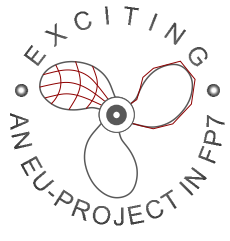
EC-Project EXCITING SCP8-2007-GA-218536

EXACT GEOMETRY SIMULATION
FOR OPTIMIZED DESIGN
OF VEHICLES AND VESSELS

http://exciting-project.eu



EXCITING Report No. 6.2

Isogeometric Toolbox. Basic Geometric Support Tools

Vibeke Skytt, SINTEF

Xu Gang, Bernard Mourrain, Régis Duvigneau, INRIA

Editor(s): Geir Skeie & Pål Bergan, DNV

September 2009

Dissemination level: PU

SINTEF

Contents

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 first software delivery of the toolbox. The delivery contains basic geometric support tools. The toolbox is based on the already existing software packages SISL, GoTools and Axel from the partners SINTEF and INRIA.

This report is divided into parts. The first part contains some considerations on the migration from a collection of CAD related geometry libraries to a toolbox for isogeometry. This topic is handled in section 2 to 6. The presentation is kept at a high level without too much detail. The intention is to define the isogeometric toolbox in the landscape of CAD, geometry modeling, isogeometric analysis and finite element analysis.

The second part is a documentation on the current version of the toolbox. It is mainly a single patch toolbox including curves, surfaces and volumes. The current toolbox has an emphasis on geometry. The documentation can be found in section 7 to 17. This part of the document is much more detailed than the first part as it is intended to, together with the doxygen documentation following the code, serve as a guide on how to use the current toolbox in implementation.

The third part from section 18 to 20 is related to the tools developed for the visualisation of shapes and results of simulation in the modeler Axel. We describe the specific plugins that have been developed for a first test case of an isogeometric solver and its integration in an optimisation loop and for the placement of inner control points for the parametrisation of computational domains.

In the last part of this document, see section 21, we identify the next toolbox extensions in a path towards a multi block isogeometric toolbox which includes also numerical analysis.

2 The Concept of an Isogeometric Toolbox

In isogeometric analysis, the spline space is used to describe both the geometry and the solution. One aspect of an isogeometric toolbox is the ability to a simple and effective exploration of the spline space. The most relevant functionality is

- Evaluation of the basis functions including a number of derivatives
- Knot insertion
- Degree elevation

Evaluation is used in the context of numerical integration. Gaussian quadrature is the most relevant integration method. The spline space is often evaluated in a regular grid of quadrature points. Thus, methods for efficient grid evaluation and compressed storage of the evaluation results are highly relevant.

The geometry enters the computations with the Jacobi determinant in the expression to be integrated. In this context, grid evaluation with respect to the geometry entities are of interest.

The spline space of the solution is either the same as the spline space of the corresponding geometry entity or it is a refined version of the geometry spline space. Data structures

that ensure consistence between the spline spaces and emphasize the correspondence can be very useful.

Geometry entities suitable for isogeometric analysis are not identical with CAD entities. We will discuss this topic in section 3. The isogeometric toolbox needs to contain tools to support creation of an analysis suitable geometry model and translation from a CAD model into an isogeometric representation of the same model.

Visualization is also an obvious ingredient in the toolbox, both with respect to geometry and the analysis results.

The isogeometric toolbox is not performing isogeometric analysis. It supports analysis. Thus, the aim of the toolbox is to provide tools that can be combined by an application. The tools should be complete, flexible and well organized. Furthermore, the tools should be of good quality. This gives an application a good starting ground for doing complex isogeometric analysis and make experiments with regard to the effect of different choices regarding analysis results and performance. The toolbox is currently lean, but will be extended with more tools throughout the project.

3 CAD versus Isogeometric Analysis

Isogeometric analysis uses the same function space to represent the mesh and the solution as CAD uses to represent free form geometry, namely the spline space. The spline space has many similarities with the traditional function space used in finite element methods, but there are also differences:

- Both spaces have got compactly supported basis functions which implies that the computational matrices will become sparse and banded.
- In both cases all basis functions in a point sum up to unity
- The standard finite element basis functions are polynomials, the basis functions of the spline space are piecewise polynomials or NURBS
- Both function spaces allow for refinement and degree raising
- The basis functions in the spline space is always positive, this is not the case for the finite element basis functions
- The partition of unity and the positivity of the basis function lead to the convex hull property for the spline space
- The function is represented as $f(\mathbf{x}) = \sum_i^n a^i \phi_i(\mathbf{x})$ where the basis functions ϕ_i, $i = 1, \ldots, n$ is multiplied with coefficients in the isogeometric case and nodal points in the finite element case. The nodal points interpolates the function. The coefficients do not.
- The spline space is variation diminishing. In contrast to the finite element space, it tends to reduce oscillations.
- The spline space has higher order continuity between element, the finite element space often has C^0 continuity. Elements of higher polynomial degree can give C^1 continuity, but the continuity is not as easily achieved as in the spline case.

The properties that distinguish the spline space from the finite element space implies that higher polynomial degrees can be used in the computation than what is normally used for traditional elements. Thus, using the isoparametric approach, the geometry model can be described with much higher precision in the isogeometric case than using traditional finite element method. Moreover, the function space is the same as the function space used in geometrical modelling. This implies that the possibility for doing computations on the exact geometric model is highly increased. Still, isogeometric analysis is not able to work with standard CAD model. Dokken et. al. has looked at the gap between CAD and numerical analysis, even for isogeometry, in [4]. We will look further at the different geometrical models in the remainder of this section.

3.1 Representing the Geometric Model

CAD represents surfaces as elementary surfaces (plane, cylinder, cone, torus, and sphere) or NURBS. The elementary surface patches can be accurately converted to trimmed NURBS surfaces, while keeping the patch structure of the CAD model. However, trimmed surfaces will frequently have approximate trimming curves as the exact closed form of the trimming curve can not be represented by NURBS. A NURBS represented CAD model will in general have trimmed surfaces, with many approximate trimming curves and gaps between adjacent surfaces patches.

In Finite Element Analysis the shape is represented by structures of Finite Elements, where each element is a trivariate (volumetric) parametric polynomial most typical of degrees two or lower, but higher degree elements are also used. Compared to CAD representation, the outer surface quality in FEA is (in most cases) of lower shape quality, but the models are geometrically watertight.

An isogeometric NURBS based analysis model is composed from regular blocks where each block is described by a tri-variate NURBS represented volume. Consequently, the construction of an isogeometric NURBS based model has much in common with constructing a block-structured grid. The main difference is that the inside of the block is described by one tri-variate NURBS volume rather than a regular grid of elements. In general the NURBS volume will be of higher degree than traditional finite elements and better adapted to the real object geometry.

An isogeometric model using spline geometry has a potential to be closer to the design intent or an initial CAD model with regard to shape than a traditional finite element mesh normally is. Refinement of the function spaces does not alter the geometrical shape of an object. No additional approximation step in the geometry representation is required when applying adaptivity in numerical analysis. This may be convenient. Furthermore, when geometry optimization is an aim of the computations, geometry update is more easily achieved than in traditional FEA since there is a tighter link between the geometry representation and the computational mesh. However,an isogeometric model is normally not identical with a CAD model.

A volume in a CAD model is represented by its boundaries and the material surrounded by these boundaries is implicitly assumed to belong to the volume. FEA needs a more explicit representation of volumes. Thus, a volumetric entity of spline type must exist in an isogeometric toolbox. Further, a path leading from a boundary represented CAD model to an isogeometric volumetric representation would be beneficial. This volumetric description must necessarily be multi block as few models can be adequately described

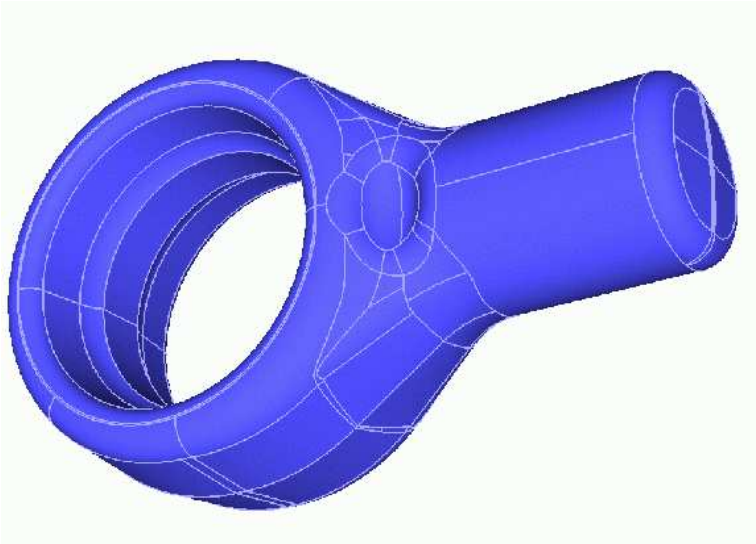


Figure 1: A boundary represented CAD model

by one box like or degenerate volume. The blocks, or volumes, should meet with exact continuity to satisfy the requirements from FEA. Moreover, the structure of the CAD model with trimmed surfaces and a division into surfaces that can depend just as much of the process of designing the model as the actual shape, is not necessarily appropriate as a guide to defining the block structured model. Figure 1 shows a CAD model where the division of the model, gives little information on the real structure of object. It is definitely not trivial to find this path.

Topological data structures and adjacency information must be extended to handle real volumetric models. There will be boundary surfaces that represent the common boundary between two volumes and boundary curves of these surfaces that are common for a number of surfaces. CAD type topology does normally not account for more than two surfaces meeting in an edge. Numerical analysis of thin plate constructions can be performed using a volumetric model as in [7] or by applying shell analysis. A shell representation of a solid model may, as a volume model, be non-manifold, i.e. more than two surfaces meet along a common edge. Experiments in [7] and[8] indicate that shell analysis for thin plate constructions is less required than in traditional finite element analysis. Ideally, an isogeometric toolbox should support it. That would also facilitate comparisements between a solid thin plate model and a shell version of the same model. However, shell analysis is not planned to be performed in Exciting, and it will thus not be prioritized in the toolbox. Still, new data structures should be planned also with this application in mind. Consequently, the topological data structures in the isogeometric toolbox must be very flexible and must be developed considerably compared to CAD topology.

A CAD model is intended to serve many purposes. It should support manufacturing, marketing, documentation, long time storage and a number of different numerical analysis computations. The different purposes require different level of detail. This is also the case for analysis with regard to different aspects of a model. Typically, much of the preprocessing before the analysis is concerned about creating an analysis suited model with appropriate level of detail. However, poor approximation properties and long computation times may force too much removal of detail. Tom Hughes has made experiments that

indicate that an isogeometric approach allows for more details in a model than traditional FEA. Still, it is not likely that one model can serve all purposes. A hierarchy of models with various levels of details where an update of one model automatically is distributed to the other models could be adequate. This is a concept that must be considered for the isogeometric toolbox. However, the analysis model and model hierarchy can not be created without the analysts skill. Again, the toolbox can only provide useful tools and data structures. The work has to be done by the application.

Seamless integration of CAD and analysis is the long term goal for isogeometric analysis. At this stage, the aim should be to collect building blocks and experience. We can work towards the final goal, and at the same time make the transition between CAD and a computational mesh, isogeometric or not, gradually more and more simple.

3.2 Isogeometric Modelling

SISL and GoTools are libraries that are related to CAD kernels. SISL support spline curves and non-trimmed spline surfaces while GoTools also allows trimmed surfaces and elementary curves and surfaces. SISL was initially implemented to be the spline part of a CAD system kernel.

A CAD kernel is different from an isogeometric toolbox although operations on the spline space, curves and surfaces are common. CAD probably requires a richer set of operations on the geometry entities than isogeometric analysis do. On the other hand, depending on the purpose of the CAD model, CAD can be very forgiving regarding the representation of a model as long as it visually looks good. An isogeometric model **must** be well represented.

CAD systems tend to hide the mathematical representation of their entities. Surface sizes are normally accessible, but the parameterization of these surfaces, represented by the spline spaces, are normally not. This makes the basis functions inaccessible for the user. Internally in the CAD system, however, this information is present. FEA is dependent on a good and easy access to the basis functions. The isogeometric toolbox, which is at the level of a CAD system core, provides this access.

The parameterization of a geometric entity and the knot vector is important for the quality of a curve or a surface. This is illustrated in section 5. Elaine Cohen et. al. [3] have studied the influence of the geometry quality on the isogeometric analysis performance and results. The knot vector is among the important factors. The knot vector and related spline space as they appear in SISL and GoTools are described in the sections 9.3 and 9.1. The influence from the parameterization on the analysis results, is not yet obvious. Thus, it is important to let the isogeometric toolbox facilitate various choices. Experiments with different geometry representations are important in the isogeometric analysis work at this stage. It should for instance be full freedom in how a knot vector is refined. Uniform refinement is not necessarily optimal.

Several modeling operations may be adequate for defining a geometry entity. Should a surface be created by loft or sweep, or maybe it is better to define all boundaries a priori and use a Coons patch type operation? Should an additional smoothing operation be applied to redistribute the inner surface coefficients? The different choices may lead to surfaces of fairly similar shape, but different representation. Some representations will be better suited to perform isogeometric analysis than others, but again the toolbox should offer full flexibility in geometry creation. This puts the responsibility for choosing the right

method on the user, but it also allows the user to experiment with the various methods and find the one that gives the best result.

Some operations, for instance loft, require a unified spline space for the defining curves or surfaces. Thus, a merging of spline spaces is invoked if the spline spaces are not initially identical. This may lead to unbalanced knot vectors or unintended matching of the input entities. The control over the parameterization is required at all stages of a geometry construction.

3.3 Multi Block Models

A complex geometry model can not be represented as one non-trimmed surface or volume. A multi block approach where each block is a surface or volume and where these blocks meet with exactly positional continuity, is able to represent more complex models.

FEA often finds it convenient to have corresponding nodes at sub block boundaries to ensure C^0 continuity between blocks, although hanging nodes can be handled with an increased effort. This can be translated to the concept of isogeometry by preferring correspondence of control points between adjacent surfaces or volumes at block boundaries. This wish can be fulfilled by propagating knot lines whenever surfaces or volumes with different spline spaces meet, and the isogeometric toolbox should provide the functionality to do so. However, such an approach has two drawbacks: the data size of the model can become very large and the unification of knot vectors can, as in the loft case, lead to unbalanced knot vectors and dense knots.

The concept of hanging nodes can be replaced by block boundaries where the surfaces or volumes meet with exact positional continuity, but where the spline spaces differ. Then there exist a refined spline space where the spline spaces of both adjacent entities are sub spaces. Thus, it is possible to define linear relations between coefficients on the boundaries of the two entities that ensure exact continuity. The situation where one adjacent spline space is a refinement of the other is also covered by this set up. An analysis application would get some tools to handle the isogeometric version of hanging nodes if the isogeometric toolbox has adequate tools to compute the relations between coefficients at block boundaries. Hanging nodes is not trivial in numerical analysis, but forcing a representation free of it is also not without drawbacks for complex models. The toolbox should, to some extent, support both options.

Figure 2 shows a planar part with many holes. An isogeometric multi block represented model gets unreasonably complex in these cases since each hole will be surrounded by 4 blocks. Alternative formulations like T-splines or other hierarchical representations may give a contribution in such cases. It is still an open question whether any alternative formulations will be covered by the toolbox.

4 Model Quality

A number of standards for CAD quality exist. One is the JAMA/JAPIA PDQ Guideline which relates to the automobile industry. In this standard, the severity of common CAD model problems with respect to different subsequent operations is evaluated. Quality problems that are found to have influence on numerical analysis are:

- Gaps between adjacent curves or surfaces lead to bad mesh quality

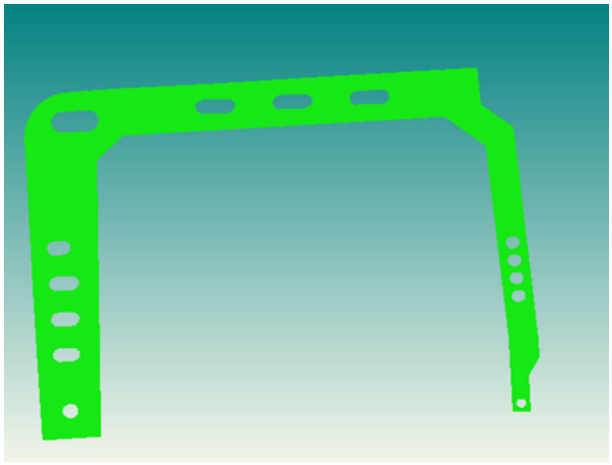


Figure 2: A simple model requiring a high number of blocks

- Tiny curves and surfaces and narrow surfaces also constitute a problem with regard to mesh quality and lead to high computation times. The same is the case with surfaces that are narrow compared to their neighbours
- Non-multiple knots that are hard to distinguish lead to similar problems
- Self intersecting curves and surfaces lead to bad mesh quality
- Embedded curves and surfaces lead to gaps between meshes
- Small curvature radius may lead to minute meshes, and consequently high computation time
- Degenerate surface boundaries may lead to unsatisfactory mesh quality and high computation time
- Degenerate surface boundaries can lead to similar problems as degenerate boundaries
- Folded surfaces may also give meshes of poor quality
- Gaps between an edge and the corresponding face or a vertex and the face give poor mesh quality

These guidelines are based on the assumption that the model is meshed based on the structure of the CAD model. Poor mesh quality with respect to a FEM mesh means large variation in mesh size, long, thin triangles and excessively skewed elements. In an isogeometric approach, the link between the quality of the geometric model and analysis is much stronger. Quality criteria for an isogeometric mesh is a topic of research, but some results have appeared.

Clearly a good mesh is one that performs well in numerical analysis, both with respect to the quality of the result and efficiency. Cohen et. al. studied in [3], the effect of different parameterization of a B-spline represented line segment in the context of computing longitudinal vibrations. They concluded that the distribution of the knot vector had an influence on the results, and a uniform knot vector was not necessarily the best choice.

Different choices of degeneracy while representing a disc was discussed in the same paper. This topic has also been a subject for other studies, and so far it seems that corner degeneracies is a better choice than letting an edge degenerate to a point in the center of the disc.

Hughes et. al. represented a hemisphere with a stiffener as one B-spline volume with multiple control points at sharp corners in [7]. In [8], the same model was represented with multiple volume patches to get better quality meshes.

In CAD, it is a preference to force discontinuities and degeneracies to surface boundaries to get an easily detectable localization. Moreover, a partly degenerate edge is not recommended. Such edges behaves badly in the context of for instance closest point iterations using a Newton type approach. They act as black holes, and the iteration would never leave this area after first entering it. The iteration problem occurs also at completely degenerate boundaries, but these cases are easier to detect and to handle specifically.

Highly non-uniform knot vectors where the size of the knot interval changes drastically from knot to knot behave badly in subsequent operations. It can lead to numerical instabilities especially in computing derivatives, and operations depending on that will be unstable and inefficient.

Mesh quality is different from CAD quality. Still it is not unreasonable to believe that spline spaces and surfaces that behave poorly in CAD operations would also be a poor choice for isogeometric analysis. The next section looks at how the parameterization of a curve or a surface affects the geometry of that entity.

5 The Aspect of Parameterization

Figure 3 illustrates the effect of the parameterization of interpolation points. The red points is to be implemented by a cubic spline curve. For the blue curve, the points are parameterized by chord length parameterization, the brown curve uses centripetal parameterization and the green one is uniformly parameterized. None of the curves are well behaved, and the bad results are due to the attempt of fitting a smooth curve to a discrete point set. Still, the different parameterizations give highly different results. Chord length parameterization gives a smooth curve, but has a tendency towards self intersection. The curve with centripetal parameterization oscillates a bit, but captures the vertical areas nicely. Uniform parameterization leads to large oscillations. This figure illustrates two important rules:

- The continuity of the geometry model and the solution space must reflect the continuity of the problem.
- Mind the parameterization. This becomes even more important for surfaces and volumes.

In figure 4 points sampled from a smooth surface are seen as a set of scattered points. In the second picture, the points are parameterized by projecting them onto a surface interpolating the boundaries of the original surface. The surface is generated by a least squares approximation with a smoothing term. The result is clearly not satisfactorily. In the last picture, a parameterization method that attempts to let the distances between points in the parameter domain match distances between points in geometry space, is applied, see [6].

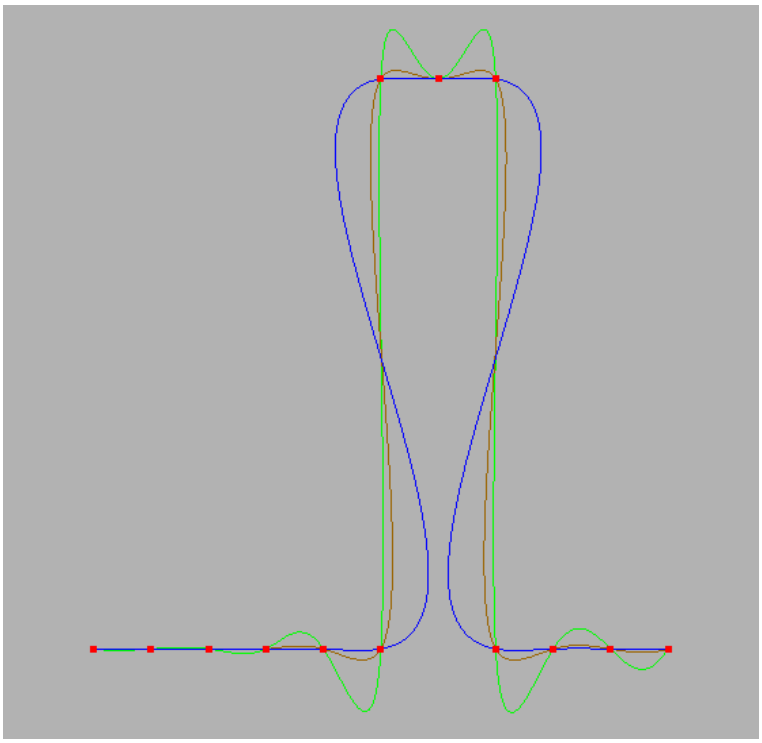


Figure 3: Interpolation using chord length, centripetal and uniform parameterization

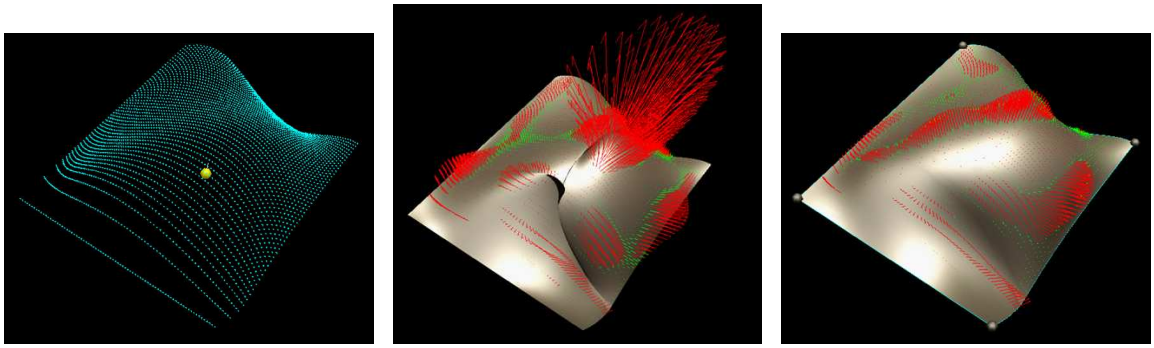


Figure 4: Surfaces approximating a scattered point set

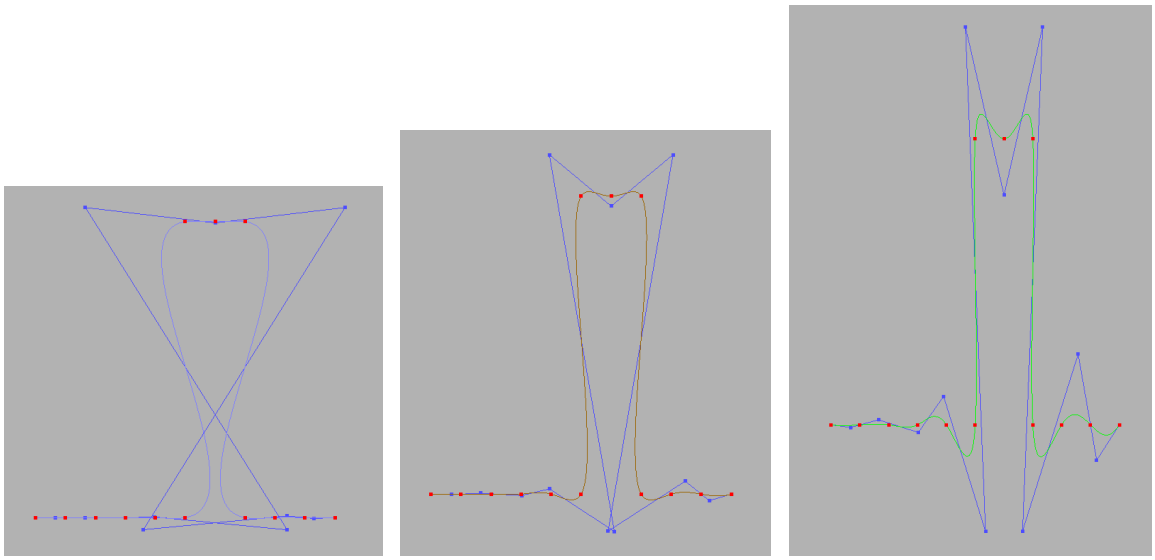


Figure 5: The control polygon of the interpolating curves

The surface is constructed as in the first case. The red and green arrows are exaggerated distances between the surfaces and the initial points. Again, the parameterization plays an important role regarding the quality of the geometry entity.

Let $f(\Omega) = \sum_i a^i \phi_i(\Omega)$ be the solution of a given differential equation. It maps from the geometry space to the solution space, i.e. $f : \Omega \to \mathbb{R}$. Thus, the geometry model in some sense parameterize the solution. Just as the parameterization and the knot vector influences the behavior of the curves and surfaces, the regularity of the geometry entity will influence the result of the isogeometric analysis.

Figure 5 shows the control polygon of the interpolating curves seen in figure 3 for the curves with chord length, centripetal and uniform parameterization, respectively. The control polygon corresponding to the uniformly parameterized curve has large oscillations while the other two self intersects. The distances between the control points are quite non-uniform. Consider figure 6. Here the spline space of the uniformly parameterized curve have been increased by adding knots where the knot vector was furthest from uniform, and where the curve has high curvature. Now, the control polygon lies much closer to the curve and has an improved behavior. Would such a simple adjustment of the geometry representation alter the result or performance of isogeometric analysis?

6 Splines for arbitrary topology local refinement

Complex geometry shape is often involved in CAD systems. Hence, it is important to develop isogeometric analysis system for geometry with arbitrary topology. Moreover, refinement steps are also needed in the analysis and construction of good approximations of the solution of PDE systems. The representation of computational domains by tensor product patches are not well-suited for local refinement, because the insertion of nodes on one parameter dimension of a patch introduces network of control points along the other parameter direction. T-splines solve partially this problem but have similar drawbacks for refinement along *diagonal directions*. We propose a new type of spline constructions, based

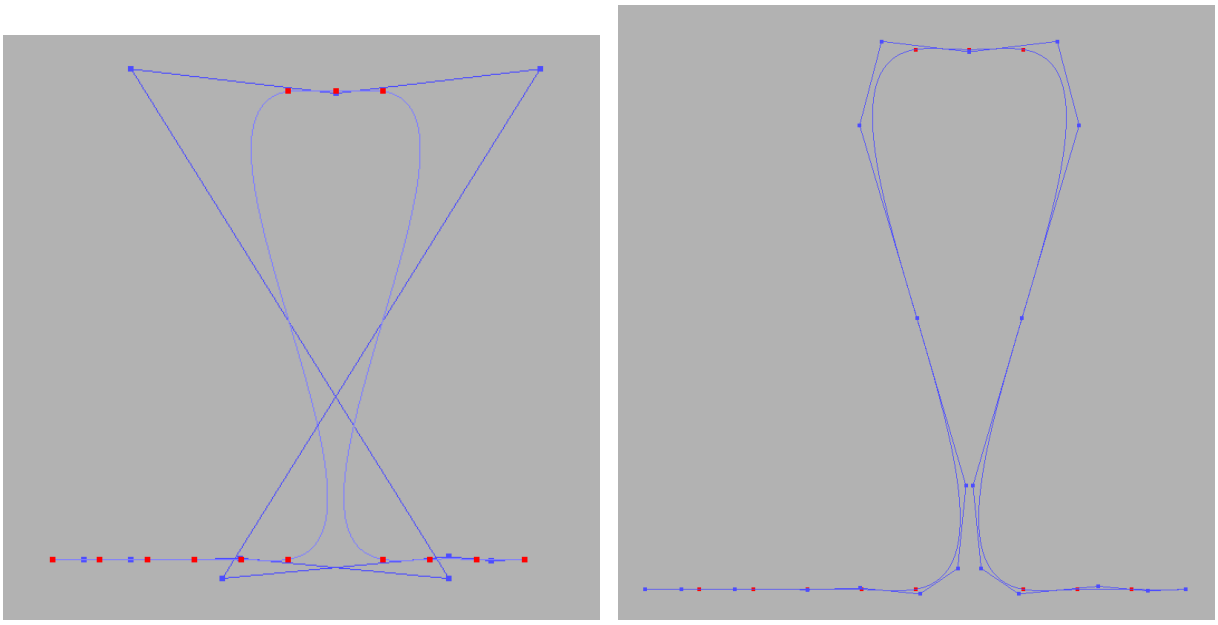


Figure 6: Refining the spline space corresponding to the chord length parameterized curve

on triangular primitives and well-suited for local refinement and arbitrary topology.

6.1 DMS-spline surfaces

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 $\boldsymbol{V}=[\boldsymbol{x}_0,\ldots,\boldsymbol{x}_{n+3})$, depending on the $n+4$ knots $\boldsymbol{x}_i\in\mathbb{R}^2, i=0,1,\ldots,n+3$. The basis function of simplex splines can be defined recursively as follows. When $n=0$,

$$M(\boldsymbol{x}|\boldsymbol{x}_0,\ldots,\boldsymbol{x}_{n+3}) = \begin{cases} \frac{1}{|\mathrm{Area}_{\mathbb{R}^2}(\boldsymbol{x}_0,\ldots,\boldsymbol{x}_2)|}, & \boldsymbol{x}\in[\boldsymbol{x}_0,\ldots,\boldsymbol{x}_2), \\ 0, & \text{otherwise.} \end{cases}$$

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

$$M(\boldsymbol{x}|\boldsymbol{x}_0,\ldots,\boldsymbol{x}_{n+3}) = \sum_{j=0}^{2}\lambda_j(\boldsymbol{x}|\,\boldsymbol{W})M(\boldsymbol{x}|\mathbf{V}\backslash\{\boldsymbol{x}_{k_j}\}),$$

where $\sum_{j=0}^{2}\lambda_j(\boldsymbol{x}|\,\boldsymbol{W})=1$ and $\sum_{j=0}^{2}\lambda_j(\boldsymbol{x}|\,\boldsymbol{W})\boldsymbol{x}_{k_j}=\boldsymbol{x}$. 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 nondegenerate.

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

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

where β is the 3-tuple(i, j, k, l),

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

$d(\mathbf{p}_i, \mathbf{q}_j, \mathbf{r}_k)$ 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

$$\boldsymbol{S}(\boldsymbol{u}) = \sum_{\boldsymbol{I}\in\Omega} \sum_{|\beta|} \boldsymbol{c}_\beta^I N_\beta^I(\boldsymbol{u}). \tag{-2}$$

where $\boldsymbol{c}_\beta^{\boldsymbol{I}} \in \mathbb{R}^3$ 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:

$$D_{\boldsymbol{v}}\boldsymbol{S}(\boldsymbol{u}) = n \sum_{\boldsymbol{I}\in\Omega} \sum_{|\beta|-1} \check{\boldsymbol{c}}_\beta^{\boldsymbol{I}}(\boldsymbol{v}) N_\beta^{\boldsymbol{I}}(\boldsymbol{u}) \tag{-2}$$

and

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

Important functionalities to be provided by the toolbox are 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

6.2 DMS-spline volumes

DMS-spline volume can be defined in a similar way with DMS-spline surfaces.

A trivariate simplex spline $M(\boldsymbol{x}|\boldsymbol{x}_0,\ldots,\boldsymbol{x}_{n+3})$ of degree n is a function of $\boldsymbol{x}\in\mathbb{R}^3$ over the half open convex hull of a point set $\boldsymbol{V}=[\boldsymbol{x}_0,\ldots,\boldsymbol{x}_{n+3})$, depending on the $n+4$ knots $\boldsymbol{x}_i\in\mathbb{R}^3, i=0,1,\ldots,n+3$. The basis function of trivariate simplex splines can be defined recursively as follows. When $n=0$,

$$M(\boldsymbol{x}|\boldsymbol{x}_0,\ldots,\boldsymbol{x}_{n+3}) = \begin{cases} \frac{1}{6|\mathrm{Vol}_{\mathbb{R}^3}(\boldsymbol{x}_0,\ldots,\boldsymbol{x}_3)|}, & \boldsymbol{x}\in[\boldsymbol{x}_0,\ldots,\boldsymbol{x}_3), \\ 0, & \text{otherwise.} \end{cases}$$

When $n>0$, select four points $\boldsymbol{W}=[\boldsymbol{x}_{k_0},\boldsymbol{x}_{k_1},\boldsymbol{x}_{k_2},\boldsymbol{x}_{k_3}]$ from $\boldsymbol{V}$, such that $\boldsymbol{W}$ is affine independent, then

$$M(\boldsymbol{x}|\boldsymbol{x}_0,\ldots,\boldsymbol{x}_{n+3}) = \sum_{j=0}^{3}\lambda_j(\boldsymbol{x}|\,\boldsymbol{W})M(\boldsymbol{x}|\mathbf{V}\backslash\{\boldsymbol{x}_{k_j}\}),$$

where $\sum_{j=0}^{3}\lambda_j(\boldsymbol{x}|\,\boldsymbol{W})=1$ and $\sum_{j=0}^{3}\lambda_j(\boldsymbol{x}|\,\boldsymbol{W})\boldsymbol{x}_{k_j}=\boldsymbol{x}$. 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 volume: let Ω be an arbitrary proper tetrahedralization of $\mathbb{R}^3$ or some bounded domain $D\subset\mathbb{R}^3$. The proper tetrahedralization means that every pair of tetrahedral domain are disjoint, or share exactly one vertex, one edge, or one face. 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 tetrahedron $\boldsymbol{I}=(\boldsymbol{p},\boldsymbol{q},\boldsymbol{r},\boldsymbol{s})$, we require

- all the tetrahedron $[\boldsymbol{p}_i,\boldsymbol{q}_j,\boldsymbol{r}_k,\boldsymbol{s}_l]$ with $i+j+k+l\le n$ are nondegenerate.
- the set

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

satisfies

$$Z\neq\emptyset$$

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

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

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

where β is the 4-tuple(i,j,k,l),

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

$d(\mathbf{p}_i,\mathbf{q}_j,\mathbf{r}_k,\mathbf{s}_l)$ is six times of the volume of $(\boldsymbol{p}_i,\boldsymbol{q}_j,\boldsymbol{r}_k,\boldsymbol{s}_l)$.

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

$$\boldsymbol{S}(\boldsymbol{u}) = \sum_{\boldsymbol{I}\in\Omega}\sum_{|\beta|}\boldsymbol{c}_\beta^I N_\beta^I(\boldsymbol{u}). \tag{-4}$$

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

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

$$D_{\boldsymbol{v}}\boldsymbol{S}(\boldsymbol{u}) = n \sum_{\boldsymbol{I}\in\Omega} \sum_{|\beta|-1} \check{\boldsymbol{c}}_{\beta}^{\boldsymbol{I}}(\boldsymbol{v}) N_{\beta}^{\boldsymbol{I}}(\boldsymbol{u}) \tag{-4}$$

and

$$\check{\boldsymbol{c}}_{\beta}^{\boldsymbol{I}}(\boldsymbol{v}) = \sum_{j=0}^{3} \boldsymbol{c}_{\beta+e^j}^{\boldsymbol{I}} \lambda_j(\boldsymbol{v}|[(\boldsymbol{p}_i, \boldsymbol{q}_j, \boldsymbol{r}_k, \boldsymbol{s}_l)])$$

Important functionalities to be provided by the toolbox are listed below:

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

We have employed DMS-spline volumes as deformation tools for freeform deformation in [13]. The evaluation algorithm and computation of derivatives are also implemented. Next step is to introduce DMS-splines into isogeometric analysis for CAD models with arbitrary topology.

7 The Toolbox

In the previous sections, we have elaborated the concept of an isogeometric toolbox, and looked at the role it ideally should play. The current version of the toolbox contains basic geometric support tools. Topological data structures will be added in the next version. The migration from CAD to isogeometry is still in its early stages. The remainder of this report is devoted to documenting the current version of the toolbox.

Currently, the isogeometric toolbox consists of SISL and the GoTools libraries; gotools-core, trivariate, igeslib and viewlib. The code originates from earlier work and parallel isogeometric projects. Exciting funding has been used for restructuring and documenting the toolbox libraries.

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 these entities. Trimmed surfaces are also included in the module. It does also contain conversion between SISL and GoTools data structures.

trivariate Spline volume including some creation methods

igeslib Read from and write to iges format.

viewlib Viewer for curves and surfaces. The viewer uses openGl and Qt. Qt is developed by Trolltech.

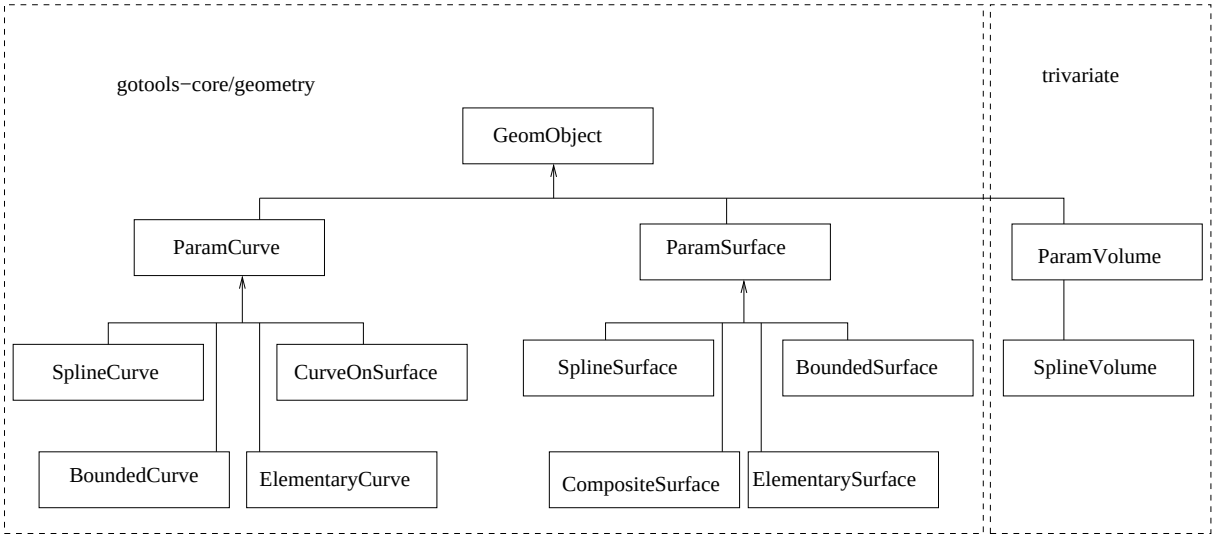


Figure 7: Simplified overview of the geometry class hierarchy

All GoTools modules are equipped with doxygen documentation. The level of detail in the documentation may vary. Anyway, the user can generate documentation of all classes and methods. The doxygen documentation is very close to the implementation. More basic information will be presented in the remaining parts of this document.

8 Data Structures

SISL is a library written in c, and has a very lean data structure. The main entity is SISLObject which can by of type SISLPoint, SISLCurve or SISLSurface. SISLCurve is a spline curve and SISLSurface is a spline surface. Spline entities are described in some detail in section 9 and 10.

Figure 7 shows the main geometric classes in GoTools and how they are divided between the modules. The current version of the isogeometric toolbox is to a large extent concerned with the entities in this figure, operations on and construction of these entities.

All geometric objects are of type GeomObject. This class has a function called instanceType, and by calling this function, it is possible to check the concrete type of a given object.

ParamCurve is the base class for all parametric curves in GoTools and defines a common interface. Many operations do not need to distinguish between different types of parametric curves. A parametric curve has functionality of type:

- operations on the parameter interval; fetch start and end parameter, change or reverse the parameter interval
- evaluation
- closest point
- compute the bounding box
- compute the directional cone surrounding all tangent directions of this curve

- length measures of the curve
- fetch a sub curve of this curve
- closest point computations
- append another curve to the current curve
- check for degeneracy
- make a copy of itself

More information can be found in the doxygen documentation corresponding to ParamCurve.

ParamSurface is the base class for a parametric surface and defines the common interface for all parametric surfaces. The functionality is roughly

- parameter domain operations; fetch the domain and the rectangular surrounding domain, check if a parameter pair lies in the domain of a surface, turn the directions of the parameter domain
- evaluation
- fetch constant parameter curves
- get the boundary curves surrounding this surface
- compute bounding box
- compute the directional cone surrounding all surface normal directions of the current surface
- area computations
- fetch a sub surface of this surface
- check for degeneracy
- make a copy of itself

ParamVolume defines the interface to all volumes. Currently only SplineVolume exists, but the hierarchy will be extended with a bounded volume and possibly also with some elementary volumes before the next release of the toolbox. The functionality so far is concerned with parameter space inquiries, evaluation and closest point computation.

9 B-spline Curves

A B-spline curve is represented by the SISL entity SISLCurve or SplineCurve in gotools-core/geometry.

The curve is defined by the formula

$$\mathbf{c}(t) = \sum_{i=1}^{n} \mathbf{p}_i B_{i,k,\mathbf{t}}(t).$$

The dimension of the curve $\mathbf{c}$ is equal to that of its *control points* $\mathbf{p}_i$. For example, if the dimension of the control points is one, the curve is a function, if the dimension is two, the curve is planar, and if the dimension is three, the curve is spatial. Usually the dimension of the curve will be at most three, but both SISL and GoTools allow for higher dimensions.

A B-spline curve is a linear combination of a sequence of B-splines $B_{i,k,\mathbf{t}}$ (called a B-basis) uniquely determined by a knot vector $\mathbf{t}$ and the order k. Order is equivalent to polynomial degree plus one. For example, if the order is two, the degree is one and the B-splines and the curve c they generate are (piecewise) linear. If the order is three, the degree is two and the B-splines and the curve are quadratic. Cubic B-splines and curves have order 4 and degree 3, etc.

The parameter range of a B-spline curve $\mathbf{c}$ is the interval

$$[t_k, t_{n+1}],$$

and so mathematically, the curve is a mapping $\mathbf{c} : [t_k, t_{n+1}] \rightarrow \mathbf{R}^{dim}$, where dim is the Euclidean space dimension of its control points.

The complete representation of a B-spline curve consists of

dim : The dimension of the underlying Euclidean space, $1, 2, 3, \ldots$.

n : The number of vertices (also the number of B-splines)

k : The order of the B-splines.

$\mathbf{t}$: The knot vector of the B-splines. $\mathbf{t} = (t_1, t_2, \ldots, t_{n+k})$.

$\mathbf{p}$: The control points of the B-spline curve. $p_{d,i}$, $d = 1, \ldots, dim$, $i = 1, \ldots, n$. e.g. when $dim = 3$, we have $\mathbf{p} = (x_1, y_1, z_1, x_2, y_2, z_2, \ldots, x_n, y_n, z_n)$.

The data in the representation must satisfy certain conditions:

- The knot vector must be non-decreasing: $t_i \leq t_{i+1}$. Moreover, two knots t_i and t_{i+k} must be distinct: $t_i < t_{i+k}$.
- The number of vertices should be greater than or equal to the order of the curve: $n \geq k$.

To understand the representation better, we will look at three parts of the representation: the B-splines (the basis functions), the knot vector and the control polygon.

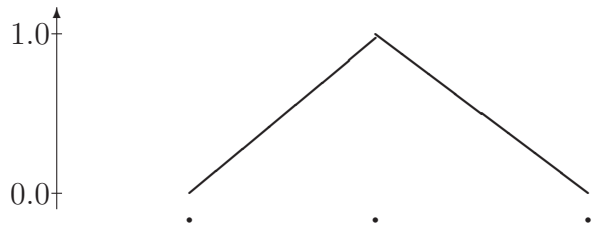


Figure 8: A linear B-spline (order 2) defined by three knots.

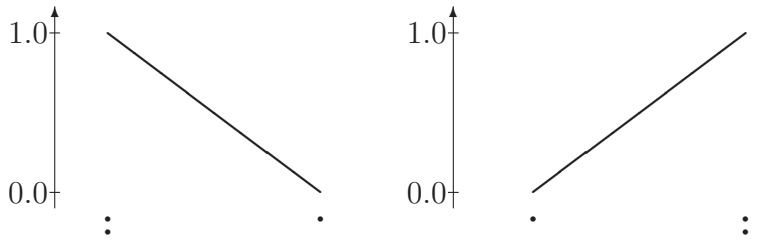


Figure 9: Linear B-splines of with multiple knots at one end.

9.1 B-spline Basis Functions

In SISL, the spline space is represented by data in SISLCurve. In GoTools, the spline space is represented by the class BsplineBasis. A set of B-spline basis functions is determined by the order k and the knots. For example, to define a single basis function of degree one, we need three knots. In figure 8 the three knots are marked as dots. Knots can also be equal as shown in figure 9. By taking a linear combination of the three basis functions shown in figures 8 and 9 we can generate a linear spline function as shown in figure 10.

A quadratic B-spline basis function is a linear combination of two linear basis functions. Shown in figure 11 is a quadratic B-spline defined by four knots. A quadratic B-spline is the sum of two products, the first product between the linear B-spline on the left and a corresponding line from 0 to 1, the second product between the linear B-spline on the right and a corresponding line from 1 to 0; see figure 11. For higher degree B-splines there is a similar definition. A B-spline of order k is the sum of two B-splines of order $k-1$, each weighted with weights in the interval [0,1]. In fact we define B-splines of order 1 explicitly

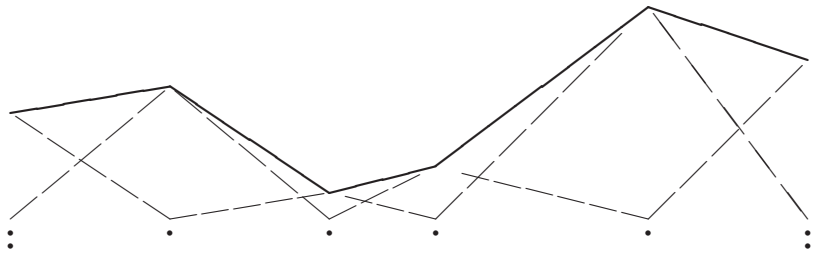


Figure 10: A B-spline curve of dimension 1 as a linear combination of a sequence of B-splines. Each B-spline (dashed) is scaled by a coefficient.

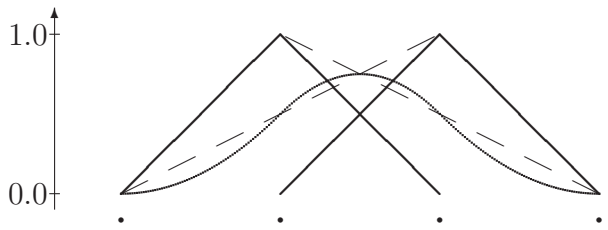


Figure 11: A quadratic B-spline, the two linear B-splines and the corresponding lines (dashed) in the quadratic B-spline definition.

as box functions,

$$B_{i,1}(t) = \begin{cases} 1 & if \quad t_i \leq t < t_{i+1}; \\ 0 & otherwise, \end{cases}$$

and then the complete definition of a k-th order B-spline is

$$B_{i,k}(t) = \frac{t - t_i}{t_{i+k-1} - t_i} B_{i,k-1}(t) + \frac{t_{i+k} - t}{t_{i+k} - t_{i+1}} B_{i-1,k-1}(t).$$

B-spline basis functions satisfy some important properties for curve and surface design. Each basis function is non-negative and it can be shown that they sum to one,

$$\sum_{i=1}^{n} B_{i,k,\mathbf{t}}(t) = 1.$$

These properties combined mean that B-spline curves satisfy the *convex hull property*: the curve lies in the convex hull of its control points. Furthermore, the support of the B-spline basis function $B_{i,k,\mathbf{t}}$ is the interval $[t_i, t_{i+k}]$ which means that B-spline curves has *local control*: moving one control point only alters the curve locally.

With k multiple knots at the ends of the knot vector, B-spline curves also have the *endpoint property*: the start point of the B-spline curve equals the first control point and the end point equals the last control point, in other words

$$\mathbf{c}(t_k) = \mathbf{p}_1 \qquad \text{and} \qquad \mathbf{c}(t_{n+1}) = \mathbf{p}_n.$$

9.2 The Control Polygon

The control points $\mathbf{p}_i$ define the vertices. The *control polygon* of a B-spline curve is the polygonal arc formed by its control points, $\mathbf{p}_0, \mathbf{p}_1, \ldots, \mathbf{p}_n$. This means that the control polygon, regarded as a parametric curve, is itself a piecewise linear B-spline curve (order two). If we increase the order, the distance between the control polygon and the curve increases (see figure 12). A higher order B-spline curve tends to smooth the control polygon and at the same time mimic its shape. For example, if the control polygon is convex, so is the B-spline curve.

Another property of the control polygon is that it will get closer to the curve if it is redefined by inserting knots into the curve and thereby increasing the number of vertices; see figure 13. If the refinement is infinite then the control polygon converges to the curve.

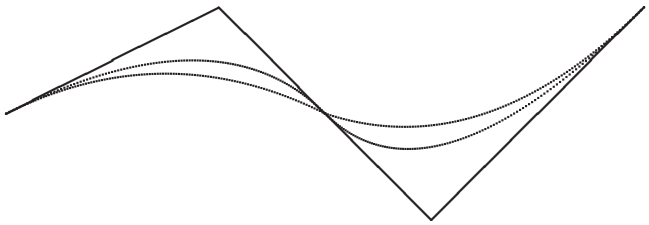


Figure 12: Linear, quadratic, and cubic B-spline curves sharing the same control polygon. The control polygon is equal to the linear B-spline curve. The curves are planar, i.e. the space dimension is two.

Figure 13: The cubic B-spline curve with a redefined knot vector.

9.3 The Knot Vector

The knots of a B-spline curve describe the following properties of the curve:

- The parameterization of the B-spline curve
- The continuity at the joins between the adjacent polynomial segments of the B-spline curve.

In figure 14 we have two curves with the same control polygon and order but with different parameterization.

This example is not meant as an encouragement to use parameterization for modelling, rather to make users aware of the effect of parameterization. Something close to curve length parameterization is in most cases preferable. For interpolation, chord-length parameterization is used in most cases.

The number of equal knots determines the degree of continuity. If k consecutive internal knots are equal, the curve is discontinuous. Similarly if $k-1$ consecutive internal knots are equal, the curve is continuous but not in general differentiable. A continuously differentiable curve with a discontinuity in the second derivative can be modelled using $k-2$ equal knots etc. (see figure 15). Normally, B-spline curves in SISL and GoTools are expected to be continuous. For many algorithms, curves should be continuously differentiable (C^1).

9.4 NURBS Curves

A NURBS (Non-Uniform Rational B-Spline) curve is a generalization of a B-spline curve,

$$\mathbf{c}(t) = \frac{\sum_{i=1}^{n} w_i \mathbf{p}_i B_{i,k,\mathbf{t}}(t)}{\sum_{i=1}^{n} w_i B_{i,k,\mathbf{t}}(t)}.$$

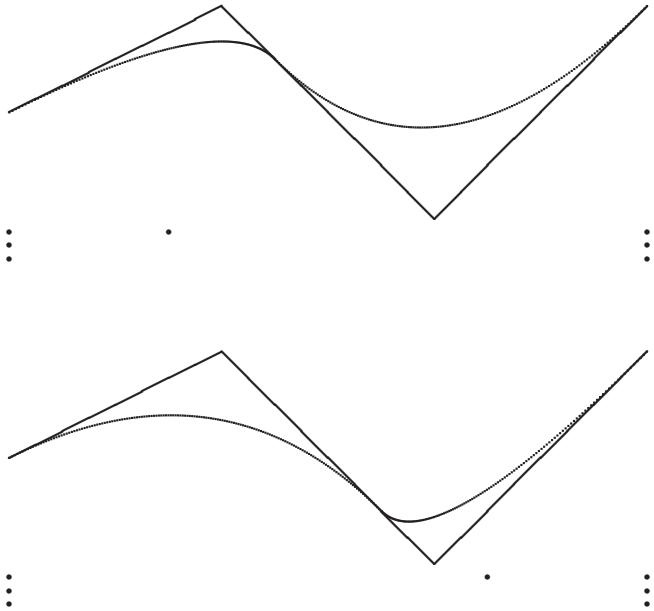


Figure 14: Two quadratic B-spline curves with the same control polygon but different knot vectors. The curves and the control polygons are two-dimensional.

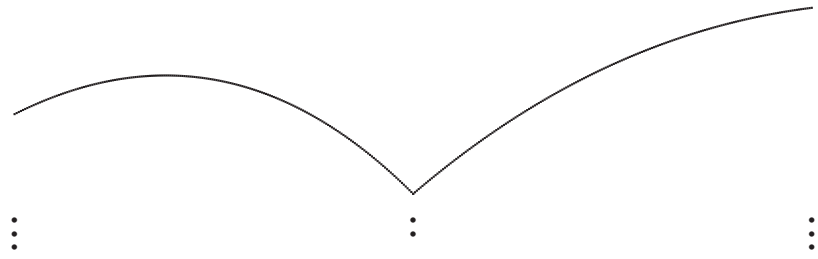


Figure 15: A quadratic B-spline curve with two equal internal knots.

In addition to the data of a B-spline curve, the NURBS curve $\mathbf{c}$ has a sequence of weights $w_1, \ldots, w_n$. One of the advantages of NURBS curves over B-spline curves is that they can be used to represent conic sections exactly (taking the order k to be three). A disadvantage is that NURBS curves depend nonlinearly on their weights, making some calculations, like the evaluation of derivatives, more complicated and less efficient than with B-spline curves.

The representation of a NURBS curve is the same as for a B-spline except that it also includes

$\mathbf{w}$: A sequence of weights $\mathbf{w} = (w_1, w_2, \ldots, w_n)$.

We make the assumption that the weights are strictly positive: $w_i > 0$.

Under this condition, a NURBS curve, like its B-spline cousin, enjoys the convex hull property. With k-fold knots at the ends of the knot vector, also NURBS curves have the endpoint interpolation property.

A NURBS curve in SISL and GoTools is stored in the same entities as polynomial curves. Note that the constructors of these entities assume that the NURBS coefficients are given in the format: $w_i p_{i,1}, \ldots w_i p_{i,dim}, w_i$ for $i = 1, \ldots n$, i.e. the coefficients are multiplied with the weights.

9.5 Spline Curve Functionality

In GoTools, SplineCurve inherits ParamCurve and has thereby the functionality specified for ParamCurve objects. Functionality specific for a B-spline curve can be found in the doxygen information. This functionality includes:

- Compute the derivative curve corresponding to a given curve
- Fetch information related to the spline space
- Fetch the control polygon of a spline curve
- Insert new knots into the knot vector of the curve and update the curve accordingly
- Increase the polynomial degree of the curve
- Make sure that the curve has got an open knot vector, i.e. knot multiplicity equal to the order in the ends of the curve

10 B-spline Surfaces

A tensor product B-spline surface is in SISL represented by SISLSurf and in GoTools by the class SplineSurface.

the B-spline surface is defined as

$$\mathbf{s}(u, v) = \sum_{i=1}^{n_1} \sum_{j=1}^{n_2} \mathbf{p}_{i,j} B_{i,k_1,\mathbf{u}}(u) B_{j,k_2,\mathbf{v}}(v)$$

with control points $\mathbf{p}_{i,j}$ and two variables (or parameters) u and v. The formula shows that a basis function of a B-spline surface is a product of two basis functions of B-spline curves (B-splines). This is why a B-spline surface is called a tensor-product surface. The following is a list of the components of the representation:

dim : The dimension of the underlying Euclidean space.

n_1 : The number of vertices with respect to the first parameter.

n_2 : The number of vertices with respect to the second parameter.

k_1 : The order of the B-splines in the first parameter.

k_2 : The order of the B-splines in the second parameter.

$\mathbf{u}$: The knot vector of the B-splines with respect to the first parameter, $\mathbf{u} = (u_1, u_2, \ldots, u_{n_1+k_1})$.

$\mathbf{v}$: The knot vector of the B-splines with respect to the second parameter, $\mathbf{v} = (v_1, v_2, \ldots, v_{n_2+k_2})$.

$\mathbf{p}$: The control points of the B-spline surface, $c_{d,i,j}$, $d = 1, \ldots, dim$, $i = 1, \ldots, n_1$, $j = 1, \ldots, n_2$. When $dim = 3$, we have $\mathbf{p} = (x_{1,1}, y_{1,1}, z_{1,1}, x_{2,1}, y_{2,1}, z_{2,1}, \ldots, x_{n_1,1}, y_{n_1,1}, z_{n_1,1}, \ldots, x_{n_1,n_2}, y_{n_1,n_2}, z_{n_1,n_2})$.

The data of the B-spline surface must fulfill the following requirements:

- Both knot vectors must be non-decreasing.
- The number of vertices must be greater than or equal to the order with respect to both parameters: $n_1 \geq k_1$ and $n_2 \geq k_2$.

The properties of the representation of a B-spline surface are similar to the properties of the representation of a B-spline curve. The control points $\mathbf{p}_{i,j}$ form a *control net* as shown in figure 16. The control net has similar properties to the control polygon of a B-spline curve, described in section 9.2. A B-spline surface has two knot vectors, one for each parameter.

10.1 The Basis Functions

A basis function of a B-spline surface is the product of two basis functions corresponding to B-spline curves,

$$B_{i,k_1,\mathbf{u}}(u) B_{j,k_2,\mathbf{v}}(v).$$

Its support is the rectangle $[u_i, u_{i+k_1}] \times [v_j, v_{j+k_2}]$. If the basis functions in both directions are of degree one and all knots have multiplicity one, then the surface basis functions are pyramid-shaped (see figure 17). For higher degrees, the surface basis functions are bell shaped.

10.2 NURBS Surfaces

A NURBS (Non-Uniform Rational B-Spline) surface is a generalization of a B-spline surface,

$$\mathbf{s}(u, v) = \frac{\sum_{i=1}^{n_1} \sum_{j=1}^{n_2} w_{i,j} \mathbf{p}_{i,j} B_{i,k_1,\mathbf{u}}(u) B_{j,k_2,\mathbf{v}}(v)}{\sum_{i=1}^{n_1} \sum_{j=1}^{n_2} w_{i,j} B_{i,k_1,\mathbf{u}}(u) B_{j,k_2,\mathbf{v}}(v)}.$$

In addition to the data of a B-spline surface, the NURBS surface has a weights $w_{i,j}$. NURBS surfaces can be used to exactly represent several common 'analytic' surfaces such

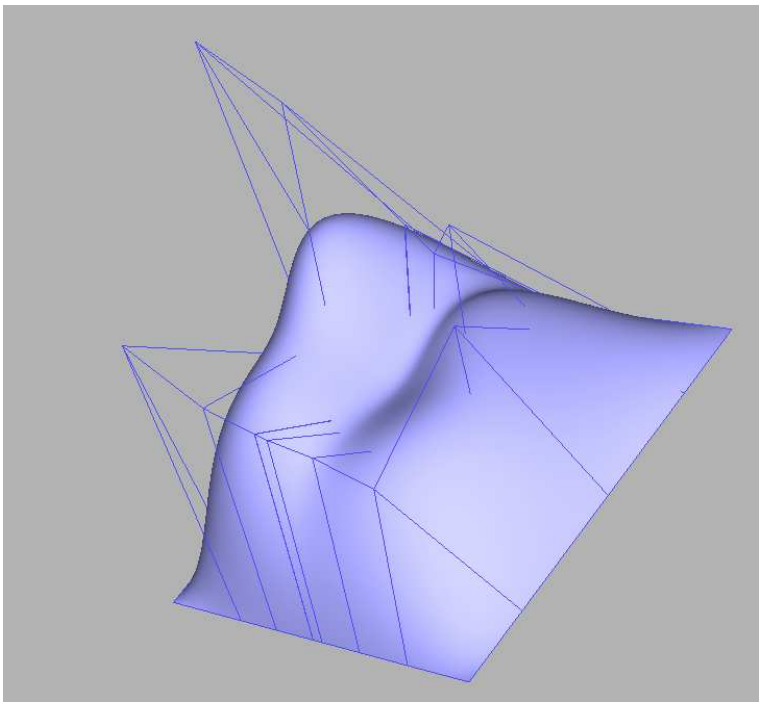


Figure 16: A B-spline surface and its control net.

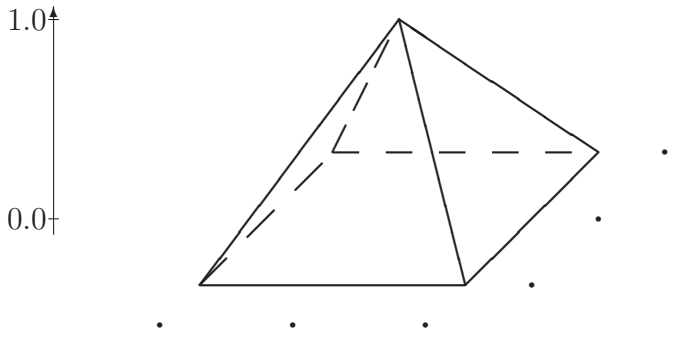


Figure 17: A basis function of degree one in both variables.

as spheres, cylinders, tori, and cones. A disadvantage is that NURBS surfaces depend nonlinearly on their weights, making some calculations less efficient.

The representation of a NURBS surface is the same as for a B-spline surface except that it also includes

w : The weights of the NURBS surface, $w_{i,j}$, $i = 1, \ldots, n_1$, $j = 1, \ldots, n_2$, so $\mathbf{w} = (w_{1,1}, w_{2,1}, \ldots, w_{n_1,1}, w_{1,2}, w_{2,2}, \ldots, w_{n_1,n_2})$.

The weights are required to be strictly positive: $w_{i,j} > 0$.

The NURBS surface is represented by SISLSurf and SplineSurface in SISL and GoTools, respectively. As for the curve case, the constructors of SISLSurf and SplineSurface expect the coefficients to be multiplied with the weights.

10.3 Spline Surface Functionality

SplineSurface inherits ParamSurface in the GoTools data structure and implements the functionality described for ParamSurface. Important additional functionality is listed below:

- Compute the derivative surface and normal surface corresponding to a given surface
- Fetch information related to the spline spaces
- Fetch the control polygon of a spline surface
- Fetch all weights of a NURBS surface
- Grid evaluation of the basis functions related to a surface
- Insert new knots into the knot vectors of the surface and adapt the surface description accordingly
- Increase the polynomial degrees of the surface
- Fetch information related to the boundary curves of a surface

11 B-spline Volumes

A B-spline volume is represented in SplineVolume in the GoTools module trivariate.

The volume is defined by the formula

$$\mathbf{V}(u, v, w) = \sum_{i=1}^{n_1} \sum_{j=1}^{n_2} \sum_{h=1}^{n_3} \mathbf{p}_{i,j,h} B_{i,k_1,\mathbf{u}}(u) B_{j,k_2,\mathbf{v}}(v) B_{h,k_3,\mathbf{w}}(w)$$

with control points $\mathbf{p}_{i,j,h}$ and three variables (or parameters) u, v and w. A basis function of a B-spline volume is a product of three basis functions of B-spline curves (B-splines).

The following is a list of the components of the representation:

dim : The dimension of the underlying Euclidean space.

n_1 : The number of vertices with respect to the first parameter.

n_2 : The number of vertices with respect to the second parameter.

n_3 : The number of vertices with respect to the third parameter.

k_1 : The order of the B-splines in the first parameter.

k_2 : The order of the B-splines in the second parameter.

k_3 : The order of the B-splines in the third parameter.

$\mathbf{u}$: The knot vector of the B-splines with respect to the first parameter, $\mathbf{u} = (u_1, u_2, \ldots, u_{n_1+k_1})$.

$\mathbf{v}$: The knot vector of the B-splines with respect to the second parameter, $\mathbf{v} = (v_1, v_2, \ldots, v_{n_2+k_2})$.

$\mathbf{w}$: The knot vector of the B-splines with respect to the third parameter, $\mathbf{w} = (w_1, w_2, \ldots, w_{n_3+k_3})$.

$\mathbf{p}$: The control points of the B-spline volume, $c_{d,i,j,h}$, $d = 1, \ldots, dim$, $i = 1, \ldots, n_1$, $j = 1, \ldots, n_2$, $h = 1, \ldots, n_3$. When $dim = 3$, we have $\mathbf{p} = (x_{1,1,1}, y_{1,1,1}, z_{1,1,1}, x_{2,1,1}, y_{2,1,1}, z_{2,1,1}, \ldots, x_{n_1,1,1}, y_{n_1,1,1}, z_{n_1,1,1}, \ldots, x_{n_1,n_2,1}, y_{n_1,n_2,1}, z_{n_1,n_2,1}, \ldots x_{n_1,n_2,n_3}, y_{n_1,n_2,n_3}, z_{n_1,n_2,n_3})$.

The data of the B-spline volume must fulfill the following requirements:

- All knot vectors must be non-decreasing.
- The number of vertices must be greater than or equal to the order with respect to all three parameters: $n_1 \geq k_1$, $n_2 \geq k_2$ and $n_3 \geq k_3$.

The properties of the representation of a B-spline volume are similar to the properties of the representation of a B-spline curve or surface. The control points $\mathbf{p}_{i,j,h}$ form a *control net*. The control net has similar properties to the control polygon of a B-spline curve, described in section 9.2. A B-spline volume has three knot vectors, one for each parameter.

11.1 The Basis Functions

A basis function of a B-spline volume is the product of three basis functions corresponding to B-spline curves,

$$B_{i,k_1,\mathbf{u}}(u) B_{j,k_2,\mathbf{v}}(v) B_{h,k_3,\mathbf{w}}(w).$$

Its support is the box $[u_i, u_{i+k_1}] \times [v_j, v_{j+k_2}] \times [w_h, w_{h+k_3}]$.

11.2 NURBS Volumes

A NURBS (Non-Uniform Rational B-Spline) volume is a generalization of a B-spline volume,

$$\mathbf{V}(u, v, w) = \frac{\sum_{i=1}^{n_1} \sum_{j=1}^{n_2} \sum_{r=1}^{n_3} h_{i,j,r} \mathbf{p}_{i,j,r} B_{i,k_1,\mathbf{u}}(u) B_{j,k_2,\mathbf{v}}(v) B_{r,k_3,\mathbf{w}}(w)}{\sum_{i=1}^{n_1} \sum_{j=1}^{n_2} \sum_{r=1}^{n_3} h_{i,j,r} B_{i,k_1,\mathbf{u}}(u) B_{j,k_2,\mathbf{v}}(v) B_{r,k_3,\mathbf{w}}(w)}.$$

In addition to the data of a B-spline surface, the NURBS surface has a weights $h_{i,j,r}$. NURBS volumes can be used to exactly represent volumes that have common 'analytic' surfaces such as spheres, cylinders, tori, and cones as boundary surfaces. A disadvantage

is that NURBS volume depend nonlinearly on their weights, making some calculations less efficient.
The representation of a NURBS volume is the same as for a B-spline volume except that it also includes

h : The weights of the NURBS volume, $h_{i,j,r}$, $i = 1, \ldots, n_1$, $j = 1, \ldots, n_2$, $r = 1, \ldots, n_3$, so $\mathbf{h} = (h_{1,1,1}, h_{2,1,1}, \ldots, h_{n_1,1,1}, h_{1,2,1}, \ldots, h_{n_1,n_2,1}, \ldots h_{n_1,n_2,n_3})$.

The weights are required to be strictly positive: $h_{i,j,r} > 0$.
The NURBS volume is represented by SplineVolume. As for the curve and surface cases, the constructor expects the coefficients to be multiplied with the weights.

11.3 Spline Volume Functionality

The functionality of a spline volume to a large extend corresponds to the functionality of a spline surface. Important functionality is:

- Evaluation and grid evaluation
- Grid evaluation of basis functions
- Compute the derivative volume corresponding to a volume
- Fetch a sub volume of a given volume
- Fetch information related to the spline spaces
- Swap and reverse parameter directions in a volume
- Fetch the control polygon of the volume
- Fetch all weights of a NURBS volume
- Insert knots into the spline spaces of the volume and adapt the volume description accordingly
- Increase the polynomial degree of the volume in one parameter direction
- Fetch a constant parameter surface from the volume
- Fetch all boundary surfaces surrounding a volume
- Check for periodicity and degeneracy

12 Other Curve Types

As shown in figure 7, a spline curve is not the only parametric curve in GoTools although it is the most important one. There are also other curves that share the public interface:

BoundedCurve A bounded curve is a restriction of a parametric curve to a given interval. The interval can be specified either in parameter space, in geometry space or both. In the last case, it must be specified whether the parameter space bound or the geometry space bound is the master.

ElementaryCurve An elementary curve is a container for a conic section or a line. This entity will be described in some detail later.

CurveOnSurface This class represents a curve lying on a surface.

12.1 Curve On Surface

The CurveOnSurface entity is often related to a bounded, or trimmed, surface. A bounded surface is limited by a curve loop where each curve in the loop most often is represented by a CurveOnSurface. However, a CurveOnSurface is an entity in its own right. It simply represents a curve lying on a surface. It can for instance originate from intersecting the surface with a plane.
A CurveOnSurface consists of a parametric surface and two corresponding curves, one curve in the parameter domain of the surface and one spatial curve. The master representation is specified. Some operations may require the existence of one particular curve, most often the parameter curve is requested. It exists functions to compute the missing curve from the existing one.

12.2 Elementary Curve

An elementary curve is a fairly new concept in GoTools, and until now such curves have been entered as entities in an IGES or STEP file. The elementary curves may also be represented as spline curves although non-bounded curves must be given a finite extension. However, the knowledge about a curve being elementary implies that some properties of the curve already is known. Operations involving this curve may be made more efficiently. The elementary curves are of the types:

Circle The circle is defined by its center and radius and the plane in which it lies if the dimension of the geometry space is larger than 2. A circle is parameterized in terms of the angle. This is a bounded parameterization.

Ellipse An ellipse is represented by a center, the direction of one semi-axis and the length of the two semi-axis. The plane in which the ellipse lies is required if the dimension of the geometry space is larger than 2. An ellipse is parameterized in terms of the angle which gives a bounded parameterization.

Line A line is given by a point and a direction. It has an unbounded parameterization, $t \in [-\infty, \infty]$.

Parabola A parabola is given by a position, C, a focal distance, F, and a coordinate system, $(\mathbf{x}, \mathbf{y}, \mathbf{z})$. It is described as $f(t) = C + F(t^2\mathbf{x} + 2t\mathbf{y})$ and has an unbounded parameterization, $t \in [-\infty, \infty]$.

13 Other Surface Types

Similar to the curve case, GoTools supports also other types of parametric surfaces than spline surfaces, see figure 7. The additional surface types are bounded surface, elementary surface and composite surface. The elementary surfaces are conic surfaces, plane and torus.

A composite surface consists of a number of spline surfaces where the associated parameter domains are mapped into one composite domain. The class is not complete in the sense that it does not support all functions specified in the interface for a parametric surface.

13.1 Bounded Surface

A bounded surface is a trimmed surface defined by an underlying surface and a trimming loop. The underlying surface is a parametric surface and the loop consists of a closed sequence of ordered parametric curves, often of type CurveOnSurface. The trimming loop must lie on the surface and be continuous with respect to a given tolerance.

13.2 Elementary Surface

Also elementary surfaces are quite new in GoTools, and have been entered as entities from IGES or STEP. The elementary surfaces may also be represented as spline surfaces although non-bounded surfaces must be given a finite extension. Operations involving this surface may, however, be made more efficiently by the knowledge of the surface type. The elementary surfaces are of the types:

Cone A cone is described by a location, the cone axis, the radius of the cross section corresponding to the location and the cone angle. The parameterization is given by the angle and the distance along the axis, thus it is unbounded in one parameter direction.

Cylinder A cylinder is given by its center and radius and the cylinder axis. It is parameterized by the angle around the circle and the distance in the axis direction.

Plane A plane is given by a point and a normal. It has an unbounded parameterization in both parameter directions.

Sphere A sphere is given by its center and radius. It has an angular parameterization in both parameter directions and thereby bounded.

Torus A torus is given by the minor and major radius, a location and an axis. The parameterization is angular in both parameter directions.

The elementary surfaces described above is placed in a coordinate system to facilitate the parameterization. In addition to the geometric information given for each entity, remaining coordinate system information must be specified.

14 Geometry Construction

The combination of SISL and GoTools provide a lot of methods for geometry construction. In the first delivery of wp6 information about methods for interpolation and approximation was included.
SISL has got many construction methods for curves and surfaces. They are described in the SISL manual, see [12].

The classes for parametric curves, surfaces and volumes in GoTools do not contain construction facilities. They are solely for operating on these entities. The construction is performed in separate classes.

14.1 Curve Construction

Construction of elementary curves are being done by their definition or they are read from an IGES file. The curve construction methods in GoTools are concerned with spline curves. The methods are split between two sub modules in gotools-core, namely geometry and creators. The relevant classes are

HermiteInterpolator The class is placed in geometry. Interpolates a set of points where each point is associated a tangent vector. The points must be parameterized. This is a local curve construction method.

SplineInterpolator A set of methods to interpolate a set of points. Some of the methods accept tangent vectors associated to some points. The points must be parameterized. The class gives the possibility to set particular conditions in the endpoints of the curve. This is a global method for curve construction.

SplineApproximator Approximation in the least squares sense. Parameterized points and a spline space must be given. This class and the two proceeding ones inherit the same class, Interpolator, and share parts of the interface.

ApproxCurve Make a curve approximating a set of points with a given accuracy. The points must be parameterized and it is possible to give an initial spline space. The method iteratively applies the method in the class SmoothCurve. This class and the following ones lie in the sub module creators.

SmoothCurve Approximate a set of parameterized data points with a spline curve using a least squares approximation with a smoothing term. The spline space must be given. The method can also be used to perform smoothing on a given curve.

HermiteAppC Approximate an evaluator based curve represented by a sub class of the class EvalCurve, by a spline curve. An hermite method is used, thus a C^1 continuous curve will be generated. The existing evaluator based curves are

- A curve in the parameter domain of a surface is represented as the projection of a 3D curve into a given surface.
- Given data describing a surface-surface intersection curve, the curve lying given offset distances from the two surface involved in the intersection and parameterized similar to the intersection curve, is represented. This construction can be used to create a rolling ball blend.
- Given a curve in the parameter domain of a surface, the geometry space curve is evaluated.
- Represent an offset curve from a given space curve. The direction of the offset is given as an expression of corresponding curves.

HermiteAppS Approximate an evaluator based curve set represented by a sub class of the class EvalCurveSet. A number of curves defined in the same spline space are generated. One concrete evaluator based curve set exists currently:

- Given a surface, a curve in geometry space and a corresponding cross tangent curve, the class evaluates the parameter curve representing the projection of the geometry curve into the surface and the corresponding projection of the cross tangent in the parameter domain of the surface.

14.2 Surface Construction

Similar to the curve construction methods, these methods are placed in the sub modules geometry and creators.

SweepSurfaceCreator The class lies in geometry. The class contains two methods; sweep a curve along another curve to create a surface and rotational sweep.

ApproxSurf This class and the following lie in creators. A set of parameterized scattered data are approximated by a spline surface. The boundary curves to the surface can be defined à priori. The spline space must be given. The method iteratively applies the method in SmoothSurface and performs parameter iteration.

SmoothSurface Approximate a set of parameterized scattered data points using a least squares approach with a smoothing term. The spline space must be given. The method can also be used to smooth a given surface.

CoonsPatchGen Construct a surface interpolating 4 boundary curves. The curves may be attached cross tangent curves. To achieve interpolation, the boundary conditions must be consistent.

HahnsSurfaceGen Fill an n-sided hole with n surfaces. $n = 3$, 5 or 6.

14.3 Volume Construction

Construction of spline volumes is taking place in the GoTools module trivariate. Three classes currently exist for this purpose:

LoftVolumeCreator Make a volume that interpolates a set of surfaces. The surfaces may or may not be parameterized in the interpolation directions. If the given surfaces are represented in different spline spaces, the spline spaces are unified.

SweepVolumeCreator Two sweep methods are implemented; sweep a surface along a given curve or sweep a surface along a circle.

CoonsPatchVolumeGen Make a volume interpolating 6 boundary surfaces. Some effort are made to organize the given surfaces in the parameter directions and ensure consistent spline spaces if the input surfaces are not already ordered.

15 IGES Converter

The IGES converter read an IGES file and represents its entities in the internal data structure of GoTools. It can also write a model represented in GoTools to an IGES file or convert between an IGES file and the internal file format of GoTools.
GoTools represent only geometric entities. Thus, IGES entities like annotation, structure, property, associativity, view, drawing and figure will be neglected. Neither are constructive solid geometry or finite element modelling entities handled. If such entities exist in a file read by the IGES converter, warning messages will be issued.
The topological entities specified in IGES 5.3, see[9] is not handled by the current version of the IGES converter. Thus, the entities vertex, edge, edge list, loop, face and shell is not handled. However, the geometric entities corresponding to these topological entities will be read. The IGES reader are able to read trimmed surfaces and they are stored as BoundedSurface in the GoTools geometry entity hierarchy. Color information is read.
The content of an IGES file is transferred to the application as a vector of GeomObjects, see section 7. By checking the type of each object and acting thereafter, the model can be stored and handled in the GoTools environment.
To write an IGES file, the file entities are added one by one to the IGES converter using the function addGeom which takes a GeomObject as parameter. The actual file is written by the command writeIGES.

16 GoTools Viewer

goview is an application program in the module viewlib. It is not a part of the isogeometric toolbox, but a utility to support visualization of geometry. goview is not intended for visualization of the simulation results as INRIA is working with visualization for the isogeometric toolbox. The GoTools viewer is included to provide some viewing opportunities until this isogeometric viewer is available.
goview can read curves and surfaces from an IGES file or from the GoTools internal file format, g2, and visualize those. Currently, goview is not able to visualize volumes. In the volume case, it is recommended to pick the boundary surfaces corresponding to the volume and draw them.
goview uses Qt for representing the GUI and openGl for graphics. Curves and surfaces are tessellated in the submodule tessellate in the module gotools-core. According to their type, curves and surfaces are approximated by triangles or line segments that are convenient for visualization using openGl.
The model in the viewer is manipulated using the mouse keys. The left one rotates the model, the middle one is for zooming and the right one for translation of the model.
goview has got a graphical user interface. It has a graphical window, a window containing an object list and a number of pull down menus:

file commands to read and write geometry and to close the current session and make the viewer ready to read a new geometry file.

view options to choose shaded or wire frame mode for visualization. A highlight modus may be toggled and some focusing facilities exist.

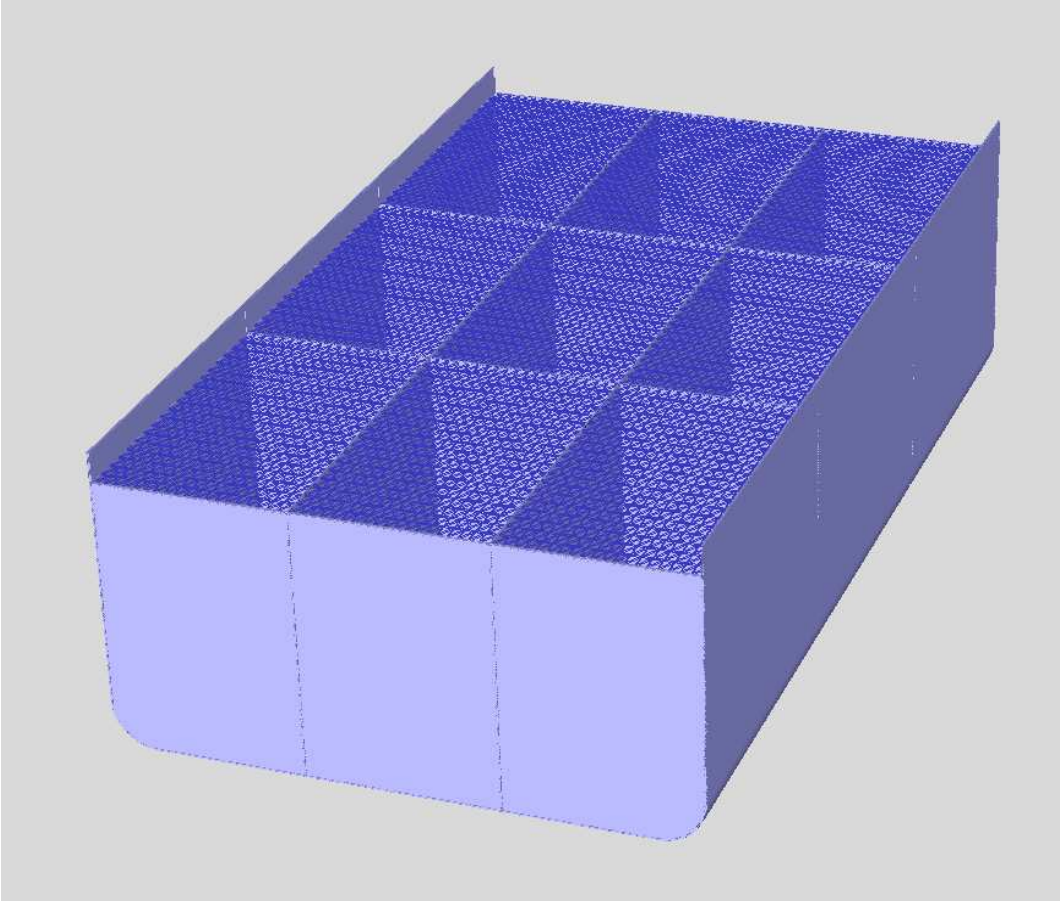


Figure 18: An isogeometric model of a midship with hull, deck and stiffeners

select Selection of entities can be done using this menu, in the object list or by using the control key in combination with the left mouse key.

group grouping of objects. This menu is not really used

object this menu offers the possibility to alter the resolution of curves and surfaces and to enable/disable selected entities from the view.

Some commands have a key pad short cut.
goview is a utility and not a product. This implies unfortunately that the help functionality is not implemented. Visualization of trimmed surfaces can have some flaws, but they are normally repaired or minimized by increasing the resolution.

17 Toolbox Operations

Isogeometric analysis relates to models that consists of one or more spline surfaces or volumes. Trimmed geometry and elementary surface are out of scope, at least for the finalized isogeometric model.
Figure 18 shows a model created by geometry construction facilities existing in the current toolbox. Although the toolbox is still in its childhood, reasonable models can be made.

However, currently most considerations with regard to choosing building operations and matching operations in a way that gives a well behaved model with a good parameterization is left to the user. Moreover, adjacency information is not a part of the current toolbox.
Curves and surface stored in an IGES file may be read into the isogeometric toolbox and serve as a starting point in building an isogeometric model.
Once a model exist, the toolbox offers good functionality with regard to the most important operations for isogeometric analysis, namely

- Evaluation and grid evaluation of the model
- Access to the basis functions
- Evaluation and grid evaluation of the basis functions
- Knot insertion
- Degree elevation

18 Visualization tools for isogeometric analysis in AXEL

Axel 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 standalone or connected together with external tools and therefore become a graphical frontend. Its modular architecture using plugins makes easy the wrapping of external libraries or the connection with other tools. This will be detailed in section 19. First we recall the design choice of the algebraic-geometric modeler Axel and the visualization capability that it provides.

18.1 Hierarchy of objects for the visualization

The design of Axel modeler is following a hierarchy of classes of data-structures and a way to implement specializations at each level in this hierarchy, the most specific implementation of an algorithm is chosen at run-time, providing effectiveness and ease of coding.
This hierarchy is extending the hierarchy of geometric objects of the isogeometry toolbox by providing new functionalities for visualisation purposes. The implementation is using the mechanism of virtual hierarchy of C++ classes to select the more specialised implementation. It is described in the last report WP6.1.
The following classes of objects have been embedded into the geometric modeler:

- **Points** and *point sets* are the primal geometric entities that can be manipulated in the modeler. They are given by their 3D coordinates and possibly with a color information.
- **Curves** in two or three dimensions, such as
  - *Polygonal curves* represented by a set of points and a set of segments connecting these points;

  - *Parametric curves* which are the image of a map $\phi : t \mapsto \left(\frac{x(t)}{w(t)}, \frac{y(t)}{w(t)}, \frac{z(t)}{w(t)}\right)$ defined on an interval $[a, b] \subset \mathbb{R}$ where x, y, z, w are polynomials or B-spline functions, based on the isogeometry toolbox representation. Axel features an edition mode which allow to dynamically modify the control polygon of the curve by moving its control points. Whenever a parametric curve is selected, a widget shows up, representing the parameter space of the curve, allowing the user to query any parameter and dynamically view the corresponding image in the euclidean space;
  - *Implicit curve* defined as the set of real solutions of an equation of the form $p(x, y) = 0$.

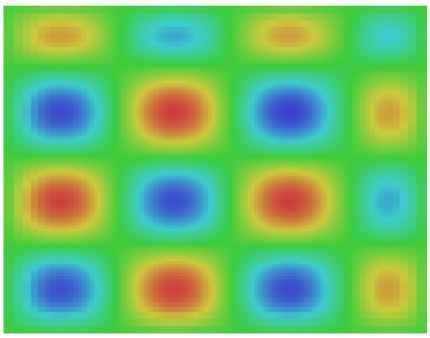
- **Surfaces**, which includes
  - *Meshed surfaces*, represented as a set of points and a set of faces, which are triangles or more general polygons;
  - *Parametric surfaces* which are the image of maps $\phi : (s, t) \mapsto \left(\frac{x(s,t)}{w(s,t)}, \frac{y(s,t)}{w(s,t)}, \frac{z(s,t)}{w(s,t)}\right)$ defined on a box $[a, b] \times [c, d] \subset \mathbb{R}^2$ where x, y, z and w are either polynomials or bivariate B-spline functions. The same features of edition and visualisation of the parameter space as with curves are available. The B-spline function representation is the one provided by the isogeometry toolbox.
  - *Implicit surfaces* (or algebraic surfaces), are the real zero set of a given (polynomial) function in three variables with real coefficients $p(x, y, z)$. The surface is not assumed to be smooth, and we use a dedicated subdivision topology algorithms, to dynamically displays it.

- **Volumes** which can be classified in a similar way as surfaces, that is meshed, parametric or implicit. For the purpose of Exciting project, we consider parametric volumes which are the image of maps $\phi : (s, t, u) \mapsto \left(\frac{x(s,t,u)}{w(s,t,u)}, \frac{y(s,t,u)}{w(s,t,y)}, \frac{z(s,t,u)}{w(s,t,u)}\right)$ defined on a box $[a, b] \times [c, d] \times [c, d] \subset \mathbb{R}^3$ where x, y, z and w are trivariate B-spline functions. The package used to display and edit the B-spline volumes is provided by the isogeometry toolbox .

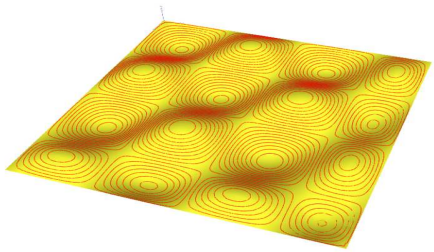
18.2 Visualization of functions on geometric data

In isogeometric analysis, we often simulate some measures on the CAD models, such as temperature and pressure. In order to show the simulation results clearly and meaningfully, we have developed some visualization tools by using color map and iso-value curves in algebraic geometric modeler AXEL [1]. The code below is the interface implementation for visualization of a color-map.

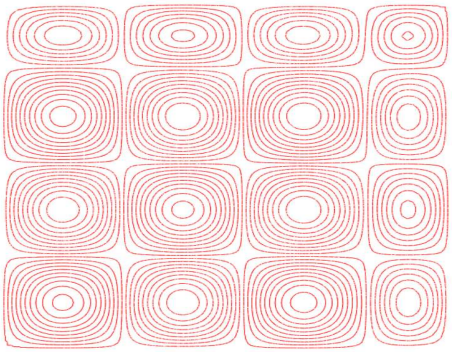
```
m_menu_visu = m_menu->addMenu(tr("Visualization"));

m_isovalueCurve_action = new QAction("Show iso-value curves", m_menu_visu) ;
m_menu_visu->addAction(m_isovalueCurve_action) ;
connect(m_isovalueCurve_action, SIGNAL(triggered()), this, SLOT(showIsovalueCurve())) ;
```



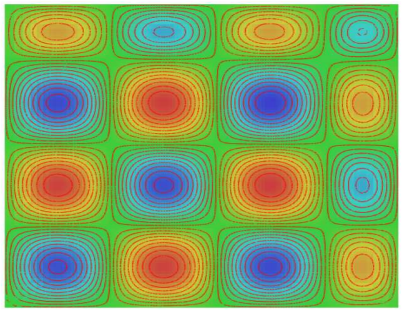
(a) color map



(b) isovalue curves on solution surface



(c) isovalue curves on 2D plane



(d) color map and isovalue

Figure 19: Color map and isovalue curves for visualization

```
m_colorMap_action = new QAction("Show color map", m_menu_visu) ;
m_menu_visu->addAction(m_colorMap_action) ;
connect(m_colorMap_action, SIGNAL(triggered()), this, SLOT(showColorMap())) ;
```

The color map is used for showing the distribution of temperature values on the solution surface. Red color is used for rendering areas with biggest temperature value, green color is used for rendering areas with median temperature value, and blue color is used for rendering areas with smallest temperature value. See Fig 19 (a)(d).
The iso-value curve is used for showing the points with the same temperature value on the solution surface. See Fig 19(b)(c)(d). These curves are obtained by intersecting the corresponding B-spline surface representing the graph of a function with a plane corresponding to the iso-value.

19 Isogeometric solver plugin in AXEL

19.1 Plugin in AXEL

Axel can be extended through plugins, an extension that interacts with it to provide specific functionalities. There are several reasons to use plugins. First, this enables third-party developers to extend the application. Second it allows separating source code from the application because of incompatible software licenses or dependencies.
Making the application extensible through plugins involves to provide an abstract interface from which the plugin must inherit, which consists in a set of methods it should overload. The code below is an example of a very simple plugin interface.

```
class PluginInterface
{
public :
    virtual ~PluginInterface() {}

    virtual void        init(CoreInterface *) const = 0 ;
    virtual String      name(void)            const = 0 ;
    virtual StringList  features(void)        const = 0 ;
} ;
```

Since C++ does not provide interfaces in the sense Java or Objective-C do with *interfaces* or *protocols*, a plugin interface is an abstract class which can not be instantiated. This interface is the link between the application and the plugin.
In the code presented above, the interface is made up of an initialization function which gives the plugin an access to a structure containing pointers to the application main entities in order for example to let the plugin add entries to the application interface and associate it with callbacks on the plugin internal functions.
In addition, this very simple plugin interface permits the application to obtain information about the plugin such as its name and the set of features it provides.
The plugins of Axel can be used as a glue between the modeler and any third party computational library, which enables the use of Axel as a graphical frontend to the glued library. These plugins can be developed by anyone, following the plugin documentation. Some of these plugins are mandatory to use Axel, since they provide most of the interesting functionalities. Currently there are six such plugins which are called:

- `QGeometricKernel` which implements the connection with B-spline representation for parametric curves, surfaces and volumes;
- `QAlgebraicKernel` which implements the connection with algebraic tools for the manipulation of algebraic curves and surfaces;
- `QphysicalSolverPlugin` which implements the connection with an isogeometric solver of heat conduction equations (see section 19.2);
- `QMinimalSurface` which implements the connection with an tools for the placement of inner control points minimizing a geometric cost-function (see section 20).

and `QCUDAImplicitRayCaster` for ray-casting of implicit algebraic surfaces on GPU, `QComplexInvariantsPlugin` for the computation of knot invariants attached to singular points.

19.2 Isogeometric solver plugin

We have developed a plugin called `QphysicalSolverPlugin` in AXEL, as a *general framework for isogeometric-based Finite-Element methods and related design optimization.* This plugin consists of three parts:

1. PDE's solver;
2. shape optimizer;
3. parametrization. For the parametrization part, we will discuss it in Section 20.

As example, a heat conduction solver and two design optimization algorithms have been implemented . The heat conduction solver is detailed in the following section.

19.2.1 Problem statement

We consider the following problem: Given a domain Ω closed by the boundary $\Gamma = \Gamma_D \bigcup \Gamma_N$:

$$\begin{aligned} &\nabla(\kappa(\boldsymbol{x})\nabla \boldsymbol{T}(\boldsymbol{x})) = 0 \quad in \quad \Omega \\ &\boldsymbol{T}(\boldsymbol{x}) = \boldsymbol{T}_0(\boldsymbol{x}) \quad on \quad \Gamma_D \\ &\kappa(\boldsymbol{x})\frac{\partial \boldsymbol{T}}{\partial \boldsymbol{n}}(\boldsymbol{x}) = \Phi_0(\boldsymbol{x}) \quad on \quad \Gamma_N \end{aligned}$$

where $\boldsymbol{x}$ are the Cartesian coordinates, $\boldsymbol{T}$ represents the temperature field and κ is the thermal conductivity. Dirichlet and Neumann boundary conditions are applied on Γ_D and Γ_N respectively, $\boldsymbol{T}_0$ and Φ_0 being the imposed temperature and thermal flux ($\boldsymbol{n}$ is unit vector normal to the boundary).

19.2.2 PDE's solver

The whole algorithm for solving the PDE problem can be described as follows (see [5] for details):

1. Construction of the CAD representation of the computational domain (see details in Section 20)
2. Computation of the matrix coefficients. For each knot span:
   - Computation of the B-spline function and gradient values at the Gauss points;
   - Computation of the transformation matrix and Jacobian at the Gauss points ;
   - Calculation of the stiffness matrix coefficients ;
   - Assembly of the matrix ;

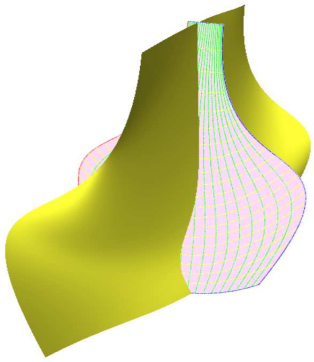


Figure 20: A computational example of the solver

3. Computation of the right-hand side terms (source terms and Neumann boundary contributions) ;

4. Imposition of values for boundaries with Dirichlet conditions ;

5. Solving of the linear system ;

6. Possible refinement process:

   - Refinement for the geometry ;
   - Refinement for the solution field ;

   Go to step 2 until a stopping criterion is satisfied ;

7. Reconstruction of the solution field (post-process).

Fig 20 shows a computational example, in which the yellow surface are the solution field, and the pink surface are the computational domain.

19.2.3 Shape optimizer

On the basis of the solver, a design optimization procedure has been developed to improve the performance of a given system. The graphical interface is used to set all the parameters required to define the design optimization problem. In particular: the choice of the design parameters (control points of a starting B-Spline curve for 2D problems) and their degrees of freedom (x or y), the choice of the objective function and the possible constraints, the choice of the optimizer and its parameters. Then a function can be called that solves the optimization problem, yielding two new B-Spline surfaces: one for the optimized B-Spline curve and one for the corresponding physical solution.
We have initiated the development of a C++ library based on the "ask & tell" architecture. Especially, two algorithms have been implemented:

- A Steepest Descent (SD) method including an adaptive line search technique and a centered finite-difference approximation of the gradients;

- An Evolution Strategy (ES) based on a Gaussian sampling of the search space and a $\lambda + \mu$ population recombination.

Algorithmic details and illustrations can be found in WP5 report "Optimization proof of concept".

19.3 Implementation for the plugin

In this subsection, we present some interface code to show the implementation for the isogeometric solver plugin.

```
class QphysicalSolverPlugin : public QObject, public QInitializablePluginInterface
{
  Q_OBJECT
  Q_INTERFACES(QAxel::QPluginInterface QAxel::QInitializablePluginInterface)

public:

  QStringList features() const;
  QString name() const;
  void init(MainWindow * window);
  void close(MainWindow * window);

private slots:

  void selectBoundaryCondition(void);
  void selectSolverParameters(void);
  void computeSolution(void);
  void storeBoundaryCondition(void);
  void set1DProblem(void);
  void set2DProblem(void);
  void set3DProblem(void);
  void storeSolverParameters(void);
  void showColorMap(void);
  void showIsovalueCurve(void);
  void project2D(void);
  void selectDesignParameters(void);
  void storeDesignParameters(void);
  void selectOptimizerParameters(void);
  void storeOptimizerParameters(void);
  void selectObjectiveParameters(void);
  void storeObjectiveParameters(void);
  void computeOptimal(void);
  void parametrizeComputationalDomainShapeOpt(void);

private:

  int dimension;

  MainWindow *m_main;
```

```
QMenu      *m_menu;

QAction    *m_1D_action;
QAction    *m_2D_action;
QAction    *m_3D_action;
QMenu      *m_menu_dimension;

QMenu      *m_menu_solver;
QAction    *m_boundary_action;
QAction    *m_paramsolver_action;
QAction    *m_solve_action;

QMenu      *m_menu_optimizer;
QAction    *m_design_action;
QAction    *m_optimizer_action;
QAction    *m_objective_action;
QAction    *m_optimize_action;

QMenu      *m_menu_visu;
QAction    *m_colorMap_action;
QAction    *m_isovalueCurve_action;

QMenu      *m_menu_tools;
QAction    *m_project_action;
QMenu      *m_menu_optPara;
QAction    *m_optPara_action;

QWidget     *bndwindow;
QVBoxLayout *bndmainlayout;
BoundaryBox *bndBox;
QTabWidget  *bndtabWidget;
QPushButton *bndvalidButton;
QString     *bndlabeltab;

QWidget     *paramwindow;
QVBoxLayout *parammainlayout;
QGridLayout *paramlayout;
ParamBox    *paramBox;
QPushButton *paramvalidButton;
QString     *paramlabeltab;

QWidget     *designwindow;
QVBoxLayout *designmainlayout;
DesignBox   *designBox;
QTabWidget  *designtabWidget;
QPushButton *designvalidButton;
QString     *designlabeltab;

QWidget     *optimwindow;
```

```
  QVBoxLayout *optimmainlayout;
  QGridLayout *optimlayout;
  OptimBox    *optimBox;
  QPushButton *optimvalidButton;
  QString     *optimlabeltab;

  QWidget     *objwindow;
  QVBoxLayout *objmainlayout;
  QGridLayout *objlayout;
  ObjBox      *objBox;
  QPushButton *objvalidButton;
  QString     *objlabeltab;

  BoundaryData *boundary;
  SolverParameters *solverparam;
  FiniteElementSolver *FEsolver;
  OptimizationParameters *optimparam;
  Optimizer *optimizer;
  ObjectiveFunction *objectivefunction;
  SourceFunction srcfunction;
  double computeErrorFunc(QBSplineSurface* updatesol);

} ;
```

20 A plugin for the parameterization of computational domain for 2D problems in isogeometric analysis

For the isogeometric analysis framework, the computational domain is exactly described using the same representation as that employed in the CAD process. Hence, once the CAD object (such as B-spline curve) is given, we need to construct the same CAD representation of the whole computational domain that surrounds the CAD object. For 2D problems (the CAD object consists of B-spline curves), the computational domain is represented as a 2D B-spline surface on a plane. Given four planar boundary B-spline curves, for different placement of inner control points, the corresponding planar B-spline surface has different parametrization (see Fig 21 For different parametrization of computational domain), we can obtain different results in isogeometric analysis. Hence, how to construct the computational domains (2D B-spline surface) from the given CAD objects (four boundary B-spline curves), is a key issue for the isogeometric analysis.
Two kinds of methods are developed in `QphysicalSovlerPlugin` for parametrization of computational domains. One is a constraint optimization method for the general 2D isogeometric problem, and another one is an optimization based method for isogeometric problem with exact solutions.

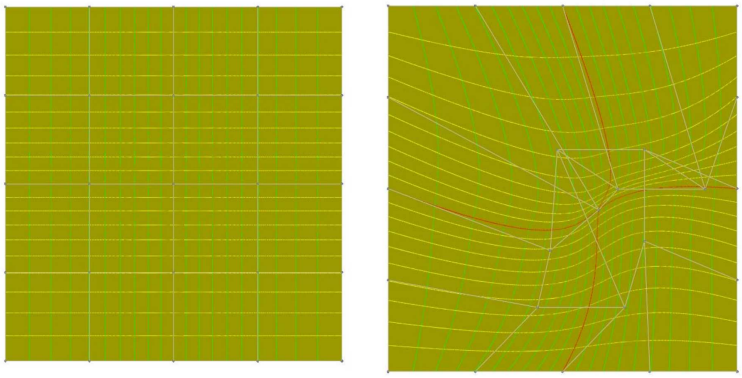


Figure 21: Two different parametrization of computational domain

20.1 Constraint optimization method for parametrization of computational domain

We have developed a constraint optimization method to get ideal parametrization of computational domain. The constraint term is for no self-intersection, and the optimization term is for uniform and orthogonal grid formed by the isoparametric curves.

20.1.1 Planar B-spline surface

Planar B-spline surface $\boldsymbol{F}(u,v)$ can be defined as follows,

$$\boldsymbol{F}(u,v) = \sum_{i=0}^{m}\sum_{j=0}^{n} N_i^p(u) N_j^q(v) \boldsymbol{P}_{ij} \tag{-3}$$

with $\boldsymbol{U} = \{0,\ldots,0,u_{p+1},\ldots,u_{r-p-1},1,\ldots,1\}$, $\boldsymbol{V} = \{0,\ldots,0,v_{p+1},\ldots,v_{r-p-1},\ 1,\ldots,1\}$ and $\boldsymbol{P}_{ij} = (P_{ij}^x, P_{ij}^y)$.
Hence, it can be considered as a mapping function from $[0,1]\times[0,1]$ to $\boldsymbol{F}(u,v)$ (see Fig 22).
The first-order derivatives of $\boldsymbol{F}(u,v)$ is a B-spline surface with lower degree, that is,

$$\begin{aligned}
\boldsymbol{F}_u(u,v) &= \sum_{i=0}^{m-1}\sum_{j=0}^{n} \boldsymbol{P}_{ij}^{(1,0)} N_{i,p-1}(u) N_{i,q}(v), \\
\boldsymbol{F}_v(u,v) &= \sum_{i=0}^{m}\sum_{j=0}^{n-1} \boldsymbol{P}_{ij}^{(0,1)} N_{i,p}(u) N_{i,q-1}(v),
\end{aligned}$$

where

$$\boldsymbol{P}_{ij}^{(1,0)} = p\frac{\boldsymbol{P}_{i+1,j} - \boldsymbol{P}_{i,j}}{u_{i+p+1} - u_{i+1}} \tag{-2}$$

$$\boldsymbol{P}_{ij}^{(0,1)} = q\frac{\boldsymbol{P}_{i,j+1} - \boldsymbol{P}_{i,j}}{v_{j+q+1} - v_{j+1}} \tag{-1}$$

The product of two B-spline basis function can be represented as a B-spline basis function with higher degrees, that is,

$$N_{i,p}(u) N_{j,q}(u) = \omega_{i,j} N_{k,p+q}(u) \tag{0}$$

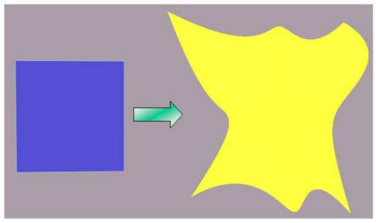


Figure 22: Planar B-spline surface is a 2D mapping function.

where $\omega_{i,j}$ can be computed as in [11].

20.1.2 Constraint condition for no self-intersection

Planar B-spline surfaces can be considered as a mapping function from $[0,1] \times [0,1]$ to $\boldsymbol{F}(u,v)$ (see Fig 2). In order to obtain the constraint condition for no self-intersection, the mapping function should be injective. From [10], we have the following proposition.

Proposition 20.1 *Suppose that* $\mathbf{F}$ *is a mapping from a connected compact domain* $\Omega_0 \subset \mathbb{R}^n$ *to a domain* $\Omega_1 \subset \mathbb{R}^n$*, if* $\mathbf{F}$ *is injective on the connected boundary* $\partial\Omega_0$ *and the Jacobian determinant of* $\mathbf{F}$ *is non-zero, then* $\mathbf{F}$ *is injective on the interior domain of* Ω_0*.*

From Proposition, we deduce that if the Jacobian determinant $\boldsymbol{J}(\boldsymbol{F})$ of the planar B-spline surface satisfies $\boldsymbol{J}(\boldsymbol{F}) > 0$ and the surface parameterisation is injective on the boundary, then $\boldsymbol{F}(u,v)$ has no self-intersections.
From (-2), (-1), (0), the Jacobian determinant of a B-spline surface $\boldsymbol{F}$ can be computed as follows

$$\begin{aligned} \boldsymbol{J}(\boldsymbol{F}) &= \begin{vmatrix} \boldsymbol{F}_u^x & \boldsymbol{F}_v^x \\ \boldsymbol{F}_u^y & \boldsymbol{F}_v^y \end{vmatrix} \\ &= \boldsymbol{F}_u^x \boldsymbol{F}_v^y - \boldsymbol{F}_v^x \boldsymbol{F}_u^y \\ &= \sum_{i=0}^{2m-1} \sum_{j=0}^{2n-1} G_{ij} N_{i,2p-1}(u) N_{i,2q-1}(v) \end{aligned}$$

Hence, the Jacobian determinant of B-spline surface can be represented in the form of B-spline surface with higher degrees. From the convex hull property of B-spline surface, we have the following theorem.

Theorem 20.2 *If* $G_{ij} \geq 0$*, then* $\mathbf{J}(\mathbf{F}) > 0$*, that is,* $\mathbf{F}(u,v)$ *has no self-intersections.*

20.1.3 Optimization term for uniform and orthogonal grid

In order to obtain B-spline surface with uniform and orthogonal isoparametric grid, we use the following optimization term as objective function,

$$min \iint (\| \boldsymbol{F}_{uu} \|^2 + \| \boldsymbol{F}_{vv} \|^2 + 2 \| \boldsymbol{F}_{uv} \|^2) + \omega(\| \boldsymbol{F}_u \|^2 + \| \boldsymbol{F}_v \|^2) dudv$$

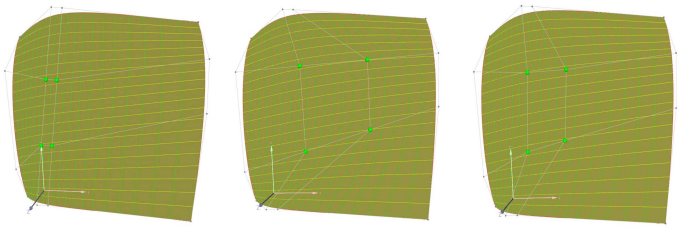


Figure 23: Construction of initial solution:

It has the following geometric interpretation: the first part is for minimal length change in four directions on isoparametric grid, and the second part is for orthogonal isoparametric curves.

20.1.4 Initial construction of inner control points

In order to solve this constraint optimization problem by using SQP method, initial construction of inner control points is required. We construct the initial control points as follows (see Fig 23 for an example),

- construct $(n-2)$ line segments by connecting the corresponding points on opposite boundary polygons along u-direction
- divide each of the generated $n-2$ pieces of line segments into $n-1$ portions of equal length, generate $(m-2)\times(n-2)$ points $\boldsymbol{P}_{ij}^1$.
- construct $(n-2)$ line segments by connecting the corresponding points on opposite boundary polygons along v-direction
- divide each of the generated $n-2$ pieces of line segments into $n-1$ portions of equal length, generate $(m-2)\times(n-2)$ points $\boldsymbol{P}_{ij}^2$.
- initial control points are midpoints of $\boldsymbol{P}_{ij}^1$ and $\boldsymbol{P}_{ij}^2$

$$\boldsymbol{P}_{ij}^0 = \frac{\boldsymbol{P}_{ij}^1 + \boldsymbol{P}_{ij}^2}{2} \tag{1}$$

20.1.5 Overview of the algorithm

Input: four coplanar boundary B-spline curves
Output: inner control points and the corresponding planar B-spline surfaces

- construct the initial inner control points as in subsection 20.1.4
- compute G_{ij} as in subsection 20.1.2
- solve the constraint optimization problem by using SQP method

$$\begin{aligned} &min \quad \iint (\| \boldsymbol{F}_{uu} \|^2 + \| \boldsymbol{F}_{vv} \|^2 + 2 \| \boldsymbol{F}_{uv} \|^2) \\ &+\omega(\| \boldsymbol{F}_{u} \|^2 + \| \boldsymbol{F}_{v} \|^2) dudv \\ &s.t. \quad G_{ij} \geq 0 \end{aligned}$$

- generate corresponding planar B-spline surface $\boldsymbol{F}(u,v)$ as computational domain

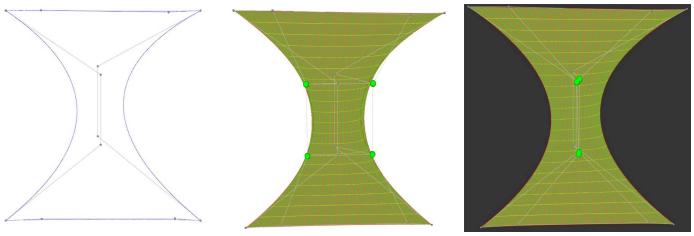


Figure 24: Example 1

20.1.6 Results and comparison

In this section, we will present some construction results and compare them with the initial solution by using the test model, which is the heat conduction problem solved in our physical solver plugin.
Fig 24 (a) present the given boundary curves, Fig 24(b) shows the initial result, the optimization result is shown in Fig 24(c). There are some self-intersections in the initial result, and our optimization result avoids self-intersection and gives uniform iso-parametric grid. We present the simulation results in Fig 25 for different parametrization. The simulation result of initial solution are shown in the left column, and the simulation result of final solution are presented in the right column. We can find that there are also some self-intersections on the initial solution surface. The corresponding color map and isovalue curves on the solution surfaces are also shown in Fig 25.

20.2 Optimal parametrization of computational domain for isogeometric problem with exact solution

Motivated by the idea of shape optimization, we proposed a new approach for the optimal parametrization of computational domain for isogeometric problem with exact solution. In this method, the optimization variables are the inner control points rather than control points of specialized boundary B-spline curve. Then we can use the shape optimization method in subsection19.2.3 to optimize the inner control points for optimal parametrization of computational domain.

21 Plans for the Next Period

Even though the toolbox is already usable, the contrast between the idealized picture of an isogeometric toolbox presented in the first part of this report and the current version of the toolbox presented in the last part, is large. During the project period, the toolbox will be extended in several stages and become more complete.
The current toolbox has got good tools to handle single block models, i.e. models consisting of one spline surface or one spline volume. To work with multi block models, adjacency information is crucial. GoTools possesses topological data structures for surface models that compute and represent neighborhood information in a surface set. The data structures are currently quite CAD oriented. They must be extended to present adjacency information in a way that is convenient for isogeometric analysis, i.e. correspondence between coefficients

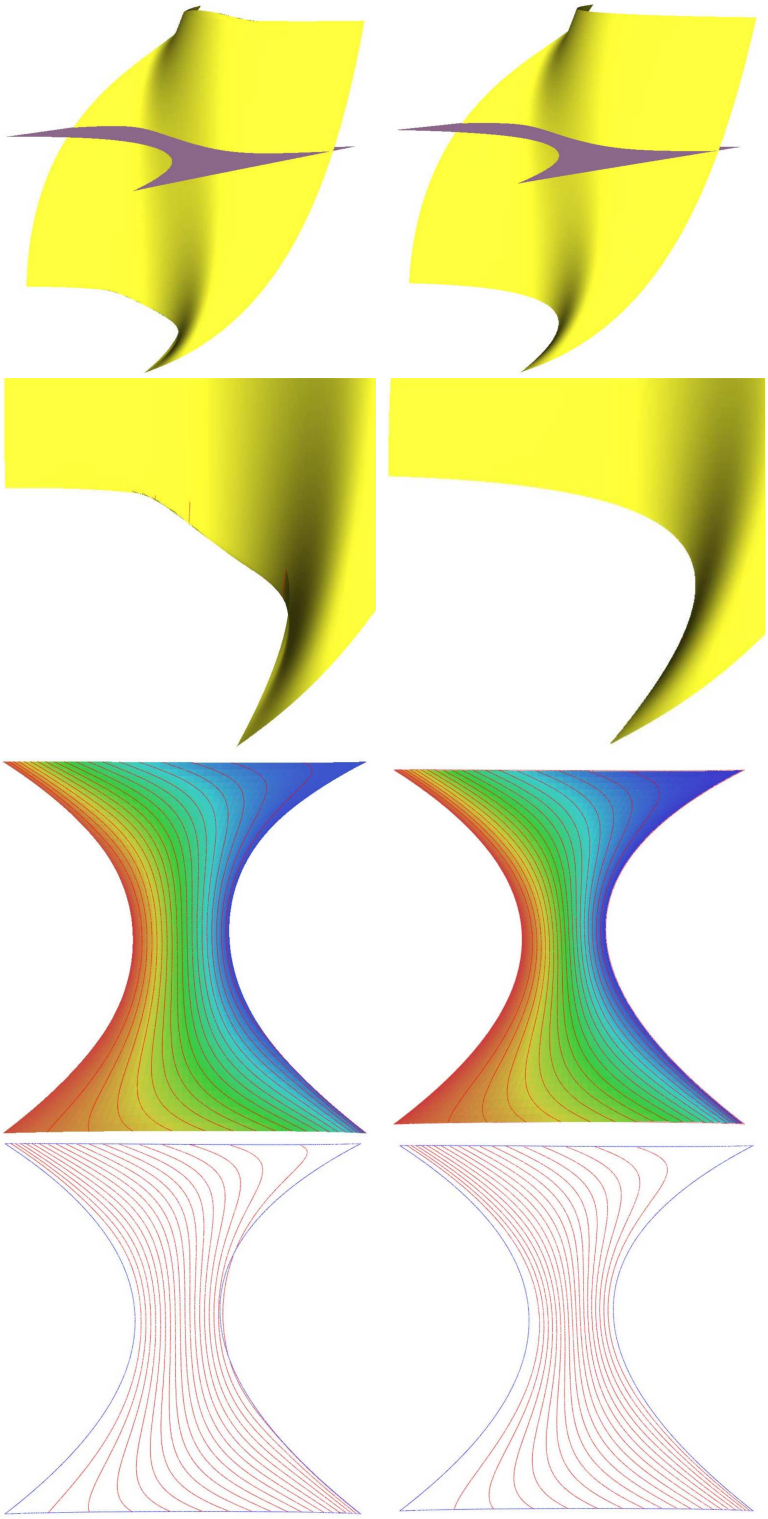


Figure 25: Example 1

on surfaces sharing a common boundary must be easily achievable. Furthermore, similar data structures must be developed for volumetric models.
GoTools are able to represent the geometry of a model using surfaces and volumes. It is also possible to fetch the spline spaces related to the geometry entities and use those spline spaces to define a solution entity represented as a SplineSurface or a SplineVolume. However, no data structures exist to represent the connection between corresponding geometry and solution entities. Similarly, it lacks data structures to connect material properties, boundary conditions and forces to the geometry and solution. In the next period, a study will be made to define the correct data structures. Then these structures will be implemented.
The current isogeometric toolbox is highly connected to geometry. When the more specific isogeometric data structures are implemented, the toolbox should incorporate more elements from numerical analysis. These contributions must be provided by several partners, particularly the partners performing isogeometric analysis.
The current version of the toolbox has some methods for curve, surface and volume generation. The set of generation methods will be extended, particularly on the volume side. A new, more elaborate method for swept volume computation is planned and described in [2]. Furthermore, not all existing methods handle rational geometry. It will be considered whether the method may be extended with NURBS, or alternatively searched for alternative methods to create a particular entity.
The current toolbox will also be extended by triangular splines, which will arbitrary topology as well as good refinement properties. The functionalities needed to compute efficiently with this surface and volume representations will completed to be used in isogeometric analysis.
For the visualisation tools, we will complete the integration of existing geometric models into the algebraic-geometric modeler AXEL. A more systematic mechanism for the integration of external representation will be developed. We also plan to introduce dynamic objects, which allow to recompute and visualise geometric objects depending on parameters. This will be used in optimisation loops where the cost function is computed by an isogeometric solvers.

References

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

[2] M. Aigner, C.Heinrich, B. Jüttler, E. Pilgerstorfer, B.Simeon and A.-V. Vuong. Swept Volume Parameterization for Isogeometric Analysis. Preprint

[3] E. Cohen, T. Martin, R.M. Kirby, T.Lyche and R.F. Riesenfeld. Analysis-aware Modeling: Understanding Quality Considerations in Modeling for Isogeometric Analysis. To appear.

[4] T. Dokken and V. Skytt. Isogeometric Representation and Analysis – Bridging the Gap between CAD and Analysis. 47th AIAA Aeorospace Sciences Meeting (2009)

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

[6] M.S. Floater, Parameterization and smooth approximation of surface triangulations. Computer Aided Geometric Design 14 (1997), 231 – 250

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

[8] J.A. Cottrell, T.J.R. Hughes, A. Reali. Studies of refinement and continuity in isogeometric structural analysis. Computer methods in applied mechanics and engineering 196 (2007), 4160 – 4183

[9] Initial Graphics Exchange Specification IGES 5.3. IGES/PDES Organization

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

[11] Morken K. Some identities for products and degree raising of splines. Constructive Approximation 1991;7(2):195-208.

[12] SISL. The SINTEF Spline Library Reference Manual, version 4.1.

[13] Gang Xu, Guozhao Wang, Xiaodiao Chen. Free form deformation with rational DMS-spline volumes. Journal of Computer Science and Technology, 2008, 23(5): 862-873