EC-Project EXCITING SCP8-2007-GA-218536

EXACT GEOMETRY SIMULATION
FOR OPTIMIZED DESIGN
OF VEHICLES AND VESSELS

http://exciting-project.eu

EXCITING Report No. 6.3

First version of isogeometric analysis framework library

Vibeke Skytt, SINTEF, Gang Xu, Bernard Mourrain, INRIA

Editor(s): Geir Skeie, DNV

June 2010

Dissemination level: PU

Contents

1 Introduction **3**

2 Block structured meshes in an isogeometric context **4**

3 Model overview **5**

4 Topology **5**

4.1 Face sets 6

4.2 Volume models 8

4.3 Shell models and mixed descriptions 10

5 Isogeometric block structured models **11**

6 Extensions to the geometry tools **13**

6.1 Translating a trimmed face into a number of 4-sided surfaces 14

6.2 Volume fairing 15

7 Parameterization tools for computational domain **17**

7.1 Easy-to-check tools for injectivity parameterization 17

7.1.1 Linear constraint for injectivity. 19

7.2 Optimal analysis-ware parameterization of computational domain 20

7.2.1 Problem statement 20

7.2.2 Shape optimization method 20

7.2.3 Examples and comparison 21

7.3 Variational harmonic spline method for parameterization of computational domain 22

8 Simplex spline tools for isogeometric analysis **27**

9 PHT spline tools for isogeometric analysis **29**

9.1 Hierarchical T-mesh 30

9.2 PHT-spline space and basis function construction 31

10 Visualization tools for isogeometric model and results **32**

10.1 Color map tools 32

10.2 Iso-value tools 35

11 Toolbox modules **35**

12 Toolbox development **37**

13 Application of the isogeometry toolbox for simulation problems **38**

13.1 Isogeometric solver for heat conduction 38

13.2 A structural solver 39

13.3 A finite volume method applied on NURBS geometry for FSI problems . . 40

13.4 Fitting of blade surfaces 42

1 Introduction

The aim of WP6 in Exciting is to assemble an isogeometric toolbox to support isogeometric analysis and shape optimization. This report corresponds to the second software delivery of the toolbox, which is named the first version of an isogeometric analysis framework library.

In isogeometric analysis the spline space is used as the function space in numerical analysis. Thus, the toolbox needs to support functionality to exploit the spline space. This was already the case for the first software delivery.

The second delivery is more demanding with respect to functionality. It is, however, not obvious how to define a general Isogeometric Programming Interface (that is a toolbox) up front or at an early stage of our development. We are not sure about the spline spaces to use etc. So this is a challenging task and we need to define a toolbox that fits our current needs without, hopefully, not closing the future. There are also two very different approaches to solving PDEs in the Exciting project; boundary and volume methods.

In Exciting we are considering Isogeometric models, that is, the same functions used in geometry is used in the approximation of the solution fields of the PDE. This is at the patch level. Further, the global solution field has to be associated with an appropriate function space, which is one with sufficient regularity. This is especially important for multi-patch models which are probably the rule rather than the exception. In many (our) cases the regularity of the solution space can be inherited from the geometry space. Thus the definition of a global model is crucial.

The toolbox contains structure and functionality for representation and creation of isogeometric models. An isogeometric model consists of one or more NURBS surfaces in the 2D case and NURBS volumes in 3D. In the previous delivery, the focus was models with only one surface or volume, i.e. a single block model. This delivery concentrates on multiblock models. A number of different model views and representations are extensively discussed in the report thus providing the global model view. Block structured meshes are described in section 2. In order to represent multiblock models topological information is required. An overview of the various realizations of multiblock models described in this document is given in section 3. Topology structures for surface models and volume models are described in section 4 while a data structure tailored towards isogeometric models is the topic of section 5.

Section 6 gives an overview of non-topological extensions to the toolbox which is related to GoTools. Section 7 presents some parameterization tools for the computational domain, section 8 focuses on the use of simplex splines for isogeometric analysis, and a tool for local refinement of spline patches is presented in section 9. Tools for visualization of the model and simulation results are presented in section 10.

An overview of the structure of current version of the toolbox is given in section 11. In section 12 a summary of the current toolbox and some plans for the further development are presented. Finally, a number of applications of the isogeometric toolbox are presented to show that Milestone 6.1 is reached.

2 Block structured meshes in an isogeometric context

A structured mesh has regular connectivity, that is each point in the mesh has the same number of neighbours. In an unstructured mesh the points have different numbers of neighbours. A spline surface or volume is structured. However, a complex geometrical shape in 2D or 3D is not necessarily easily mapped to a rectangular shape. If the physical region is broken into pieces, each piece can have a relatively simple mapping into a regular grid. These pieces can then be fitted with some degree of continuity in to the physical domain. Each piece of the domain is named a block and the total domain is described by a multiblock mesh.

In an isogeometric context each block is represented by a NURBS surface or volume depending on the spatial dimension. Adjacent blocks meet with C^0 continuity. The blocks meet in a corner-to-corner configuration, i.e. T-joints are not allowed. This configuration is not required in order to construct a watertight model with spline blocks since the requested continuity can be achieved with more topological flexibility. However, in order to maintain the same continuity for the corresponding solution field, additional constraints must be added in order to solve the differential equation. Thus, the increased modelling flexibility are easily outweighed by the increased solver complexity.

Adjacent NURBS blocks are expected to have corresponding coefficients. As for the corner-to-corner condition, this is not due to spline constraints, but to simplicity for the solver. This constraint may be loosened in later versions of the toolbox as it can lead to high data size for complex models.

The NURBS surfaces and volumes in an isogeometric model are non-trimmed, but may have degeneracies. This facilitates the representation of domains without corners, or domains with less than 4 or 8 corners for surfaces or volumes, respectively. In a degenerate block corner, the Jacobian corresponding to the problem at hand, becomes zero. Thus, isogeometric analysis must be prepared for such configurations.

A block structured isogeometric model may be constructed using toolbox functionality. The toolbox contains some facilities for multiblock construction, and single blocks can with care be constructed in such a manner that they fit into a multiblock setting. Examples of construction methods are:

- Sweep a multiblock surface model to create a multiblock volume model.
- Mirror a multiblock surface model with respect to a given plane, and create a multiblock volume model by lofting.
- Fill a hole in a multiblock model by a Coons patch construction. The method applies both to surfaces and volumes. An entity created as a Coons patch may be preprocessed by a smoothing step where the boundary is kept fixed. This will lead to a better distribution of coefficients internal in the object. Section 6.2 gives some details on this topic.
- Fetch one boundary curve or surface from an already constructed block and use this boundary entity in the construction of a new block by for instance lofting or sweep.
- Split a trimmed surface into a number of 4-sided trimmed surfaces and approximate each such surface by a spline surface maintaining the C^0 or G^1 continuity between the surfaces. In Figure 8 the first part of this process is illustrated, see also section 6.1.

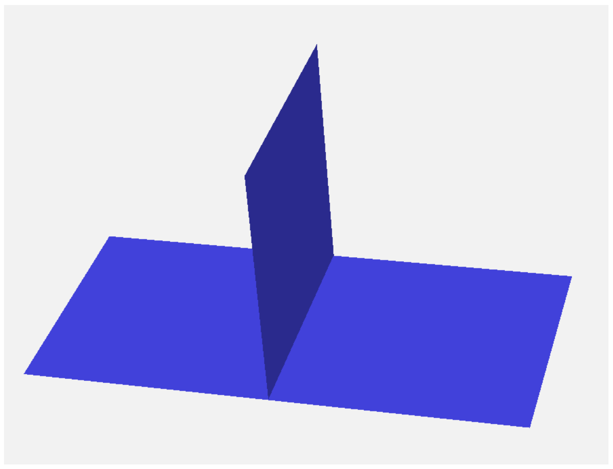


Figure 1: A non-manifold face set

More detailed descriptions of the construction methods in the toolbox can be found in the toolbox repository.

3 Model overview

In the following sections, the realization of a multiblock model in the GoTools environment is explained. We describe three model implementations:

1. A CAD type boundary representation model for face sets and solids, this model is described in section 4.1.
2. Trivariate composite models where each entity is a volume with a geometric description, this model is described in section 4.2.
3. A representation of a multiblock model where each block is a bivariate or trivariate NURBS object. This representation is tailored towards isogeometric use and the model also contains information about solution fields corresponding to the geometry entity. The model is described in section 5.

The two first models are general in respect to the geometry entities not being restricted to be of a particular type, and the purpose of the model is not defined. In an isogeometric setting, a geometry model should be created within the framework of model 1 and 2. Then the user should ensure that the objects in the model satisfy the requirements for a block structured isogeometric model as described in section 2. Functionality exist to perform these checks. Finally, the initial model is used as input to the isogeometric model (item 3 listed above).

4 Topology

Topological structures stores adjacency between geometric entities and provide the possibility to achieve a united treatment of composite entities.

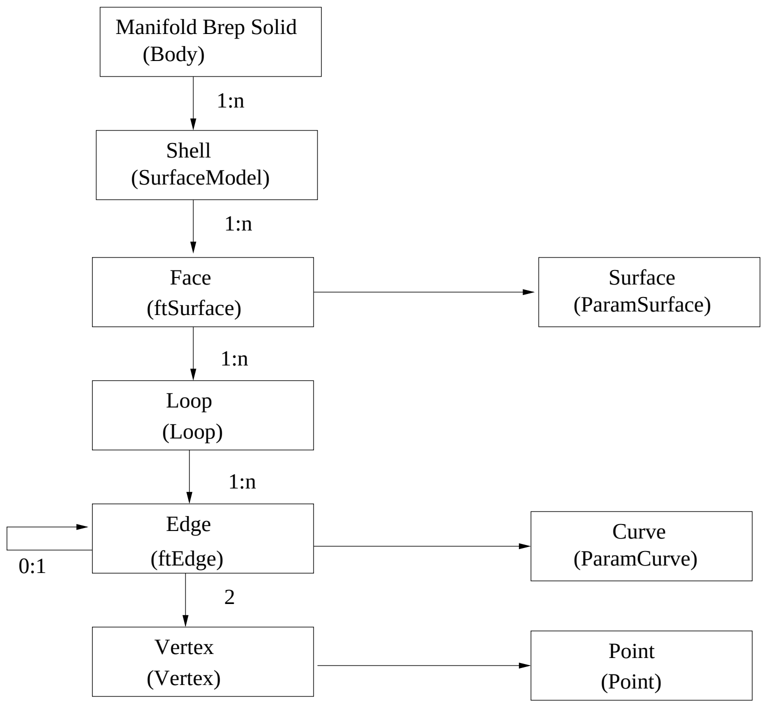


Figure 2: Topology structures for a boundary represented solid, corresponding GoTools names are shown in brackets

Topological structures may be manifold or non-manifold. A manifold is a topological space where at each point there is a neighbourhood that is topologically the same as the open unit ball in the relevant dimension. This implies that the configuration shown in Figure 1 cannot occur in a manifold face set. Only two faces can meet in an edge, otherwise the model is not homeomorphic to a disc along the edge. The collection of all boundary surfaces to all blocks in a multiblock volume model will give rise to a non-manifold model, while the outer boundary of the volume model normally will be manifold.

4.1 Face sets

Block structured isogeometric models in 2D, face sets in 3D and boundary represented solids in 3D may all be represented using the topology structures in Figure 2. A solid is surrounded by a set of faces collected in a shell. Normally, the solid has only an outer boundary, but it may also have voids. Thus, the solid may point to more than one shell. The solid is implicitly represented by its boundaries, and thus have no geometric description. A face set or an isogeometric model in 2D is represented by one shell. The solid entity is not used. In GoTools the shell is represented by the class SurfaceModel and the solid by the class Body.

A shell consist of a number of faces, in GoTools represented by the class ftSurface. The prefix *ft* has historical reasons. The face has a geometrical description represented by a parametric surface. The GoTools class is called ParamSurface and it may be a NURBS surface, an elementary surface or a trimmed version thereof.

A face is limited by one or more loops. The first loop corresponds to the boundary of the parametric surface or the outer trimming loop. Subsequent loops represent inner

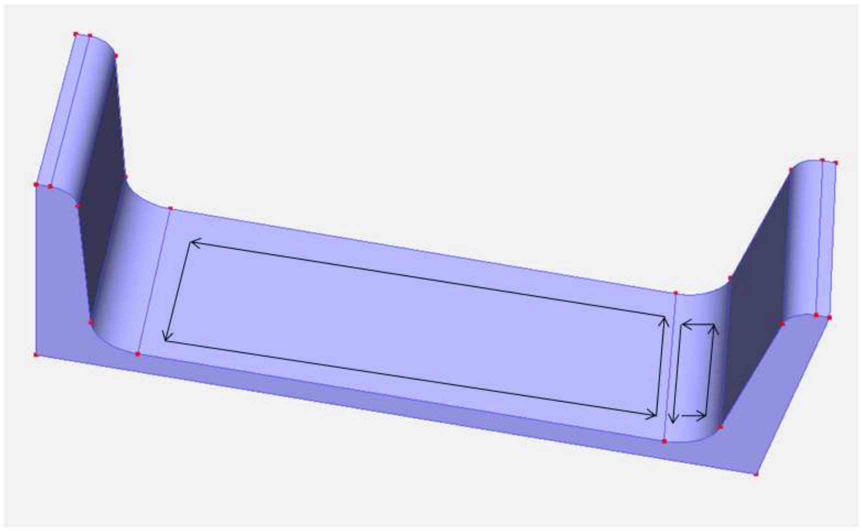


Figure 3: A boundary represented solid

trimming curves. A non-trimmed NURBS surface has only one loop.

A loop is described by a closed sequence of edges. In GoTools, the edge is represented by the class ftEdge. ftEdge is a half edge. An edge can be seen to represent a boundary between two faces and should thus contain information about the spacial representation of this boundary, the two faces involved and often the parameter representation of the boundary in these faces. A half edge contains information related only to one face. It holds a face pointer, the parameter curve corresponding to this face and a spatial curve. It also has information about the other half edge representing the boundary. The other half edge may or may not point to the same representation of the spatial curve. Different representations of the same spatial curves may lead to gaps between surfaces. The two half edges meeting in an edge have opposite orientation. A half edge structure can only represent manifold models as no more than two faces can meet in an edge.

Figure 3 shows a boundary represented solid. The faces are light blue, the edges are blue and the vertices are red. Each edge is represented by two half edges, i.e. ftEdge. The half edges join into loops, and the arrows indicate the orientation of the half edges for two adjacent faces.

The edge has a geometric representation as the boundary curve or trimming curve corresponding to a surface. This curve will, in GoTools, often be of type CurveOnSurface which has information about the boundary curve, the corresponding surface and the parameter representation of this curve in the surface. Thus, the relation between an edge and a face is repeated for the geometric entities.

While GoTools expects a one-to-one match between a face and the corresponding surface, several edges may correspond to one geometric curve. This is represented by letting the edge be limited by curve parameters.

GoTools is to some extent parameter space based. The parameter curve corresponding to an edge is used for tessellation, point-in-face queries and several other operations. Thus, if the parameter representation of an edge is not given initially, which is often the case for analytic geometries, it will at some stage be computed. An approximation to the true curve will be found.

An edge is limited by to vertices described by points in the geometry space. The vertices should be the spatial representations of the curve parameters limiting the edge. The two half edges representing one edge is limited by the same vertex entities, and the vertex has information about the relation between half edges. A number of edges meet in a vertex.

The data structure is equipped with a number of back pointers not shown in Figure 2. The face points to the solid, the loop and half edges point to the corresponding face and the vertex points to adjacent edges. The half edges have double pointers around the loop which correspond to the information given in the loop. Furthermore, there is a certain duplication in topology and geometry information. Thus, fetching information is fast and easy due to information redundancy, but changes in the structure must be done carefully to maintain information consistency.

SurfaceModel is a topological shell, but also an entity that allows a unified treatment of a face set. As such, the class belongs to a hierarchy called composite model. It is possible to compute for instance a bounding box, a closest point, the tessellation or some intersections for a SurfaceModel as one operation. More information about the SurfaceModel operations can be found in the documentation in the toolbox repository.

4.2 Volume models

The topology structures required to describe a block structured mesh in 3D where each block is a NURBS volume, is different from the ones described in the previous section. The boundary surfaces of a volume model do not define a manifold surface model, and going from a boundary represented solid to sets of trivariate solids requires some extensions to the data structures.

A multiblock trivariate model is represented as a VolumeModel. The class VolumeModel is, like SurfaceModel, a composite model. The model consists of a number of ftVolumes. An ftVolume is a solid with geometry description. The geometry is represented as a ParamVolume. Currently this means a NURBS volume, but the data structure does also allow other volume types.

An ftVolume is surrounded by a number of shells just like Body in section 4.1. In fact, ftVolume inherits Body. Currently, a volume will have only an outer boundary, but the implementation allows also inner trimming.

The shells are represented as SurfaceModel as before, and each shell is a collection of faces. Now, there is more than one solid, and the boundary faces of one solid will normally coincide with boundary faces of other solids. Thus, adjacency information must be represented at the face level. The solution is to extend ftSurface to become a half face in the same manner as ftEdge is a half edge.

An ftSurface is equipped with a back pointer to the Body and a pointer to the corresponding half face (ftSurface) on the adjacent Body. This provides a representation of adjacency. Unlike ftEdge, no dedicated parameter information for the half face exists

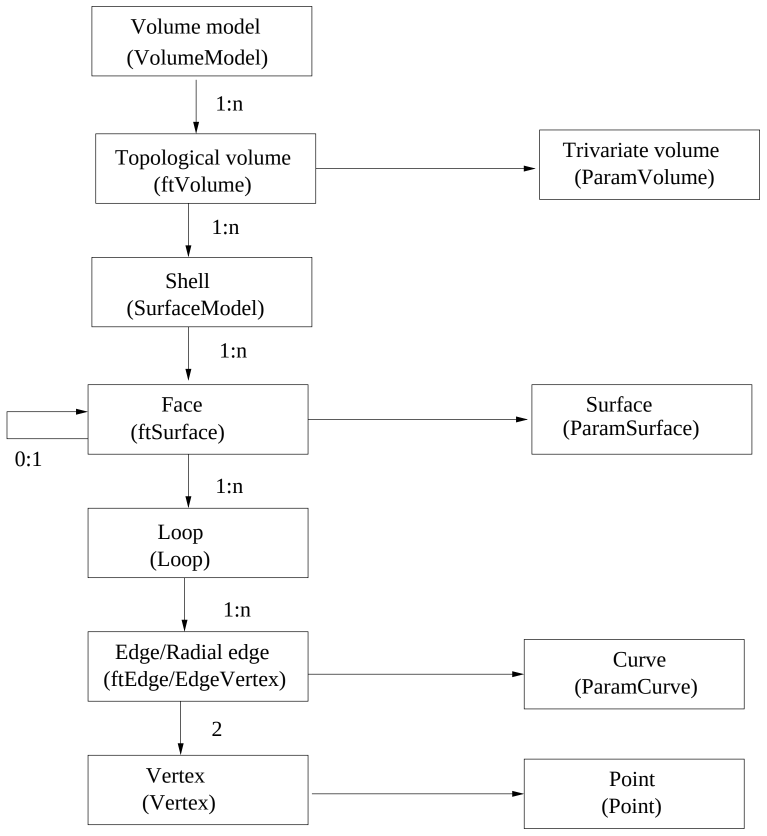


Figure 4: Topology structures for a volume model

in ftSurface. However, the geometry representation of the face will normally hold this information.

The volume model is not a manifold model. More than two half edges meet in common boundaries. This is represented by introducing the concept of a radial edge. The radial edge does not keep any edge information. It is simply a collection of coincident ftEdges. In GoTools the radial edge class is named EdgeVertex. The name indicates that the object is an edge that in some sense behaves like a vertex.

In Figure 5, two simple blocks lie adjacent and the half faces related to the common boundary is shown in red and blue. They coincide, but are shown with some distance for visualization purposes. Pointers are set between these faces to represent the adjacency. At the boundary curves of the common face, two pairs of half edges, one for each block, coincides. This pairs are collected in an EdgeVertex entity.

The radial edge is added to the structure in addition to the manifold half edge. The half edge is used in the topology structures of the shells which are assumed to be manifold. In addition, the half edge is extended with a pointer to a radial edge. If the shell is a part of a volume model where adjacency between volumes are found, information about all coincident edges is stored in the radial edge instance while the half edge stores connectivity related to the boundary faces of one single volume.

A restriction to the current structure is that the object in Figure 6 cannot be represented as shown in the picture. The boundary surfaces corresponding to one volume are expected to be represented in the half edge based topology structure. However, along the seem of the rotational volume elements, there are two adjacent boundary surfaces. This implies

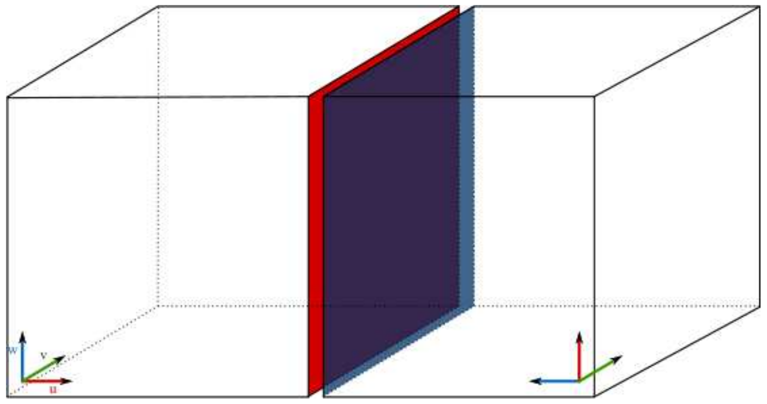


Figure 5: A volume model with two blocks and common boundary surfaces emphasized

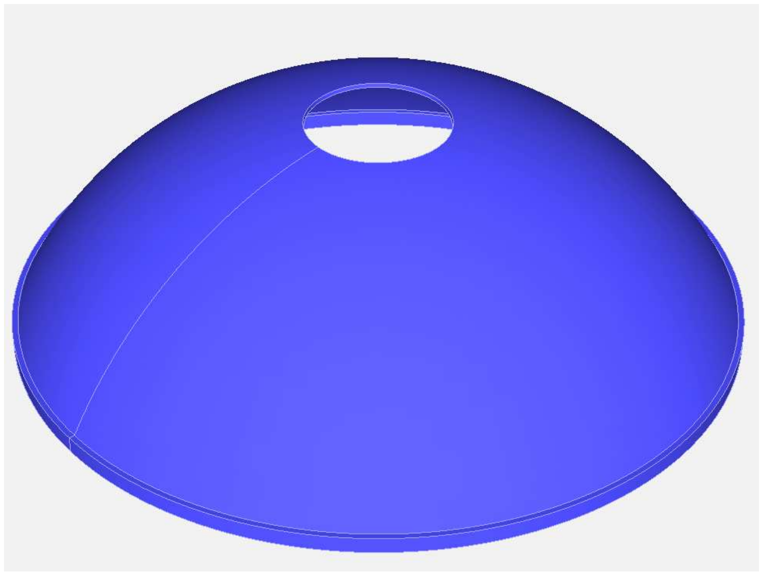


Figure 6: A volume model, the emphasized curves show boundaries

that 4 edges meet at two of the boundaries of the seem surface. This is not a manifold construction. Thus, the boundary surface corresponding to the rotational volumes cannot be represented in the half edge structure. The current solution is to split the rotational entities into two. A more long term solution would be to extend the data structure and use the radial edge entity also for the topology of the shells surrounding volume elements.

4.3 Shell models and mixed descriptions

Thin plate objects are often described as shell models. This enables lower computation times for numerical analysis even though extra degrees of freedom lead to additional complexity. Models combining volumetric entities with shells are also used to improve the analysis performance. In both cases, non-manifold surface models appear. Also for isogeometric analysis, such models can be relevant, but they are currently not handled by the topology structures in GoTools.

5 Isogeometric block structured models

A block structured model contains information about each block and the relation between adjacent blocks. This can be modelled using a SurfaceModel or a VolumeModel depending on the spatial dimension. In an isogeometric context, these concepts have, however, two drawbacks:

- An isogeometric model is a collection of isogometric blocks where each block consists of a NURBS surface or volume. A surface or volume model, as described in section 4, is a collection of surfaces or volumes. Currently the volume is a NURBS volume, but the possible volume types will be extended in the future. The surface can be an elementary surface or a NURBS surface. It can be trimmed or not. Moreover, it is no restriction on these models with respect to the object configuration. A corner-to-corner configuration may be represented, but more general configurations may also be represented. The SurfaceModel and VolumeModel concepts are, in fact, too flexible for being ideal for an isogeometric block structured model.

- Isogeometric analysis is concerned with a partial differential equation solved by finite element analysis, a physical domain represented by a geometric model and a solution to the equation. Both the domain and the solution field is represented in the same function space, namely the spline space. Moreover, the spline spaces of the geometry entity and one or more solutions related to that entity is the same up to a possible varying degree of refinement. Thus, a data structure that is able to combine a geometry entity and one or more corresponding solutions including boundary conditions, seems preferable.

The SurfaceModel and VolumeModel classes are not obsolete. They are valuable during model build where an extended flexibility with regard to geometry types and topological configuration is needed, but a more tailor-made data structure is feasible for the analysis phase.

The isogeometric_model module is designed to host a data structure for isogeometric block structured surface- and volume models. The physical domain is modelled using the functionality related to SurfaceModel and VolumeModel and the domain is represented using these classes. At the finishing stage of the modelling phase it is ensured that all geometric entities are described as NURBS entities and that the corner-to-corner configuration condition is satisfied. Then the model is ready for being translated into an isogeometric model.

Figure 7 illustrates the data structure of the isogeometric model. A model can be of either surface or volume type, i.e. an IsogeometricSfModel entity or an IsogeometricVolModel entity. A model entity points to at least one block being represented by the classes IsogeometricSfBlock and IsogeometricVolBloc. The module takes a SurfaceModel or a VolumeModel entity as input, respectively, and distributes the content to a number of IsogeometricSfBlock or IsogeometricVolBlock entities. The blocks knows about their neighbours, at most 4 for surfaces and at most 6 for volumes. A block contains information about a number of solution entities related to the part of the physical domain represented in the block. Normally, there is one solution associated with a block, but it may be more. A solution has knowledge about boundary conditions and point wise

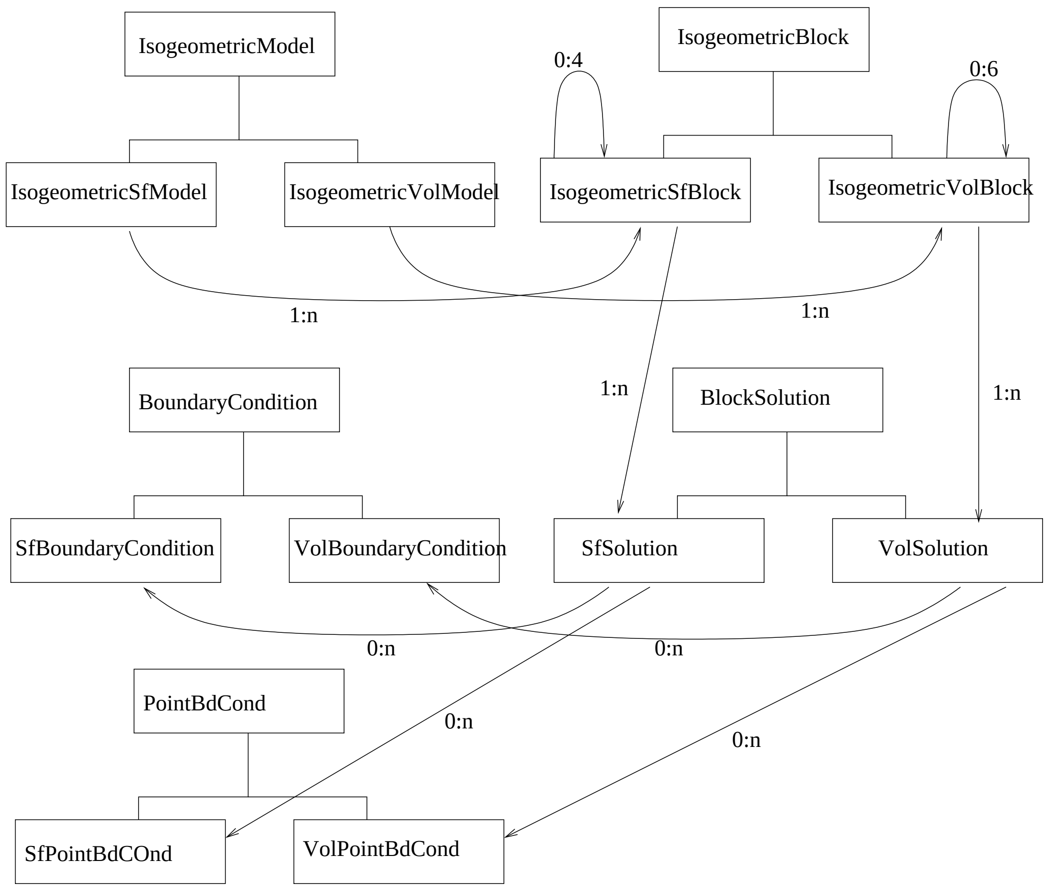


Figure 7: The data structure of the isogeometric model

boundary conditions. These conditions are assigned to the models, and they are further distributed to the appropriate block and solution.

The various classes belonging to the isogeometric_model module provide an interface to the spline objects that is convenient when used in the setting of isogeometric analysis. It is for instance possible to:

- Refine a solution space or perform degree elevation on it
- Fetch information about neighbouring blocks given one block
- Ensure that all adjacent solution spaces share the same spline space to get correspondence of coefficients at the boundary
- Fetch information about coefficient correspondence at block boundaries, using the local enumeration of coefficients within a block
- Fetch information about coefficients belonging to the model boundaries, again using local enumeration
- Store boundary conditions along with the corresponding geometry model for easy access to the condition
- Assign Dirichlet boundary conditions to the model boundary representing the boundary condition in the spline space of the corresponding blocks
- Perform pre evaluation, store the result and return the information as appropriate in order to utilize the tensor product structure of spline surfaces and volumes
- Update the description of the physical domain when appropriate in for instance fluid structure interaction

Further some support functions may be beneficial; we need to build a global system of equations. We need to iterate over the model to enable this. We need to be able to map local variables to global variables in the algebraic system of equations. Numerical quadrature is at the core of this assembly process. The isogometric model keeps track of correspondence of local variables, i.e. spline coefficients, but do not perform any global enumeration. Numerical quadrature is supported by pre evaluation.

The isogeometric model structure is relatively static. Limited geometric updates can be performed from the application. No topological changes are allowed. The solutions fields are refined whenever the geometry entities are refined, but the geometry entities are refined with respect to the solution entities only when this is issued from the application.

The isogeometric_model module was intended to be a part of the current deliverable, but this part of the deliverable has to be postponed slightly. The module will be added to the toolbox as soon as it is completed.

6 Extensions to the geometry tools

One emphasis of this delivery is multiblock models. This extensions mainly involves topology structures. However, also the geometry tools are extended. The remainder of this

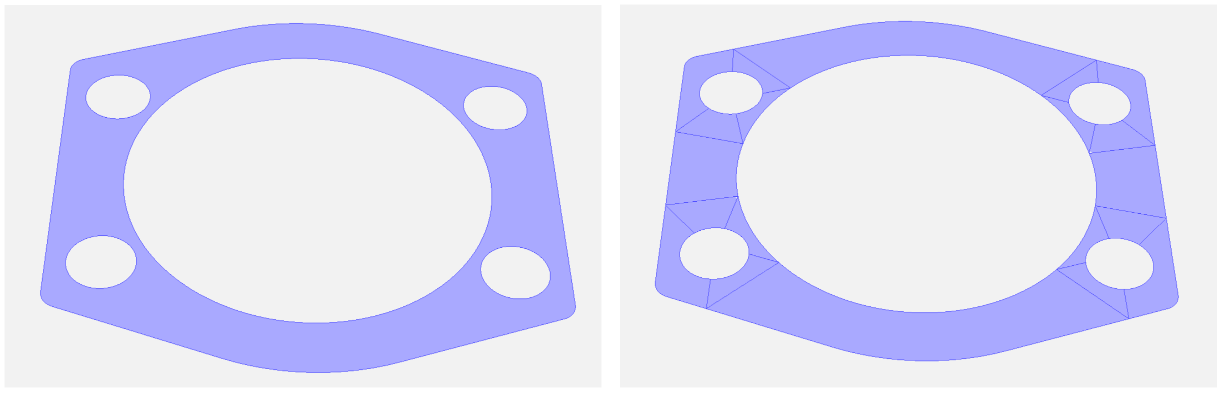


Figure 8: Replacing a surface with outer and inner trimming with a number of four-sided surfaces

section focus on extensions related to GoTools while the sections 7 to 10 focus on extensions to other parts of the isogeometric toolbox.

The previous version of the toolbox contained functionality to approximate an evaluator based curve or set of curves by one or more C^1 continuous spline curves. This functionality has been extended with two new evaluator based curves

- An intersection curve represented as two corresponding curve-on-surface curves
- The projection of a 3D curve onto a 3D surface

The class AdaptCurve represents an alternative method to approximate an evaluator based curve. While the previous method uses an Hermite approach, this class performs least squares approximation with smoothing. This method allows for higher order continuity in the resulting curve.

A method for smoothing of a set of curves is included. Each curve may interpolate or approximate a number of points and derivatives in addition to smoothing. Moreover, constraints can be set to ensure equality of crossing curves in certain curve parameters. This functionality is implemented in the class SmoothCurveSet.

The class ApproxSurf aims at approximating a set of scattered points with a spline surface. In this delivery, the class has been extended with the possibility to refine the spline space if this is required to meet a given approximation accuracy.

6.1 Translating a trimmed face into a number of 4-sided surfaces

A multiblock isogeometric model consists of non-trimmed NURBS blocks. A CAD model is to a large extent a collection of trimmed surfaces. Moreover, trimming is a convenient tool in the design process. Thus, a tool that can translate trimmed surfaces into a set of non-trimmed NURBS surfaces is highly convenient. Moreover, a multiblock surface model may be used as a building block in the process of creating a multiblock volume model.

Figure 8 shows a trimmed surface being translated into a set of 4-sided surfaces. The procedure is as follows:

- Identify all holes

- Sort the holes. In this case the midpoint of the large hole is taken as midpoint of the model and the other holes are sorted radially around this midpoint
- Isolate the smaller holes by intersecting the face with planes through the midpoint of the large hole, and reorganize the initial trimming curve and the intersection curves to create new trimming loops. The position of the planes are set in order to split between the holes, but also to place the hole in the vicinity of midpoint of a sub surface.
- Split sub surfaces with holes by intersecting with planes from corner vertices to the mid point of the current hole.

Throughout the process, the topology information of the face set is maintained in order to avoid creating T-junctions when surfaces are divided. Whenever possible, we choose to split through existing vertices.

Given a set of 4-sided trimmed surface, each surface may be replaced by a non-trimmed spline surface. The boundaries are interpolated using a Coons patch approach and the inner of the surfaces are approximated by adapting the new surfaces to a dense point set sampled from the original surface. In this context least squares approximation with a smoothing term is applied. Parameter iteration with respect to the point set is performed. C^0 continuity of the resulting surface set is guaranteed by this approach.

6.2 Volume fairing

Volume fairing is an extension to surface fairing which already is a part of GoTools and thereby the isogeometric toolbox, see [13]. The idea is to minimize a functional consisting of a number of terms, in the volume case:

- Least squares approximation
- A smoothing term
- Periodicity in one or two parameter directions

Let the NURBS volume be

$$V(u,v,w) = \sum_{i,j,k} V_{i,j,k} c_{i,j,k}$$

where $c_{i,j,k}$ are the volume coefficients and

$$V_{i,j,k} = \frac{B_{i,d_1,u}(u)B_{j,d_2,v}(v)B_{k,d_3,w}(w)}{\sum_{i',j',k'} h_{i',j',k'} B_{i',d_1,u}(u)B_{j',d_2,v}(v)B_{k',d_3,w}(w)}$$

is the rational basis functions corresponding to the coefficients. Here B denote the B-spline basis functions in each parameter direction, d_r is the polynomial degree in the three parameter directions and h are the rational weights. In the polynomial case all weights are equal to 1.

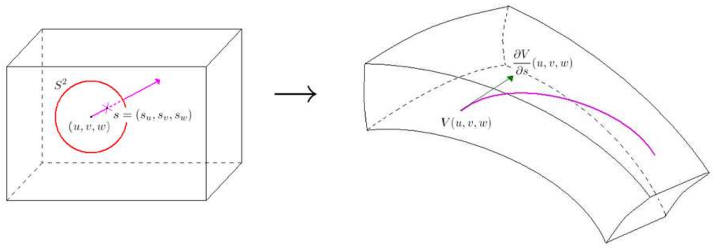


Figure 9: Directions of differentiation in the domain and NURBS volume

Figure 9 illustrates the mapping from the trivariate parameter domain to the volume. At each point in the volume, the volume is differentiated in all directions on the unit sphere.

$$
\begin{aligned}
\frac{\delta^n V}{(\delta s)^n} &= \frac{d^n}{(dt)^n}(t \to V((u, v, w) + ts))_{t=0} \\
&= \sum_{n_1+n_2+n_3=n} \binom{n}{n_1, n_2, n_3} \frac{\delta^n V}{(\delta u)^{n_1}(\delta v)^{n_2}(\delta w^{n_3})}(u, v, w)
\end{aligned}
$$

$n_1 > 0,\ n_2 > 0,\ n_3 > 0$. These derivatives are then integrated around the unit sphere and over the domain of the current volume:

$$
E_n = \int_u \int_v \int_w \iint_{s \in S^2} \left(\frac{\delta^n V}{(\delta s)^n}(u, v, w) \right)^2 ds\, du\, dv\, dw
$$

The final smoothing term becomes

$$
E = \sum_{n=1}^{3} \omega_n E_n
$$

where $\omega_n, \quad n = 1, 2, 3$ are weights associated to the smoothing of the expression in the corresponding derivative.

The smoothing term may be added a least squares term and a periodicity term to define the final fairness functional. The periodicity term minimizes discontinuities of derivaties up to second order across the seem. A number of coefficients at the boundary are normally kept fixed while the coefficients in the inner of the volume are modified to improve the parameterization of the volume.

Figure 10 shows the mean curvature of an iso surface in a spline volume before and after smoothing.

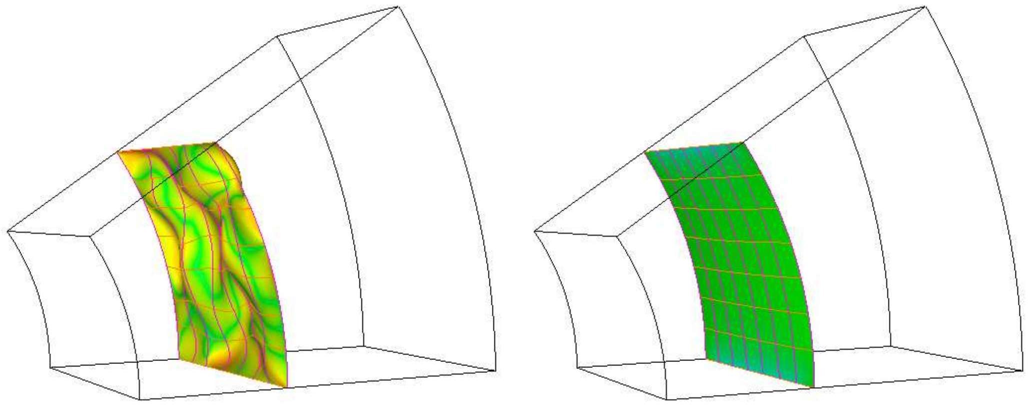


Figure 10: Volume smoothing

7 Parameterization tools for computational domain

In finite element analysis (FEA), mesh generation, which generates discrete geometry as computational domain from given CAD object, is a key and the most time-consuming step. In IGA framework, parameterization of computational domain, which corresponds to the mesh generation in FEA, also has some impact on analysis result and efficiency. Moreover, in FEA, one can perform arbitrary refinements on the computational mesh, but in IGA using tensor product B-splines, the refinement is not arbitrary, we can only perform refinement operations in u direction and v direction by knot insertion or degree evaluation. Hence, parameterization of computational domain is more important in IGA.

The parameterization of a computational domain in IGA is determined by control points, knot vectors and the degrees of B-spline objects. For IGA problem of two dimension, the knot vectors and the degree of computational domain are determined by the given boundary curves. Hence, finding the optimal placement of inner control points for a specified physical problem, is a key issue in IGA. A basic requirement of resulting parameterization for IGA is that it doesn't have self-intersections. In this section, we first propose a linear and easy-to-test sufficient condition for injectivity of planar B-spline parameterization. Then we show that different parameterization of computational domain has different impact on the simulation results in IGA. For problems with exact solutions, a shape optimization method is proposed to obtain an optimal parameterization of computational domain. For problems with unknown exact solutions, we proposed a harmonic parameterization method to construct the inner control points. Some examples and comparisons are presented based on the heat conduction problem to show the effectiveness of the proposed method.

7.1 Easy-to-check tools for injectivity parameterization

The main idea of the isogeometric approach is to use the same representation for the geometry and the physical solutions we are interested in. Schematically, the geometry Ω involved in the physical problem can be a surface or a volume in a three-dimensional space $\mathbb{R}^3$. Let us call $\mathbf{x} = (x, y, z)$ the coordinates associated to this space. In our case, this geometry will be represented by a parameterization σ for a domain $\mathcal{P}$ of the parameter space. Let us call $\mathbf{u}$ the coordinates of this parameter domain, which could be of dimension 2 for a surface or 3 for a volume. This parameterization will be given by B-spline functions with knots in $\mathcal{P}$ and control points in $\mathbb{R}^3$.

The concept of isogeometry consists in representing the physical quantities $\Phi \in \mathbb{R}^p$ on the geometry Ω using the same type of B-spline representation as for the geometry Ω. In other words, given a point $\mathbf{x} = \sigma(\mathbf{u}) \in \Omega$ with $\mathbf{u} \in \mathcal{P}$, we associate to it the physical quantities $\Phi(\mathbf{u})$ where $\Phi(\mathbf{u})$ is a B-spline function with nodes in $\mathcal{P}$ and control points in $\mathbb{R}^p$. This means that the map $\mathbf{x} \in \Omega \mapsto \Phi \in \mathbb{R}^p$ is defined *implicitly* as $\mathbf{x} \mapsto \Phi \circ \sigma^{-1}(\mathbf{x})$.

Consequently, the framework of isogeometry is thus valid when the parameterization σ of the geometry is *injective* (or bijective on its image). We are going to describe sufficient and easy-to-check conditions for the injectivity of σ. We will consider this problem in the context of finding a "good" parameterization of a domain when its boundary is given. In [4], a general sufficient condition is proposed for injective parameterization.

Proposition.1 Suppose that σ is a C^1 parameterization from a compact domain $\mathcal{P} \subset \mathbb{R}^n$ with a connected boundary to a geometry $\Omega \subset \mathbb{R}^n$. If σ is injective on the boundary $\partial\mathcal{P}$ of $\mathcal{P}$ and its Jacobian J_σ does not vanish on $\mathcal{P}$, then σ is injective.

For a parameterization σ from $[a,b] \times [c,d]$ to $\Omega \subset \mathbb{R}^2$, we define the boundary curves as the image of $\{a\} \times [c,d], \{b\} \times [c,d], [a,b] \times \{c\}, [a,b] \times \{b\}$ by σ. We say that σ defines a *regular boundary* if these curves do not intersect pairwise, except at their end points and if they have no self-intersection.

As a consequence of the previous proposition, we get the following injectivity test for standard B-spline tensor product parameterization of a planar domain.

Proposition.2 Let σ be a C^1 parameterization from $[a,b] \times [c,d]$ to $\Omega \subset \mathbb{R}^2$ which defines a regular boundary. If its Jacobian J_σ does not vanish on $[a,b] \times [c,d]$, then σ is injective.

These tests involve injectivity conditions on the boundary, which can be checked recursively using the same techniques, non-intersection tests for boundary curves and surfaces which are provided for instance by geometric (subdivision) algorithms and the local injectivity condition corresponding to the non-vanishing of the Jacobian. This last condition requires to test on all the domain Ω that the Jacobian does not vanish. Hereafter we propose a sufficient and easy-to-test condition to ensure the local injectivity condition.

We consider first the case of a planar parameterization

$$\sigma : \mathbf{u} \in \mathcal{P} := [a,b] \times [c,d] \mapsto \sigma(\mathbf{u}) := \sum_{0 \le i \le l_1, 0 \le j \le l_2} \mathbf{c}_{i,j} N_{i,j}(\mathbf{u}),$$

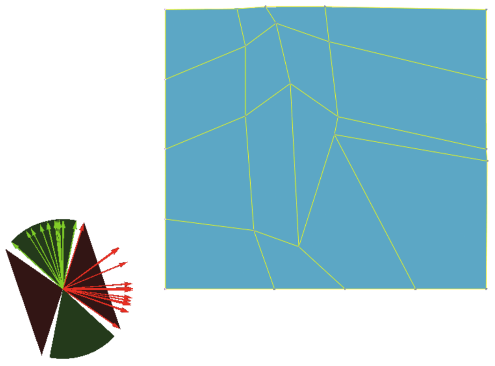
where $\mathbf{c}_{i,j} \in \mathbb{R}^2$ are the control points and $N_{i,j}(\mathbf{u})$ are the B-spline basis functions. The derivative of $\sigma(\mathbf{u})$ with respect to $\mathbf{u}_1$ can be expressed in terms of the differences $\Delta^1_{i,j} := \mathbf{c}_{i+1,j} - \mathbf{c}_{i,j}$:

$$\partial_{u_1}\sigma(\mathbf{u}) := \sum_{0 \le i \le l_1 - 1, 0 \le j \le l_2} \omega^1_{i,j} \Delta^1_{i,j} N^1_{i,j}(\mathbf{u}),$$

where $N^1_{i,j}$ is the B-spline basis function with one degree less in $\mathbf{u}_1$, $\omega^1_{i,j}$ is a positive factor. We denote by $\mathcal{C}_1(\mathbf{c})$ the convex cone of $\mathbb{R}^2$ generated by the half rays $\mathbb{R}_+ \cdot \Delta^1_{i,j}$.

Similarly, the derivative of $\sigma(\mathbf{u})$ with respect to $\mathbf{u}_2$ can be expressed in terms of the differences $\Delta^2_{i,j} := \mathbf{c}_{i,j+1} - \mathbf{c}_{i,j}$:

$$\partial_{u_2}\sigma(\mathbf{u}) := \sum_{0 \le i \le l_1 - 1, 0 \le j \le l_2 - 1} \omega^2_{i,j} \Delta^2_{i,j} N^2_{i,j}(\mathbf{u}),$$


(a) transverse cone

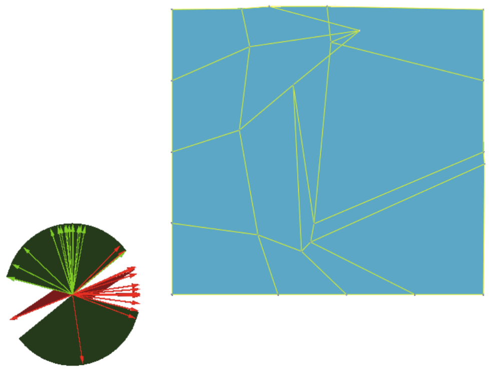

(b) non-transverse cone

Figure 11: Injectivity test by cones.

where $N^2_{i,j}$ is the B-spline basis with one degree less in $\mathbf{u}_2$, $\omega^2_{i,j}$ is a positive factor. We denote by $\mathcal{C}_2(\mathbf{c})$ the convex cone of $\mathbb{R}^2$ generated by the half rays $\mathbb{R}_+ \cdot \Delta^2_{i,j}$. If there exist two opposite vectors, which are on a straight line, we define $\mathcal{C}_i(\mathbf{c})$ as a half-plane.

We say that two cones $\mathcal{C}_1, \mathcal{C}_2$ are *transverse* if $\mathbb{R} \cdot \mathcal{C}_1$ and $\mathbb{R} \cdot \mathcal{C}_2$ intersect only at $\{0\}$.

Theorem 1. Let σ be a B-spline parametrisation, which is at least C^1 from $\mathcal{P} := [a, b] \times [c, d]$ to $\Omega \subset \mathbb{R}^2$ given by the control points $\mathbf{c}$. If the boundary curves do not intersect and have no self-intersection point and the cones $\mathcal{C}_1(\mathbf{c}), \mathcal{C}_2(\mathbf{c})$ are transverse, then σ is injective on $\mathcal{P}$.

Proof. We check first that the transversality of the cones $\mathcal{C}_1(\mathbf{c}), \mathcal{C}_2(\mathbf{c})$ implies that the Jacobian of σ is not vanishing. This Jacobian $J_\sigma(\mathbf{u})$ is obtained by taking the determinant $|\partial_{\mathbf{u}_1}\sigma, \partial_{\mathbf{u}_2}\sigma|$ which expands as

$$\sum_{0 \le i \le l_1-1, 0 \le j \le l_2} \sum_{0 \le i' \le l_1-1, 0 \le j' \le l_2-1} |\Delta^1_{i,j}, \Delta^2_{i',j'}| \omega^1_{i,j} \omega^2_{i',j'} N^1_{i,j}(\mathbf{u}) N^2_{i',j'}(\mathbf{u}).$$

Since the cone $\mathcal{C}_1(\mathbf{c})$ and $\mathcal{C}_2(\mathbf{c})$ are transverse, the determinants $|\Delta^1_{i,j}, \Delta^2_{i',j'}|$ have a constant sign for $\Delta^1_{i,j} \in \mathcal{C}_1(\mathbf{c})$, $\Delta^2_{i',j'} \in \mathcal{C}_2(\mathbf{c})$. As the basis functions and the factors are positive, the Jacobian $J_\sigma(\mathbf{u})$ cannot vanish at $\mathbf{u} \in \mathcal{G}$, except if all the $N^1_{i,j}(\mathbf{u})\, N^2_{i',j'}(\mathbf{u})$ vanish, which is not possible.

The map σ is locally injective on $\mathcal{P}$. By Proposition 7.1, we deduce that σ is globally injective on $\mathcal{P}$. $\square$

Figure 11 shows two examples of the injectivity testing method. In Figure 11 (a), it satisfies the sufficient condition in our method, but it does not satisfy the sufficient condition of the method proposed in [8]. Hence, our method is an improved version of the method presented in [8].

7.1.1 Linear constraint for injectivity.

This condition can be used to devise an algorithm which constructs an injective parameterization for given boundary control points. We first consider the planar case. Given four planar boundary curves described by the controls points $\mathbf{c}_{i,0}, \mathbf{c}_{i,l_2}, \mathbf{c}_{0,j}, \mathbf{c}_{l_1,j}$, with $0 \le i \le l_1, 0 \le j \le l_2$, we define the boundary cone $\mathcal{C}^0_1(\mathbf{c})$ (resp. $\mathcal{C}^0_2(\mathbf{c})$) as the cone generated by the vectors $\Delta^1_{i,0}(\mathbf{c})$, $\Delta^1_{i,l_2}(\mathbf{c})$ for $0 \le i \le l_1 - 1$ (resp. $\Delta^2_{0,j}(\mathbf{c})$, $\Delta^2_{l_2,j}(\mathbf{c})$ for $0 \le j \le l_2 - 1$). We assume that these boundary curves form a regular boundary and that the two boundary cones $\mathcal{C}_1^0(\mathbf{c})$, $\mathcal{C}_2^0(\mathbf{c})$ are transverse. $\mathbb{R} \cdot \mathcal{C}_1^0(\mathbf{c})$ is the cone defined by $F_1^+(\mathcal{C}_1^0(\mathbf{c})) \leq 0, F_1^-(\mathcal{C}_1^0(\mathbf{c})) \leq 0$, where F_1^+ and F_1^- are the linear equations defining the boundary of $\mathbb{R} \cdot \mathcal{C}_1^0(\mathbf{c})$. We defined similarly F_2^+, F_2^- for $\mathcal{C}_2^0(\mathbf{c})$.

To apply Proposition 7.1, the inner control points $\mathbf{c}_{i,j}$ should satisfy the following linear constraints for injective parameterization:

$$\begin{cases} F_1^+(\mathbf{c}_{i+1,j} - \mathbf{c}_{i,j}) \leq 0, F_1^-(\mathbf{c}_{i+1,j} - \mathbf{c}_{i,j}) \leq 0, 0 \leq i < l_1, 0 < j < l_2 \\ F_2^+(\mathbf{c}_{i,j+1} - \mathbf{c}_{i,j}) \leq 0, F_2^-(\mathbf{c}_{i,j+1} - \mathbf{c}_{i,j}) \leq 0, 0 < i < l_1, 0 \leq j < l_2. \end{cases} \tag{1}$$

The linear condition in (1) is a rather restrictive condition, and it is sufficient to require that the two cones constructed from the first derivative vectors are separated. Inspired from [9], the following constraints are proposed as alternative condition

$$\begin{cases} F_2^+(\mathbf{c}_{i+1,j} - \mathbf{c}_{i,j}) + F_1^-(\mathbf{c}_{i+1,j} - \mathbf{c}_{i,j}) \leq 0, F_2^-(\mathbf{c}_{i+1,j} - \mathbf{c}_{i,j}) + F_1^+(\mathbf{c}_{i+1,j} - \mathbf{c}_{i,j}) \leq 0, \\ F_2^+(\mathbf{c}_{i,j+1} - \mathbf{c}_{i,j}) + F_1^-(\mathbf{c}_{i,j+1} - \mathbf{c}_{i,j}) \leq 0, F_2^-(\mathbf{c}_{i,j+1} - \mathbf{c}_{i,j}) + F_1^+(\mathbf{c}_{i,j+1} - \mathbf{c}_{i,j}) \leq 0, \end{cases}$$

where $0 < i < l_1, 0 \leq j < l_2$.
Remarks 1. For 3D case, the 3D convex cones can be also constructed from the derivative vectors in three parametric directions. The difference is that the cross product condition should be considered in the injectivity condition as in [8]. These conditions provide an easy-to-check method for the injectivity of a parameterization.

7.2 Optimal analysis-ware parameterization of computational domain

7.2.1 Problem statement

The problem studied in this section can be stated as follows: given four coplanar boundary B-spline curves, find the inner control points such that the parameterization of a computational domain is optimal for an IGA problem with known exact solution. The extension of the proposed method to isogeometric problems without known exact solution is one of our ongoing work.

7.2.2 Shape optimization method

The shape optimization problem consists in finding the shape which is optimal in that it minimizes a certain cost function while satisfying given constraints. The purpose of shape optimization in CAE is to optimize the CAD object for some physical problem, and the design variables are the control points of the CAD object. For 2D isogeometric shape optimization problem, the design variables are the control points of boundary B-spline curves.

Inspired from the idea of shape optimization, in order to obtain optimal parameterization of computational domain, we should let the inner control points, rather than boundary control points, be the design variables for the shape optimization, and find the best placement of inner control points to make the value of a cost function as small as possible.
Initial construction of inner control points.

As the shape optimization problem, we need to construct an initial placement of inner control points as starting point in the iteration process. We rely on the discrete Coons method presented in [7] to generate inner control points as initial value from boundary control points.

Given the boundary control points $\boldsymbol{P}_{0,j}, \boldsymbol{P}_{n,j}, \boldsymbol{P}_{i,0}, \boldsymbol{P}_{i,m}$, $i = 0, \ldots, n, j = 0, \ldots, m$, the inner control points $\boldsymbol{P}_{i,j}$ $(i = 1, \ldots, n-1, j = 1, \ldots, m-1)$ can be constructed by the discrete Coons method as follows:

$$\begin{aligned}\boldsymbol{P}_{i,j} &= (1-\frac{i}{n})\boldsymbol{P}_{0,j} + \frac{i}{n}\boldsymbol{P}_{n,j} + (1-\frac{j}{m})\boldsymbol{P}_{i,0} + \frac{j}{m}\boldsymbol{P}_{i,m} \\ &\quad -[1-\frac{i}{n} \quad \frac{i}{n}]\begin{pmatrix} \boldsymbol{P}_{0,0} & \boldsymbol{P}_{0,m} \\ \boldsymbol{P}_{n,0} & \boldsymbol{P}_{n,m} \end{pmatrix}\begin{pmatrix} 1-\frac{j}{m} \\ \frac{j}{m} \end{pmatrix}\end{aligned}$$

Remarks 1. Since the sum of the coefficients equals 1, the resulting inner control points lie in the convex hull of the boundary control points.
Remarks 2. For some given boundary curves, this construction may cause some self-intersections, and lead to an improper parameterization for IGA. We use the linear injectivity condition proposed in Section 7.1.1 to check the injectivity of initial parameterization. If it does not satisfy the condition, the linear programming method is used to produce another initial parameterization.
Optimization method. In the proposed approach, we minimize the error computed from the IGA solution and the exact solution, by moving inner control points of the computational domain. Therefore, we consider as optimization variables the coordinates of the inner control points and as cost function the error of the IGA solution. The optimization algorithm used for this study is a classical steepest-descent method in conjunction with a back-tracking line-search. For this exercise, the gradient of the cost function is approximated using a centered finite-differencing scheme.

Each iteration k of the optimization algorithm can be summarized as follows, starting from a point x_k in the variable space:

1. Evaluation of perturbed points $x_k + \epsilon e_k$

2. Estimation of the gradient $\nabla f(x_k)$ by finite-difference

3. Define search direction $d_k = -\nabla f(x_k)$

4. Line search : find ρ such as $f(x_k + \rho d_k) < f(x_k)$

These steps are carried out until a stopping criterion is satisfied.

7.2.3 Examples and comparison

In this section, we will present some parameterization results and compare them with the initial solution with respect to the following heat conduction problem :

Given a domain Ω with $\Gamma = \partial\Omega_D \cup \partial\Omega_N$, we consider the following thermal conduction problem:

$$\begin{aligned}&\boldsymbol{\nabla}(\kappa(\mathbf{x})\boldsymbol{\nabla}T(\mathbf{x})) = f(\mathbf{x}) \quad \text{in } \Omega \\ &T(\mathbf{x}) = T_0(\mathbf{x}) \text{ on } \partial\Omega_D \\ &\kappa(\mathbf{x})\frac{\partial T}{\partial \mathbf{n}}(\mathbf{x}) = \Phi_0(\mathbf{x}) \text{ on } \partial\Omega_N,\end{aligned} \tag{2}$$

where $\mathbf{x}$ are the Cartesian coordinates, T represents the temperature field and κ the thermal conductivity. Dirichlet and Neumann boundary conditions are applied on $\partial\Omega_D$ and $\partial\Omega_N$ respectively, T_0 and Φ_0 being the imposed temperature and thermal flux ($\mathbf{n}$ unit vector normal to the boundary). f is a user-defined function that allows to generate problems with an analytical solution, by adding a source term to the classical heat conduction equation.
Example I . The fist example is for the parameterization of the domain

$$\Omega(x,y) = \{(x,y)| -1 \le y \le x^2, 0 \le x \le 1\}$$

by Bézier surface with degree 3×6. The parabola is represented by degenerate cubic Bézier curve. For the problem with boundary condition $\boldsymbol{T}_0(\boldsymbol{x}) = 0$ and $\Phi_0(\boldsymbol{x}) = 0$ in (2), we can construct an exact solution $\boldsymbol{T}(x,y)$ as follows

$$\boldsymbol{T}(x,y) = \sin(\pi(y-x^2))\sin(\pi x)\sin(\pi y)$$

The initial placement of inner control points is produced by the discrete Coons method as shown in Figure 12 (a). The final parameterization results and some comparisons are also shown in Figure 12. We can find that there are some self-intersections on the control mesh in Figure 12 (b). However, there is no self-intersection on the final parameterization as shown in Figure 12 (c). During the optimization, the initial error is reduced by 14.65% as shown in Figure 12 (g). The error history during refinement operation is presented in Figure 12 (h).
Example II. The second example is for the parameterization of the domain $\Omega = [0,3]\times[0,6]$ by cubic B-spline surface with knot vector $\{0,0,0,0,1,2,3,4,4,4,4\}$ in the u direction and knot vector $\{0,0,0,0,1,1,1,1\}$ in the v direction. We test these two parameterizations on the heat conduction problem (2) with source term

$$\boldsymbol{f}(x,y) = -\frac{4}{9}\pi^2\sin(\frac{\pi x}{3})\sin(\frac{\pi y}{3}).$$

For this problem with boundary condition $\boldsymbol{T}_0(\boldsymbol{x}) = 0$ and $\Phi_0(\boldsymbol{x}) = 0$, the exact solution over the computational domain $[0,6]\times[0,6]$ is

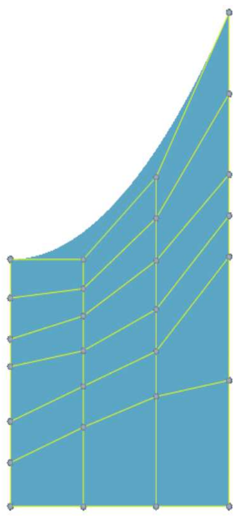
$$\boldsymbol{T}(x,y) = 2\sin(\frac{\pi x}{3})\sin(\frac{\pi y}{3}).$$

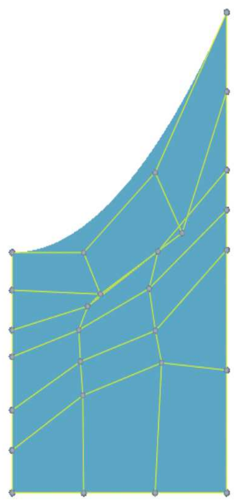
The initial placement of inner control points is non-uniform as shown in Figure 13 (a), and the final parameterization result and some comparison are also shown in Figure 13. During the optimization, the initial error is reduced by 3.31% as shown in Figure 13 (g). The error history during refinement operation is presented in Figure 13 (h).

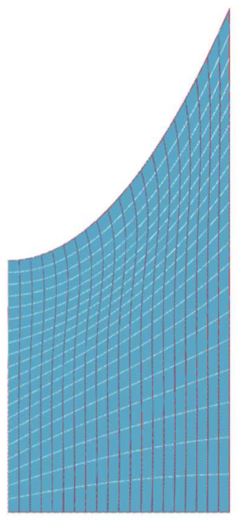
More details and examples can be checked in [6].

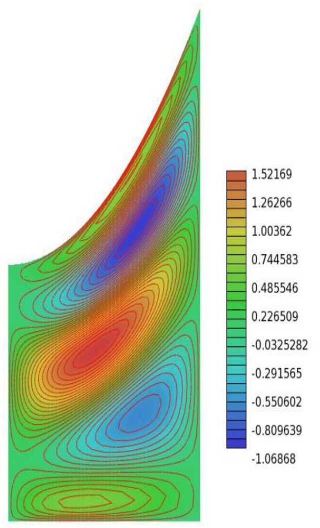
7.3 Variational harmonic spline method for parameterization of computational domain

Injectivity is a basic requirement in parameterization of computational domain for IGA problems. In this section, a variational harmonic spline method is proposed to obtain injective parameterization with high quality.

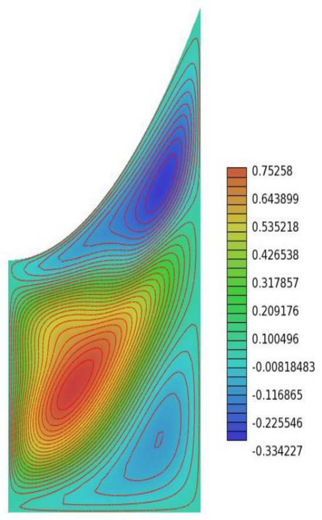

(a) initial control mesh and surface


(b) final control mesh and surface

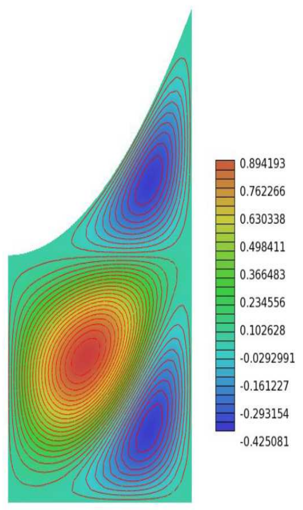

(c) final parameterization



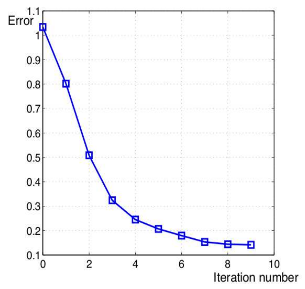
(d) color map and iso-temperature lines of initial solution surface



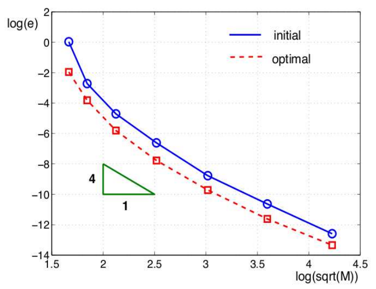
(e) color map and iso-temperature lines of final solution surface



(f) color map of exact solution surface

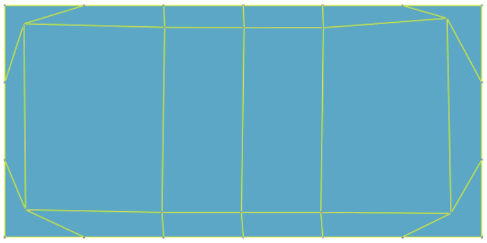


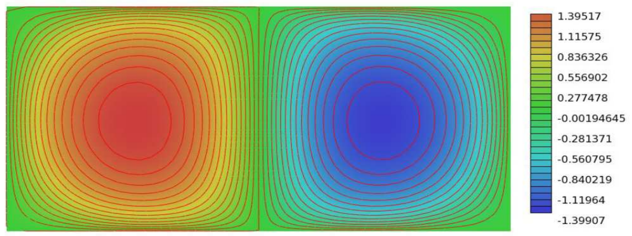
(g) error history during the optimization process



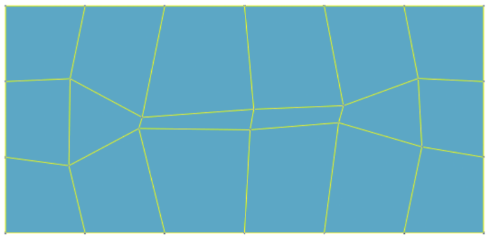
(h) error history during the refinement process

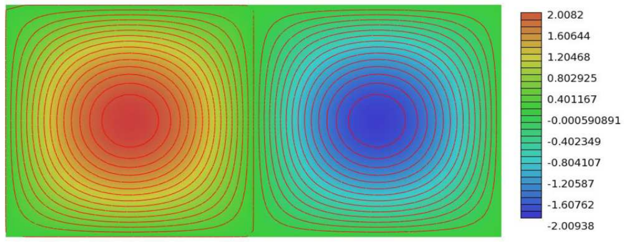
Figure 12: Example I.


(a) initial control mesh and surface

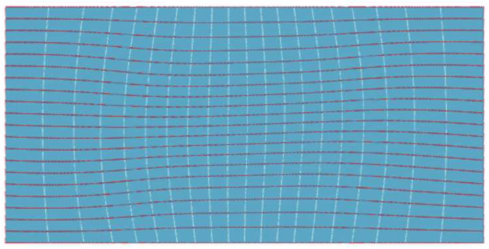


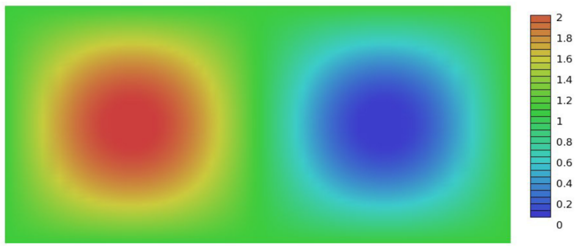
(b) color map of initial solution surface


(c) final control mesh and surface

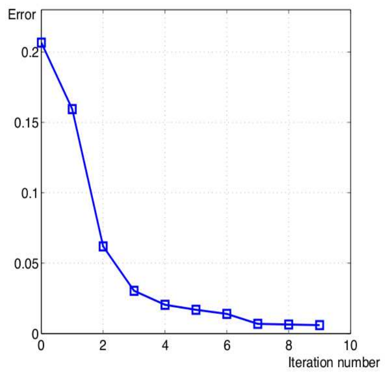


(d) color map and iso-temperature lines of final solution surface

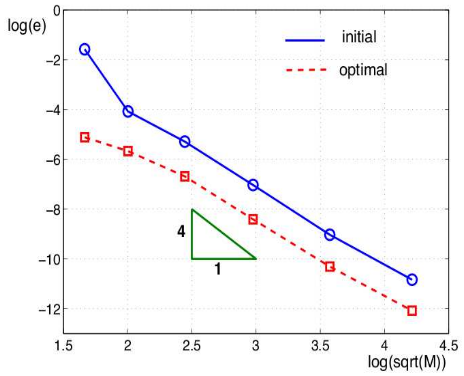

(e) final parameterization



(f) color map of exact solution surface



(g) error history during the optimization process



(h) error history during the refinement process

Figure 13: Example II.

In the field of mesh generation in FEA, a general method is based on partial differential equations. The grid points are the solution of an elliptic partial differential equation system with Dirichlet boundary conditions on all boundaries. There are several advantages in elliptic mesh generation. The theory of partial differential equations guarantees that the mapping between physical and transformed regions will be one-to-one. Another important property is the inherent smoothness in the solution of elliptic systems. A disadvantage of elliptic method is that there will be some non-uniform grid elements near convex (concave) parts of the boundary. Different from the previous elliptic mesh generation method, the proposed method focus on isogeometric version(the coordinates of interior control points are unknown variables), and converts the elliptic PDE into a nonlinear constraint optimization problem. A regular term is integrated into the optimization formulation to achieve more uniform grid.

Consider a simply connected bounded domain $\mathcal{S}$ in two dimensional space with Cartesian coordinates $(x;y)^T$. Suppose that $\mathcal{S}$ is bounded by four B-spline curves $\mathbf{S}(\xi,0), \mathbf{S}(\xi,1), \mathbf{S}(1,\eta), \mathbf{S}(0,\eta)$. The parametric domain $\mathcal{P}$ of $\mathcal{S}$ should be a rectangular in two dimensional space with coordinates ξ,η, which is determined by the knot vector of boundary B-spline curves in ξ and η direction. The mapping from parametric space $\mathcal{P}$ to physical space $\mathcal{S}$ can be described as a B-spline surface $\mathbf{S}(\xi,\eta) = (x(\xi,\eta), y(\xi,\eta)) = \sum_{i=0}^{n}\sum_{j=0}^{m} N_i^p(\xi)N_j^q(\eta)\mathbf{P}_{i,j}$ with given four B-spline curves as boundaries. $N_i^p(\xi)$ and $N_j^q(\eta)$ are B-spline basis functions, $\mathbf{P}_{i,j}$ are control points. Assume that this mapping is prescribed which maps the boundary of $\mathcal{P}$ one-to-one on the boundary of $\mathcal{S}$. The parameterization problem of computational domain can be stated as: given four boundary B-spline curves, find the placement of inner control points such that the resulted planar B-spline surface is a good computational domain for isogeometric analysis.

Harmonic mapping, which is a one-to-one transformation both for 2D and 3D regions, will be used in our parameterization method. Let $\sigma : \mathcal{S} \mapsto \mathcal{P}$ be a harmonic mapping from $\mathcal{S}$ to $\mathcal{P}$. That is,

$$\Delta\xi(x,y) = \xi_{xx} + \xi_{yy} = 0$$

$$\Delta\eta(x,y) = \eta_{xx} + \eta_{yy} = 0$$

By chain rules, we have

$$d\xi = \xi_x dx + \xi_y dy = \xi_x(x_\xi d\xi + x_\eta d\eta) + \xi_y(y_\xi d\xi + y_\eta d\eta) \tag{3}$$

$$d\eta = \eta_x dx + \eta_y dy = \eta_x(x_\xi d\xi + x_\eta d\eta) + \eta_y(y_\xi d\xi + y_\eta d\eta) \tag{4}$$

From (3) and (4), we have

$$\xi_x x_\xi + \xi_y y_\xi = \eta_x x_\eta + \eta_y y_\eta = 1$$

$$\xi_x x_\eta + \xi_y y_\eta = \eta_x x_\xi + \eta_y y_\xi = 0$$

By solving above two linear systems, we obtain

$$\xi_x = \frac{y_\eta}{J}, \xi_y = \frac{-x_\eta}{J} \tag{5}$$

and

$$\eta_x = \frac{-y_\xi}{J}, \eta_y = \frac{x_\xi}{J} \tag{6}$$

where

$$J = x_\xi y_\eta - x_\eta y_\xi$$

is Jacobian of transformation from $\mathcal{P}$ to $\mathcal{S}$.

From (5)(6)and $\frac{\partial}{\partial x} = \xi_x \frac{\partial}{\partial \xi}$, $\frac{\partial}{\partial y} = \xi_y \frac{\partial}{\partial \xi}$, we have

$$\begin{aligned} J\Delta\xi(x,y) &= J\frac{\partial}{\partial x}\xi_x + J\frac{\partial}{\partial y}\xi_y \\ &= (J\xi_x\frac{\partial}{\partial \xi} + J\eta_x\frac{\partial}{\partial \eta})\xi_x + (J\xi_y\frac{\partial}{\partial \xi} + J\eta_y\frac{\partial}{\partial \eta})\xi_y \\ &= (y_\eta\frac{\partial}{\partial \xi} - y_\xi\frac{\partial}{\partial \eta})(\frac{y_\xi}{J}) + (-x_\eta\frac{\partial}{\partial \xi} + x_\xi\frac{\partial}{\partial \eta})(\frac{-x_\eta}{y}) = 0 \end{aligned}$$

After differential computation, we have

$$-y_\eta Lx + x_\eta Ly = 0 \tag{7}$$

where

$$L = (x_\eta^2 + y_\eta^2)\frac{\partial^2}{\partial \xi^2} - 2(x_\xi x_\eta + y_\xi y_\eta)\frac{\partial^2}{\partial \xi\eta} + (x_\xi^2 + y_\xi^2)\frac{\partial^2}{\partial \eta^2}$$

Similarly, from $J\Delta\eta(x,y) = 0$, we have

$$y_\xi Lx - x_\xi Ly = 0 \tag{8}$$

Then from (7) and(8), we have

$$Lx(\xi,\eta) = Ly(\xi,\eta) = 0$$

that is,

$$\| L\mathbf{S}(\xi,\eta) \|^2 = (Lx)^2 + (Ly)^2 = 0$$

This is a nonlinear system in terms of inner control points. The final grid obtained by this method usually has non-uniform elements near convex(concave) boundary region. In order to solve this problem, we consider the uniform issues and minimize the following energy function,

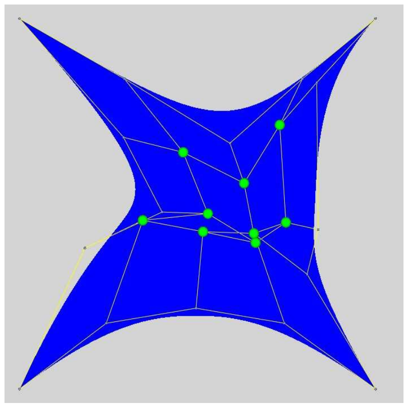
$$\iint \lambda_1(\| \mathbf{S}_{\xi\xi} \|^2 + \| \mathbf{S}_{\eta\eta} \|^2 + 2 \| \mathbf{S}_{\xi\eta} \|^2) + \lambda_2(\| \mathbf{S}_\xi \|^2 + \| \mathbf{S}_\eta \|^2) dudv \tag{9}$$

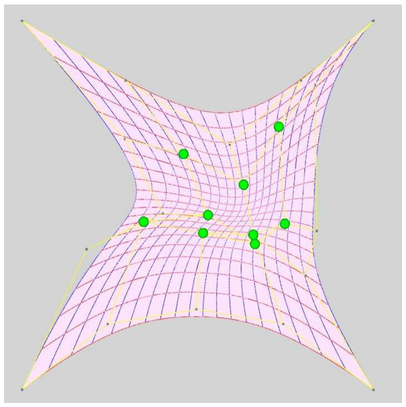
Then by adding the harmonic mapping term into (9), we obtain a new energy function for parameterization of computational domain

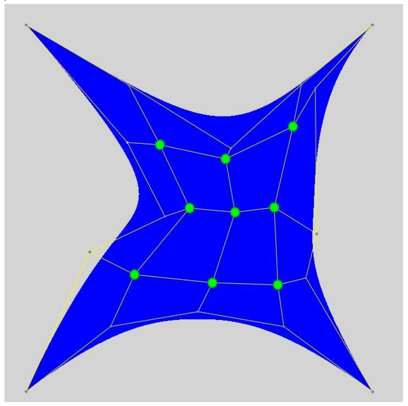
$$\iint \| L\mathbf{S}(\xi,\eta) \|^2 + \lambda_1(\| \mathbf{S}_{\xi\xi} \|^2 + \| \mathbf{S}_{\eta\eta} \|^2 + 2 \| \mathbf{S}_{\xi\eta} \|^2) + \lambda_2(\| \mathbf{S}_\xi \|^2 + \| \mathbf{S}_\eta \|^2) dudv$$

where λ_1 and λ_2 are positive weights to control the final parameterization results.

The Discrete Coons method in (2) is used to construct initial inner control points, and the Steepest Descent method is employed to minimize the energy function. Figure 14(a)(b) present the initial parameterization and Figure 14(c)(d) show the final parameterization using the variational harmonic spline method.


(a) initial coons parameterization


(b) initial iso-parametric net


(c) final variational harmonic parameterization

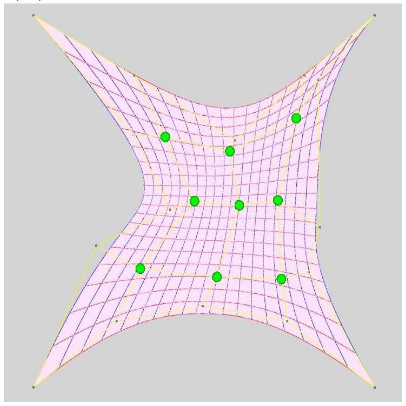

(d) final iso-parametric net

Figure 14: Examples of variational harmonic parameterization.

8 Simplex spline tools for isogeometric analysis

DMS-splines, introduced by Dahmen, Micchelli and Seidel in [1], are based on the simplex splines.

A simplex spline $M(\boldsymbol{x}|\boldsymbol{x}_0,\ldots,\boldsymbol{x}_{n+3})$ of degree n is a function of $\boldsymbol{x}\in\mathbb{R}^2$ over the half open convex hull of a point set $\boldsymbol{V}=[\boldsymbol{x}_0,\ldots,\boldsymbol{x}_{n+2})$, depending on the $n+3$ knots $\boldsymbol{x}_i\in\mathbb{R}^2, i=0,1,\ldots,n+2$. The basis function of simplex splines can be defined recursively as follows. When $n=0$,

$$\begin{aligned}&M(\boldsymbol{x}|\boldsymbol{x}_0,\ldots,\boldsymbol{x}_{n+2})\\&=\begin{cases}\frac{1}{|\mathrm{Area}_{\mathbb{R}^2}(\boldsymbol{x}_0,\ldots,\boldsymbol{x}_2)|}, & \boldsymbol{x}\in[\boldsymbol{x}_0,\ldots,\boldsymbol{x}_2),\\ 0, & \text{otherwise.}\end{cases}\end{aligned}$$

When $n>0$, select three points $\boldsymbol{W}=[\boldsymbol{x}_{k_0},\boldsymbol{x}_{k_1},\boldsymbol{x}_{k_2}]$ from $\boldsymbol{V}$, such that $\boldsymbol{W}$ is affine independent, then

$$M(\boldsymbol{x}|\boldsymbol{x}_0,\ldots,\boldsymbol{x}_{n+2})=\sum_{j=0}^{2}\lambda_j(\boldsymbol{x}|\,\boldsymbol{W})M(\boldsymbol{x}|\mathbf{V}\backslash\{\boldsymbol{x}_{k_j}\}),$$

where $\sum\limits_{j=0}^{2}\lambda_j(\boldsymbol{x}|\,\boldsymbol{W})=1$ and $\sum\limits_{j=0}^{2}\lambda_j(\boldsymbol{x}|\,\boldsymbol{W})\boldsymbol{x}_{k_j}=\boldsymbol{x}$. In fact, $\lambda_j(\boldsymbol{x}|\,\boldsymbol{W})$ are the barycentric coordinates of $\boldsymbol{x}$ with respect to $\boldsymbol{W}$.

In the following, we will introduce the definition of DMS-spline surface: let Ω be an arbitrary proper triangulation of $\mathbb{R}^2$ or some bounded domain $D \subset \mathbb{R}^2$. The proper triangulation means that every pair of triangle domain are disjoint, or share exactly one vertex or one edge. Next, with every vertex $\boldsymbol{x}$ of Ω, we associate a cloud of knots $[\boldsymbol{t}_0, \boldsymbol{t}_1, \cdots, \boldsymbol{t}_n]$, where $\boldsymbol{t}_0 = \boldsymbol{t}$. For every triangle$\boldsymbol{I} = (\boldsymbol{p}, \boldsymbol{q}, \boldsymbol{r})$, we require

• all the triangles $[\boldsymbol{p}_i, \boldsymbol{q}_j, \boldsymbol{r}_k]$ with $i + j + k \leq n$ are non-degenerate.

• the set

$$Z = interior(\bigcap_{i+j+k\leq n} [\boldsymbol{p}_i, \boldsymbol{q}_j, \boldsymbol{r}_k])$$

satisfies

$$Z \neq \emptyset$$

• if $\boldsymbol{I}$ has a boundary edge, the knots associated to the boundary edge must lie outside of Ω.

Then DMS-spline basis function $N^{\boldsymbol{I}}_\beta(\boldsymbol{u})$ is defined by means of simplex spline $M(\boldsymbol{u}|\, \boldsymbol{V}^{\boldsymbol{I}}_\beta)$ as

$$N^{\boldsymbol{I}}_\beta(\boldsymbol{u}) = |d(\boldsymbol{p}_i, \boldsymbol{q}_j, \boldsymbol{r}_k)| M(\boldsymbol{u}|\, \boldsymbol{V}^{\boldsymbol{I}}_\beta).$$

where β is the 3-tuple(i, j, k, l),

$$\boldsymbol{V}^{\boldsymbol{I}}_\beta = [\boldsymbol{p}_0, \cdots, \boldsymbol{p}_i, \boldsymbol{q}_0, \cdots, \boldsymbol{q}_j, \boldsymbol{r}_0, \cdots, \boldsymbol{r}_k].$$

$d(\mathbf{p}_i, \mathbf{q}_j, \mathbf{r}_k)$ is the area of $(\boldsymbol{p}_i, \boldsymbol{q}_j, \boldsymbol{r}_k)$.

A degree n DMS-spline surface $\boldsymbol{S}(\mathbf{u})$ over Ω is then defined as

$$\boldsymbol{S}(\boldsymbol{u}) = \sum_{\boldsymbol{I}\in\Omega} \sum_{|\beta|} \boldsymbol{c}^I_\beta N^I_\beta(\boldsymbol{u}).$$

where $\boldsymbol{c}^{\boldsymbol{I}}_\beta \in \mathbb{R}^3$ are the control points.

DMS-spline surface has the following properties which are similar with the properties of B-spline surfaces.

- affine invariance
- convex hull property
- local support
- DMS-spline surfaces are C^{n-1} continuous if the knots are in general position, where n is the degree of DMS-spline.

The directional derivative of $\boldsymbol{S}(\boldsymbol{u})$ with respect to a vector $\mathbf{v}$ can be calculated as follows:

$$D_{\boldsymbol{v}}\boldsymbol{S}(\boldsymbol{u}) = n \sum_{\boldsymbol{I}\in\Omega} \sum_{|\beta|-1} \check{\boldsymbol{c}}^{\boldsymbol{I}}_\beta(\boldsymbol{v}) N^{\boldsymbol{I}}_\beta(\boldsymbol{u})$$

and

$$\check{\boldsymbol{c}}^{\boldsymbol{I}}_\beta(\boldsymbol{v}) = \sum_{j=0}^{2} \boldsymbol{c}^{\boldsymbol{I}}_{\beta+e^j} \lambda_j(\boldsymbol{v}|[(\boldsymbol{p}_i, \boldsymbol{q}_j, \boldsymbol{r}_k)])$$

Important functionality is listed below:

- evaluation algorithm for DMS-spline surface
- compute the derivatives of DMS-spline surface
- insert new knots into the knot clouds of the surface and adapt the surface description accordingly
- refinement operation for DMS-spline surface

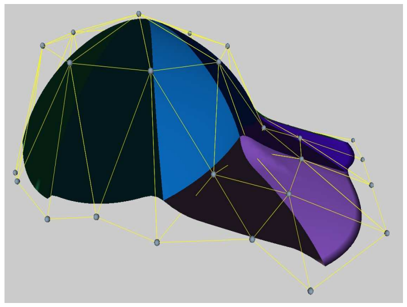
For above functionality, we have developed a plugin called "QSimplexSplinePlugin" in AXEL. For example, the class "QSimplexsplineSurface" is defined as follows,

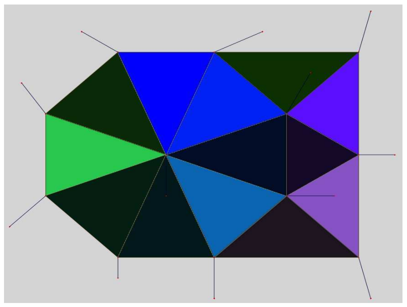
```
class QSimplexsplineSurface : public QCADSurface
{
  Q_OBJECT

 public:
  QSimplexsplineSurface(void) ;
  ~QSimplexsplineSurface(void) { /* freeSurf in glue */ }

  void draw(void) ;
  void drawWithNames(void) ;
  void draw_patch(int k, int res); //draw k-th patch with "res" samplings
  void draw_control(int k);//draw control mesh of k-th patch

 public slots:

  QPoint3f * eval(float u, float v) ;
  QPoint3f * evalFu(float u, float v) ;
  QPoint3f * evalFv(float u, float v) ;
  double3 BasisFunction(int index, int j, float u, float v);
  void compute(void) ;

 public :

  int degree ;      //!< Degree of simplex spline surface
  int number ;      //!< Number of control points
  int res;        //resolution

 protected:
  static int id_ ;
  bool dirty;

} ;
```

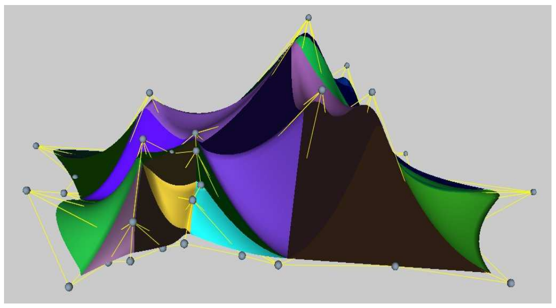
Figure 15 presents two examples of simplex spline surfaces and their parametric domain.

9 PHT spline tools for isogeometric analysis

A new type of piecewise bi-cubic polynomial splines called PHT spline was introduced over hierarchical T-meshes in [10]. It can be extended easily to high-degree cases. The


(a) simplex spline surface I


(b) simplex spline domain I


(c) simplex spline surface II

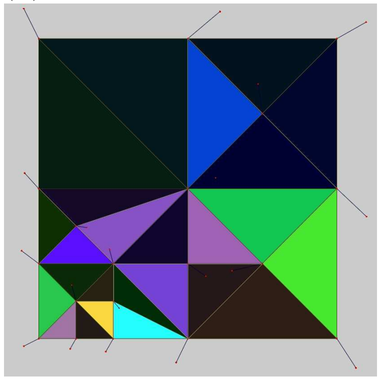

(d) simplex spline domain II

Figure 15: Examples of simplex spline surface and parametric domain.

basis functions of PHT-splines have many important properties as B-splines, such as non-negativity, local support and partition of unity. Compared with Sederberg's T-splines [11], they have two main advantages : first, the basis functions are polynomials rather than rational functions, which makes the computation efficient; second, they have nice local refinement operation by using cross insertion. In addition, there is a fast conversion between NURBS and PHT-splines, while the conversion between NURBS and T-splines is more complicated.

9.1 Hierarchical T-mesh

A T-mesh is basically a rectangular partition of a domain that allows T-junctions. The end points of each grid line in the T-mesh must be on two other grid lines, and each cell or facet in the grid must be a rectangle. A grid point in a T-mesh is also called a vertex of the T-mesh. If a vertex is on the boundary of the domain, then is called a *boundary vertex*. Otherwise, it is called an interior vertex. Interior vertices have two types. One is called *crossing vertex*, the other one is called *T-vertex* respectively. The line segment connecting two adjacent vertices on a grid line is called an edge of the T-mesh.

A hierarchical T-mesh is a special type of T-mesh with a level structure. It is defined in a recursive fashion. One generally starts from a tensor-product mesh (level 0). From level k to level $k+1$, one subdivide a cell at level k into four sub cells which are cells at level $k+$. For simplicity, we subdivide each cell by connecting the middle points of the opposite edges with two straight lines. Figure 16 gives an example of hierarchical T-mesh.

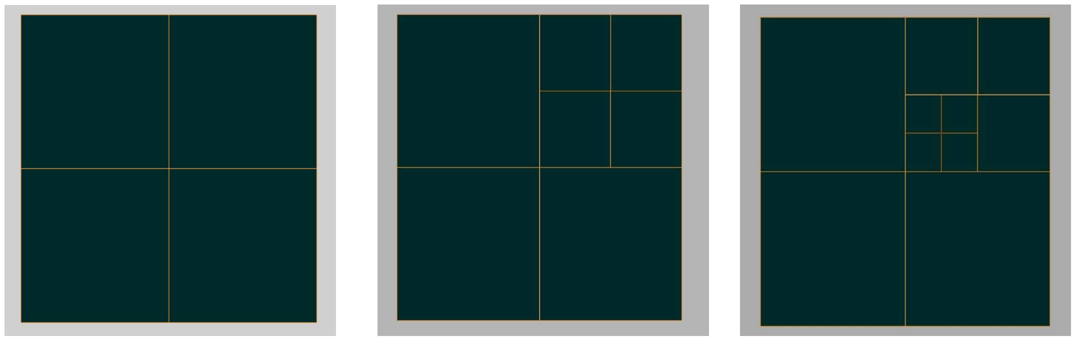

(a) first level (b) second level (c) third level

Figure 16: Hierarchical T-mesh.

9.2 PHT-spline space and basis function construction

Let $\Omega \in \mathbb{R}$ be a rectangular domain with boundary $\partial\Omega$ in R^2 . Denote by $\mathcal{T} = \cup K$ a hierarchical T-mesh over domain Ω, where Ks are cells of the mesh, then define

$$S(m, n, \alpha, \beta, \mathcal{T}) := \{s(x) \in C^{\alpha,\beta}(\Omega) | s(x) \in \mathbb{P}_{mn} \quad over \quad any \quad element \quad K \in \mathcal{T}\},$$

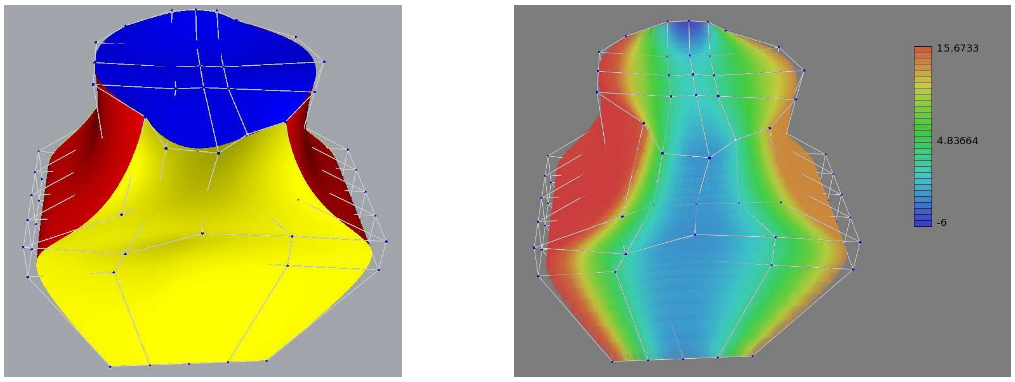
where $\mathbb{P}_{mn}$ is the space of all the bivariate polynomials with degree (m, n), and C denotes the space of all the bivariate functions which are continuous in with order (α, β). And $S(m, n, \alpha, \beta, \mathcal{T})$ is therefore a spline space over $\mathcal{T}$. In [17], a dimension formula for the spline space $S(m, n, \alpha, \beta, \mathcal{T})$, with $m \geq 2\alpha + 1$ and $n \geq 2\beta + 1$, is provided. Here, we set $m = n = 3$, $\alpha = \beta = 1$, so that

$$dim\mathbf{S}(3, 3, 1, 1, \mathcal{T}) = 4(V_b + V_+)$$

where V_b and V_+ represent the number of boundary and crossing vertices, respectively. From the dimension formula, we can see that each basis vertex (boundary vertex or crossing vertex) is associated with four basis functions, which can be constructed in a hierarchical manner.

For the initial level (level 0, denoted as $\mathcal{T}_0$), the standard bi-cubic tensor-product B-splines are used as basis functions. For simplicity, we set the initial mesh to be uniform rectangular gird. Suppose the grid is $[x_1, x_2, x_3, ..., x_s] \times [y_1, y_2, y_3, ..., y_t]$, and there are four basis functions to be defined on any vertex (x_i, y_j), since all the vertices are either crossing vertices or boundary vertices. Each basis function at (x_i, y_j) has support $[x_{i-1}, x_{i+1}] \times [y_{j-1}, y_{j+1}]$. These four basis functions are defined to be B-spline basis functions with knots $[x_{i-1}, x_{i-1}, x_i, x_i, x_{i+1}] \times [y_{j-1}, y_{j-1}, y_j, y_j, y_{j+1}]$, $[x_{i-1}, x_i, x_i, x_{i+1}, x_{i+1}] \times [y_{j-1}, y_{j-1}, y_j, y_j, y_{j+1}]$, $[x_{i-1}, x_i, x_i, x_{i+1}, x_{i+1}] \times [y_{j-1}, y_j, y_j, y_{j+1}, y_{j+1}]$, $[x_{i-1}, x_{i-1}, x_i, x_i, x_{i+1}] \times [y_{j-1}, y_j, y_j, y_{j+1}, y_{j+1}]$, respectively, such that their function values and derivatives vanish outside $[x_{i-1}, x_{i+1}] \times [y_{j-1}, y_{j+1}]$.

We now proceed to the mesh refinement of $\mathcal{T}_0$. We denote the T-mesh at level k as $\mathcal{T}_k$. Inductively, suppose that the basis functions b_j^k, $j = 1, ..., d_k$, on $\mathcal{T}_k$ have been constructed, the basis functions on $\mathcal{T}_{k+1}$ can then be constructed from two sources: some from the modifications of the old basis functions on $\mathcal{T}_k$, and others from the new basis functions associated with the new basis vertices of $\mathcal{T}_{k+1}$. Notice the fact that a basis function can be represented by specifying its 16 Bézier ordinates (coefficients) in every cell within the support of the basis function. When a cell is refined by adding a cross vertex, the cell



(a) boundary B-spline surface (b) solution field

Figure 17: 3D colormap.

is subdivided into four sub cells. Each sub cell supports the original basis function, and there are 16 Bézier ordinates on it. After this subdivision, the basis function remains the same, only that it is defined on the refined mesh. By adding a cross vertex, we get 5 new vertices, some of which are new basis vertices. Then we have to reset all the associated Bézier ordinates associated with the new basis vertices to zero.Other than the modification of old basis functions, there are some new basis vertices. These new basis vertices may be crossing vertices; or may be some T-vertices from the mesh of the previous level, which become basis vertices due to the addition of cross points to the neighbouring cells. For these kind of basis vertices, we simply build basis functions over their support cells as in the initial level. One may check [10] for more details on the construction of basis functions.

10 Visualization tools for isogeometric model and results

From the idea of isogeometry, in 2D isogeometric analysis, the solution field is represented by B-spline surface with 3D control points; similarly, for 3D isogeometric analysis , B-spline volume with k-D control points is used to represent the solution field, where k is the number of kind of unknown solution variables.

Visualization of isogeometric model and results is an important issue in the isogeometric toolbox. For isogeometric model, we can use AXEL as an interface for visualization, which has been described in WP-6-2. For the visualization of analysis results, the color-map and iso-value objects are used.

10.1 Color map tools

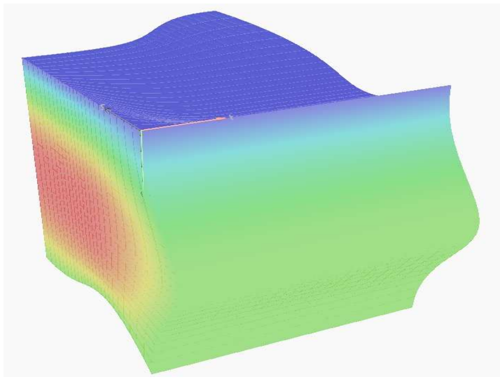
In order to visualize the analysis results in user-interface, a kind of unknown solution field of specified physical simulation problem, can be represented by color-map tools(also known as pseudo-coloring). This involves assigning a different color to each value in the functions range, so that color visually conveys numeric information. For points in the domain other than the input grid points, the function can be interpolated, and the resulting scalar can be used with the color map.

A C++ class named "QHSVcolormap" is added to convert the solution value into HSV or RGB color information. The following code shows how to construct and add the colormap into the object list.

```
// define surfaceColorMap by QBSpineSurface "solsurf"
QSurfaceColorMap  *colormap = new QSurfaceColorMap(solsurf);
// add QSurfaceColorMap into QViewer
ObjectManager::instance()->addObject(colormap) ;
```

For B-spline volume with 4D control points, we have a similar class:

```
class QBSplineVolume4f : public QParametricVolume
{
     Q_OBJECT

public:
     QBSplineVolume4f(void) ;
    ~QBSplineVolume4f(void) { /* freeSurf in glue */ }

    void draw(void) ;
    void drawWithNames(void) ;

public slots:
    QPoint4f * eval(float u, float v, float w) ;
    float umin(void) { return knots1[0]    ; }
    float umax(void) { return knots1[number1+order1-1] ; }
    float vmin(void) { return knots2[0]    ; }
    float vmax(void) { return knots2[number2+order2-1] ; }
    float wmin(void) { return knots3[0]    ; }
    float wmax(void) { return knots3[number3+order3-1] ; }

public slots :
    void compute(void) ;

    void push_point(QPoint4f * p) ;
    void push_point(QPoint4f * p, int position) ;

    void computeKnots1(void);
    void computeKnots2(void);
    void computeKnots3(void);
    void computeMinMaxValue(void); // compute min and max value

    QList<QPoint4f *> listSelectedControlPoints(void) ;

public :
    float  Valmin;          //min value which will be shown by color information
    float  Valmax;          //max value which will be shown by color information
    bool uniform ;
    int order1 ;     //!< Order of volume in first parameter direction
    int order2 ;            //!< Order of volume in second parameter direction
    int order3 ;            //!< Order of volume in third parameter direction
```


(a)3D solution field

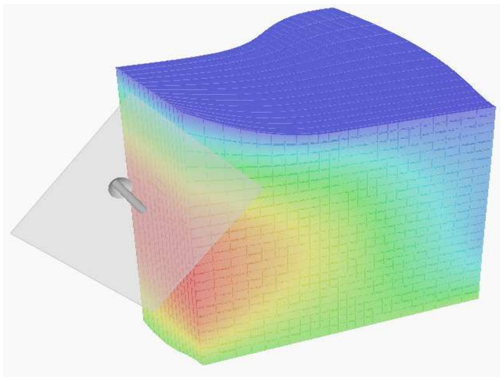
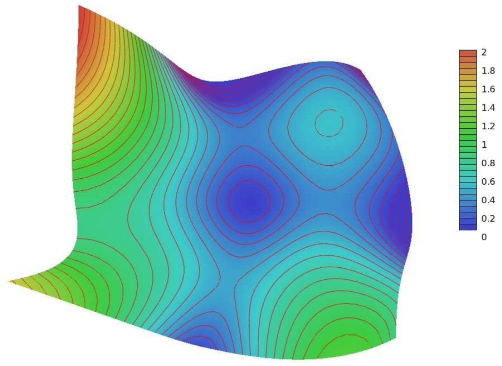
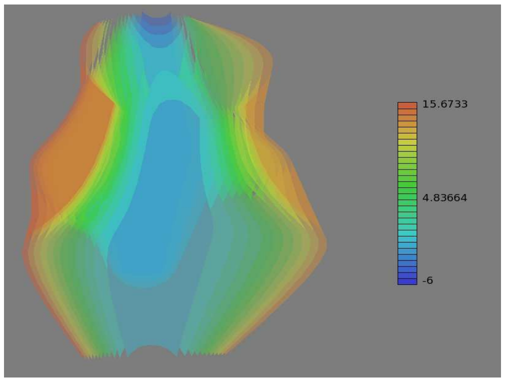

(b) interior view by clipping plane

Figure 18: Interior view by clipping plane.



(a)2D colormap and its iso-curves



(b) iso-surfaces of colormap in Fig.17(a)

Figure 19: Iso-value object.

```
    int number1;            //!< Number of coefficients in first parameter direction
    int number2;            //!< Number of coefficients in second parameter direction
    int number3;            //!< Number of coefficients in third parameter direction
    double * knots1 ;     //!< knot vector in first parameter direction
    double * knots2 ;       //!< knot vector in second parameter direction
    double * knots3 ;       //!< knot vector in third parameter direction
    QList<QPoint4f *> points ; //list of control points

protected:
    static int id_ ;
    bool dirty ;
} ;
```

Figure 17 presents a 3D example using color map to represent the solution field.

Clipping plane, which is an kind of OpenGL tools, can be used for the visualization of interior colormap in isogeometric solution volume. Figure 18 shows the clipping plane and the corresponding interior colormap.

10.2 Iso-value tools

Iso-value object has an implicit form to represent the solution part which has the same solution values. It is an important tools for field visualization. The isosurface in 3D case for a given value fTv is the set of all points in the domain for which the function has the value fTv. Under a fairly common set of conditions this set is a surface or set of surfaces. For 2D case, this concept reduces to an isocurve or isoline. The isocurve is the same thing as a contour line in topographical maps. The isosurface is the 3-d analog of an isocurve.

The classical marching square/cube method is employed for rendering iso-value object in our implementation. The following code shows the class of QIsovalueSurface.

```
class QIsovalueSurface : public QAxelObject
{
  Q_OBJECT
  public:
  QIsovalueSurface(QBSplineVolume4f * vol, int sample) ;
  ~QIsovalueSurface(void) { delete vertices;   delete colors; delete index;   }

  virtual void draw(void) ;
  virtual void drawWithNames(void) ;

 public:

  void marching_cube(void);
  void vMarchCubes(int cubeindex, float fTv); // compute iso-surface in "cubeindex"-th
                                                 cube for isovalue "fTv"
  QBSplineVolume4f * vol;
  int sample;    //sample for computation
  int isosample; // number of isosurface
  float vmax;
  float vmin;
  QList<QCubeIGA *> cubes ;
  float* vertices;
  float* colors;
  int * index;

 protected:

  static int id_ ;
  HSVColor hsv_colormap;//draw color map
  void getcolor(double c,  float* mc);
} ;
```

Figure 19 presents 2D and 3D iso-value objects for visualization of solution field.

11 Toolbox modules

The current toolbox deliverable contains the following packages:

axel : Interface and object definition for software AXEL

gotools : NURBS spline and geometry library

optimization : Isogeometric solver, optimizer and visualization tools

phtspline : PHT spline tools for isogeometric analysis

simplexspline : Simplex spline tools for isogeometric analysis

sisl : Complement for gotools

Axel is an algebraic-geometric modeler developed in GALAAD team in INRIA Sophia-antipolis. It provides a unified framework for both the visualization and manipulation of geometric objects with semi-algebraic representation, manipulating exact data to provide certified results or computing with approximations in a robust way. The application is designed to be either used stand alone or connected together with external tools and therefore become a graphical front end. Its modular architecture using plugins makes easy the wrapping of external libraries or the connection with other tools.

Much of the geometry related software are implemented as extensions to the background software SISL and GoTools from SINTEF. The current toolbox deliverable in gotools contains the functionality for:

- Spline space exploration
- Univariate, bivariate and trivariate geometry description, creation and operations upon these entities
- Single block and multiblock models

The toolbox modules related to SISL and GoTools are:

sisl SINTEF's spline library. It is partly replaced by GoTools, but act as a complement whenever appropriate.

gotools-core Parametric curves and surfaces including construction methods and operations on the entities. Trimmed surfaces are included in the module.

trivariate Spline volume including some creation methods

igeslib Read from and write to iges format. Conversion between sisl and GoTools data structures

compositemodel View a set of surfaces as one entity and perform operations on the model. Represents an isogeometric model in 2D during the construction phase and may represent it also in later phases.

trivariatemodel View a set of volumes as one entity and perform operations on the model. Represents an isogeometric model in 3D during the construction phase and may represent it also in later phases.

model_toolbox Some functionality to modify a set of surfaces. Most important is replacement of 4-sided trimmed surfaces by approximating non-trimmed spline surfaces. This module will be improved and modified during the remainder of the Exciting project, but the current version may still be useful.

viewlib Viewer for curves and surfaces

The following GoTools modules are to some extend used by the main modules, but are not seen as top level modules in an isogeometric context.

topology Structures for computing the topology of a surface model. The module is used from compositemodel in adjacency analysis.

parametrization Parametrization of scattered data points.

implicitization Approximating spline curves and surfaces by implicit geometry.

intersection Intersections involving spline curves and surfaces, and surface self intersection.

ttl Triangulations

The module isogeometric_model which is described in section 5 will also be added to the toolbox.

A set of example programs are included in the GoTools toolbox modules. They are placed in the examples folder of gotools-core and illustrate the use of selected functions. More example programs will be added later.

All modules are equipped with doxygen documentation. The main GoTools toolbox modules in isogeometric context do in addition have an overview documentation. The document is placed in the doc folder for the relevant modules.

12 Toolbox development

The first version of the toolbox contained basic geometric support tools, see [14]. This version was mostly adapted to single block models. In the new version, multiblock models are handled although the specific data structures for multiblock models for isogeometric analysis are postponed. These structures will be added to the toolbox within short time.

The geometry tools in the toolbox are also extended since the last deliverable as explained in the section 6 to 9. The visualization tools related to Axel are also extended.

The module model_toolbox was described in the report corresponding to D6-1 [15]. This module has tools to merge surface patches into larger spline surfaces and to improve tangent plane continuity between adjacent surfaces. The report of D6-1 gives some background for this functionality. The software in model_toolbox is in a prototype stage. It has been used in an isogeometric setting to approximate trimmed 4-sided surfaces by non-trimmed spline surfaces. The experience shows that the tool is really useful, but that some improvements is required to make it sufficiently flexible and stable. This will be a topic before the next toolbox delivery.

The PHT-splines described in section 9 is a tool for local refinement of B-spline surfaces. The work with local refinement will be continued.

The current toolbox contains a couple of solvers in the optimization module. These solvers will be raised to a more visible level and more solvers will be added. This is expected to give rise to a partly reorganization of the toolbox. The reorganization will not affect the geometry tools.

13 Application of the isogeometry toolbox for simulation problems

Milestone 6.1 [15] addresses the use of the isogeometry toolbox for representative test cases. Subsection 13.1 describes an isogeometric solver for heat conduction and how the toolbox is used in this context. This solver is also a part of the toolbox and currently placed in the optimization module.

Examples related to use of the toolbox in a structural solver are given in subsection 13.2 and in fluid structure interaction in subsection 13.3. Subsection 13.4 illustrates the use of the toolbox in least squares fitting.

13.1 Isogeometric solver for heat conduction

The first solver in toolbox focus on the heat conduction problem in (2). According to a classical variational approach, we seek for a solution $T \in H^1(\Omega)$, such as $T(\mathbf{x}) = T_0(\mathbf{x})$ on $\partial\Omega_D$ and:

$$\int_\Omega \boldsymbol{\nabla}(\kappa(\mathbf{x})\boldsymbol{\nabla}T(\mathbf{x}))\ \psi(\mathbf{x})\ d\Omega = \int_\Omega f(\mathbf{x})\ \psi(\mathbf{x})\ d\Omega \quad \forall \psi \in H^1_{\partial\Omega_D}(\Omega), \tag{10}$$

where $\psi(\mathbf{x})$ are test functions. After integrating by parts and using boundary conditions, we obtain:

$$-\int_\Omega \kappa(\mathbf{x})\boldsymbol{\nabla}T(\mathbf{x})\ \boldsymbol{\nabla}\psi(\mathbf{x})\ d\Omega + \int_{\partial\Omega_N} \Phi_0(\mathbf{x})\ \psi(\mathbf{x})\ d\Gamma = \int_\Omega f(\mathbf{x})\ \psi(\mathbf{x})\ d\Omega. \tag{11}$$

According to the IGA paradigm, the temperature field is represented using B-spline basis functions. For a 2D problem, we have:

$$\mathcal{T}(\xi,\eta) = \sum_{i=1}^{n_i}\sum_{j=1}^{n_j} \tilde{N}_i^{p_i}(\xi)\, \tilde{N}_j^{p_j}(\eta)\, T_{ij}, \tag{12}$$

where $\hat{N}_i$ functions are B-Spline basis functions and $\mathbf{u} = (\xi,\eta) \in \mathcal{P}$ are domain parameters. Then, we define the test functions $\psi(\mathbf{x})$ in the physical domain such as:

$$N_{ij}(\mathbf{x}) = N_{ij} \circ \sigma^{-1}(x,y) = \tilde{N}_{ij}(\xi,\eta) = \tilde{N}_i^{p_i}(\xi)\, \tilde{N}_j^{p_j}(\eta). \tag{13}$$

where σ is the planar B-spline mapping from parametric domain $\mathcal{P}$ to physical domain Ω as follows

$$\sigma(\mathbf{u}) = \sigma(\xi,\eta) = \sum_{i=1}^{n_i}\sum_{j=1}^{n_j} N_i^{p_i}(\xi)\, N_j^{p_j}(\eta)\, \mathbf{c}_{ij}, \tag{14}$$

The weak formulation Eq. 11 reads:

$$\sum_{k=1}^{n_k}\sum_{l=1}^{n_l} T_{kl} \int_\Omega \kappa(\mathbf{x})\boldsymbol{\nabla}N_{kl}(\mathbf{x})\ \boldsymbol{\nabla}N_{ij}(\mathbf{x})\ d\Omega = \int_{\partial\Omega_N} \Phi_0(\mathbf{x})\ N_{ij}(\mathbf{x})\ d\Gamma - \int_\Omega f(\mathbf{x})\ N_{ij}(\mathbf{x})\ d\Omega.$$

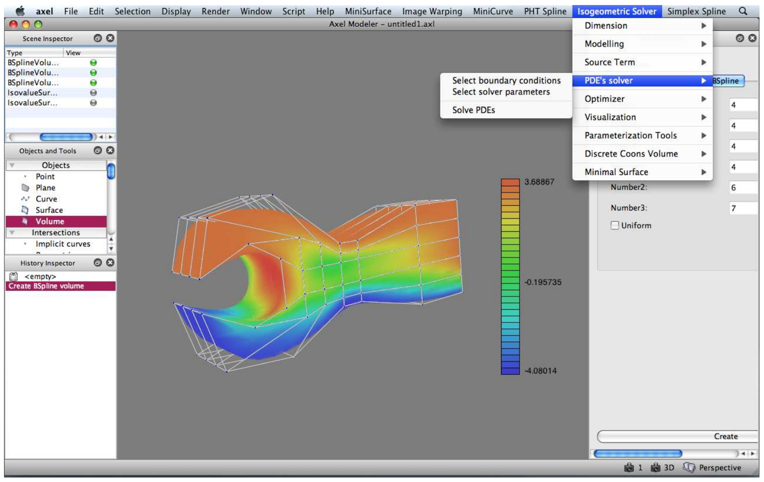


Figure 20: Interface for isogeometric solver in AXEL.

Finally, we obtain a linear system similar to that resulting from the classical finite-element methods, with a matrix and a right-hand side defined as:

$$\begin{aligned}
M_{ij,kl} &= \int_{\Omega} \kappa(\mathbf{x}) \boldsymbol{\nabla} N_{kl}(\mathbf{x}) \ \boldsymbol{\nabla} N_{ij}(\mathbf{x}) \ d\Omega \\
&= \int_{\mathcal{P}} \kappa(\mathcal{T}(\mathbf{u})) \boldsymbol{\nabla}_{\mathbf{u}} \tilde{N}_{kl}(\mathbf{u}) B(\mathbf{u})^T B(\mathbf{u}) \ \boldsymbol{\nabla}_{\mathbf{u}} \tilde{N}_{kl}(\mathbf{u}) J(\mathbf{u}) \ d\mathcal{P} \\
S_{ij} &= \int_{\partial\Omega_N} \Phi_0(\mathbf{x}) \ N_{ij}(\mathbf{x}) \ d\Gamma - \int_{\Omega} f(\mathbf{x}) \ N_{ij}(\mathbf{x}) \ d\Omega \\
&= \int_{\partial\mathcal{P}_N} \Phi_0(\mathcal{T}(\mathbf{u})) \ \tilde{N}_{kl}(\mathbf{u}) J(\mathbf{u}) \ d\tilde{\Gamma} - \int_{\mathcal{P}} f(\mathcal{T}(\mathbf{u})) \ \tilde{N}_{kl}(\mathbf{u}) J(\mathbf{u}) \ d\mathcal{P}.
\end{aligned} \tag{15}$$

where J is the Jacobian of the transformation, B^K is the transposed of the inverse of the Jacobian matrix. The above integrations are performed in the parameter space using classical Gauss quadrature rules.

Starting from a planar B-spline surface as computational domain, a general framework of an isogeometric solver for thermal conduction problem (2) has been implemented as plugins in the AXEL platform [2], yielding a B-spline surface as solution field. This plugin is written in C++ and QT, and has nice user-interface as shown in Figure 20. GoTools is used to evaluate the basis functions and their derivatives. Visualization of analysis results is done by the visualization tools in toolbox. Gauss-Seidel algorithm is employed to solve the linear system. In order to improve the simulation results, refinement operation can be performed for two parametric directions.

Additional details concerning the methods can be found in [3].

13.2 A structural solver

The application of work package 3 is car components and frames and in this context an isogeometric structural solver is being developed. Report [12] describes this solver at the stage of D3.2 and the problem formulation for isogeometric analysis related to linear elasticity. In this report a comparison between the isogeometric solver and an isoparametric FEM approach is made. The result is favorable for isogeometric analysis.

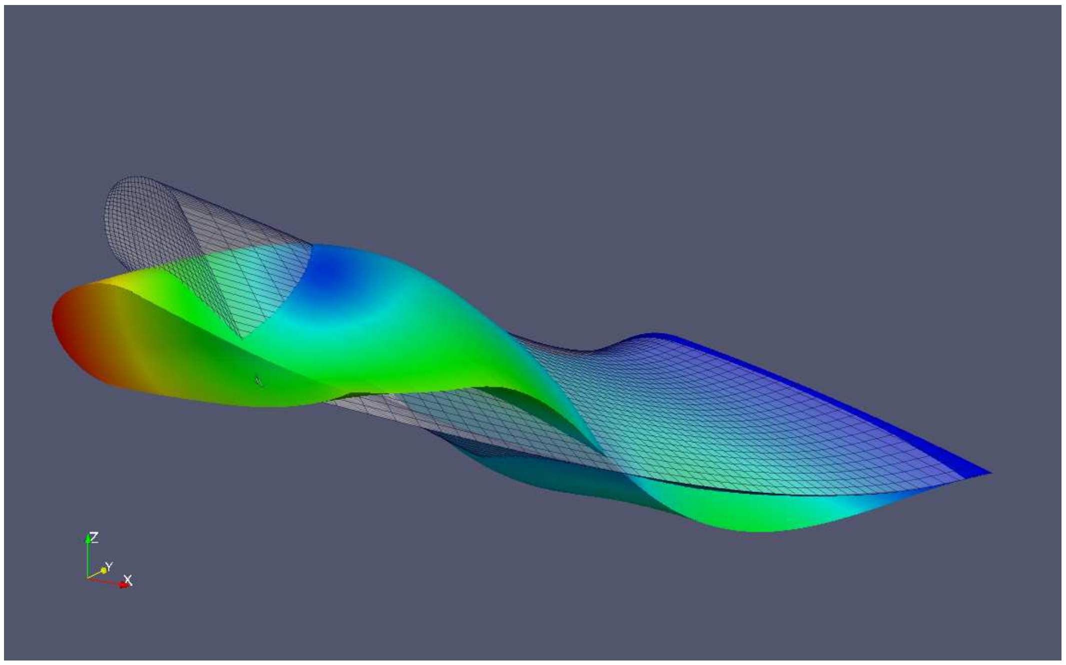


Figure 21: Eigenvalue computations of a propeller blade

The solver is using the isogeometric toolbox for functionality related to geometry and spline space exploration.

Figure 21 illustrates eigenvalue analysis for a propeller blade. Figure 22 shows von Mises stress computation on a bent half cylinder. Both test cases are computed with the structural solver.

13.3 A finite volume method applied on NURBS geometry for FSI problems

In connection with WP4, a method for applying Finite Volume Method on NURBS geometry has been developed. The idea of this approach is to solve incompressible Navier-Stokes equations on exact geometry for use in Fluid-Structure Interaction.

The idea in the modified Finite Volume Method is to transform the Navier-Stokes equation to the parametric space. The discretization of the domain being performed in FVM is done in parameter space. A number of geometric quantities appear in the transformed formulation. These are exactly computed using the isogeometric toolbox.

Figure 23 shows the mesh of a 2D example structure and fluid. The initial shape is a bent pipe with circular behaviour. This example is fetched from the presentation given by Christoph Heinrich in 7^{th} International Conference on Curves and Surfaces held in Avignon in June 2010. Figure 24 shows how a 3 refinements of the initial mesh still maintain the exact continuity between the fluid and the structure. The boundary between the fluid and the structure is updated in the initial space and again the fluid mesh is refined. During computations the number of degrees of freedom in the fluid mesh is more than 5 times as high as for the structure.

Figure 25 shows first the total velocity in the modified model, then the pressure distribution and finally the fluid and structure shown together. In Figure 26 the result of the

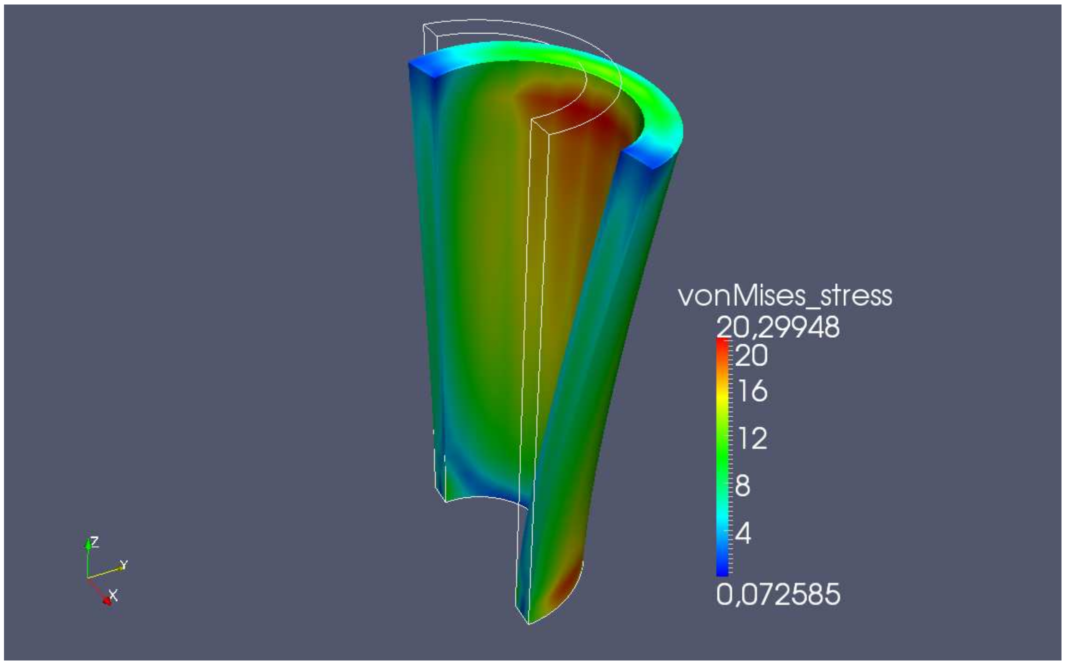


Figure 22: Von Mises stress computation

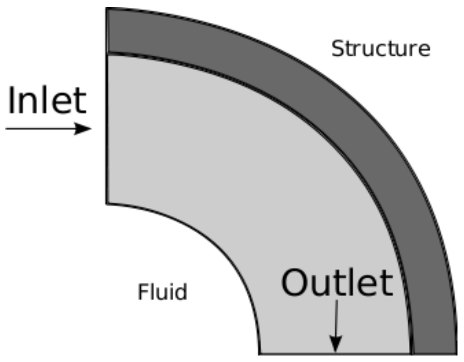


Figure 23: FSI example with a bent pipe

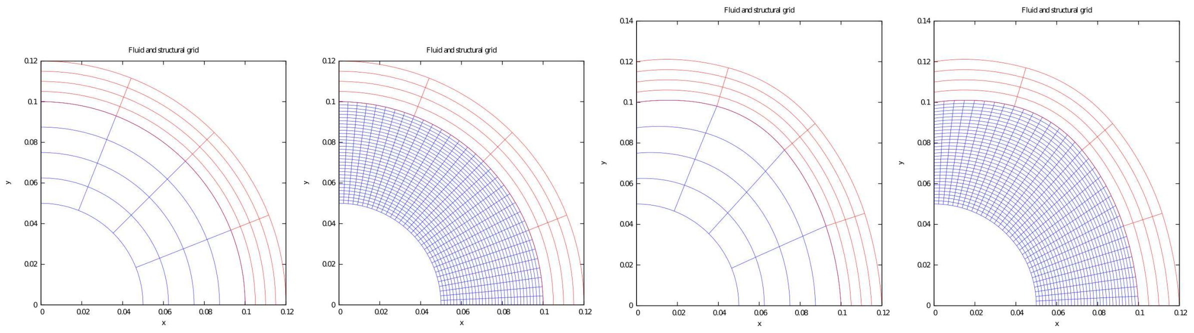


Figure 24: Fluid and structure grids

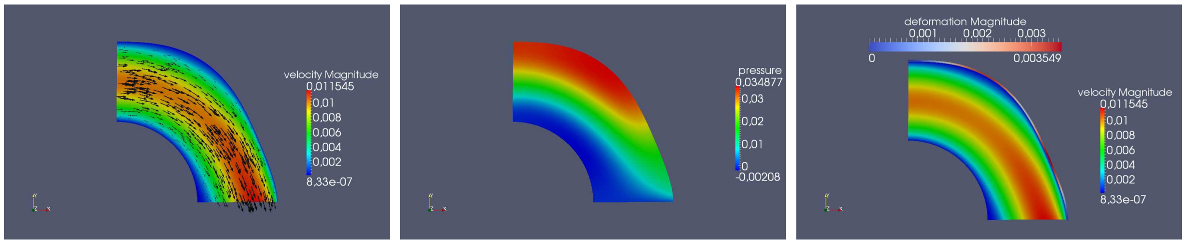


Figure 25: Results of the computations

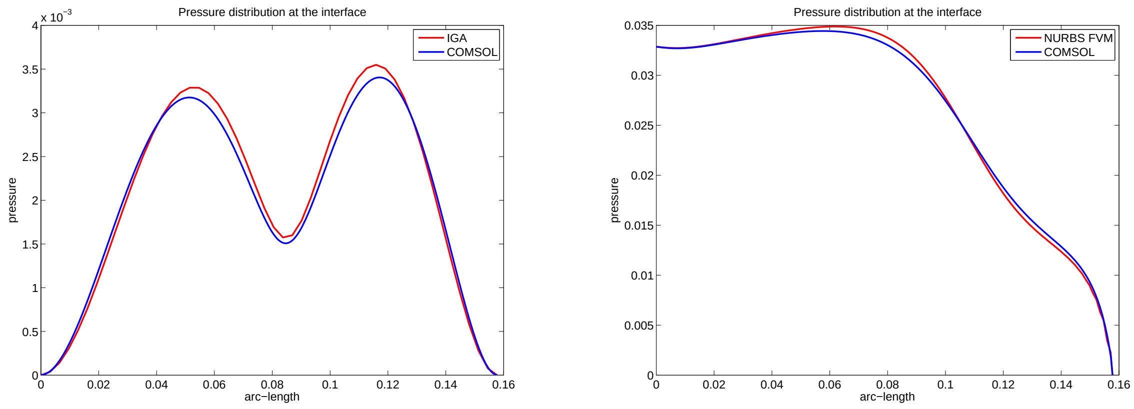


Figure 26: Comparison of result

modified Finite Volume Method on the fluid side and isogeometric analysis on the structural side are compared at the interface with the results achieved with the commercial Finite Element code COMSOL.

13.4 Fitting of blade surfaces

In WP1, the main use of the toolbox is in least squares approximation to a sets of data points by spline surfaces. Figure 27 shows the medial surface of a turbine blade together with the defining data points. Similarly, Figure 28 shows a turbine blade.

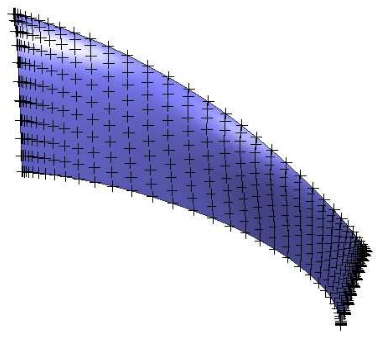


Figure 27: Medial surface fitting

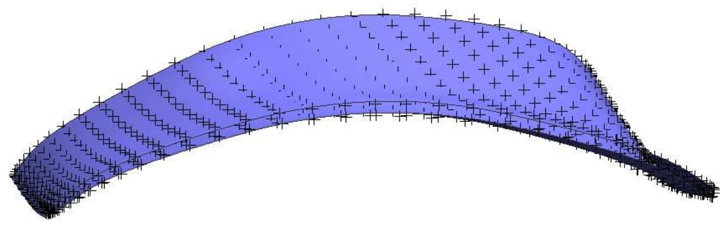


Figure 28: Blade surface fitting

References

[1] W. Dahmen, C.A Micchelli, H.P Seidel. Blossoming begets B-spline built better by B-patches. Mathematics of Computation, 1992, 59(199): 97-115.

[2] AXEL: http://axel.inria.fr/

[3] R. Duvigneau. An Introduction to Isogeometric Analysis with Application to Thermal Conduction. Rapport de recherche 6957, June 2009.

[4] H. Kestelman, Mappings with non-vanishing Jacobian, Amer. Math. Monthly 78 (1971), 662-663.

[5] G. Xu, G.Z Wang, X.D. Chen. Free form deformation with rational DMS-spline volumes. Journal of Computer Science and Technology, 2008, 23(5): 862-873

[6] G. Xu, B. Mourrain, Rgis Duvigneau, Andr Galligo. Optimal analysis-aware parameterization of computational domain in isogeometric analysis. Proc. of Geometric Modeling and Processing, LNCS 6130 (2010), 236-254

[7] G. Farin, Dianne Hansford. Discrete coons patches. Computer Aided Geometric Design, 16(7):691-700, 1999.

[8] J. E. Gain, N. A. Dodgson. Preventing self-Intersection under free-form deformation. IEEE Transactions on Visualization and Computer Graphics, 7(4), pp. 289-298, Oct.-Dec. 2001.

[9] B. Jüttler. Shape-preserving least-squares approximation by polynomial parametric spline curves. Computer Aided Geometric Design, 14: 731-747, 1997.

[10] J.S. Deng, F.L. Chen, X. Li, C.Q Hu, W.H Tong, Z.W Yang, and Y.Y Feng. Polynomial splines over hierarchical T-meshes. Graph. Models, 70(4):76-86, 2008.

[11] T. W. Sederberg, J. Zheng, A. Bakenov, and A. Nasri. T-splines and T-NURCCS. ACM Transactions on Graphics, 22(3):161-172, 2003.

[12] B. Simeon, A.-V. Vuong. EXCITING Report No. 3.2. Isogeometric structural prototype solver. September 2009.

[13] E. Mehlum, V. Skytt. Surface Editing. Numerical Methods and Software Tools in Industrial Mathematics. Dæhlen and Tveito (eds.), Birkhäuser, 1997, 381–394.

[14] V. Skytt, G. Xu, B. Morrain, R. Duvigneau. EXCITING Report No. 6.2. Isogeometric Toolbox. Basic Geometric Support Tools. September 2009.

[15] V. Skytt, G. Xu, B. Mourrain. EXCITING Report No. 6.1. First Definition of Isogeometric Toolbox, Cases, Data Structures and Methods. March 2009