonals. It is possible to overcome this problem by the use of perturbations, which would lead to a relaxing of the interface conditions and a dependence on user input [5].

Dirichlet boundary condition

Dirichlet boundary conditions are special cases of the before mentioned coupling conditions. In the following equations, Dirichlet conditions considering a prescribed displacement u_0 and rotation ω_0 are exemplary shown for the penalty approach.

$$\delta W^{disp} = -\alpha_{disp} \int_{\Gamma_e} (u^{(1)} - u_0) \cdot \delta u^{(1)} \, d\Gamma_e \tag{31}$$

$$\delta W^{rot} = -\alpha_{rot} \int_{\Gamma_e} (\omega_{T_2}^{(1)} - \omega_0) \cdot \delta \omega_{T_2}^{(1)} \, d\Gamma_e \tag{32}$$

Neumann boundary condition

B-Rep elements can also be used for introducing Neumann conditions since the B-Rep elements provide an appropriate integration domain for edges. A line load \mathbf{p} can be added to the system by an additional term for the virtual work.

$$\delta W^{load} = \int_{\Gamma_e} \mathbf{p}(\xi) \, d\Gamma_e \tag{33}$$

Mechanically motivated boundary conditions

Additional to the classical boundary conditions as shown before, the B-Rep edges can be used for imposing further mechanically motivated entities onto the boundaries. Cables [29] and beams [39] are typically used for the reinforcement of shell or membrane structures. These formulations are also based on the principle of virtual work as shown in Eq. (34). The particular element formulations have to be derived from the respective strains and stresses, e.g.

$$\delta W^{mec} = \int_{\Gamma_e} \mathbf{S} : \delta \mathbf{E} \, d\Gamma_e, \quad (34)$$

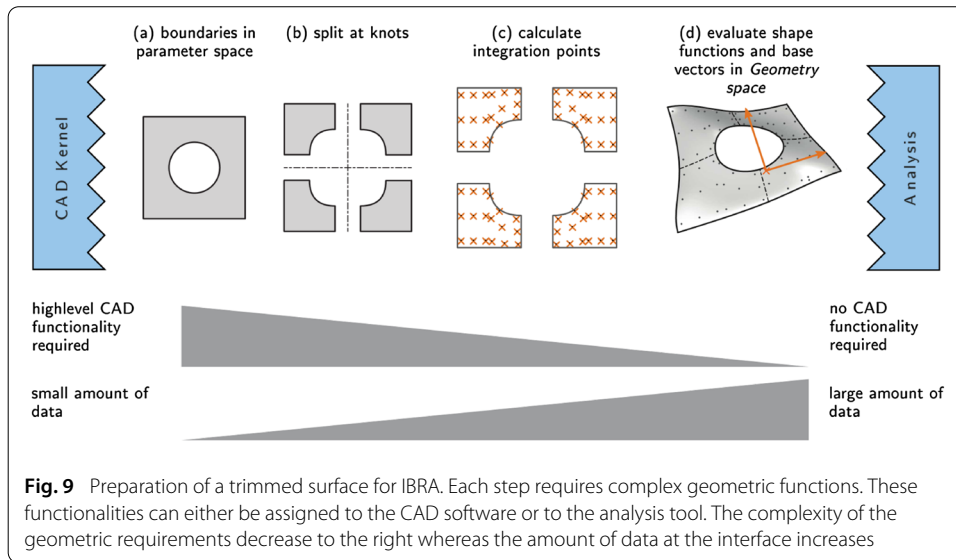
where \mathbf{S} is the stress measure and $\delta \mathbf{E}$ the energetically-conjugated virtual strain. Note that the embedded entity is fully described by the control points of the surface and its degrees of freedom. Consequently, the coupling of an independent structural element to the surface can be avoided and accompanying CAD inaccuracies do not corrupt the analysis result.

Problematic description of geometries in CAD

Although, the isogeometric B-Rep analysis is based on the exact geometries provided from the CAD software, the description might not be well-suited for analyses. The challenge in using CAD-provided geometries is, that for the design, typically a lower quality in the geometry description is acceptable. In the following some of the issues in the CAD-described context, with respect to direct use for structural analysis shall be described here:

- Despite being suitable for shell analysis, thin-walled structures may be modeled by solids, which are described by their boundary surfaces. In this context a direct evaluation with surface element formulations cannot be done on the initial geometries. A proper face description, e.g. by the middle surface has to be derived. This needs additional understanding and work load.
- CAD descriptions with high model tolerances can lead to big gaps between the given patches. The model quality can be such poor that the CAD utility cannot set up the coupling conditions within appropriate tolerances. Additionally, with large gaps between patches some coupling methods can fail easily and a proper physical description is not ensured. This requires large user input to choose tolerances and coupling method correctly.
- Badly conditioned patches with large differences in the lengths require a very special treatment. Here, a robust integration procedure has to be chosen.
- Patches with a small physical domain in relation to the patch size come up with additional challenges in the solveability. Typically, contributions associated to some degrees of freedom will be badly conditioned, which spoils the solutions.

It should be noted, that the challenges resulting from poor geometry description are intrinsically present in the geometry, and would manifest themselves equivalently in the meshing process prior to classical FEM based simulation. A special strength of the here proposed IBRA workflow is the intentional modularity, which allows the user to enhance and exchange the modules which are needed to treat the given challenges in the description of CAD geometries. Those modules will depend on the problems which evolve from the CAD design and quality of the model.



Solver design

IBRA enables direct analysis of CAD-geometries. Figure 9 shows the preparation of the CAD data for the finite element solver. A lot of complex geometric operations must be applied to provide the data for the solver. A complete IBRA solver environment would need only the basic geometrical data and do all these steps internally. The steps can also be done by uncoupled utility tools. This approach would create a large amount of data but allows conventional solvers to do IBRA without implementing additional functionality.

There are different ways to design an IBRA solver. Depending on the implemented functions the solver needs different kind of data. Using the example of the surface integration, Fig. 9 shows different entry points for an analysis tool. In principle, the whole spectrum of depicted interfaces between CAD and a solver is possible and in general it holds: A solver which is close to CAD needs a very high level of CAD functionalities and only a little amount of (complex) CAD data, whereas solvers without any dedicated build-in CAD functions require a big amount of rather basic data. Each step which reduces the CAD-related complexity of the solver (i.e. moving more “away” from CAD in direction of “classical FEM-solvers”), results in additional amount of data needed at the interface to the solver:

- a. *Read trimming data from CAD boundary curves.*
- b. *Split trimming domain at knots boundary curves for each knot span.*
- c. *Compute integration points for each span set of integration points and weights.*
- d. *Evaluate all relevant data at the integration points basis vectors, shape functions, material properties, etc.*

Thus, a complete IBRA analysis tool would only need the geometric CAD data in combination with the mechanical properties. It would be able to do the whole data processing internally (Fig. 9, leftmost).

On the other extreme, these steps can be done by an advanced CAD-geometry tool, tightly related with the CAD system which computes the required data at each integration point and stores them into exchange files. Since the numerical data is already evaluated for each object the integration point does not require a link back to the geometry.

This so called *meshless* approach allows to design a lightweight solver which only implements the mechanical behaviour of the finite elements but without any IBRA specific functionality.

In the *meshless* environment each integration point contains the following data

- *Shape function and derivatives* the evaluated values of the shape functions and derivatives respectively to each control point.
- *Integration weight* the weighting factor for the *Gaussian quadrature*.
- *Control points* the locations and weights of the control points in *Geometry space*.

Storing the data for each integration point in an exchange file leads to large datasets. As a compromise, it is proposed within the present contribution to design an analysis tool with an interface for the integration points (after step c). In this way, the amount of data is reduced and the analysis tool needs only to provide some basic IGA functionalities for evaluating the geometry at the integration points.

Therefore, at least the following data must be available for each integration point

- *Location* the location of the quadrature point in *Parameter space*.
- *Integration weight* the weighting factor for the Gaussian quadrature.
- *Control points* the locations and weights of the control points in *Geometry space*.
- *Degrees* the degrees of the shape functions in each parameter direction.
- *Knots* the knot vectors in each parameter direction.

With this information the evaluation of the shape functions and the calculation of the base vectors can be performed (step d) using Eqs. (9) and (14).

Each integration point can carry additional information depending on the element type. Coupling elements for example require the tangents of the trimming curves within the *Parameter space* of the underlying surface (see also Eq. 11).

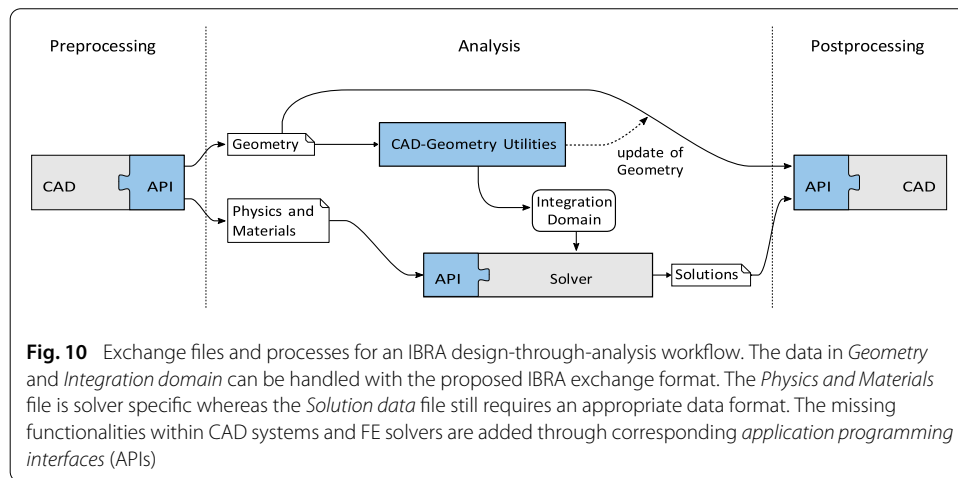
Note that the geometry refinement needs to be done in advance i.e. before creating the integration domains, since the solver is not capable to remesh the integration domain without the respective CAD functionalities. One loses the possibility of direct adaptive refinement if the given parametrization cannot resolve the structural behaviour.

Design-through-analysis workflow

The goal of IBRA is to provide a general framework for bridging the gap between different CAD systems and FE solvers (see also [1]). Thus, in the following clear data interfaces are defined for an IBRA design-through-analysis workflow which allows such a seamless communication and a simulation directly based on the CAD data.

An overview of an IBRA design-through-analysis workflow is given in Fig. 10. Note that this is only exemplary for the proposed solver design in “Solver design” section. However, the workflow can be split and categorized into the following four software components

- CAD system for preprocessing.
- CAD-Geometry Utilities for determining the integration domains (trimmed surfaces and B-Rep edges).
- FE solver for performing IBRA (with different CAD-related capabilities as described in “Solver design” section).
- CAD system for postprocessing.



For the communication between these software components it turned out that the following four data interfaces are useful

- Geometry.
- Integration domains.
- Physics and materials.
- Solutions.

This division into several interfaces provides modularity and allows that isogeometrically sophisticated as well as adapted standard FE-solvers find a docking point for processing the CAD-data. In the following, these data interfaces are explained briefly and specific exchange formats are developed and detailed in “IBRA exchange format” section.

Geometry

The *Geometry* data interface contains CAD data or more precisely NURBS-based B-Rep models. It contains exclusively geometrical information whereas each entity has an id. These ids are used to assign analysis properties to the geometrical entities specified within this file. The corresponding analysis data are specified within the *Physics and Materials* interfaces. The *Geometry* data can be written with an API in the desired CAD system. A possible format of this interface is described in “Data interface—Geometry” section. Some specific examples of the *Geometry* data file are given in “Exchange format description for geometry” section.

It can happen that during the preparation of the *Integration domain* updates in the CAD geometries occur. This is the case for e.g. refinement operations. In this case the *Geometry* data has to be updated. Thus, the *CAD-Geometry Utilities* has to provide functionalities to update *Geometry* data (see Fig. 10).

Physics and materials

The data interface *Physics and Materials* corresponds to the classical input file for an FE solver just without geometry data. Using the ids employed within the data interface *Geometry*, the geometrical entities (faces, edges, ...) resp. the corresponding integration domains can be enhanced with analysis data.

Integration domains

Since the numerical integration of trimmed surfaces and along common edges is not a trivial task, it can be useful to outsource this task into a separate specialized library the *CAD-Geometry Utilities* and to hand over just the required data for the IBRA element formulations. The required data for numerical integration of trimmed surfaces and edges (see also Eqs. (9) and (14)) are defined in the data interface – *Integration domains* which is described in “Data interface—Integration domains” section. The data can be exchanged within the applications in text files, for those the design is described in “Data interface—Integration domains” section. It can also be exchanged with more optimized interfaces as e.g. data bases if the *CAD-Geometry Utilities* and the solver are strongly connected or necessary interface connections are given.

The interface of the *Integration Domains* can be on different levels (see Fig. 9). The IBRA exchange format described in “Data interface—Integration domains” section refers to an exchange after computation of the integration points. It is advantageous if the interface happens between different software with interfaces that only allow import and export through text files. An exchange after the evaluation of shape functions and with it the base vectors became apparent to being more efficient for deeper integrated systems. All geometry data is already precomputed by the *CAD-Geometry Utilities*. This strategy is presented in “Data interface—Integration points” section.

Solutions

After computation, the solver writes the solution into files which can be used to visualize different results. The solution are either provided for the control points or for the integration points in IBRA. In contrast to classical finite element analysis, IBRA uses also ids for integration points. The IBRA solution data can be stored analogously to the *Geometry* and *Integration domain* file. The only important thing is to use the same ID-systems in order to have the results assignable to the initial files. In that case, the *Geometry* and *Integration domain* data interfaces can be enriched by the solution data e.g. displacements or stresses from the FE solver. A visualization on the CAD model within a CAD system becomes possible. Alternatively, a visualization mesh can be created from the CAD model or integration points which allows also the visualization of IBRA results within an arbitrary postprocessing tool for classical FEA.

Exemplary implementations

The IBRA design-through-analysis workflow as presented in Fig. 10 has been implemented for the finite element analysis packages *Carat++* [55] and *KRATOS-Multiphysics* [56].

- *Carat++* is a finite element software, developed at the author’s chair. It contains a complete set of CAD functionalities. Therefore it is possible to start an analysis directly from the *Geometry* and *Physics and Materials* files.
- *KRATOS-Multiphysics* [56] is an open source finite element software started in 2001 by Davvand, Rossi et al. [57,58]. An interface to the *Integration Domain* allows to do IBRA without the effort of implementing the necessary CAD functions.
- The plug-in *TeDA* has been developed for the CAD applications *Rhinoceros* [59] and *Siemens NX* [60] to provide the required data for each analysis tool. It allows

the assignment of mechanical properties to the geometric entities and the export of *Geometry* and *Physics and Materials* files as well as of *Integration Domain* files.

IBRA exchange format

Within this paper, a format for the data interfaces *Geometry* and *Integration domains* is presented. As a format for the data, a *JavaScript Object Notation (JSON)* is used because it is widely spread, flexible, easy to parse and a lot of libraries and useful tools are available for it. Even if an integrated data interface for the integration domains has big advantages e.g. performance or adaptive refinement, the proposed exchange format based on JSON is very useful in terms of the development of new B-Rep element formulations or the reduction of the implementation effort for using IBRA in a new FE solver to a minimum. Before describing the single components of the data interfaces *Geometry* and *Integration domains*, the ID system used within the exchange format is explained.

ID systems

Within the exchange format each entity requires an id, since geometry, integration domains and structural properties are separately stored. Hence, a unique link in between has to be provided in order to allow also hybrids of the workflow shown in Fig. 10. The following id systems are used

- **brep_id**: This id is used to address the geometries of the topological entities *breps*, *faces*, *edges*, and *vertices*
- **cp_id**: This id is used to address the control points to which the degrees of freedom are enhanced.
- **elem_id** and **quad_id**: This id is used to address the integration domains (elements), and quadrature points of the CAD model. Note that those ids are unique to the **cp_id**. Thus, an element and node nor quadrature point cannot have the same id.

All ids are required for the data interface *Project parameter* which enhance the geometrical entities with these ids by analysis data like boundary conditions or physical properties.

Data interface—Geometry

The data interface *Geometry* contains all geometrical information of a NURBS-based B-Rep model. It has to be as light as possible but still complete. Additionally it is desired that it is very easy to understand.

As changes can occur in the file design it gets a **version_number**. The here proposed design refers to number *1.0*.

Every CAD-model has some types of **tolerance**. It is important to know about the tolerances as the quality of the solution highly depends on it. Additionally during computation the knowledge about the tolerance is important to know about the correct convergence. As different CAD software uses distinct types of tolerances it is preferred to include a list of tolerances which can be specified by the chosen tools.

A surface B-Rep model consists of *faces*, *edges* and *vertices*. The structure and components of these entities are described in the following. Note that a model can contain more

```

{
  *version_number*: 1.0,
  *tolerances*: [
    list of tolerances
  ],
  *brep*: [
    {
      *brep_id*: int,
      *faces*: [],
      *edges*: [],
      *vertices*: []
    }
  ]
}

```

Listing 1 Geometry – B-Rep

than one B-Rep. The structure of a B-Rep model is shown in Listing 1. The following informations are necessary for each B-Rep:

- **brep_id**
- **faces** see Listing 2
- **edges** see Listing 3
- **vertices** see Listing 4.

Faces

A face is represented by a trimmed NURBS surface. An example of such a geometry is shown in Fig. 2.

An exemplary case of a *face* within the exchange format is given in Listing 2. The format consists of three main parts

- **surface** describes the shape of the face i.e. the untrimmed NURBS surface.
- **boundary_loops** describes the boundaries of the surface.
- The **embedded_loops**, **embedded_curves** and **embedded_points** define embedded entities which are used to address additional geometrical entities derived from the surface. They follow the same syntax as boundary entities (see below).

Additionally, the following information is needed

- **brep_id**
- **swapped_surface_normal** indicates if the surface normal of the surface needs to be swapped to have the correct orientation of the face.

The part **surface** describes the NURBS surface without B-Rep (trimming) information. The following information are provided by

- **is_trimmed**: In the case the boundaries trim parts of the surface this flag is set TRUE. The flag is used for efficiency reasons.
- **is_rational**: In the case the flag is false all weights of the control points are equal to one. The flag is used for efficiency reasons.
- **degrees**: The polynomial degrees for the parameters u and v
- **knot_vectors**: The knot vectors Ξ and H for the parameters u and v
- **control_points** The control points (CPs) have the id `cp_id`, their spatial position given as x-, y- and z-coordinates and their weights.

The boundaries of the face are listed within **boundary_loops**. Each loop has an

```

"faces": [{
  "brep_id": int,
  "swapped_surface_normal": bool,
  "surface":
  {
    "is_trimmed": bool,
    "is_rational": bool,
    "degrees": [int,int],
    "knot_vectors": [[double, ...],[double, ...]],
    "control_points": [
      [double cp_id,
       [double position_x, double position_y,
        double position_z, double weight]],
      ...list of control points
    ]
  }
},
"boundary_loops": [
  {
    "loop_type": String, Outer or Inner
    "trimming_curves": [
      {
        "trim_index": int,
        "curve_direction": bool,
        "parameter_curve":{
          "is_rational": bool,
          "degree": [int],
          "knot_vector": [double, ...],
          "active_range": [double, double],
          "control_points": [
            [double cp_id,
             [double position_u, double position_v, double position_w, double weight],
            ]
            ...list of control points
          ]
        }
      }
    ]
    ...list of boundary curves
  }
],
"embedded_loops": [],
"embedded_curves": [
  {
    "trim_index": int,
    "curve_direction": bool,
    "parameter_curve":{
      "is_rational": bool,
      "degree": [int],
      "knot_vector": [double, ...],
      "active_range": [double, double],
      "control_points": [
        [double cp_id,
         [double position_u, double position_v, double position_w, double weight]
        ],
        ...list of control points
      ]
    }
  }
],
"embedded_edges":
  ...list of embedded edges
],
"embedded_points": [
  {
    "trim_index": int,
    "point": [double position_u, double position_v, double position_w]
  }
],
...list of embedded vertices
],
...list of faces
]

```

Listing 2 Geometry – Faces

- **loop_type** [Outer, Inner]: indicator specifies if loop is an outer or inner loop
- **trimming_curves** contains a set of respectively ordered curves (see also Fig. 2).

The edges belonging to the corresponding face are listed in **trimming_curves**. Each edge has

- **trim_index** This index is part of a local id system for each brep.
- **curve_direction** direction of the curve in the loop
- **parameter_curve** the parameter curve describes the shape of the edge as a trimming curve in the parameter space of the corresponding face.

For the **parameter_curve** the following information is provided

- **is_rational** see above
- **degree** see above


```

"edges" : [
{
  "brep_id": int,
  "3d_curve":
  {
    "degree": [int],
    "knot_vector": [double, ...],
    "active_range": [double, double],
    "control_points": [
      [double cp_id,
       [double position_x, double position_y, double position_z, double weight]
      ],
      ...list of control points
    ],
  },
  "trimming_ranges": [
    {
      "trim_index": int,
      "range": [double, double],
    },
    ...list of trimming ranges
  ],
  "topology": [
    {
      "brep_id": int,
      "trim_index": int,
      "relative_direction": bool
    }
  ],
  "embedded_curves": [],
  "embedded_points": []
},
...list of edges points
]

```

Listing 3 Geometry – Edges

- **knot_vector** see above
- **active_range** the parameter curve is bounded by two vertices defined in the parametric coordinate of the curve.
- **control_points** the control points (CPs) have their spatial position given as u - and v -coordinates and their weights in the parameter space of its face.

Points on the face can be described within **embedded_points** as follows

- **trim_index** see above
- **parameter_point** the point is defined with the parametric coordinates of the surface u and v .

Edges

A B-Rep edge is represented by a spatial NURBS curve (see also “Trimmed NURBS surfaces” section). For an B-Rep edge the following information are provided

- **brep_id**
- **3d_curve** contains the information for describing the spatial curve representing the edge. Geometrical description given by degree, knot vector and control points. Additionally, an active range is added for trimming.
- **trimming_ranges**
- **topology** describes the topology to the underlying geometrical information of the edge (see also Fig. 2)
- **embedded_points** and **embedded_curves** see above.

The **trimming_ranges** are used for segmenting of the curve for e.g. coupling or boundary conditions and needs the following

- **trim_index** related to the brep_id of its geometry
- **range** gives the bounds for segments of the edge.

```

"vertices": [
  {
    "brep_id": int,
    "coordinates":
    [double cp_id,
    [double position_x, double position_y, double position_z, double weight]],
    "topology": [
      {
        "brep_id": int,
        "trim_index": int
      }
    ]
  },
  ...list of vertices
],

```

Listing 4 Geometry – Vertices

The **topology** needs the following

- **brep_id** related to the brep_id of its geometry
- **trim_index** related to the trim_index of its trimming entity
- **relative_direction** direction of the edge to its trimming entity.

If no **topology** is included, the here described description provides the definition of an independent curve.

Vertices

A B-Rep vertex is represented by a spatial point. A vertex has the following information

- **brep_id**
- **coordinates** x-, y-, and z-coordinates
- **topology** describes the topology to the underlying geometrical information of the vertex.

Note that only one assignment to either face or edge is necessary. Several assignments indicate a coupling.

Data interface—Integration domains

A novelty of the proposed exchange format is that it provides already the integration domains for geometrical entities like *faces* and *edges*. Thus the implementation effort on the FE solver side can be reduced to a minimum because the non-trivial numerical integration of trimmed surfaces and common edges is outsourced Listing 5.

The idea of the data structure is to provide for each quadrature point all necessary information like location, *weight*, knot vectors etc. (see e.g. Eq. 19). In addition, the integration points are grouped to elements which in their turn are grouped to B-Rep entities, i.e. faces and edges. By having an id for each entity/group the correct assignment of analysis data can be guaranteed. The creation of these integration domains is performed by the library as shown in Fig. 10.

This work only explains the data structure for surfaces and its edges. The extension to curves and volumes can be derived straight forward.

The format consists of three main parts

- **nodes** contains all control points of the data interface specified in “Data interface—Geometry” section
- **2d_elements** contains all surface elements which are grouped to B-Rep faces
- **brep_elements** contains all edge elements grouped to B-Rep edges.

```

{
  "nodes": [
    [int cp_id,
     [double position_x, double position_y, double position_z, double weight]
    ],
    ... list of nodes
  ],
  "2d_elements": [
    [int brep_id
     [
       [int element_id,
        [int degree_u, int degree_v],
        [[double knot_vector, ..],[.., ..]],
        [int cp_id, ..],
        bool swapped_normal,
        [
          [int quadrature_point_id,
           double quadrature_point_weighting,
           [double location_in_u, double location_in_v],
           [optional additional properties]
          ],
          ...list of quadrature_points
        ]
      ],
      ...list of elements
    ],
    ...list of patches
  ],
  "1d_elements": [],
  "3d_elements": [],
  "brep_elements": [
    [int brep_id,
     [
       [int brep_element_id,
        [
          [int element_id, optional element_id of slave elements],
          [[int quadrature_point_id,
           double weighting,
           [double location_in_u, double location_in_v],
           [double tangent_t_xi, double tangent_t_eta],
           [optional properties related to degrees of freedom],
           [optional location in slave element],
           [optional tangents of slave element trim]
          ]
        ]
        ...list of quadrature points
      ],
      ...list of brep_elements
    ],
    ...list of breps
  ]
}

```

Listing 5 Element Formulation

Nodes

The set of nodes is optional because the control points are already listed within the data interface “Data interface—Geometry” section (see *faces*). Nevertheless for a faster and easier parsing it can make sense to duplicate them in a separate list.

Elements

The proposed exchange format allows 1D, 2D, and 3D geometries whereas the dimension refers to the dimension of the *Parameter space*:

- **1d_elements**: curved structures, possible element types are e.g. beams or trusses.
- **2d_elements**: surfaces, possible element types are e.g. plates, membranes or shells.
- **3d_elements**: solids.

This section explains exemplarily the components of the **2d_elements**. The elements are grouped to B-Rep entities having a corresponding **brep_id** (see also data interface *Geometry*). Each element has the following information:

- **element_id**
- **degrees**
- **knot_vectors**: the part of the knot vector responsible for the specific element in both parameter directions.

- **cp_ids**: ids of the control points required for the specific element.
- **swapped_normal**: same flag as used for a *face* (see data interface *Geometry*).
- **quadrature points**:
 - **quad_id** the quadrature point id is required to address results e.g. stresses for the post processing.
 - **weighting**: The weighting factor \tilde{w}_l (see Eq. 8).
 - **location**: specifies the location (two parameters) of the quadrature point within the corresponding surface.
 - **add_properties**: This optional container can be used for additional parameters, e.g. a thickness definition.

B-Rep elements

The B-Rep edge elements are also grouped to B-Rep edges with their corresponding **brep_id** (see also data interface *Geometry*). A B-Rep element has the following information

- **brep_element_id**
- **element_ids**: These ids link to the underlying 2D elements. In the case of a free edge just one element is specified. In case of common edges two ids are provided, whereas the first id refers to the master.
- **quadrature points**:
 - **quad_id** id of the quadrature point.
 - **weighting**: The weighting factor \tilde{w}_l for integrating edges (see Eq. 13). The factor refers to the master curve.
 - **location**: the location of the quadrature point on the underlying surface of the master curve given as two parameters.
 - **tangent**: The tangent of the trimming curve given within the parameter space of the underlying master surface element.
 - **location_slave**: This information is only required for coupling elements. The location of the corresponding evaluation point on the slave side given as two parameters within the slave surface element.
 - **tangent_slave**: This information is only required for coupling elements. The tangent of the trimming curve given within the parameter space of the underlying slave surface element (see Fig. 5).

Data interface—integration points

The last interface describes the *Integration domain* directly on level of the integration points including the evaluated shape functions which is the last layer of the in Fig. 9 described steps. This level requires the lowest amount of CAD-geometry data. In this data exchange all numerical data is precomputed. The exchange happens on the level of the *integration points* thus the data is similar as needed for the creation of all needed integration points as described in “Solver design” section.

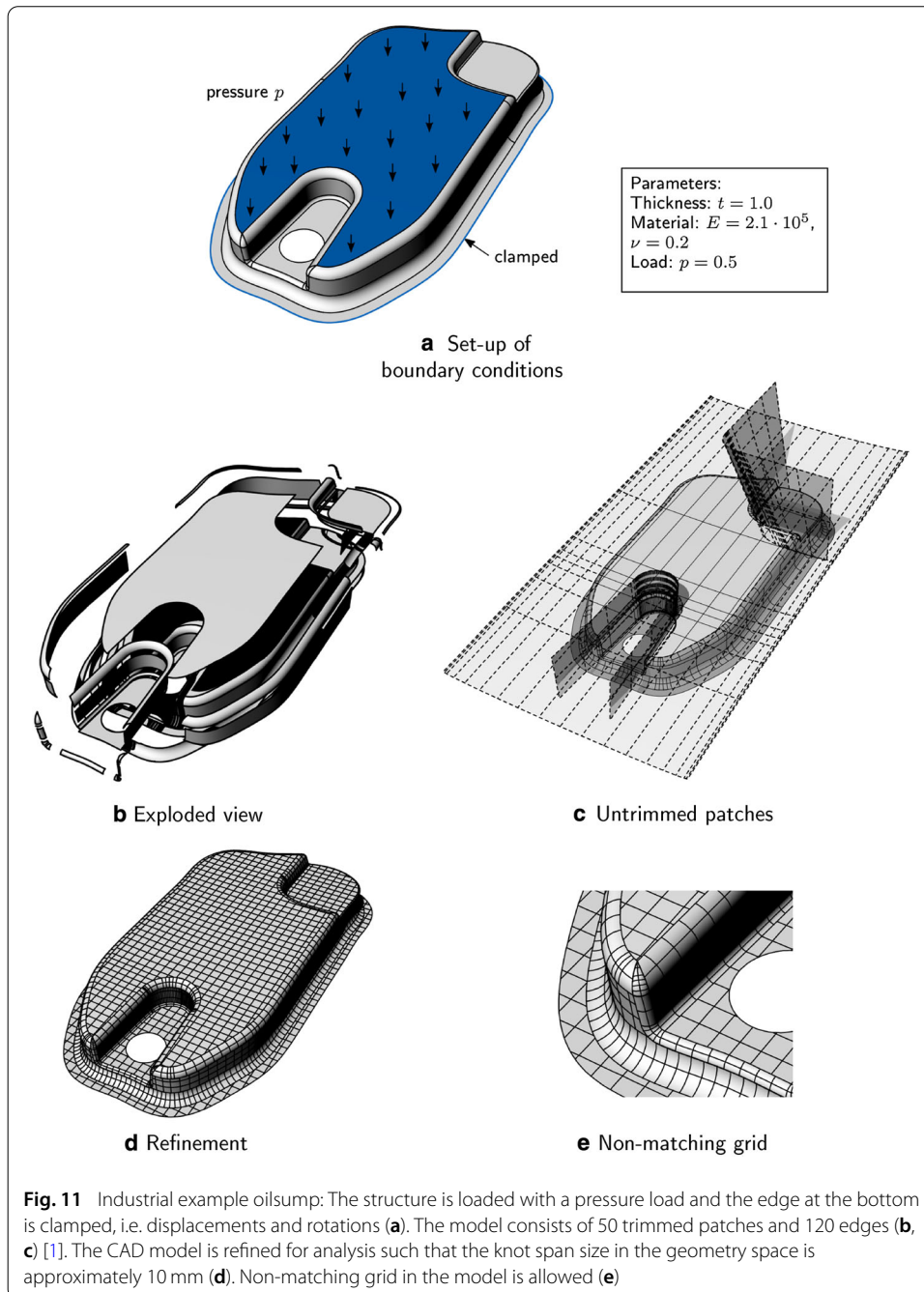
The step from the previous exchange format (see “Data interface—Integration domains” section) to this data can happen in the *Solver* or in the *CAD-Geometry Utilities*. If it happens in the latter the previous exchange will not be necessary. Since this exchange is based on a high amount of data a storing in text files is not advantageous. To avoid big data exchange

it is recommended to use more advanced interfaces as data bases. In general, this input changes w.r.t. the previously described that here all shape functions are already prepared for the computations. The exchange format from “Data interface—Integration domains” section describes the mathematical functions for the evaluation.

In the following, a proposed structure of the data base is described. The structure can vary from the used tools but the necessary data is given as follows:

- Surface integration points.
 - **ID** is used to address and structure the integration points.
 - **u, v** the location of the integration point in *Parameter space* of its underlying surface. It is necessary for the evaluation of solutions on integration points.
 - **shape functions** The pre-evaluation of shape functions allows to skip the geometry data as degrees and knot vectors.
 - **shape function derivatives** are important for many element formulations. It can also be extended to a higher order of derivatives.
 - **integration weight** is used for the integration. The computation is described in Eq. (8).
 - **control points** are needed to connect the integration point to its degrees of freedom.
- Coupling edge integration points.
 - **ID** see above.
 - **u, v** see above.
 - **shape functions master** see above.
 - **shape function derivatives master** see above.
 - **tangents master** are needed for the correct integration of the edge. See also Eq. (11). The tangents are described in *Parameter space* of the surface (see Fig. 5).
 - **shape functions slave** see above.
 - **shape function derivatives slave** see above.
 - **tangents slave** similar to tangents master.
 - **integration weight** on the edges. The computation is described in Eq. (13).
 - **control points master** see above.
 - **control points slave** see above.
- Edge integration points
 - **ID** see above.
 - **u, v** see above.
 - **shape functions** see above.
 - **shape function derivatives** see above.
 - **tangents** are needed for the correct integration of the edge. See also Eq. (11). The tangents are described in *Parameter space* of the surface (see Fig. 5).
 - **integration weight** see above.
 - **control points** see above.

Beside that, the control points related to the degrees of freedom are stored in additional lists to ensure the unique relation of the integration points to them.

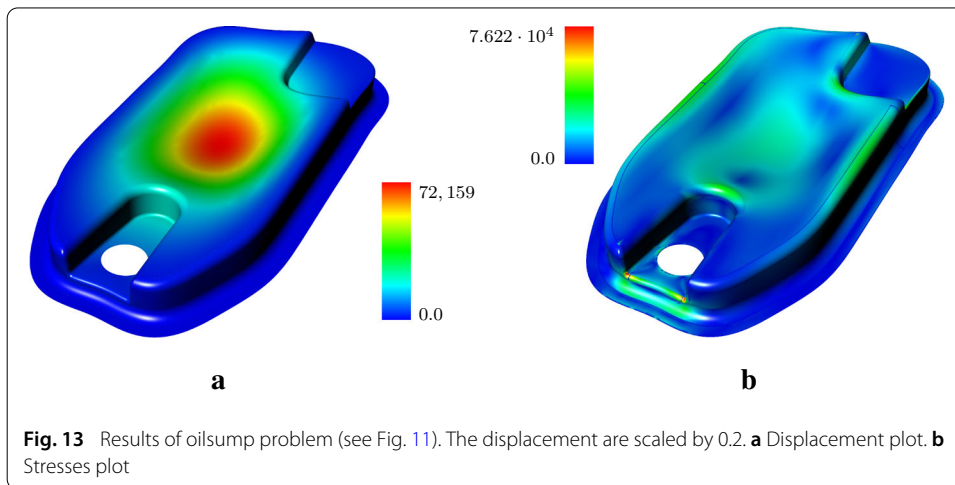
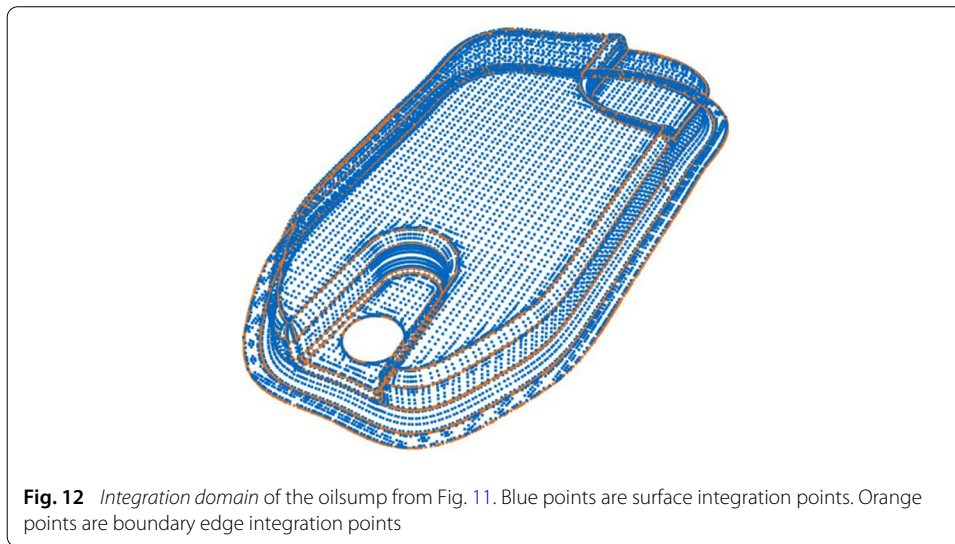


Simulation of real CAD models

The proposed format can not only be used for simple academic examples but also for real industrial problems. These examples have been solved within *KRATOS Multihysics* and *CARAT++* by using parallel solving and assembling routines. Moreover, the B-Rep elements using the penalty and Lagrange method are tested. Structural mechanics and form finding on the IBRA structures were performed.

Oilsump

The oilsump (by Breitenberger et al. [1]) is a multipatch structure. It is used to show the challenges that occur during IBRA simulations. The modeling and simulation process is



presented in Fig. 11. A pressure force (force in normal direction of the top patch) is applied on top of the structure. The boundary of it is clamped (fixed in x -, y -, and z -coordinates and blocking of rotations). The physical problem is shown in Fig. 11a.

The oilsump is designed with of 50 trimmed NURBS-patches. The patches are trimmed with outer and inner loops. In Fig. 11b is shown the exploded view of all trimmed patches of the oilsump. In Fig. 11c are shown the full patches without trimming. The later applied trimming loops are displayed as well. The problem contains 120 edges, either coupling edges or boundary edges.

All patches have to be coupled on their boundary trimming curves. Patch continuity is applied with the penalty method (see “Continuity between patches” section). Challenges in the coupling occur due to non-matching grids between the different patches (see Fig. 11e).

CAD structures usually need less control points to draw and simulate undeformed systems. Thus, it is needed to refine the structure to avoid extra stiffnesses due to too few degrees of freedom. In Fig. 11d the refined structure is shown. This geometry is used for simulation and postprocessing.

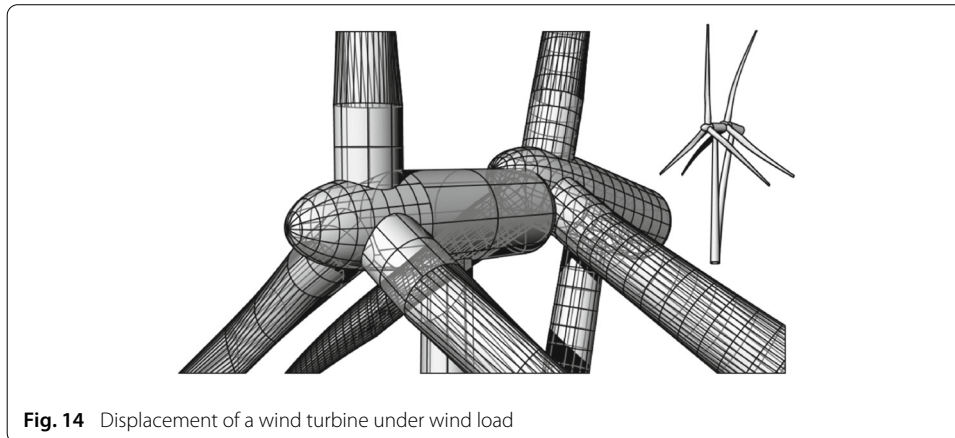


Fig. 14 Displacement of a wind turbine under wind load

The geometries as defined here are typically provided in the *Geometry* data interface (see Fig. 10). From this geometrical information a mathematical model will be produced. The outcome of the creation of the *Integration domain* is displayed in Fig. 12. Here, all integration points needed for structural mechanics problems are displayed. All integration points for surface element formulations are displayed in blue. The integration points of edges are displayed in orange. This model is exchanged to the solver. This is done within the *Integration domain* interface (see Fig. 10). This model only contains numerical information depending on degrees of freedom. There are no physics applied yet.

The structural analysis is now done on the mathematical model. Thus, physical and material entities have to be applied and the integration has to be fulfilled. In this case, the physical problem which is described in Fig. 11a will be solved on the numerical model. The solutions of the evaluation of the displacements and the occurring stresses are shown in Fig. 11. The displacements go from 0.0 to 72.159 mm. As expected the highest displacement is on top of the oilsump. The von Mises stresses go from 0.0 to $7.622 \cdot 10^4$. The highest stresses are mainly in the boundaries of the body.

In Fig. 13, one can see that continuity between the patches is provided on all layers. No gap or discontinuity can be seen.

Wind turbine

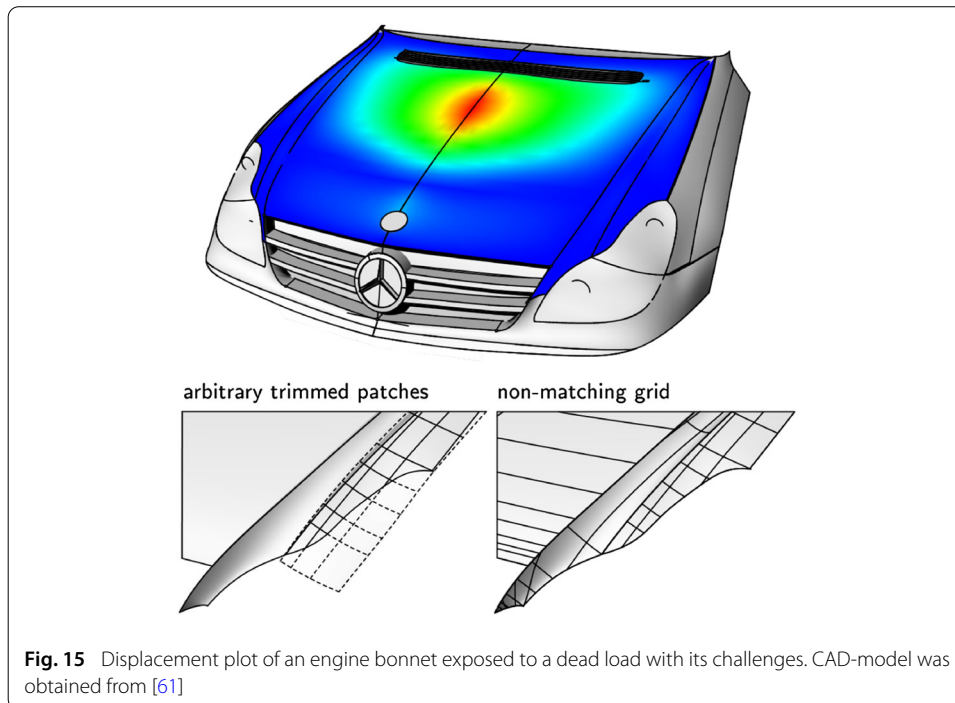
Furthermore another structure is presented to show the generality of the procedure. The wind turbine is, as the oilsump, a multipatch problem. The NURBS-patches are trimmed and non-matching grids complicate the problem.

In Fig. 14 is shown the initial geometry of the wind turbine and the deformed one. Within the simulation refinement operations are done on the NURBS-patches. The structure is described with isogeometric KL-shell elements.

Engine bonnet

The engine bonnet proves that real existing structures can be simulated with IBRA and the here described workflow.

The displacement plot of the structure is given in Fig. 15. Additionally, one of the NURBS-patches with trimming loop and the difficulties of the non-matching grids are displayed.



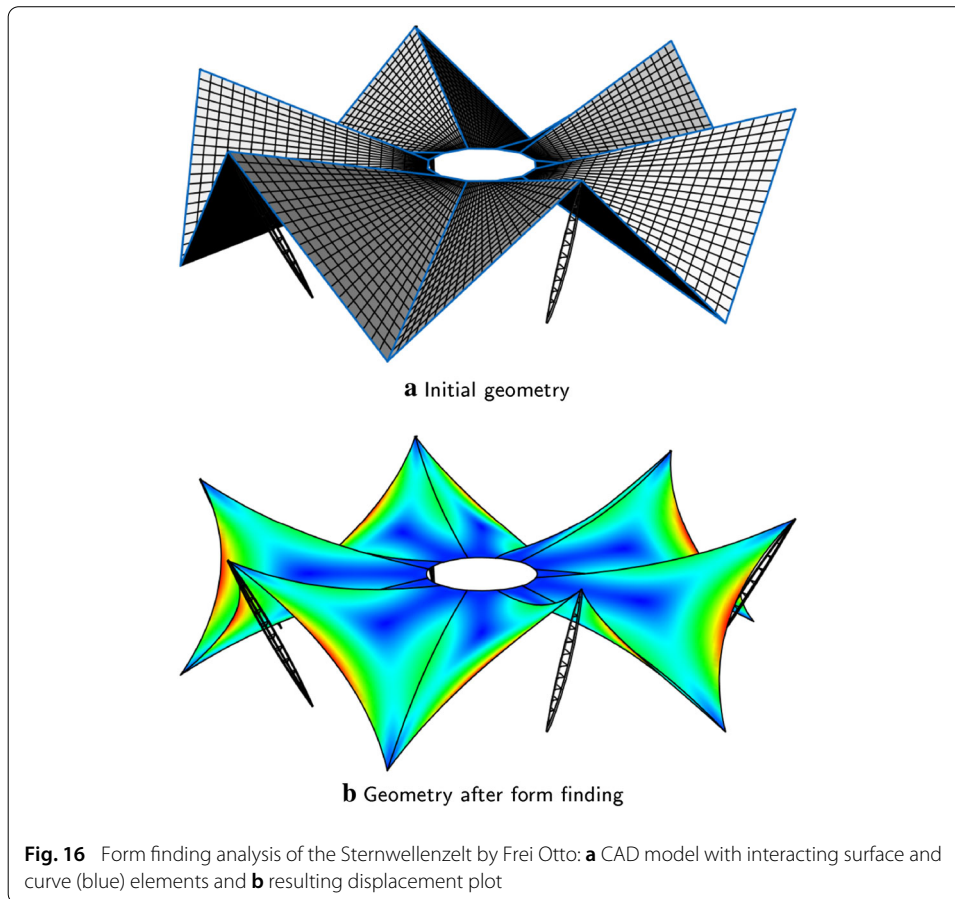
Membrane roof—Sternwellenzelt by Frei Otto

A CAD model, inspired by the Sternwellenzelt by Frei Otto, consisting of surface and curve element is presented in Fig. 16a. A form finding analysis was conducted for isogeometric membrane elements [29] under a given prestress. The edges are supported and reinforced by cables [29], which are defined by a given prestress, and elastic beam elements [62]. The technique of embedding withing NURBS as described in [29, 62] is applied for the edges cables. The analysis is based on the Updated reference strategy (URS) by [63]. The problem can be simulated with a quarter due to its symmetry. Here, this resulted in approximately 4500 degrees of freedom. The respective displacement plot was added to Fig. 16b.

Conclusion

A design-through-analysis workflow for isogeometric B-Rep analysis (IBRA) is proposed for bridging the gap between CAD and FEM. The design was elaborated with the explicit goal of being flexible to use different software environments containing structural analysis solvers with a bandwidth of isogeometric fidelity. This allows even the use of classical FE solvers for the simulation based on IBRA.

In “Isogeometric B-Rep analysis (IBRA)” section the procedure and numerical integration properties of IBRA are described. IBRA is directly considering the boundary representation of NURBS-entities within the analysis of thin-walled structures. This implies the intensive use of trimming and coupling information coming from real CAD models which follows the principles of embedding physical objects into existing background parametrizations. One challenging part in the simulation procedure is the handling of poorly defined geometries in computational structural analysis. The treatment of these so called “dirty” geometries is very important for the quality of the solutions and thus, shall be part of ongoing and future research.



Special aspects need to be considered for this approach in the design of suitable structural analysis solvers allowing a seamless design-through-analysis workflow. The full scope of possible solver designs and the consequently needed efforts of extracting NURBS-geometries and its associated data are shown in “Solver design” section. The solver properties and its specific different levels of CAD-data exchange ranging from full geometries to pure numerical information are explained. The lowest level of CAD-geometry-use is described by a new *meshless* approach, which allows the simulation without additional NURBS-based functions.

A workflow is derived which allows a distinction of CAD-geometries and numerical integration of models from the IBRA specifications. It is designed to reach maximal flexibility for possible exchange of simulation tools and it describes the entire way from designing to solving and finally towards processing solutions. This workflow is explained in “Design-through-analysis workflow” section. To allow this exchangeability, interfaces and new IBRA exchange formats from the level of CAD-geometries towards integration points with only numerically evaluated data are defined in “IBRA exchange format” section.

The design-through-analysis workflow has been tested successfully with evaluation of many different engineering problems. Some representative examples involving complex geometries in the field of structural mechanics and form finding are shown in “Simulation of real CAD models” section. In Appendix A some basic and precisely documented examples of the CAD-geometry exchange interface are provided. Those examples are designed

to show the maximal bandwidth of NURBS-entities. Some exchange properties on integration domain level are described in detail in Appendix B. Those examples demonstrate the full range of usability of the defined format.

It has been proven successfully that the proposed design-through-analysis workflow (accompanied by the identification and definition of respective interface data) enables the simulation of complex structure models based on IBRA with solvers of very distinct isogeometric fidelity. This approach allows to spread the scope of IBRA for real industrial use with fully CAD-designed problems.

Abbreviations

AGIP: adaptive Gaussian integration procedure; API: application programming interface; CAD: computer aided design; B-Rep: boundary representation; FE: finite elements; IGA: isogeometric analysis; IBRA: isogeometric B-Rep analysis; NURBS: non-uniform rational B-Spline.

Authors' contributions

All authors have prepared the manuscript. All authors read and approved the final manuscript.

Author details

¹Lehrstuhl für Statik, Technische Universität München, Arcisstr. 21, 80333 Munich, Germany, ²International Center for Numerical Methods in Engineering (CIMNE), Technical University of Catalonia, Campus Norte UPC, 08034 Barcelona, Spain.

Acknowledgements

This work was supported by "Zentrales Innovationsprogramm Mittelstand" of the *Bundesministerium für Wirtschaft und Energie (BMWi)* as part of the project "Erforschung der mathematisch-methodischen Grundlagen zur Nutzung der Isogeometrischen Analyse in der Leichtbau-Planung" (ZF4066102BZ6) and by Deutsche Forschungsgemeinschaft (DFG) as part of the SPP project "Leicht Bauen mit Beton" (BL 306/23-1) and the project "Eine Methode zur effizienten Simulation großer Mauerwerksscheiben unter exzentrischer und/oder zyklischer biaxialer Beanspruchung auf Grundlagewirklichkeitsnaher Kleinkörperversuche" (BL306/34-1). The support is gratefully acknowledged.

Competing interests

The authors declare that they have no competing interests.

Availability of data and materials

The used software is [55,56].

Consent for publication

Non-exclusive rights of use.

Ethics approval and consent to participate

Not applicable.

Funding

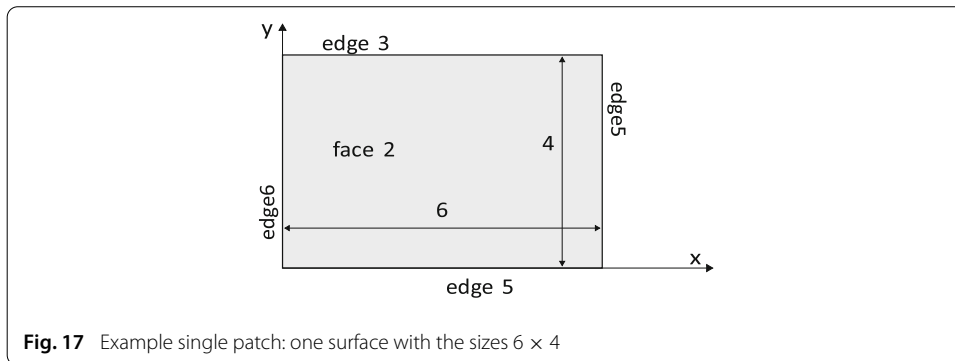
Deutsche Forschungsgemeinschaft (DFG), "Eine Methode zur effizienten Simulation großer Mauerwerksscheiben unter exzentrischer und/oder zyklischer biaxialer Beanspruchung auf Grundlagewirklichkeitsnaher Kleinkörperversuche" (BL306/34-1) Deutsche Forschungsgemeinschaft (DFG), "Leicht Bauen mit Beton" (BL 306/23-1) "Zentrales Innovationsprogramm Mittelstand" of the *Bundesministerium für Wirtschaft und Energie (BMWi)*, "Erforschung der mathematisch-methodischen Grundlagen zur Nutzung der Isogeometrischen Analyse in der Leichtbau-Planung" (ZF4066102BZ6).

Appendix A: Exchange format description for *Geometry*

This section shows some basic examples to illustrate the *Geometry* data format. The problems give exemplary surface and beam structures for big industrial geometries. Also coupling between patches and between beams are shown exemplary.

Single patch

This example shows the representation of an untrimmed rectangular patch (Fig. 17) in the *Geometry* exchange format (Listing 6).



```
{
  "brep": {
    "brep_id": 1,
    "faces": [
      {
        "brep_id": 2,
        "swapped_surface_normal": false,
        "surface": {
          "is_trimmed": true,
          "is_rational": false,
          "degrees": [ 1, 1 ],
          "knot_vectors": [
            [ 0, 0, 6, 6 ], [ 0, 0, 4, 4 ]
          ],
          "control_points": [
            [ 1, [ 0, 4, 0, 1 ] ], [ 2, [ 6, 4, 0, 1 ] ],
            [ 3, [ 0, 0, 0, 1 ] ], [ 4, [ 6, 0, 0, 1 ] ]
          ]
        },
        "boundary_loops": [
          {
            "loop_type": "outer",
            "trimming_curves": [
              {
                "trim_index": 0,
                "curve_direction": true,
                "parameter_curve": {
                  "is_rational": false,
                  "degree": 1,
                  "knot_vector": [ 0, 0, 6, 6 ],
                  "active_range": [ 0, 6 ],
                  "control_points": [
                    [ 0, 0, 0, 1 ], [ 6, 0, 0, 1 ]
                  ]
                }
              },
              {
                "trim_index": 1,
                "curve_direction": true,
                "parameter_curve": {
                  "is_rational": false,
                  "degree": 1,
                  "knot_vector": [ 0, 0, 4, 4 ],
                  "active_range": [ 0, 4 ],
                  "control_points": [
                    [ 6, 0, 0, 1 ], [ 6, 4, 0, 1 ]
                  ]
                }
              },
              {
                "trim_index": 2,
                "curve_direction": true,
                "parameter_curve": {
                  "is_rational": false,
                  "degree": 1,
                  "knot_vector": [ 0, 0, 6, 6 ],
                  "active_range": [ 0, 6 ],
                  "control_points": [
                    [ 6, 4, 0, 1 ], [ 0, 4, 0, 1 ]
                  ]
                }
              },
              {
                "trim_index": 3,
                "curve_direction": true,
                "parameter_curve": {
                  "is_rational": false,
                  "degree": 1,
                  "knot_vector": [ 0, 0, 4, 4 ],
                  "active_range": [ 0, 4 ],
                  "control_points": [
                    [ 0, 4, 0, 1 ], [ 0, 0, 0, 1 ]
                  ]
                }
              }
            ]
          }
        ]
      }
    ],
    "edges": [
      {
        "brep_id": 3,
        "3d_curve": {
          "degree": 1,
          "knot_vector": [ 0, 0, 6, 6 ],
          "control_points": [
            [ 5, [ 0, 4, 0, 1 ] ], [ 6, [ 6, 4, 0, 1 ] ]
          ],
          "active_range": [ 0, 6 ]
        },
        "topology": [
          {
            "brep_id": 2,
            "trim_index": 0,
            "relative_direction": true
          }
        ]
      },
      {
        "brep_id": 4,
        "3d_curve": {
          "degree": 1,
          "knot_vector": [ 0, 0, 4, 4 ],
          "control_points": [
            [ 7, [ 6, 4, 0, 1 ] ], [ 8, [ 6, 0, 0, 1 ] ]
          ],
          "active_range": [ 0, 4 ]
        },
        "topology": [
          {
            "brep_id": 2,
            "trim_index": 1,
            "relative_direction": true
          }
        ]
      }
    ]
  }
}
```

```

{"brep_id": 5,
 "3d_curve": {
  "degree": 1,
  "knot_vector": [-6, -6, 0, 0],
  "control_points": [
    [9, [ 6, 0, 0, 1 ]],[10,[ 0, 0, 0, 1 ]]],
  "active_range": [ -6, 0 ]},
 "topology": [{
  "brep_id": 2,
  "trim_index": 2,
  "relative_direction": true}],
 {"brep_id": 6,
 "3d_curve": {
  "degree": 1,
  "knot_vector": [-4, -4, 0, 0],
  "control_points": [
    [11,[ 0, 0, 0, 1 ]],[12,[ 0, 4, 0, 1 ]]],
  "active_range": [ -4, 0 ]},
 "topology": [{
  "brep_id": 2,
  "trim_index": 3,
  "relative_direction": true}]
}]
}
    
```

Listing 6 Example single patch: the *Geometry* exchange format provides the data for the geometry in Fig. 17

Two trimmed patches

In this example two trimmed patches are coupled by a common edge. Fig. 18 and Listing 7 show the geometry and the corresponding representation in the *Geometry* exchange format.

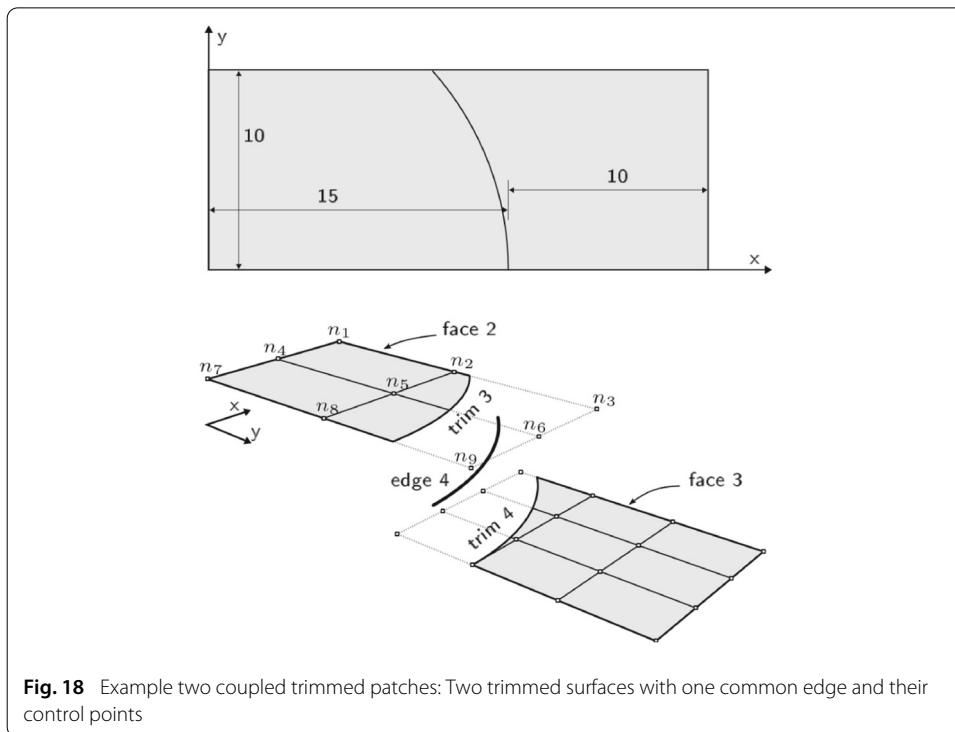


Fig. 18 Example two coupled trimmed patches: Two trimmed surfaces with one common edge and their control points

```

{
  "brep": {
    "brep_id": 1,
    "faces": [
      {
        "brep_id": 2,
        "swapped_surface_normal": false,
        "surface": {
          "is_trimmed": true,
          "is_rational": false,
          "degrees": [2, 2],
          "knot_vectors": [[0, 0, 0, 20, 20, 20],[0, 0, 0, 10, 10, 10]],
          "control_points": [
            [1,[0, 10, 0, 1]], [2,[10, 10, 0, 1]], [3,[20, 10, 0, 1]],
            [4,[0, 5, 0, 1]], [5,[10, 5, 0, 1]], [6,[20, 5, 0, 1]],
            [7,[0, 0, 0, 1]], [8,[10, 0, 0, 1]], [9,[20, 0, 0, 1]]],
          "boundary_loops": [
            {
              "loop_type": "outer",
              "trimming_curves": [
                {
                  "trim_index": 0,
                  "curve_direction": true,
                  "parameter_curve": {
                    "is_rational": false,
                    "degree": 1,
                    "knot_vector": [0, 0, 15, 15],
                    "active_range": [0, 15],
                    "control_points": [[15, 10, 0, 1],[0, 10, 0, 1]] }},
                {
                  "trim_index": 1,
                  "curve_direction": true,
                  "parameter_curve": {
                    "is_rational": false,
                    "degree": 1,
                    "knot_vector": [0, 0, 10, 10],
                    "active_range": [0, 10],
                    "control_points": [[0, 10, 0, 1],[0, 0, 0, 1]] }},
                {
                  "trim_index": 2,
                  "curve_direction": true,
                  "parameter_curve": {
                    "is_rational": false,
                    "degree": 1,
                    "knot_vector": [0, 0, 11.18, 11.18],
                    "active_range": [0, 11.18],
                    "control_points": [[0, 0, 0, 1],[11.18, 0, 0, 1]] }},
                {
                  "trim_index": 3,
                  "curve_direction": true,
                  "parameter_curve": {
                    "is_rational": false,
                    "degree": 3,
                    "knot_vector": [-10.95, -10.95, -10.95, -10.95, 0, 0, 0, 0],
                    "active_range": [-10.95, 0],
                    "control_points": [
                      [11.18, 0, 0, 1],[13.64, 2.75, 0, 1],
                      [15, 6.31, 0, 1],[15, 10, 0, 1]] }
                ]
              }
            ]
          }
        },
        "brep_id": 3,
        "swapped_surface_normal": false,
        "surface": {
          "is_trimmed": true,
          "is_rational": false,
          "degrees": [1, 1],
          "knot_vectors": [
            [0, 0, 5, 10, 15, 15],
            [0, 0, 3.33, 6.66, 10, 10]],
          "control_points": [
            [10,[10, 10, 0, 1]], [11,[15, 10, 0, 1]],
            [12,[20, 10, 0, 1]], [13,[25, 10, 0, 1]],
            [14,[10, 6.66, 0, 1]], [15,[15, 6.66, 0, 1]],
            [16,[20, 6.66, 0, 1]], [17,[25, 6.66, 0, 1]],
            [18,[10, 3.33, 0, 1]], [19,[15, 3.33, 0, 1]],
            [20,[20, 3.33, 0, 1]], [21,[25, 3.33, 0, 1]],
            [22,[10, 0, 0, 1]], [23,[15, 0, 0, 1]],
            [24,[20, 0, 0, 1]], [25,[25, 0, 0, 1]]
          ],
          "boundary_loops": [
            {
              "loop_type": "outer",
              "trimming_curves": [
                {
                  "trim_index": 4,
                  "curve_direction": true,
                  "parameter_curve": {
                    "is_rational": false,
                    "degree": 3,
                    "knot_vector": [0, 0, 0, 0, 10.95, 10.95, 10.95, 10.95],
                    "active_range": [0, 10.95],
                    "control_points": [
                      [5, 10, 0, 1],[5, 6.31, 0, 1],
                      [3.64, 2.75, 0, 1],[1.18, 0, 0, 1]] }},
                {
                  "trim_index": 5,
                  "curve_direction": true,
                  "parameter_curve": {
                    "is_rational": false,
                    "degree": 1,
                    "knot_vector": [0, 0, 13.82, 13.82],
                    "active_range": [0, 13.82],
                    "control_points": [[1.18, 0, 0, 1],[15, 0, 0, 1]] }},
                {
                  "trim_index": 6,
                  "curve_direction": true,
                  "parameter_curve": {
                    "is_rational": false,
                    "degree": 1,
                    "knot_vector": [0, 0, 10, 10],
                    "active_range": [0, 10],
                    "control_points": [[15, 0, 0, 1],[15, 10, 0, 1]] }},
                {
                  "trim_index": 7,
                  "curve_direction": true,

```

```

"parameter_curve": {
  "is_rational": false,
  "degree": 1,
  "knot_vector": [0, 0, 10, 10],
  "active_range": [0, 10],
  "control_points": [[15, 10, 0, 1 ],[5, 10, 0, 1 ]] }
}]
}],
"edges": [{
  "brep_id": 4,
  "3d_curve": {
    "degree": 3,
    "knot_vector": [0, 0, 0, 0, 10.95, 10.95, 10.95, 10.95],
    "control_points": [
      [26,[15, 0, 0, 1]], [27,[15, 3.69, 0, 1]],
      [28,[13.64, 7.25, 0, 1]], [29,[11.18, 10, 0, 1]]],
    "active_range": [ 0, 10.95 ]},
  "topology": [{
    "brep_id": 2,
    "trim_index": 3,
    "relative_direction": false },
    {"brep_id": 3,
    "trim_index": 4,
    "relative_direction": true }]},
  {"brep_id": 5,
  "3d_curve": {
    "degree": 2,
    "knot_vector": [0, 0, 0, 11.18, 11.18, 11.18],
    "control_points": [
      [30,[0, 10, 0, 1]], [31,[5.59, 10, 0, 1]], [32,[11.18, 10, 0, 1]]],
    "active_range": [ 0, 11.18 ]},
  "topology": [{
    "brep_id": 2,
    "trim_index": 2,
    "relative_direction": true }]},
  {"brep_id": 6,
  "3d_curve": {
    "degree": 2,
    "knot_vector": [-10, -10, -10, 0, 0, 0],
    "control_points": [
      [33,[0, 0, 0, 1]], [34,[0, 5, 0, 1]], [35,[0, 10, 0, 1]],
    "active_range": [-10, 0 ]},
  "topology": [{
    "brep_id": 2,
    "trim_index": 1,
    "relative_direction": true }]},
  {"brep_id": 7,
  "3d_curve": {
    "degree": 2,
    "knot_vector": [-15, -15, -15, 0, 0, 0],
    "control_points": [
      [36,[15, 0, 0, 1]], [37,[7.5, 0, 0, 1 ]], [38,[0, 0, 0, 1]]],
    "active_range": [ -15, 0 ]},
  "topology": [{
    "brep_id": 2,
    "trim_index": 0,
    "relative_direction": true }]},
  {"brep_id": 8,
  "3d_curve": {
    "degree": 1,
    "knot_vector": [0, 0, 3.33, 6.66, 10, 10],
    "control_points": [
      [39,[25, 10, 0, 1 ]], [40,[25, 6.66, 0, 1 ]],
      [41,[25, 3.33, 0, 1 ]], [42,[25, 0, 0, 1 ]]],
    "active_range": [ 0, 10 ]},
  "topology": [{
    "brep_id": 3,
    "trim_index": 6,
    "relative_direction": true }]},
  {"brep_id": 9,
  "3d_curve": {
    "degree": 1,
    "knot_vector": [-15, -15, -10, -5, -5],
    "control_points": [
      [43,[25, 0, 0, 1]], [44,[20, 0, 0, 1]], [45,[15, 0, 0, 1]]],
    "active_range": [ -15, -5 ]},
  "topology": [{
    "brep_id": 3,
    "trim_index": 7,
    "relative_direction": true }]},
  {"brep_id": 10,
  "3d_curve": {
    "degree": 1,
    "knot_vector": [1.18, 1.18, 5, 10, 15, 15],
    "control_points": [
      [46,[11.18, 10, 0, 1]], [47,[15, 10, 0, 1]],
      [48,[20, 10, 0, 1]], [49,[25, 10, 0, 1]]],
    "active_range": [ 1.18, 15 ]},
  "topology": [{
    "brep_id": 3,
    "trim_index": 5,
    "relative_direction": true }}
  ]}
}]
}

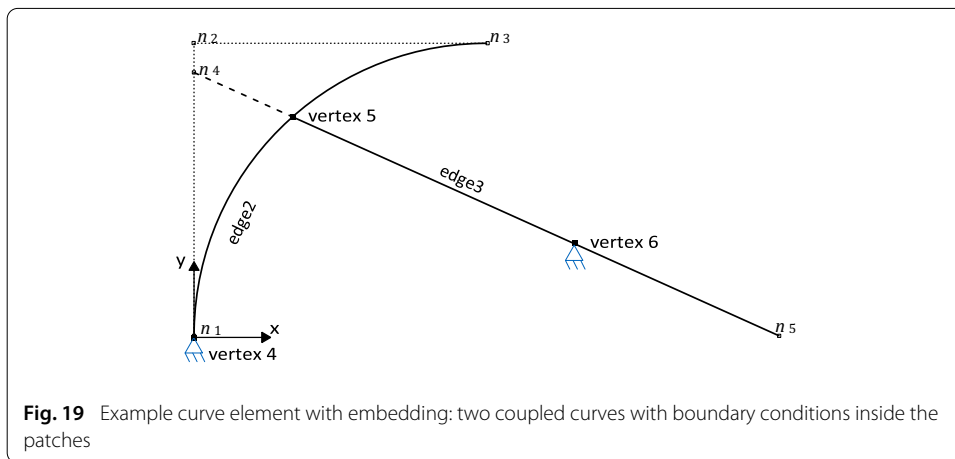
```

Listing 7 Two trimmed patches: The IBRA exchange format provides the data for the geometry in Fig. 18

Curve elements and embedding

Curve elements can also be represented by the proposed data format. Therefore, the edge-list is used. The curve is described by the 3D-curve. In contrast to the B-Rep edge, topology information is missing if it is an independent curve. The curve can be trimmed by the active range. Two coupled curves are shown in Fig. 19. Embedded points are used in the geometry file for having something to address in the project parameter and material files for the depicted boundary conditions.

Vertices are in difference to edges only written if they are needed for boundary conditions, such as coupling, supports or loads. This holds for the position in the parameter space (i.e. embedded_points-list) as well as the geometry space (i.e. vertices-list). Listing 8 shows the resulting geometry-interface.




```

{
  "brep": [
    {
      "brep_id": 1,
      "faces": [],
      "edges": [
        {
          "brep_id": 2,
          "3d_curve": {
            "degree": 2,
            "knot_vector": [0, 0, 0, 1.570796, 1.570796, 1.570796],
            "control_points": [
              [1,[0, 0, 0, 1]], [2,[0, 10, 0, 0.707]], [3,[10, 10, 0, 1]]],
            "active_range": [ 0, 1.570796 ]},
          "topology": [],
          "embedded_points": [
            { "trim_index": 1,
              "point": [0, 0, 0 ]},
            { "trim_index": 2,
              "point": [8.457079, 0, 0 ]}],
        }
      ],
      "brep_id": 3,
      "3d_curve": {
        "degree": 1,
        "knot_vector": [0, 0, 10, 10],
        "control_points": [
          [4,[0, 9, 0, 1]], [5,[20, 0, 0, 1]]],
        "active_range": [1.683992, 10]},
        "topology": [],
        "embedded_points": [
          { "trim_index": 3,
            "point": [1.683992, 0, 0]},
          { "trim_index": 4,
            "point": [6.5, 0, 0]}]
      },
      "vertices": [
        {
          "brep_id": 4,
          "3d_point": [6,[0, 0, 0, 1 ]],
          "topology": [
            {
              "brep_id": 2,
              "trim_index": 1}],
        },
        {
          "brep_id": 5,
          "3d_point": [7,[ 3.367983, 7.484407, 0, 1 ]],
          "topology": [
            {
              "brep_id": 2,
              "trim_index": 2},
            {
              "brep_id": 3,
              "trim_index": 3}],
        },
        {
          "brep_id": 6,
          "3d_point": [8,[ 13, 3.15, 0, 1 ]],
          "topology": [
            {
              "brep_id": 3,
              "trim_index": 4}],
        }
      ]
    }
  ]
}

```

Listing 8 Example curve element with embedding: the IBRA exchange format provides the data for the geometry in Fig. 19

Coupled surface and curve

The following example describes how to couple 1D and 2D-elements with each other. Figure 20 shows the geometry. One edge of the single patch in “Single patch” section is coupled to a NURBS curve.

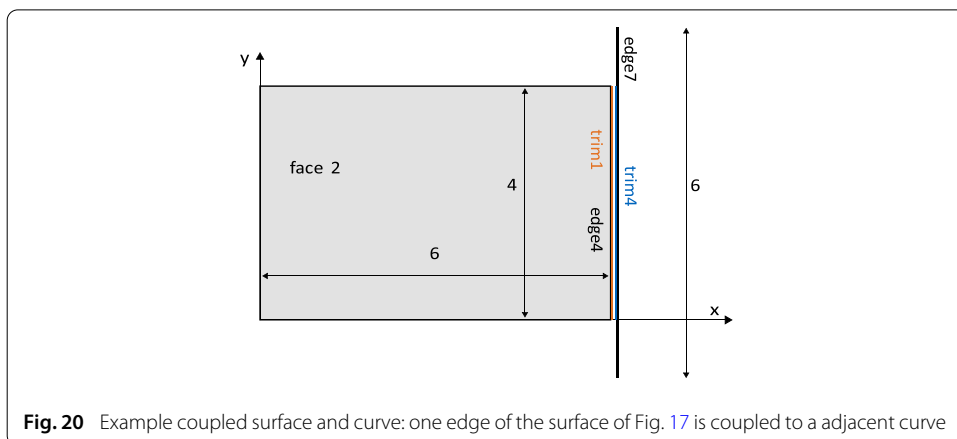


Fig. 20 Example coupled surface and curve: one edge of the surface of Fig. 17 is coupled to an adjacent curve

An additional curve is added to the list of edges for the NURBS curve. The topology information can be added to the already existing edge which correspond to respective boundary of the edge. The resulting geometry-file is exemplified in Listing 9.

```

{
  "brep": {
    "brep_id": 1,
    "faces": [
      {
        "brep_id": 2,
        "swapped_surface_normal": false,
        "surface": {
          "is_trimmed": true,
          "is_rational": false,
          "degrees": [ 1, 1 ],
          "knot_vectors": [
            [ 0, 0, 6, 6 ], [ 0, 0, 4, 4 ] ],
          "control_points": [
            [ 1, [ 0, 4, 0, 1 ] ], [ 2, [ 6, 4, 0, 1 ] ],
            [ 3, [ 0, 0, 0, 1 ] ], [ 4, [ 6, 0, 0, 1 ] ] ] ],
          "boundary_loops": [
            {
              "loop_type": "outer",
              "trimming_curves": [
                {
                  "trim_index": 0,
                  "curve_direction": true,
                  "parameter_curve": {
                    "is_rational": false,
                    "degree": 1,
                    "knot_vector": [ 0, 0, 6, 6 ],
                    "active_range": [ 0, 6 ],
                    "control_points": [
                      [ 0, 0, 0, 1 ], [ 6, 0, 0, 1 ] ] ] },
                {
                  "trim_index": 1,
                  "curve_direction": true,
                  "parameter_curve": {
                    "is_rational": false,
                    "degree": 1,
                    "knot_vector": [ 0, 0, 4, 4 ],
                    "active_range": [ 0, 4 ],
                    "control_points": [
                      [ 6, 0, 0, 1 ], [ 6, 4, 0, 1 ] ] ] },
                {
                  "trim_index": 2,
                  "curve_direction": true,
                  "parameter_curve": {
                    "is_rational": false,
                    "degree": 1,
                    "knot_vector": [ 0, 0, 6, 6 ],
                    "active_range": [ 0, 6 ],
                    "control_points": [
                      [ 6, 4, 0, 1 ], [ 0, 4, 0, 1 ] ] ] },
                {
                  "trim_index": 3,
                  "curve_direction": true,
                  "parameter_curve": {
                    "is_rational": false,
                    "degree": 1,
                    "knot_vector": [ 0, 0, 4, 4 ],
                    "active_range": [ 0, 4 ],
                    "control_points": [
                      [ 0, 4, 0, 1 ], [ 0, 0, 0, 1 ] ] ] }
              ]
            }
          ]
        }
      ]
    },
    "edges": [
      {
        "brep_id": 3,
        "3d_curve": {
          "degree": 1,
          "knot_vector": [ 0, 0, 6, 6 ],
          "control_points": [
            [ 5, [ 0, 4, 0, 1 ] ], [ 6, [ 6, 4, 0, 1 ] ] ],
          "active_range": [ 0, 6 ] },
        "topology": [
          {
            "brep_id": 2,
            "trim_index": 0,
            "relative_direction": true
          }
        ]
      },
      {
        "brep_id": 4,
        "3d_curve": {
          "degree": 1,
          "knot_vector": [ 0, 0, 4, 4 ],
          "control_points": [
            [ 7, [ 6, 4, 0, 1 ] ], [ 8, [ 6, 0, 0, 1 ] ] ],
          "active_range": [ 0, 4 ] },
        "topology": [
          {
            "brep_id": 2,
            "trim_index": 1,
            "relative_direction": true
          },
          {
            "brep_id": 7,
            "trim_index": 4,
            "relative_direction": false
          }
        ]
      },
      {
        "brep_id": 5,
        "3d_curve": {
          "degree": 1,
          "knot_vector": [ -6, -6, 0, 0 ],
          "control_points": [
            [ 9, [ 6, 0, 0, 1 ] ], [ 10, [ 0, 0, 0, 1 ] ] ],
          "active_range": [ -6, 0 ] },
        "topology": [
          {
            "brep_id": 2,
            "trim_index": 2,
            "relative_direction": true
          }
        ]
      },
      {
        "brep_id": 6,
        "3d_curve": {

```

```

"degree": 1,
"knot_vector": [-4, -4, 0, 0],
"control_points": [
  [11,[ 0, 0, 0, 1 ]],[12,[ 0, 4, 0, 1 ]]],
"active_range": [ -4, 0 ]],
"topology": [{
  "brep_id": 2,
  "trim_index": 3,
  "relative_direction": true}]
}],
{"brep_id": 7,
"3d_curve": {
  "degree": 1,
  "knot_vector": [0, 0, 6, 6],
  "control_points": [
    [13,[ 6, -1, 0, 1 ]],[14,[ 6, 5, 0, 1 ]]],
  "active_range": [ 0, 6 ]},
"topology": [],
"embedded_curves": [
  {"trim_index": 4,
  "curve_direction": true,
  "parameter_curve": {
    "is_rational": false,
    "degree": 1,
    "knot_vector": [0, 0, 4, 4],
    "active_range": [ 0, 4 ],
    "control_points": [
      [ 1, 0, 0, 1 ],[ 5, 0, 0, 1 ]]}
  }
]}
}
}
}

```

Listing 9 Example coupling of a surface with a curve: the IBRA exchange format provides the data for the geometry in Fig. 20

Appendix B: Exchange format description for integration domains

Within this section, some examples are presented which illustrate how the data of the IBRA exchange format can be used. The examples are supposed to be very simple and illustrative. Moreover, some relevant values like the length of a B-Rep edge are computed exemplary. These examples can be used as guide lines for testing new implementations within your FE solver.

Single patch

Figure 21 shows the first example which consists of one quadratic patch with 9 control points and 4 quadrature points. Please note that this would represent an under integration of the element. The corresponding exchange format data (see also “Data interface—Integration domains” section) is given in Listing 10. The control points (see nodes) have the ids from 1 to 9. The 2D element has the id 14 which consists of 4 quadrature points with the ids 10–13. The weighting factor for all quadrature points corresponds to 6.0.

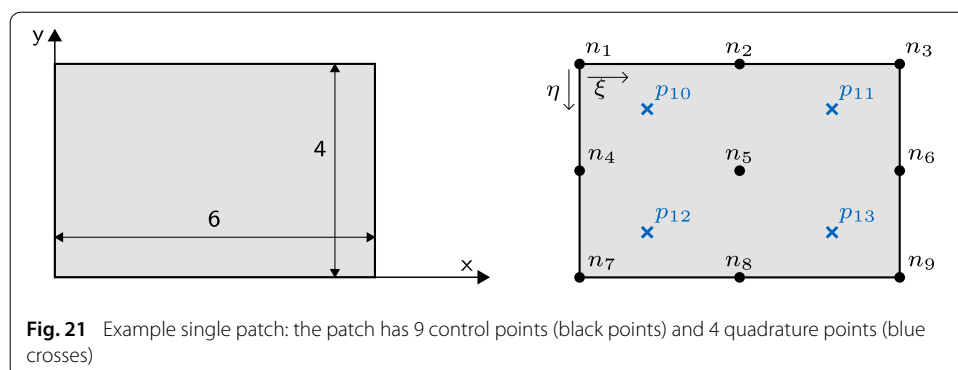


Fig. 21 Example single patch: the patch has 9 control points (black points) and 4 quadrature points (blue crosses)

```

{
  "nodes": [
    [1, [0, 4, 0, 1]], [2, [3, 4, 0, 1]], [3, [6, 4, 0, 1]],
    [4, [0, 2, 0, 1]], [5, [3, 2, 0, 1]], [6, [6, 2, 0, 1]],
    [7, [0, 0, 0, 1]], [8, [3, 0, 0, 1]], [9, [6, 0, 0, 1]]
  ],
  "2d_elements": [
    [1, ...Patch-ID
      [
        [14, ...Element-ID
          [2, 2], [[0.0, 0.0, 0.0, 6.0, 6.0, 6.0], [0.0, 0.0, 0.0, 4.0, 4.0, 4.0]],
          [1, 2, 3, 4, 5, 6, 7, 8, 9], true,
          [[10, 6.0, [1.2679, 0.8452]], [11, 6.0, [4.7320, 0.8452]],
           [12, 6.0, [1.2679, 3.1547]], [13, 6.0, [4.7320, 3.1547]]]
          ] ...end Quadrature Points
        ] ...end Element
      ] ...end List of Elements
    ] ...end Patch
  ]
}

```

Listing 10 Example single patch: the IBRA exchange format provides the data for the geometry in Fig. 21

Table 1 Example single patch: shape function values $N_i(\xi, \eta)$ for each quadrature point (see Fig. 21)

(a) Quadrature point p_{10}			
0.38689	0.20733	0.02777	
0.20733	0.11111	0.01488	
0.02777	0.01488	0.00199	
(b) Quadrature point p_{11}			
0.02777	0.20733	0.38689	
0.01488	0.11111	0.20733	
0.00199	0.01488	0.02777	
(c) Quadrature point p_{12}			
0.02777	0.01488	0.00199	
0.20733	0.11111	0.01488	
0.38689	0.20733	0.02777	
(d) Quadrature point p_{13}			
0.00199	0.01488	0.02777	
0.01488	0.11111	0.20733	
0.02777	0.20733	0.38689	

The shape functions values are computed and displayed in Table 1 for this example . As the quadrature points are located double symmetric they lead to the same function values just ordered differently.

Table 2 shows in addition the first and second derivatives evaluated for quadrature point p_{10} . These values can be used to check for a correct implementation of the shape functions.

Computation of Jacobian J_1

In the following, the Jacobian J_1 (see also Eq. 6) for the quadrature point p_{10} is computed. This Jacobian represents the mapping from *geometry* to *parameter space*.

The general formula of the Jacobian **J** is given by

$$\mathbf{J} = \begin{bmatrix} \frac{\partial x}{\partial \xi} & \frac{\partial x}{\partial \eta} \\ \frac{\partial y}{\partial \xi} & \frac{\partial y}{\partial \eta} \\ \frac{\partial z}{\partial \xi} & \frac{\partial z}{\partial \eta} \end{bmatrix} \tag{B.1}$$

Table 2 Example single patch: first and second order derivatives of the shape functions $N_i(\xi, \eta)$ evaluated on the quadrature point p_{10} (see Fig. 21) with the parameters $\xi = 1.2679$ and $\eta = 0.8452$ (see Listing 10)

cp id	$\frac{\partial N}{\partial \xi}$	$\frac{\partial N}{\partial \eta}$	$\frac{\partial N}{\partial \xi^2}$	$\frac{\partial N}{\partial \eta^2}$	$\frac{\partial N}{\partial \xi \cdot \partial \eta}$
1	-0.16352	-0.24528	0.03455	0.07775	0.10366
2	0.11970	-0.13144	-0.06911	0.04166	-0.07589
3	0.04381	-0.01761	0.03455	0.00558	-0.02777
4	-0.08763	0.17955	0.01851	-0.15550	-0.07589
5	0.06415	0.09622	-0.03703	-0.08333	0.05555
6	0.02348	0.01289	0.01851	-0.01116	0.02033
7	-0.01174	0.06572	0.00248	0.07775	-0.02777
8	0.00859	0.03522	-0.00496	0.04166	0.02033
9	0.003145	0.004718	0.002481	0.005582	0.007443

where the coordinates x, y and z in *geometry space* are mapped onto the two dimensional *parameter space* (ξ and η). The values for the Jacobian can be computed by using Eq. (B.2).

$$\frac{\partial \mathbf{x}}{\partial \xi_\alpha} = \sum_{i=1}^{n_{cp}} \frac{\partial N_i(\xi, \eta)}{\partial \xi^\alpha} \cdot \mathbf{x}_i \tag{B.2}$$

with n_{cp} being the number of control points and \mathbf{x}_i the coordinates of the control points in *geometry space*. For the example single patch the values for the derivatives $\frac{\partial N_i(\xi, \eta)}{\partial \xi^\alpha}$ of the quadrature point p_{10} can be looked up in Table 2.

The resulting Jacobian \mathbf{J} is given in Eq. (B.3) with its basis vectors \mathbf{g}_1 and \mathbf{g}_2 . Note that mapping is constant for all quadrature points since the patch geometry coincide with its parameter space. Thus basis vectors for all quadrature points p_{10-13} are the same and given in Eq. (B.3).

$$\mathbf{J} = \begin{bmatrix} 1 & 0 \\ 0 & -1 \\ 0 & 0 \end{bmatrix} \rightarrow \mathbf{g}_1 = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}, \mathbf{g}_2 = \begin{bmatrix} 0 \\ -1 \\ 0 \end{bmatrix} \tag{B.3}$$

Having the basis vectors the Jacobian J_1 can be computed using Eq. (6). For the example single patch this leads to the correct value of 1.0

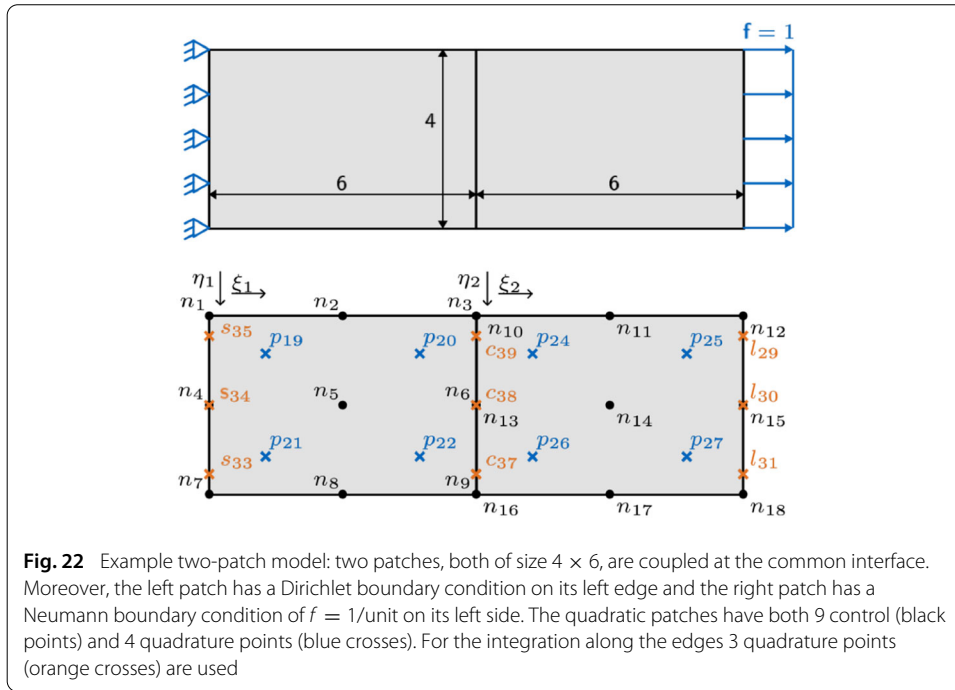
$$J_1 = \left\| \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \times \begin{bmatrix} 0 \\ -1 \\ 0 \end{bmatrix} \right\|_2 = 1.0 \tag{B.4}$$

Having the Jacobian J_1 for all quadrature points the area of the geometry can be computed.

Numerical integration of surfaces

In the following the area of the example single patch is approximated using Eq. (9). The weighting factors \tilde{w} for the quadrature points are given in Listing 10. They all have a value of 6.0. Thus the approximation of the area leads to the value 24.0, which is the exact solution.

$$A_{patch} = \sum_{l=1}^{n_{qp}} J_1 \tilde{w}_l = 4 \cdot (1.0 \cdot 6.0) = 24.0 \tag{B.5}$$



Two-patch model

The second example deals with a simple two-patch model. The problem is illustrated in Fig. 22. The corresponding data of the IBRA exchange format are given in Listing 11. The geometries of both patches are the same as in “Single patch” section. Thus, the shape function values in Table 1 and the derivatives in Table 2 are still valid for these patches. Since the evaluation of the surface area for both patches is exactly the same as already explained in “Single patch” section the focus of this example is on the integration of B-Rep edges and its element formulations.

Numerical integration of B-Rep edges

The formula for approximating the length of B-Rep edges is described in Eq. (10) and is given by

$$\Gamma_e = \sum_l^{n_{qp}} \tilde{w}_l \cdot \tilde{J}_1 \tag{B.6}$$

Here the mapping \tilde{J}_1 is defined in Eq. (11) whereas the weighting factor \tilde{w}_l is provided by the exchange format (see Listing 11).

In the following the length of the edge with the Neumann boundary condition in Fig. 22 (see also `brep_id 2002` in Listing 11) is evaluated. The Jacobian \tilde{J}_1 (see Eq. 11) can be computed as follows

$$\tilde{J}_1 = \|(\mathbf{g}_1 \cdot \tilde{t}_\xi + \mathbf{g}_2 \cdot \tilde{t}_\eta)\|_2 = \left\| \left(\begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \cdot 0.0 + \begin{bmatrix} 0 \\ -1 \\ 0 \end{bmatrix} \cdot 1.0 \right) \right\|_2 = 1.0 \tag{B.7}$$

```

{
  "nodes": [
    [1, [0, 4, 0, 1]], [2, [3, 4, 0, 1]], [3, [6, 4, 0, 1]],
    [4, [0, 2, 0, 1]], [5, [3, 2, 0, 1]], [6, [6, 2, 0, 1]],
    [7, [0, 0, 0, 1]], [8, [3, 0, 0, 1]], [9, [6, 0, 0, 1]],
    [10, [6, 4, 0, 1]], [11, [9, 4, 0, 1]], [12, [12, 4, 0, 1]],
    [13, [6, 2, 0, 1]], [14, [9, 2, 0, 1]], [15, [12, 2, 0, 1]],
    [16, [6, 0, 0, 1]], [17, [9, 0, 0, 1]], [18, [12, 0, 0, 1]]
  ],
  "2d_elements": [
    [1, [ ... Patch-ID
      [23, [2, 2], [[0.0, 0.0, 0.0, 6.0, 6.0, 6.0],
        [0.0, 0.0, 0.0, 4.0, 4.0, 4.0]],
        [1, 2, 3, 4, 5, 6, 7, 8, 9], true,
        [[19, 6.0, [1.2679, 0.8452]], [20, 6.0, [4.7320, 0.8452]],
         [21, 6.0, [1.2679, 3.1547]], [22, 6.0, [4.7320, 3.1547]]]
      ]
    ],
    [2, [ ... Patch-ID
      [28, [2, 2], [[0.0, 0.0, 0.0, 6.0, 6.0, 6.0],
        [0.0, 0.0, 0.0, 4.0, 4.0, 4.0]],
        [10, 11, 12, 13, 14, 15, 16, 17, 18], true,
        [[24, 6.0, [1.2679, 0.8452]], [25, 6.0, [4.7320, 0.8452]],
         [26, 6.0, [1.2679, 3.1547]], [27, 6.0, [4.7320, 3.1547]]]
      ]
    ],
  ],
  "brep_elements": [
    [2002, [ ... BRep-ID for Load Condition
      [32, [
        [[28], [29, 1.1111, [6.0, 0.4508], [0.0, 1.0]]],
        [[28], [30, 1.7777, [6.0, 2.0], [0.0, 1.0]]],
        [[28], [31, 1.1111, [6.0, 3.5491], [0.0, 1.0]]]
      ]
    ],
    [1001, [ ... BRep-ID for Support Condition
      [36, [ ... BRep-Element-ID
        [[23], [33, 1.1111, [0.0, 3.5491], [0.0, -1.0]]],
        [[23], [34, 1.7777, [0.0, 2.0], [0.0, -1.0]]],
        [[23], [35, 1.1111, [0.0, 0.4508], [0.0, -1.0]]]
      ]
    ],
    [1002, [ ... BRep-ID for Coupling Condition
      [43, [ ... BRep-Element-ID
        [[23, 28], [37, 1.1111, [6.0, 3.5491], [0.0, 1.0],
          [0.0, 3.5491], [0.0, -1.0]]],
        [[23, 28], [38, 1.7777, [6.0, 2.0], [0.0, 1.0],
          [0.0, 2.0], [0.0, -1.0]]],
        [[23, 28], [39, 1.1111, [6.0, 0.4508], [0.0, 1.0],
          [0.0, 0.4508], [0.0, -1.0]]]
      ]
    ],
  ],
}

```

Listing 11 Example two-patch model: the IBRA exchange format provides the data for the geometry in Fig. 22

The values \tilde{t}_ξ and \tilde{t}_η are directly read from the exchange format (see Listing 11). They are same for all three quadrature points used for this edge because it is aligned to a parameter direction. The basis vectors can be adapted from “Single patch” section because the geometry is exactly the same as used here. The approximated length of the edge can be computed as follows

$$\Gamma_e = 1.0 \cdot 1.1111 + 1.0 \cdot 1.7777 + 1.0 \cdot 1.1111 = 3.9999 \quad (\text{B.8})$$

which corresponds almost to the exact solution of 4.0. The small error evolves from the usage of numbers with restricted accuracy. The accuracy can be increased by using numbers with a higher precision.

Table 3 Example two-patch model: shape function values $N_i(\xi, \eta)$ for the integration points c_{37-39} for the master side (nodes 1–9, upper values) and slave side (nodes 10–18, values below)

(a) Point c37		
0	0	0.7872
0	0	0.2000
0	0	0.0127
0.7872	0	0
0.2000	0	0
0.0127	0	0
(b) Point c38		
0	0	0.25
0	0	0.5
0	0	0.25
0.25	0	0
0.5	0	0
0.25	0	0
(c) Point c39		
0	0	0.0127
0	0	0.2000
0	0	0.7872
0.0127	0	0
0.2000	0	0
0.7872	0	0

Continuity condition

The coupling for the edge with the brep_id 1002 (see Listing 11) is explained exemplarily. The penalty approach for displacements (see also “Continuity between patches” section) is used for the coupling formulation. All required information are provided by the exchange format in Listing 11. In the case of displacement coupling with penalty the condition can be formulated as follows

$$\alpha_{disp} \cdot \sum_k^{n_{qp}} \tilde{w}_k \cdot \tilde{J}_1^k \cdot \left(\sum_i^{n_{cp}^{(1)}} N_i^{(1)}(\xi_1, \eta_1) \cdot u_i^{(1)} - \sum_j^{n_{cp}^{(2)}} N_j^{(2)}(\xi_2, \eta_2) \cdot u_j^{(2)} \right) = 0 \quad (B.9)$$

This equation expresses that the difference between the master and slave boundary curve is penalized with a penalty factor. Thus continuity for the displacement can be achieved. The penalty factor α_{disp} represents an analysis data and is thus not provided by the exchange format but by the data interface *Project parameters* and is assigned through the brep_id to the corresponding elements. The mapping \tilde{J}_1^k can be computed as explained in “Two-patch model” section using the master curve information. The weighting factor \tilde{w}_k is provided by the exchange format (see also “Two-patch model” section). The values of the shape functions $N_i(0, \eta)$ are evaluated for each quadrature point on the coupling edge for the master and slave side. The corresponding values for the quadrature points c_{37-39} are given in Table 3.

Dirichlet boundary condition

Since Dirichlet boundary conditions are as a special case of coupling boundary conditions are treated analogously. The Dirichlet boundary condition with penalty for displacements

Table 4 Example two-patch model: shape function values $N_i(\xi, \eta)$ for the integration points s_{33-35}

(a) Point s_{33}		
0.7872	0	0
0.2000	0	0
0.0127	0	0
(b) Point s_{34}		
0.25	0	0
0.5	0	0
0.25	0	0
(c) Point s_{35}		
0.0127	0	0
0.2000	0	0
0.7872	0	0

The corresponding control points have the ids 1–9

(see also Eq. 31) can be written as

$$u = \alpha_{disp} \sum_k^{n_{qp}} \tilde{w}_i \cdot \tilde{J}_1 \cdot \left(\sum_i^{n_{cp}} N_i^{(1)}(\xi, \eta) \cdot u_i^{(1)} - u_0 \right) \tag{B.10}$$

with u_0 being the prescribed displacement. The values for the shape functions for the edge 1001 (see Listing 11) are given in Table 4. For this example just the control points n_1, n_4 and n_7 have an impact on the support because the edge corresponds to the surface boundary. In the case of trimmed patches all values can be non-equal to zero.

Neumann boundary condition

The discrete form of integrating a force along an edge is given by

$$\mathbf{f} = \sum_k^{n_{qp}} \tilde{w}_i \cdot \tilde{J}_1 \cdot N_i(\xi, \eta) \cdot \mathbf{p}, \tag{B.11}$$

where \mathbf{p} is a line load along the edge.

In the following, the force vector on the control points for the edge 2002 (see Listing 11) is computed. The integration along the edge for this element has been done already in “Two-patch model” section. In the following just the x-component of the force vector given as $f = 1.0$ is considered.

The respective load on each integration point is computed in a first step as follows

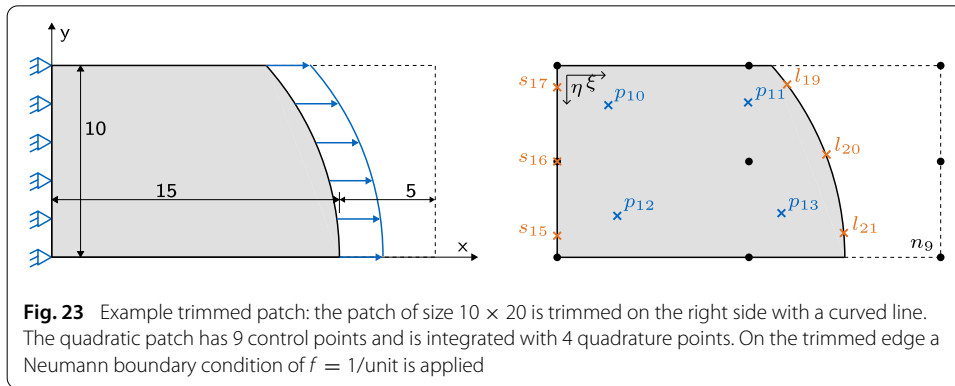
$$\begin{aligned} f_{29} &= f \cdot \tilde{w}_{29} \cdot \tilde{J}_1 = 1 \cdot 1.1111 \cdot 1.0 = 1.1111 \\ f_{30} &= 1.0 \cdot 1.7777 \cdot 1.0 = 1.7777 \\ f_{31} &= 1.0 \cdot 1.1111 \cdot 1.0 = 1.1111 \end{aligned} \tag{B.12}$$

The values for the relevant (non-zero) control points (n_{12}, n_{15} and n_{18}) can then be determined by performing the corresponding integral (see Eq. B.11). The values for the

Table 5 Example two-patch model: shape function values $N_i(\xi, \eta)$ for the integration points s_{29-31}

(a) Point /29		
0	0	0.7872
0	0	0.2000
0	0	0.0127
(b) Point /30		
0	0	0.25
0	0	0.5
0	0	0.25
(c) Point /31		
0	0	0.0127
0	0	0.2000
0	0	0.7872

The corresponding control points have the ids 10–18



shape functions are taken from Table 5.

$$\begin{aligned}
 f_{n_{12}} &= \sum_{qp=29}^{l_{qp}=31} N_{12,qp} \cdot f_{qp} \\
 &= 0.7872 \cdot 1.1111 + 0.25 \cdot 1.7777 + 0.0127 \cdot 1.1111 = 1.333 \qquad \text{(B.13)} \\
 f_{n_{15}} &= 0.2 \cdot 1.1111 + 0.5 \cdot 1.7777 + 0.2 \cdot 1.1111 = 1.3333 \\
 f_{n_{18}} &= 0.0127 \cdot 1.1111 + 0.25 \cdot 1.7777 + 0.7872 \cdot 1.1111 = 1.3333
 \end{aligned}$$

The values $f_{n_{12}}$, $f_{n_{15}}$ and $f_{n_{18}}$ are added to the force vector \mathbf{f} respectively to their degree of freedoms.

Trimmed patch

The third example deals with a trimmed NURBS surface because CAD systems usually use trimmed surfaces. The problem description is shown in Fig. 23. The corresponding data of the IBRA exchange format are given in Listing 12. For this example the focus is on the numerical integration of the trimmed edge, where a Neumann boundary with $f = 1/\text{unit}$ is applied. The integration along the trimmed edge with the id 1005 is done with the three quadrature points l_{19-21} .

```

{
  "nodes": [
    [1, [0, 10, 0, 1.0]], [2, [10, 10, 0, 1.0]], [3, [20, 10, 0, 1.0]],
    [4, [0, 5, 0, 1.0]], [5, [10, 5, 0, 1.0]], [6, [20, 5, 0, 1.0]],
    [7, [0, 0, 0, 1.0]], [8, [10, 0, 0, 1.0]], [9, [20, 0, 0, 1.0]]
  ],
  "2d_elements": [
    [1,
      [[14, [2, 2], [
        [0.0, 0.0, 0.0, 20.0, 20.0, 20.0],
        [0.0, 0.0, 0.0, 10.0, 10.0, 10.0]],
        [1, 2, 3, 4, 5, 6, 7, 8, 9], true,
        [[10, 31.0759, [2.6677, 2.0618]],
         [11, 29.7525, [9.9563, 1.9214]],
         [12, 37.7542, [3.1338, 7.8353]],
         [13, 39.6138, [11.6955, 7.6949]]
        ]
      ]
    ],
    [1002, [
      [18, [
        [[14], [15, 2.7777, [0.0, 8.8729], [0.0, -1.0]]],
        [[14], [16, 4.4444, [0.0, 5.0 ], [0.0, -1.0]]],
        [[14], [17, 2.7777, [0.0, 1.1270], [0.0, -1.0]]]
        ]
      ]
    ],
    [1005, [
      [22, [
        [[14], [19, 3.0405, [11.9927, 0.9811], [0.6192, 0.8103]]],
        [[14], [20, 4.8648, [14.0450, 4.6352], [0.3489, 0.9135]]],
        [[14], [21, 3.0405, [14.9514, 8.7270], [0.0786, 1.0168]]]
        ]
      ]
    ]
  ]
}

```

Listing 12 Example trimmed patch: the IBRA exchange format provides the data for the geometry in Fig. 23

Table 6 Example trimmed patch: shape function values $N_i(\xi, \eta)$ for the integration points 519–21

(a) Point s19			
	0.1303	0.3905	0.2924
	0.0283	0.0849	0.0636
	0.0015	0.0046	0.0034
(b) Point /20			
	0.0255	0.1203	0.1419
	0.0440	0.2079	0.2452
	0.0190	0.0898	0.1059
(c) Point /21			
	0.0010	0.0061	0.0090
	0.0141	0.0838	0.1241
	0.0485	0.2874	0.4256

The corresponding control points have the ids 1–9

Neumann boundary condition

The Neumann boundary condition is applied to a trimming curve, which does not correspond to the surface boundary. Thus, the quadrature points along the edge are influenced by the entire set of control points of the underlying surface element. The shape functions for the three quadrature points /19 – 21 of this edge are shown in Table 6. Note that in contrast to untrimmed surfaces (see previous examples) no zero values occur.

The basis vectors \mathbf{g}_1 and \mathbf{g}_2 for this example can be computed as follows

$$\mathbf{J} = \begin{bmatrix} 1 & 0 \\ 0 & -1 \\ 0 & 0 \end{bmatrix} \rightarrow \mathbf{g}_1 = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}, \mathbf{g}_2 = \begin{bmatrix} 0 \\ -1 \\ 0 \end{bmatrix} \tag{B.14}$$

Taking the tangents of the trimming curves for each quadrature point from the exchange format (see Listing 12) the corresponding Jacobian $\tilde{\mathbf{J}}_1$ can be determined as follows

$$\begin{aligned} \tilde{\mathbf{J}}_1^{(l19)} &= \left\| \left(\begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \cdot 0.6192 + \begin{bmatrix} 0 \\ -1 \\ 0 \end{bmatrix} \cdot 0.8103 \right) \right\|_2 = 1.0198 \\ \tilde{\mathbf{J}}_1^{(l20)} &= 0.9779 \quad \tilde{\mathbf{J}}_1^{(l21)} = 1.0198 \end{aligned} \tag{B.15}$$

With these Jacobians the force vector for each control point can be determined using Eq. (B.11).

In the following this is done exemplarily for the control point n_9 .

$$\begin{aligned} \mathbf{f}_{9,x} &= N_9(\xi_{19}, \eta_{19}) \cdot \tilde{\mathbf{J}}_1^{(l19)} \cdot \tilde{\mathbf{w}}_{19} \cdot \mathbf{f}_{line,x} \\ &\quad + N_9(\xi_{20}, \eta_{20}) \cdot \tilde{\mathbf{J}}_1^{(l20)} \cdot \tilde{\mathbf{w}}_{20} \cdot \mathbf{f}_{line,x} \\ &\quad + N_9(\xi_{21}, \eta_{21}) \cdot \tilde{\mathbf{J}}_1^{(l21)} \cdot \tilde{\mathbf{w}}_{21} \cdot \mathbf{f}_{line,x} \end{aligned} \tag{B.16}$$

$$\begin{aligned} \mathbf{f}_{9,x} &= 0.0034 \cdot 1.0198 \cdot 3.0405 \cdot 1.0 \\ &\quad + 0.1059 \cdot 0.9779 \cdot 4.8648 \cdot 1.0 \\ &\quad + 0.4256 \cdot 1.0198 \cdot 3.0405 \cdot 1.0 \\ &= 1.8339 \end{aligned}$$

Publisher’s Note

Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Received: 3 January 2018 Accepted: 9 May 2018

Published online: 04 July 2018

References

1. Breitenberger M. Cad-integrated design and analysis of shell structures. Dissertation. München: Technische Universität München; 2016.
2. Sederberg TW, Cardon DL, Zheng J, Lyche T. T-Spline simplification and local refinement. ACM Trans Graph. 2004;23:276–83.
3. Bazilevs Y, Calo VM, Cottrell JA, Evans JA, Hughes TJR, Lipton S, Scott MA, Sederberg TW. Isogeometric analysis using t-splines. Comput Methods Appl Mech Eng. 2010;199(5–8):229–63. <https://doi.org/10.1016/j.cma.2009.02.036>.
4. Breitenberger M, Apostolatos A, Philipp B, Wüchner R, Bletzinger K-U. Analysis in computer aided design: nonlinear isogeometric B-Rep analysis of shell structures. Comput Methods Appl Mech Eng. 2015;284(284):401–57. <https://doi.org/10.1016/j.cma.2014.09.033>.
5. Apostolatos A, Schmidt R, Wüchner R, Bletzinger K-U. A Nitsche-type formulation and comparison of the most common domain decomposition methods in isogeometric analysis. Int J Numer Methods Eng. 2014;97(7):473–504. <https://doi.org/10.1002/nme.4568>.
6. Apostolatos A, Breitenberger M, Wüchner R, Bletzinger K-U. Springer. In: Jüttler B, Simeon B, editors. Domain decomposition methods and Kirchhoff-love shell multipatch coupling in isogeometric analysis, vol. 107., Lecture Notes in Computational Science and Engineering isogeometric analysis and applications 2014: Cham and Heidelberg and New York; 2015. p. 73–101. https://doi.org/10.1007/978-3-319-23315-4_4.
7. Burman E, Claus S, Hansbo P, Larson MG, Massing A. Cutfem: discretizing geometry and partial differential equations. Int J Numer Methods Eng. 2015;104(7):472–501. <https://doi.org/10.1002/nme.4823>.
8. Düster A, Parvizian J, Yang Z, Rank E. The finite cell method for three-dimensional problems of solid mechanics. Comput Methods Appl Mech Eng. 2008;197(45–48):3768–82. <https://doi.org/10.1016/j.cma.2008.02.036>.
9. Schillinger D, Ruess M, Zander N, Bazilevs Y, Düster A, Rank E. Small and large deformation analysis with the p- and B-spline versions of the finite cell method. 2012. <https://doi.org/10.1007/s00466-012-0684-z>.

10. Reed K, Harrod D, Conroy W. The initial graphics exchange specification (IGES) version 5.0. NISTIR, vol. 4412. Gaithersburg: National Institute of Standards and Technology; 1990.
11. Productions M. Iges 5.3. Productions, MerAI 1992.
12. ISO 10303. <http://www.WikiSTEP.org>. STEP (Standard for the Exchange of Product model data).
13. Piegl LA, Tiller W. The NURBS book. 2nd ed. New York: Springer; 1997.
14. Cox MG. The numerical evaluation of b-splines. *IMA J Appl Math.* 1972;10(2):134–49. <https://doi.org/10.1093/imamat/10.2.134>.
15. de Boor C. On calculation with b-splines. *J Approx Theory.* 1972;6:50–62.
16. Piegl L, Tiller W. Geometry-based triangulation of trimmed nurbs surfaces. *Comput Aided Design.* 1998;1(30):11–8.
17. Renner G, Weiß V. Exact and approximate computation of b-spline curves on surfaces. *Comput Aided Design.* 2004;36(4):351–62. [https://doi.org/10.1016/S0010-4485\(03\)00100-3](https://doi.org/10.1016/S0010-4485(03)00100-3).
18. Wüchner R, Breitenberger M, Bauer AM, Bletzinger K-U. Isogeometric structural analysis and design, München; 2017.
19. Guo Y, Ruess M. Weak dirichlet boundary conditions for trimmed thin isogeometric shells. *Comput Math Appl.* 2015;70(7):1425–40. <https://doi.org/10.1016/j.camwa.2015.06.012>.
20. Hughes TJR, Reali A, Sangalli G. Efficient quadrature for nurbs-based isogeometric analysis. *Comput Methods Appl Mech Eng.* 2010;199(5–8):301–13. <https://doi.org/10.1016/j.cma.2008.12.004>.
21. Marussig B, Hughes TJR. A review of trimming in isogeometric analysis: challenges, data exchange and simulation aspects. *Arch Comput Methods Eng.* 2017;30(8):657. <https://doi.org/10.1007/s11831-017-9220-9>.
22. Marussig B, Zechner J, Beer G, Fries T-P. Stable isogeometric analysis of trimmed geometries. *Comput Methods Appl Mech Eng.* 2017;316:497–521. <https://doi.org/10.1016/j.cma.2016.07.040>.
23. Nagy AP, Benson DJ. On the numerical integration of trimmed isogeometric elements. *Comput Methods Appl Mech Eng.* 2015;284:165–85. <https://doi.org/10.1016/j.cma.2014.08.002>.
24. Ruess M, Schillinger D, Özcan AI, Rank E. Weak coupling for isogeometric analysis of non-matching and trimmed multi-patch geometries. *Comput Methods Appl Mech Eng.* 2014;269:46–71. <https://doi.org/10.1016/j.cma.2013.10.009>.
25. Schmidt R, Wüchner R, Bletzinger K-U. Isogeometric analysis of trimmed nurbs geometries. *Comput Methods Appl Mech Eng.* 2012;241–244:93–111. <https://doi.org/10.1016/j.cma.2012.05.021>.
26. Zhu X-F, Ma Z-D, Hu P. Nonconforming isogeometric analysis for trimmed cad geometries using finite-element tearing and interconnecting algorithm. *Proce Instit Mech Eng C.* 2017;231(8):1371–89. <https://doi.org/10.1177/0954406216688491>.
27. Guo Y, Heller J, Hughes TJR, Ruess M, Schillinger D. Variationally consistent isogeometric analysis of trimmed thin shells at finite deformations, based on the STEP exchange format. *Comput Methods Appl Mech Eng.* 2018;336:39–79. <https://doi.org/10.1016/j.cma.2018.02.027>.
28. Chen L, Nguyen-Thanh N, Nguyen-Xuan H, Rabczuk T, Bordas SPA, Limbert G. Explicit finite deformation analysis of isogeometric membranes. *Comput Methods Appl Mech Eng.* 2014;277:104–30. <https://doi.org/10.1016/j.cma.2014.04.015>.
29. Philipp B, Breitenberger M, D'Auria I, Wüchner R, Bletzinger K-U. Integrated design and analysis of structural membranes using the isogeometric B-Rep analysis. *Comput Methods Appl Mech Eng.* 2016;303:312–40. <https://doi.org/10.1016/j.cma.2016.02.003>.
30. Benson DJ, Bazilevs Y, Hsu MC, Hughes TJR. Isogeometric shell analysis: the Reissner–Mindlin shell. *Comput Methods Appl Mech Eng.* 2010;199(5–8):276–89. <https://doi.org/10.1016/j.cma.2009.05.011>.
31. Benson DJ, Bazilevs Y, Hsu M-C, Hughes TJR. A large deformation, rotation-free, isogeometric shell. *Comput Methods Appl Mech Eng.* 2011;200(13–16):1367–78. <https://doi.org/10.1016/j.cma.2010.12.003>.
32. Benson DJ, Hartmann S, Bazilevs Y, Hsu M-C, Hughes TJR. Blended isogeometric shells. *Comput Methods Appl Mech Eng.* 2013;255:133–46. <https://doi.org/10.1016/j.cma.2012.11.020>.
33. Kiendl JM. Isogeometric analysis and shape optimal design of shell structures. Aachen: Shaker; 2011.
34. Echter R, Oesterle B, Bischoff M. A hierarchic family of isogeometric shell finite elements. *Comput Methods Appl Mech Eng.* 2013. <https://doi.org/10.1016/j.cma.2012.10.018>.
35. Dornisch W, Klinkel S, Simeon B. Isogeometric Reissner–Mindlin shell analysis with exactly calculated director vectors. *Comput Methods Appl Mech Eng.* 2013;253:491–504. <https://doi.org/10.1016/j.cma.2012.09.010>.
36. Dornisch W, Müller R, Klinkel S. An efficient and robust rotational formulation for isogeometric Reissner–Mindlin shell elements. *Comput Methods Appl Mech Eng.* 2016;303:1–34. <https://doi.org/10.1016/j.cma.2016.01.018>.
37. Oesterle B, Ramm E, Bischoff M. A shear deformable, rotation-free isogeometric shell formulation. *Comput Methods Appl Mech Eng.* 2016;307:235–55. <https://doi.org/10.1016/j.cma.2016.04.015>.
38. Oesterle B, Sachse R, Ramm E, Bischoff M. Hierarchic isogeometric large rotation shell elements including linearized transverse shear parametrization. *Comput Methods Appl Mech Eng.* 2017;321:383–405. <https://doi.org/10.1016/j.cma.2017.03.031>.
39. Bauer AM, Breitenberger M, Philipp B, Wüchner R, Bletzinger K-U. Embedded structural entities in nurbs-based isogeometric analysis. *Comput Methods Appl Mech Eng.* 2017;325:198–218. <https://doi.org/10.1016/j.cma.2017.07.010>.
40. Coox L, Greco F, Atak O, Vandepitte D, Desmet W. A robust patch coupling method for nurbs-based isogeometric analysis of non-conforming multipatch surfaces. *Comput Methods Appl Mech Eng.* 2017;316:235–60. <https://doi.org/10.1016/j.cma.2016.06.022>.
41. Dornisch W, Stöckler J, Müller R. Dual and approximate dual basis functions for b-splines and nurbs—comparison and application for an efficient coupling of patches with the isogeometric mortar method. *Comput Methods Appl Mech Eng.* 2017;316:449–96. <https://doi.org/10.1016/j.cma.2016.07.038>.
42. Embar A, Dolbow JE, Harari I. Imposing Dirichlet boundary conditions with Nitsche's method and spline-based finite elements. *Int J Numer Methods Eng.* 2010;194:3.
43. Guo Y, Ruess M. Nitsche's method for a coupling of isogeometric thin shells and blended shell structures. *Comput Methods Appl Mech Eng.* 2015;284:881–905. <https://doi.org/10.1016/j.cma.2014.11.014>.

44. Guo Y, Ruess M, Schillinger D. A parameter-free variational coupling approach for trimmed isogeometric thin shells. *Comput Mech*. 2017;59(4):693–715. <https://doi.org/10.1007/s00466-016-1368-x>.
45. Jiang W, Annavarapu C, Dolbow JE, Harari I. A robust Nitsche's formulation for interface problems with spline-based finite elements. *Int J Numer Methods Eng*. 2015;104(7):676–96. <https://doi.org/10.1002/nme.4766>.
46. de Lorenzis L, Wriggers P, Zavarise G. A mortar formulation for 3d large deformation contact using nurbs-based isogeometric analysis and the augmented Lagrangian method. *Computat Mech*. 2012;49(1):1–20. <https://doi.org/10.1007/s00466-011-0623-4>.
47. Hansbo A, Hansbo P. An unfitted finite element method, based on Nitsche's method, for elliptic interface problems. *Comput Methods Appl Mech Eng*. 2002;191:5537–52.
48. Griebel M, Schweitzer MA. A particle-partition of unity method-part V: boundary conditions. Berlin: Springer; 2002.
49. Fritz A, Hüber S, Wohlmuth BI. A comparison of mortar and nitsche techniques for linear elasticity. *CALCOLO*. 2004;41:115–37.
50. Sanders JD, Dolbow JE, Laursen TA. On methods for stabilizing constraints over enriched interfaces in elasticity. *Int J Numer Methods Eng*. 2009;78:1009–36.
51. Sanders JD, Laursen TA, Puso MA. A Nitsche embedded mesh method. *Comput Mech*. 2012;49:243–57.
52. Ruess M, Schillinger D, Bazilevs Y, Varduhn V, Rank E. Weakly enforced essential boundary conditions for nurbs-embedded and trimmed nurbs geometries on the basis of the finite cell method. *Int J Numer Methods Eng*. 2013;95(10):811–46. <https://doi.org/10.1002/nme.4522>.
53. Brezzi F. A discourse on the stability conditions for mixed finite element formulations. *Comput Methods Appl Mech Eng*. 1990;82:27–57.
54. Brivadis E, Buffa A, Wohlmuth B, Wunderlich L. Isogeometric mortar methods. *Comput Methods Appl Mech Eng*. 2015;284:292–319. <https://doi.org/10.1016/j.cma.2014.09.012>.
55. Carat++. <http://www.st.bgu.tum.de/lehre0/forschung/carat/>.
56. KRATOS-Multiphysics. <http://www.cimne.com/kratos/>.
57. Dadvand P, Rossi R, Oñate E. An object-oriented environment for developing finite element codes for multi-disciplinary applications. *Arch Comput Methods Eng*. 2010;17(3):253–97. <https://doi.org/10.1007/s11831-010-9045-2>.
58. Dadvand P. A framework for developing finite element codes for multi-disciplinary applications. Dissertation. Barcelona: Universidad Politecnica de Catalunya; 2007.
59. Rhinoceros. <http://www.rhino3d.com>. McNeel.
60. Siemens NX. <http://www.plm.automation.siemens.com>. Siemens.
61. GrabCAD. 2016. <https://grabcad.com/library/mercedes-16>.
62. Bauer AM, Breitenberger M, Philipp B, Wüchner R, Bletzinger K-U. Nonlinear isogeometric spatial Bernoulli beam. *Comput Methods Appl Mech Eng*. 2016;303:101–27. <https://doi.org/10.1016/j.cma.2015.12.027>.
63. Bletzinger K-U, Ramm E. A general finite element approach to the form finding of tensile structures by the updated reference strategy. *Int J Space Struct*. 1999;14(2):131–45. <https://doi.org/10.1260/0266351991494759>.

Submit your manuscript to a SpringerOpen[®] journal and benefit from:

- Convenient online submission
- Rigorous peer review
- Open access: articles freely available online
- High visibility within the field
- Retaining the copyright to your article

Submit your next manuscript at ► springeropen.com
