EC-Project EXCITING SCP8-2007-GA-218536

EXACT GEOMETRY SIMULATION
FOR OPTIMIZED DESIGN
OF VEHICLES AND VESSELS

http://exciting-project.eu

EXCITING Report No. 6.1

First definition of isogeometric framework, cases, data structures and methods

Vibeke Skytt, SINTEF
Xu Gang & Bernard Mourrain, INRIA

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

March 2009

Dissemination level: PU

Contents

1 Introduction 4

2 GoTools and SISL 4
- 2.1 Geometry Structures . . . 6
- 2.2 Topology Structures for Surface Models . . . 8
- 2.3 Topology Structures for Volume Models . . . 9

3 The Geometric Modeler Axel 10
- 3.1 The generic framework . . . 11
- 3.2 The Geometric primitives in Axel . . . 12
  - 3.2.1 Points . . . 12
  - 3.2.2 Curves . . . 13
  - 3.2.3 Surfaces . . . 14
  - 3.2.4 Volumes . . . 15
- 3.3 The plugin system . . . 15
- 3.4 Interface . . . 16

4 The Relation between the Isogeometry Toolbox and the other Work Packages, Test Cases 18
- 4.1 Work Package 1 . . . 18
- 4.2 Work Package 2 . . . 18
- 4.3 Work Package 3 . . . 19
- 4.4 Work Package 4 . . . 20
- 4.5 Work Package 5 . . . 20

5 B-spline Basics 21

6 The Isogeometric Toolbox 23
- 6.1 Grid Evaluation . . . 24
- 6.2 Refinement and Degree Elevation . . . 24
- 6.3 Numerical Integration . . . 25
- 6.4 Parameterization of Computational Domains . . . 25
- 6.5 Volume Creation . . . 26
- 6.6 Intersection, Self-intersection, Topology and Arrangements . . . 27
- 6.7 Curve and Surface Fitting . . . 28
- 6.8 Data Reduction . . . 28
- 6.9 Merging of Surface Patches . . . 29
- 6.10 Tangent Plane Continuity between Surfaces . . . 29
- 6.11 Modification of Sets of Surfaces . . . 32
- 6.12 Graphical rendering of surfaces and volume object . . . 32

7 From a Brep CAD Model to a Multi-block NURBS Volume Model 33

8 Conclusion 35

A WP6 **37**

A.1 Introduction 37

A.2 Exact methods for curve and surface generation 37

A.2.1 Curves 37

A.2.2 Surfaces 37

A.3 SmoothCurve and ApproxCurve 38

A.4 SmoothSurf and ApproxSurf 39

A.5 The parameterization module 41

A.6 Surface creation from point sequences 41

A.7 Approximation of evaluator based curves 42

A.8 Modification of a surface set 42

B WP2 **44**

C WP3 **45**

D WP4 **46**

1 Introduction

Exciting focuses on optimized design of functional free-form surfaces. Isogeometric analysis is used to evaluate the performance of a given design. Isoparametric analysis is a well known concept that means that the solution space of the physical field and the space in which the geometry is represented, is the same. In isogeometric analysis, NURBS is used both to represent the geometry and as a solution space for the differential equation. This choice of solution space reduces or even eliminates the need for an approximate geometry which is normal in current methods for numerical simulation and analysis.

In WP6 the goal is to develop a set of tools for isogeometry based analysis and shape optimization. As the isogeometric approach to analysis is a fairly new concept, no isogeometric toolbox is currently available. The ambition level of the Exciting Isogeometry Toolbox is to coordinate the toolbox activities and support the projects need for

- CAD model import and export
- CAD model checking and repair
- Building of isogeometric volumes from a watertight surface model
- Refine the element structure of the isogeometric model
- Numerical integration
- Export of analysis models to visualization tools

Much of the software in the toolbox will be developed outside of the Toolbox work package, either from other Exciting work packages or from outside of Exciting.

The tools focus on geometry and NURBS functionality to support the PDE solvers. Furthermore, it will focus on the type of geometry which is relevant to Exciting, namely ship hulls and propellers in naval sector and car components, frames and turbochargers in automotive and wheel-rail sectors. Thus, the toolbox is not expected to cover all required functionality to take a general boundary represented CAD model and produce a a set of NURBS volumes representing the same model. Moreover, restrictions on the complexity of the surface model is expected.

SINTEF and INRIA are the main participants in work package 6. SINTEF offer the background software GoTools and SISL while INRIA has got Axel.

The document is organized as follows. First GoTools and Axel are presented. Then the cases to be investigated in Exciting are referenced. These cases will to a large extent act as a requirement specification to the toolbox. It must provide the basic functionality for being able to handle the cases. A summary of B-spline properties are given next. Finally, methods for the planned functionality are described. Background information to this document is collected in appendices.

2 GoTools and SISL

SISL is SINTEF's spline library. It is a mature library written in c. It handles NURBS curves and surfaces. It contains a number of methods to define geometry, but has its strength in operations related to curves and surfaces, in particular intersection functionality. SISL does not handle trimmed surfaces and has no topology structures.

GoTools is a collection of geometry libraries. Some of them provide well defined operations related to geometry, others are related to specific projects or applications. The division into modules or libraries are partly based on history and partly on conscious modularization.

GoTools does partly overlap with SISL. Both libraries contain creation of NURBS curves and surfaces and operations upon them, but the functionality is not exactly the same. GoTools offers in addition trimmed surfaces and topological structures. Moreover, GoTools has some functionality for operating on surface sets as one unity and for doing quality control of a CAD model. GoTools is written in C++.

SISL and parts of GoTools are made public as GNU GPL code. The libraries are documented, doxygen is used in GoTools. SISL is expected to work on most platforms. GoTools supports Linux and Windows with the Visual Studio compiler.

GoTools is under continuous development. Currently, SINTEF is, in addition to Exciting, involved in two Norwegian research projects in the field of isogeometry. This produces sideground software which is of interest for Exciting. The software will be added to the Isogeometric Toolbox, but the ownership remains at SINTEF.

GoTools modules that are relevant for the isogeometric toolbox are:

utils Independent classes for handling of points, bounding boxes, directional cones and similar aspects. Some methods that are independent of splines

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

ttl Triangulations

geometry Parametric curves and surfaces including construction methods and operations on these entities. Trimmed surfaces are included in the module.

topology Structures for computing the topology of a surface model. Some computations are expected to be performed by the application.

trivariate Spline volume including some creation methods

parametrization Parametrization of scattered data points. This library is dependent only on the basic modules.

implicitization Approximating spline curves and surfaces by implicit geometry.

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

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

steplib Step reader. The reader uses a commercial EXPRESS parser from EPM Technology, see *http://www.epmtech.jotne.com*. This functionality is only available on Windows.

compositemodel View a set of surfaces as one entity and perform operations on the model

model_toolbox Some functionality to modify a set of surfaces

qualitymodule Quality testing of CAD models

viewlib Viewer for curves and surfaces

surfjoin Graphical test interface related to model_toolbox

GUV Viewer frame work.

The viewers use openGl and Qt. Qt is developed by Trolltech, see *http://www.qtsoftware.com*. GUV does also use Coin from Kongsberg Sim, see *http://www.km.kongsberg.com/sim*. As already mentioned, GoTools has been collected during many previous or ongoing projects. This development history does not always imply the best organization of the code. Some restructuring of modules will be performed before GoTools is put into the Isogeometric Toolbox. The geometry library, as described in the list, is currently 4 libraries which will be merged into one.

The model_toolbox library will be extracted from a module implemented during the EU-project Fantastic. Documentation can be found in [1] and [2]. Parts of this module has already been taken out to create the library compositemodel. model_toolbox will perform operations on a set of surfaces that imply modifications in the surface set. Currently, the main operation in this library is merging of a set of surface patches into one large surface. This operation will be explained in some detail later in the document. Effort will be applied to make model_toolbox up to date with the other libraries in GoTools and for stabilization.

GoTools uses boost for shared pointers. Implicitization uses newmat which is a free software library. We have a policy to avoid such dependencies in the code, but it is not likely that this link will be removed before the Isogeometric Toolbox is initiated.

The libraries multivariate_splines and structured_blocks are template based and handles multivariate, tensor product, non-rational splines. These libraries will not be added to the toolbox unless a particular need for it arises.

The existing data structures of GoTools as well as Axel lie premises for the data structures of the Isogeometric Toolbox. New functionality will adapt to the existing data structures. In the remainder of this section, we will describe the main GoTools classes related to representation of geometry and topology.

2.1 Geometry Structures

Figure 1 gives a simplified view on the geometry classes. Additional classes exist, but are less relevant in this context. Curves, surfaces and volumes are all geometric objects. The classes ParamXXX provides an interface to parametric curves, surfaces and volumes. The concrete curves are most often of type SplineCurve or CurveOnSurface. The latter is related to BoundedSurface which is a boundary trimmed surface. CurveOnSurface knows the surface it belongs to and does most often have a representation both in the parameter domain of this surface and in geometry space. SplineSurface is a non-trimmed tensor product surface and may be rational or not. SplineVolume is a non-trimmed tensor product

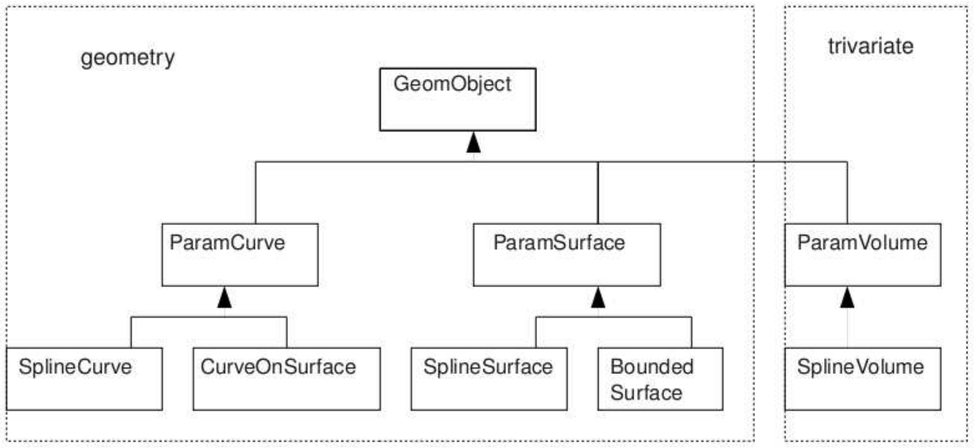


Figure 1: Simplified overview of the geometry class hierarchy

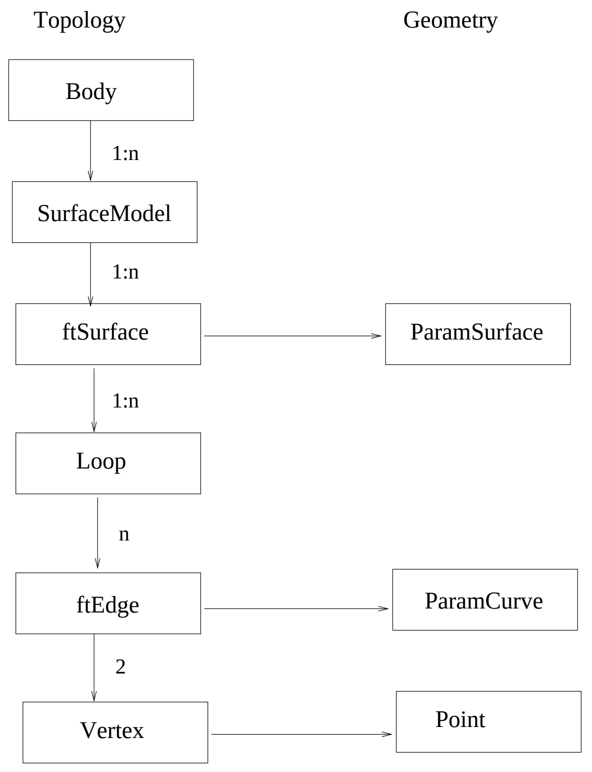


Figure 2: Overview of the topology classes related to surface models

volume. ParamVolume is currently redundant, but is included for improved flexibility in the future.

Isogeometric analysis is performed using one or more NURBS surfaces or volumes. Trimming is not applicable. However, initial data may contain trimmed surfaces as is the case for the container ship, see figure 11. Thus, it is important to be able to represent and handle trimmed surfaces.

The geometry classes is split between two modules where geometry contains curves and surfaces. The volume classes lie in a module called trivariate. trivariate is dependent on geometry, but not the other way around.

2.2 Topology Structures for Surface Models

Each geometry entity represents one curve, one surface or one volume. However, a model is seldom representable by one entity. Then the need for sets of entities, and analysis and representation of adjacency arises. There is a need for topological structures. Figure 2 shows the current topology classes in GoTools.

Body is a boundary represented solid model. It is a virtual volume limited by one or more shells, one outer and possibly a number of inner ones. The shells are represented by the class SurfaceModel which is also the entity used to represent a face set. In addition to being a topological entity, SurfaceModel provides an interface which views the set of surfaces as one entity. Thus, an operation like closest point computation can be performed without caring about which surface is closest to the point.

The surface model consists of a set of faces represented by the class ftSurface. The face represents the abstract idea of a bounded surface, but it also has a pointer to the geometric representation of this surface. The geometric surface is a ParamSurface. Currently, the ParamSurface is either of type SplineSurface or BoundedSurface. The face is bounded by one or more loops, one outer and possibly a number of inner ones. The inner loops represents trimming. In that case the geometric surface will be a BoundedSurface. All faces have an outer loop which either represents an outer trimming curve in the case of a BoundedSurface or the surface boundary for SplineSurfaces.

A loop consists of a number of edges represented by ftEdge. The edges have a geometric representation by a ParamCurve. This is the level where adjacency information related to faces or surfaces is stored. An ftEdge is a half edge. It knows the face to which it belongs and the geometry of the curve it corresponds to. This curve is normally a CurveOnSurface curve. Then, both parameter and geometry information are available. The ftEdges has a pointer to a twin edge. This edge belongs to the adjacent ftSurface meeting the current one along the boundary represented by the current ftEdge. This construction is a very flexible one, but it does not automatically ensure C^0 continuity as the curves pointed at by the two twin edges, are normally not the same.

An edge is limited by two vertices which has a geometric representation as a point. A vertex has knowledge about all edges meeting in this point.

2.3 Topology Structures for Volume Models

The existing topology structures will be extended to include volumes. These structures will be sideground software and developed in the Norwegian research projects on isogeometry. The following aspects have been considered:

- The topology structure should be easy to access
- Topology related to volumes should fall in line with already existing topology structures in GoTools
- The structure should support C^0 continuity between adjacent volumes
- Easy access to information about continuity between volumes
- Support for adaptive gridding
- Support for shape optimization
- High flexibility in the operation of building a volume representation of a model. The restriction imposed by requiring adjacent volumes to have common corners, is not wanted
- The structure should have room for context information, for instance the initial CAD surfaces that was involved in building a particular volume

We want to build on the existing GoTools topology structures. This implies that the structure lies closer to a CAD representation than to a FEM representation. The suggestion is as follows:

- Create a class ftVolume. ftVolume points to a ParamVolume which for all practical purposes is equal to SplineVolume, but the distinction adds room for some extra flexibility.

- ftVolume is surrounded by a set of ftSurfaces. A SplineVolume has 6 boundary surfaces, but the number of ftSurfaces corresponding to these surfaces may be higher depending on whether or not adjacent volumes meet corner-to-corner. In contrast to the ftSurface/ftEdge relationship, there is no natural sequencing of boundary elements in this case.

- ftSurface is equipped with a pointer to a ftVolume or to some super class of ftVolume. In non-volumetric applications, this pointer will have no content.

- ftSurface gets a twin pointer to the ftSurface of an adjacent volume at the common boundary. Similar to the previous case, this pointer do not need to point at anything. Moreover, if the current volume lies at the boundary of the complete set, the twin pointer points to null.

The classes ftSurface which corresponds to a face and ftEdge which is a half edge, have got the prefix ft for historical reasons. The ftVolume gets its name to fit in with this pattern.

The suggested solution has both good and bad features. The good ones are:

- The structure is flexible

- It is compatible with ftSurface, ftEdge, etc.

- Reuse of code reduces effort and risk

- Can be extended, for instance to handle trimmed volumes

There is also a few drawbacks:

- There is no explicit C^0 continuity between adjacent volumes. Test functionality for continuity must be implemented. Also functionality to enforce C^0 continuity can be relevant.

- The topology structure is relatively complex. It is necessary to provide a simple interface for the cases where the complete flexibility is not needed.

We will consider ftSurface while defining the functionality of ftVolume. However, ftSurface has got a rich set of functionality. Some of it is concerned with representation of adjacency and adjacency analysis. Other functions fall in the classes of quality testing and surface repair. Not all of the ftSurface functionality have a natural counterpart in ftVolume. ftVolume needs functionality related to adjacency with neighboring volumes. This includes information about linear side constraints involving coefficients of volumes on both sides of a common boundary surface to ensure or maintain a certain continuity during computations. ftVolume must also be able to fetch the corresponding ParamVolume or SplineVolume.

3 The Geometric Modeler Axel

Axel aims at providing a unified framework for both the visualization and manipulation of geometric objects with semi-algebraic representation, manipulating exact data to provide certified results or computing with approximations in a robust way.

The application is designed to be either used stand alone or connected together with external tools and therefore become a graphical frontend. Its modular architecture using plugins makes easy the wrapping of external libraries or the connection with other tools. As

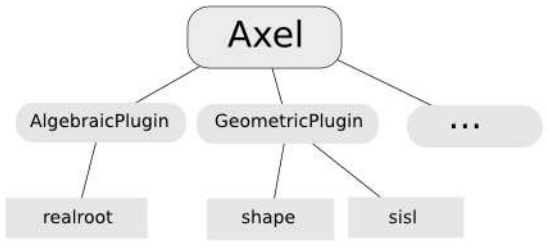


Figure 3: Axel's internal architecture

shown in figure 3, Axel is organized as a hierarchy of layers. At the topmost level, the user interacts with the application using files, scripts or the graphical interface, implemented in two distinct layers. Objects and tools are defined within the virtual layer which consists in hierarchies of virtual classes which feature a selection mechanism and allow to interface functionalities that can be found in bottommost layers of Axel, a set of plugins. Some plugins are mandatory for the application to provide its main functionalities, these are the *geometric kernel* and the *algebraic kernel*. The first one provides an interface between geometric objects manipulated in Axel and their definition in the supporting libraries. The second one is used to furnish additional tools on the algebraic data structures used for the representation of objects.

3.1 The generic framework

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 always chosen at run-time, providing effectiveness and ease of coding. We describe the internal layer of Axel which allows such a combination.

Geometric as well as algebraic objects can be organized within different yet very dense taxonomies. Pragmatically, this taxonomy of objects ends up in a tree data structure in which internal nodes correspond to abstract classes. A parent child relationship in the

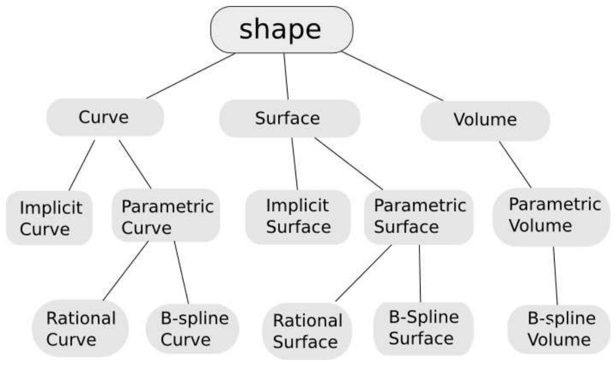


Figure 4: Virtual hierarchy of objects

tree corresponds to an inheritance relationship. The overall structure is called a *virtual hierarchy*.

A leaf node corresponds to the most specific way to *see* an object, and the path from the root to the leaf in this virtual hierarchy of objects contains all possible ways to *see* this object from the most generic to the most specific.

At each level of this path, the object gains new attributes, which makes possible the definition of new functions or the specialization (sometimes referred to as *overloading* in object programming) of existing functions.

Besides the fact that a taxonomy is a very natural concept corresponding to the reality, there are two main interests of organizing objects such a way. First, this hierarchy permits to dispatch algorithms, to choose an implementation that is the most adapted to the problem in hand. Second, this hierarchy allows a developer to "hang" its own new data type to any class of the hierarchy by inheritance and therefore benefit from all algorithms and functionalities defined at upper levels of the hierarchy.

The selection mechanism can be achieved by several means. Indeed, many languages such as C++ come with their own selection mechanism (using the `virtual` keyword in this case). Otherwise, it is possible to use dedicated design patterns such as the Concept-View-Interface pattern. In this pattern, the virtual hierarchy is built using a template wrapper to link an interface to its implementation. The selection mechanism is achieved by a combination of overloading and the use of namespaces in which global functions are defined to work on the representation of an object.

Figure 4 shows a hierarchy of geometric objects similar to the one found in Axel.

Geometric tools (or algorithms), can also be classified in a *virtual hierarchy of tools*, closely related to the *virtual hierarchy of objects*. The selection mechanism, usually used with functionalities of an object (a method of a class in object programming), works just the same on the hierarchy of tools, or algorithms.

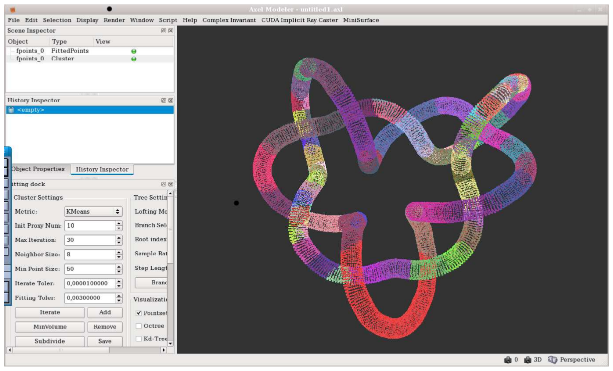
These hierarchies are used to connect the views on the geometric objects to the action to be performed on them. This former computation is provided by external libraries following a plugin mechanism. To embed dedicated plugins, we have to take care of techniques to link these hierarchies together with functionalities of an external application. In section 3.3, we describe the method that we adopt to build dynamic modules, combining generic and special code through a transparent mechanism.

3.2 The Geometric primitives in Axel

Among elementary geometric entities such as points, planes *etc.* Axel aims at providing an interface for the visualization and manipulation of points, curves, surfaces and volumes with various representations: implicit, parametric or piecewise linear.

3.2.1 Points

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. See Figure 5.


Axel Modeler - untitled1.axl
File Edit Selection Display Render Window Script Help Complex Invariant CUDA Implicit Ray Caster MiniSurface
Scene Inspector
History Inspector
Object Properties
History Inspector
Iterate
Add
MinVolume
Remove
Subdivide
Save
3D
Perspective

Object	Type	View
fpoints_0	FittedPoints	
fpoints_0	Cluster	

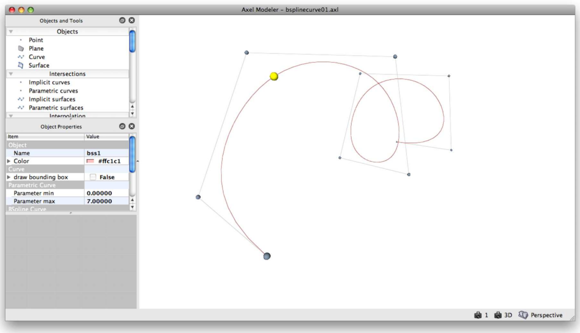
<empty>

Cluster Settings

Metric:	KMeans
Init Proxy Num:	10
Max Iteration:	30
Neighbor Size:	8
Min Point Size:	50
Iterate Toler:	0,0000100000
Fitting Toler:	0,00300000

✓ Pointset
Octree
Kd-Tree

Figure 5: Point set with color information.



Item	Value
Object	
Name	bss1
Color	#ffc1c1
Curve	
draw bounding box	False
Parametric Curve	
Parameter min	0.00000
Parameter max	7.00000
BSpline Curve	

Figure 6: Editing a B-spline curve in Axel

3.2.2 Curves

Curves in two or three dimensions are widely used in computational mathematics and geometric modeling either to build surfaces or to model surface sections. They are the main ingredient of technical drawing software.

Polygonal curves. They are represented by a set of points and a set of segments connecting these points.

Parametric curves. Parametric curves 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 $[a, b] \subset R$ where x, y, z, w are polynomials or B-spline functions. Figure 6 shows a curve, which definition can be found in the Axel data file shown in section 3.4. 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.

The package used to display and edit the B-spline curves is currently the library `sisl` and will be replaced by the toolbox developed in the WP6.

Implicit curves. An algebraic planar implicit curve is represented by a polynomial $f(x, y)$ in two variables with coefficients in Q. See Figure 7. An algebraic space curve is given two (or more) polynomials in three variables with real coefficients $f(x, y, z)$ and

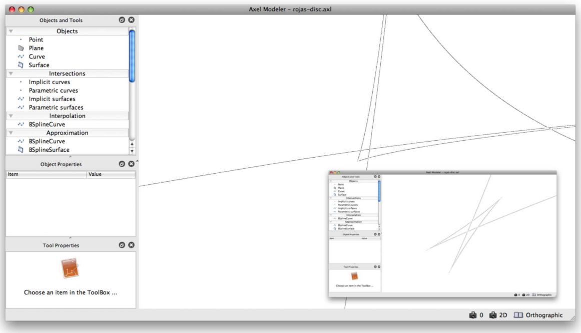


Figure 7: Topology of an implicit curve of degree 47 with hidden cusp points.

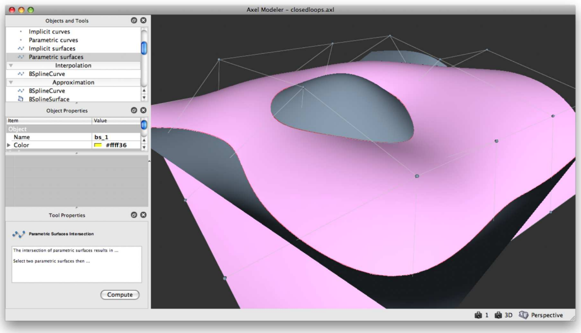


Figure 8: Computing an intersection of generic parametric surfaces in Axel.

$g(x, y, z)$, with $\gcd(f, g) = 1$. Its visualization also requires tools for solving polynomial equations and topology algorithms.

We use dedicated tools from the plugin `shape` and `realroot`, for computing the topology of these implicit curves, in particular for detecting and correctly handling singularities.

3.2.3 Surfaces

Surfaces are nearly ubiquitous for Computer-Aided Design (CAD), Manufacturing (CAM), and Engineering (CAE). In Axel, they are classified in three main families: piecewise linear, parametric and implicit surfaces.

Meshed surfaces They are represented as a set of points and a set of faces, which are triangles or more general polygons.

Parametric surfaces. As in the curve case, there 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 $[a, b] \times [c, d] \subset \mathrm{R}^2$ where x, y, z and w are either polynomials or bivariate B-spline functions. See Figure 8. The package used to display and edit the B-spline surfaces is currently the library `sisl` and will be replaced by the toolbox developed in the WP6.

Implicit surfaces Implicit surfaces (or algebraic surfaces), are the real zero set of a given polynomial in three variables with real coefficients $p(x, y, z)$. The surface is not assumed

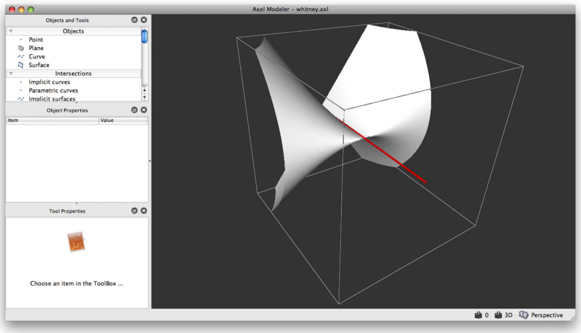
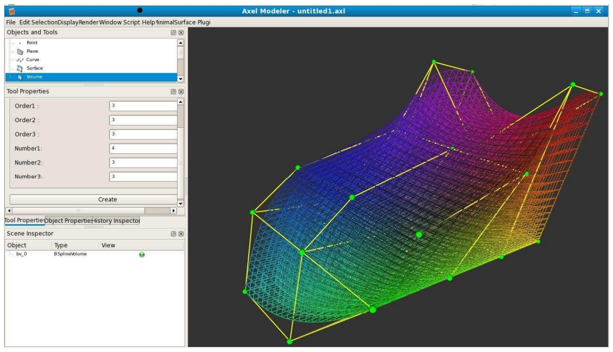


Figure 9: An implicit surface dynamically displayed along with its singular locus.


Axel Modeler - untitled1.axl
File Edit Selection Display Render Window Script Help MinimalSurface Plugi
Objects and Tools
Point
Plane
Curve
Surface
Volume
Tool Properties
Create
Tool Properties Object Properties History Inspector
Scene Inspector

Order1 : 3
Order2 : 3
Order3 : 3
Number1: 4
Number2: 3
Number3: 3

Object	Type	View
bv_0	BSplineVolume	

Figure 10: Displaying and editing a B-spline volume.

to be smooth, and we use a dedicated subdivision topology algorithms, to dynamically displays it. See Whitney umbrella in figure 9. This computation is also using the package `realroot` and `shape`.

3.2.4 Volumes

The volumes 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 $[a,b]\times[c,d]\times[c,d] \subset \mathrm{R}^3$ where x, y, z and w are trivariate B-spline functions. See Figure 10. The package used to display and edit the B-spline volumes is provided by a preliminary version of the toolbox to be developed in WP6.

3.3 The plugin system

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. Current there are two such plugins which are called the geometric kernel and the algebraic kernel.

The geometric kernel contains all advanced functionalities that are related to the geometric operations. Each time an object is created in Axel, the geometric kernel maintains a map between the geometric abstraction of the object in the application and a virtual interface to the object in the geometric kernel. The computation is performed by the external packages `shape` for algebraic objects or by `sisl` for B-spline curves or surfaces. They are glued together in the geometric kernel.

The algebraic kernel provides the lowest level abstraction to handle the representation of geometric objects. The algebraic kernel contains all advanced functionalities that are related to the algebraic representation of geometric objects found in Axel, based on the package `realroot`. This includes the polynomial solvers and algebraic operations on the data representing the objects.

3.4 Interface

Axel proposes three main types of interaction: its graphical user interface, a data file formalism and an interactive scripting environment.

Graphical user interface The modeler features a main viewer which comes with a large amount of mouse and keyboard bindings. It allows an efficient navigation in the 3D space without populating the screen with useless views. For example, it is possible to align the view with the axis, to zoom on a given region, to set key positions for the camera and even to run animations.

In the viewer, each object is associated to a *frame*, a three dimensional equivalent of *layers* that can be found in image manipulation programs. The object can be manipulated using usual translation, rotation and scaling operations either relatively to its own frame which defines a local coordinate system, or, relatively to scene, corresponding to a global coordinate system. Frames can also be organized within a hierarchy, providing parent-child relationships.

The rest of the graphical user interface can be dynamically arranged as a set of collaborating docks. For example, when an object or a tool is invoked in the main *Objects and Tools Dock*, the corresponding *Object Properties Dock* and *Tool Properties Dock* are dynamically updated to provide the user with an additional set of settings.

This is through the object properties dock that the user can access the algebraic representation of object, eventually changing some elements of it directly within the widget. This dock is also enriched with many intuitive widgets, such as, a view of the parameter space associated to any parametric curve or surface, which can be queried to dynamically display the image of the set of parameters picked by the user within the widget.

A *Scene Inspector Dock* allows to keep track of objects present in the current scene and to eventually hide them in order to focus on a part of interest, and an *History Dock* permits to keep track of the incremental construction of an algebraic model.

File interface Axel comes with a data file formalism which allows the user to specify a set of objects or to save and share a model. The formalism is a subset of the extended markup language (XML) which makes it easily readable and writable. The following example illustrates how to specify the B-spline curve shown in figure 6.

```
<axl>
   <curve type="bspline" name="bsc1">
      <dimension>3</dimension>
      <number>10</number>
      <order>4</order>
      <knots>0 0 0 0 1 2 3 4 5 6 7 7 7 7</knots>
      <points>
         0 0 0 1 0 0.5 1 1 1 0 1 1.5
         0 0 2 1 0 2.5 1 1 3 0 1 3.5
         0 0 4 1 0 4.5
      </points>
   </curve>
</axl>
```

The formalism also permits to embed scripts that can be executed when the file has been parsed by the application or to save information related to the viewer state or the camera position, field of view *etc.*

Axel also allows the user to parse data file formats which are heavily used in many research fields such as `OFF`, `PLY` *etc.*

Script interface Axel comes with an embedded script interpreter, which allows to dynamically interact with the application using Axel Application Programming Interface (API) to access the program's internals and greatly expand Axel's functionalities.

The scripting language is ECMAScript which has been enriched with objects which allow an interaction with the main application, the viewer or different managers that come with Axel such as the object manager or the tool manager.

A more advanced extension language may be used in the future to guide the geometric computation to be done in the modeler.

4 The Relation between the Isogeometry Toolbox and the other Work Packages, Test Cases

The activities in the work packages 1 to 5 are the main source of premises for the Isogeometric Toolbox. Thus, the test cases used in these work packages are very important for the design of this toolbox.

4.1 Work Package 1

This work package handles propellers and turbochargers. Currently, a number of different geometry representations of a propeller is used for CFD analysis, stress analysis, NC tool patch generation etc. Most of these representation will be replaced by one single geometric model.

The propeller surfaces will be modelled by approximating point sets with one or more surfaces, see section 6.7 and appendix A for methods in GoTools covering this functionality. The current version of GoTools should be satisfactory for this purpose.

Stress analysis is performed using a volume mesh generated from a NURBS Brep model. The surface model generated by point approximation will we well suited as a basis for mesh generation. Alternatively, a volume model of the propeller can be generated from methods in GoTools. Currently the module trivariate contains a limited set of functionality for construction of one volume from surfaces. The set of volume construction methods in GoTools is planned to be extended, see section 6.5.

4.2 Work Package 2

In WP2, the aim is to perform optimization of ship hulls with respect to a number of design parameters.

The test cases are two hull forms provided by DNV. Both are container ships. One is MS Exciting which is shown in figure 11. It consists of 309 surfaces. Most surfaces are simple non-rational B-spline surfaces, but there are also trimmed surfaces in the data set. The body lines of both ships will be provided.

The design parameters are associated with global dimensions of the ship, the bow, the midsection and the stern. Examples of parameters are length, breadth, radius of bilge, bow radius and transom slope. All parameters, restrictions on the parameters and more information about the ships can be found in appendix B.

Linking these parameters with the actual ship description is the process of parameterization. This is a task of WP2. WP6 should ease the process by offering tools. It might be feasible to simplify the ship description by using fewer, but larger surfaces. The model_toolbox module has functionality for this, but it need some improvements and stabilization. More information can be found in section 6.9 and 6.10.

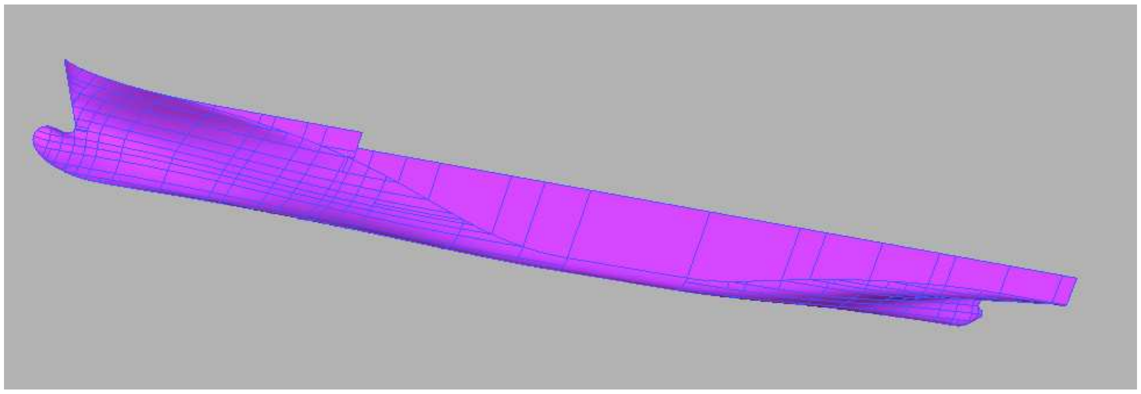


Figure 11: The container ship MS Exciting

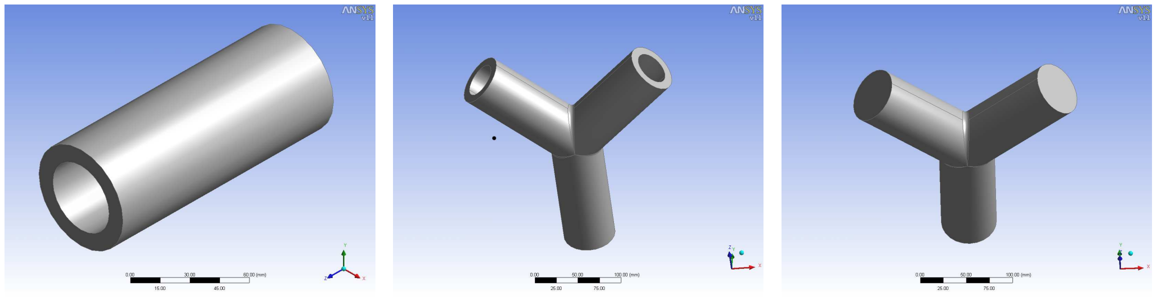


Figure 12: Different combinations of tubes and cylinders

Some parameters might be more easily associated with characteristic curves in the ship hull than the actual surface description. Optimization of the parameters may imply a modification of a set of characteristic curves which again is the source of updating the hull surfaces. Such a procedure indicates heavy use of curve and surface fitting, functionality GoTools is well equipped with, see section 6.7. See section 6.11 for considerations on modifications of surface sets.

4.3 Work Package 3

WP3 addresses isogeometric analysis in the car industry. Figure 12 and 13 shows relevant models for this work package. In appendix C, Vuong and Simeon shortly describe the work to take place in WP3 and the relation between WP3 and the Isogeometric Toolbox.

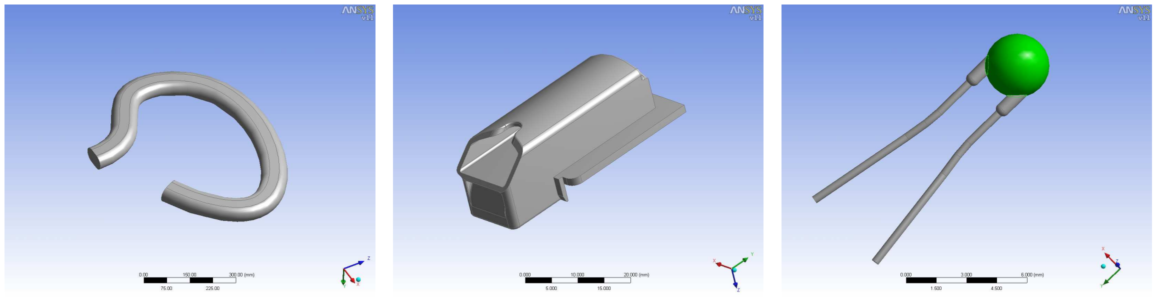


Figure 13: More CAD models

Functionality of the Toolbox needed from WP3 is the following:

- Simple geometry generation. Possible methods can be found in section 6.5.
- Evaluation of NURBS/B-Splines and derivatives in three dimensions. Section 6.1 is related to this requirement.
- Knot insertion for NURBS volumes. This functionality is shortly described in 6.2.
- Degree elevation for NURBS volumes, see 6.2.
- Visualization of NURBS volumes. This is a task either for AXEL or the GoTools viewers GUV or goview. See also section 6.12.
- Visualization of a scalar or vector field on a NURBS volume. Also the field is represented by NURBS. AXEL has got functionality in this direction.
- Generation of a volume mesh from its surfaces (or a standard CAD format)

The requirements listed above fall into three categories. The three first points are extensions of normal NURBS functionality to trivariate volumes. The next two deal with visualization. The last point in the list is concerned with conversion from a standard boundary represented CAD model to a multi-block NURBS volume model fit for isogeometric analysis. The last point is by far the most challenging one. A discussion on this topic can be found in section 7.

4.4 Work Package 4

WP4 works with fluid-structure interaction. Two different physical fields have contact at a common boundary. This boundary, or interface, will be updated during the process. WP4 depends on solvers which are developed in WP1 and WP3 and will use these solvers in iterative steps updating the interface at each step. More information about the process can be found in appendix D.

The test cases which are proposed in the appendix are standard benchmark examples in the field. They consist of two 2D and one 3D example. The geometry itself has limited complexity, but the cooperation between the two solvers adds to the complexity of the problem. The two solvers will not necessarily need the same degree of refinement in the spline space in which the geometry and the field are described. Refinement of a spline space and data reduction are relevant methods, see the sections 6.2 and 6.8.

This work package has particular challenges related to information transfer since two solvers will be used. Prior to delivery 6.2, an action between the work packages is required to define the need for file formats and other formats related to this information transfer.

4.5 Work Package 5

WP5 will do design optimization and is tightly linked to WP2. The test cases are the same as for that work package. One ingredient in the optimization process is a parameterization tool which represents complex shapes with a few number of design parameters. This tool is the responsibility of WP2 and WP5, but the Isogeometric Toolbox should support it.

Hierarchical definitions of geometry and physical solutions imply the possibility to use hierarchical algorithms in the optimization which again reduces computational costs. Thus, hierarchical descriptions are highly desirable. However, it is currently not covered in the plans for WP6.

5 B-spline Basics

Before going into details on the toolbox methods, we will give a summary of the B-spline format and some B-spline properties. A polynomial B-spline curve is a piecewise polynomial curve represented with B-spline functions as basis. This basis has some properties that ensure numerical stability, i.e. the basis functions are always non-negative, and at a given parameter all non-zero basis function sum up to one. The basis functions have local support, and thus moving one coefficient of a curve changes the curve only locally. The joints between the polynomial pieces of a spline curve are defined by a knot vector. Normally the degree of continuity of a curve is equal to the polynomial degree minus one, but multiplicity of knots can be used to decrease the continuity of the curve in the joints. A polynomial B-spline curve, f, is described as follows:

$$f(t) = \sum_{i=1}^{n} c_i B_{i,k,\tau}(t)$$

where $\tau = \{\tau_1, \tau_2, ..., \tau_{n+k+1}\}$ is the knot vector of the curve. $\tau_i \leq \tau_{i+1}$. k is the polynomial degree of the curve. $t \in [\tau_{k+1}, \tau_{n+1}]$ and $\{B_{i,k,\tau}\}_{i=1}^{n}$ are the basis functions of the curve. $\{c_i\}_{i=1}^{n}$, $c_i \in R^d$ where d is the dimension of the geometry space, are the coefficients of the curve.

A rational B-spline curve is a spline curve with rational components. This curve can be written as

$$g(t) = \frac{\sum_{i=1}^{n} c_i h_i B_{i,k,\tau}(t)}{\sum_{i=1}^{n} h_i B_{i,k,\tau}(t)}, \quad h_i \in R_+, \quad i = 1, ..., n$$

The curve can alternatively be written as $g(t) = \sum_{i=1}^{n} c_i R_{i,k,\tau}(t)$ where $R_{i,k,\tau}(t) = \frac{h_i B_{i,k,\tau}(t)}{\sum_{i=1}^{n} h_i B_{i,k,\tau}(t)}$. This basis function possesses many of the same properties as the B-spline basis. It has a local support, it is always non-negative, and the basis functions sum up to unity in every parameter within the parameter interval. A rational B-spline curve has many of the same operations as a polynomial B-spline curve, but some operations will involve more work, for instance differentiation. Rational splines does not in general have the same numerical stability as polynomial ones, but keeping the size of the denominator well above zero and avoiding a large variation in the size, the stability is good.

Let $F(u, v)$ be a tensor product B-spline surface, $u \in [u_0, u_N]$ and $v \in [v_0, v_M]$. The surface has degree k_1 in the first parameter direction and k_2 in the second, and is defined on the knot vectors τ_1 and τ_2.

The polynomial B-spline surface is given by

$$G(u, v) = \sum_{i=1}^{n} \sum_{j=1}^{m} c_{i,j} B_{i,k_1,\tau_1}(u) B_{j,k_2,\tau_2}(v)$$

A rational B-spline surface is not a tensor-product. This can be seen from the expression of the surface.

$$S(u,v) = \frac{\sum_{i=1}^{n}\sum_{j=1}^{m} h_{i,j}c_{i,j}B_{i,k_1}(u)\,B_{j,k_2}(v)}{\sum_{i=1}^{n}\sum_{j=1}^{m} h_{i,j}B_{i,k_1}(u)B_{j,k_2}(v)} = \sum_{i=1}^{n}\sum_{j=1}^{m} c_{i,j}R_{ij}(u,v)\,,$$

$$c_{i,j} \in R^d, \quad h_{i,j} \in R_+$$

Here

$$R_i(u,v) = \frac{h_{i,j}B_{i,k_1}(u)\,B_{j,k_2}(v)}{\sum_{i=1}^{n}\sum_{j=1}^{m} h_{i,j}B_{i,k_1}(u)B_{j,k_2}(v)}$$

We see that the basis functions are not separated in the parameter directions of the surface. This fact makes for instance smoothing of a rational surface using a variational approach much more time consuming than for a polynomial B-spline surface, see section 6.7. It may also have influence on the performance of the numerical integration related to the Finite Element Method when this is applied to spline elements.

The weights give extra flexibility for a rational curve or surface compared to a polynomial one. The traditional use of this flexibility is to be able to represent conic arcs and surfaces exactly. The weights could also give large degrees of freedom in a design or modification process, but using the weights of a rational curve or surface as design parameter typically lead to a non-linear optimization problem.

NURBS curves and surface belong to a well known industry standard that possesses a rich set of methods. They are covered by standards for data transfer, and due to the properties of the basis functions they are numerically stable.

A NURBS surface is 4-sided. This implies that it is not always convenient to fill a mesh with tensor product B-spline surfaces. These surfaces are not well suited for irregular meshes. Degenerate surfaces may be defined to fill irregular areas in a mesh. Possible degeneracies are:

- Edge collapse, i.e. one edge collapses into a point.
- Parallel (or anti-parallel) derivatives in a corner of a surface.

Degeneracies lead to a vanishing normal in the degenerate point. The surface can still have a consistent tangent plane.

The local control makes B-spline curves and surfaces well suited for modification. Parts of a surface may be altered without changing the complete surface, and only a few coefficients influence each polynomial area of the surface. The spline properties ensure continuity between the polygonal pieces even if some coefficients are changed.

A non-rational trivariate spline volume is represented as follows:

$$V(u,v,w) = \sum_{i=1}^{n_1}\sum_{j=1}^{n_2}\sum_{k=1}^{n_3} c_{i,j,k}B_{i,o_1,\tau_1}(u)B_{j,o_2,\tau_2}(v)B_{k,o_3,\tau_3}(w)$$

where $c_{i,j,k}$ are volume coefficients, $c_{i,j,k} \in R^3$ and $B_{i,o_1,\tau_1}(u)$, $i = 1,\cdots n_1$ is basis of the spline space in the first parameter direction. The space has degree o_1 and knot vector τ_1. The spline spaces in the other two parameter are defined similarly.

A rational volume is represented by

$$V(u,v,w) = \frac{\sum_{i=1}^{n_1}\sum_{j=1}^{n_2}\sum_{k=1}^{n_3} h_{i,j,k}c_{i,j,k}B_{i,o_1,\tau_1}(u)B_{j,o_2,\tau_2}(v)B_{k,o_3,\tau_3}(w)}{\sum_{i=1}^{n_1}\sum_{j=1}^{n_2}\sum_{k=1}^{n_3} h_{i,j,k}B_{i,o_1,\tau_1}(u)B_{j,o_2,\tau_2}(v)B_{k,o_3,\tau_3}}$$

The weights $h_{i,j,k} \in R$ are always positive.

A NURBS volume is block like. This leads to the same concerns on flexibility of shape as for surfaces. Non-trivial weights lead to extra flexibility and more complex computations similar to the case for curves and surfaces.

6 The Isogeometric Toolbox

The Isogeometric Toolbox will to a large extent be based on background software from the partners, in particular SINTEF and INRIA, and sideground software, i.e. software developed in parallel isogeometric projects. The heavy use of background software puts clear premises on the language, C++, and the data structures used in the toolbox.

The isogeometric toolbox work will be in the following areas

- Add background and sideground software to the toolbox
- Stabilize and extend existing software
- Integrate software from other partners into the toolbox
- Develop new software

New software will to a large extent have to be developed in the other work packages. It can be added to the toolbox if appropriate. New development directly into the toolbox must be governed by the need of the other parts of the project. The proposed test cases give some indication on the particular functionality to prioritize.

The toolbox will contain functionality for specific tasks in the isogeometry analysis process. The structuring will be based on the structure of the background software. However, specific needs or especially contributions from the other work packages may imply a need to consider some restructuring. We aim for GNU GPL open source code.

Basic routines for solving linear systems will be available. They can be based on existing library such as Lapack library or provided by the toolbox to ensure the independence of the software.

Tools to produce automatically plugins for Axel will be provided.

A svn project has been opened for the toolbox at

`http://gforge.inria.fr/projects/isogeometry/`

The remainder of this section will give some details on the toolbox methods. The presentation is based on GoTools/SISL and Axel. The list is not complete. New knowledge of the needs will occur during the project and obvious B-spline functionality is not described.

6.1 Grid Evaluation

The value of a NURBS surface in a given parameter pair (u_0, v_0) is

$$S(u_0, v_0) = \frac{\sum_{i=1}^{n} \sum_{j=1}^{m} h_{i,j} c_{i,j} B_{i,k_1}(u_0)\, B_{j,k_2}(v_0)}{\sum_{i=1}^{n} \sum_{j=1}^{m} h_{i,j} B_{i,k_1}(u_0) B_{j,k_2}(v_0)}$$

We see that if we choose to evaluate the surface in a number of parameter values (u_0, v_r) where $r = 0, \ldots, R$, the contribution to the result from the spline basis in the u-direction is always the same. This property is taken advantage of in the grid evaluator for surfaces and volumes. All evaluations are done simultaneously and the value and eventual derivatives of the basis functions in one parameter value is evaluated only once.

GoTools has got a number of grid evaluators. The functionality applies to both surfaces and volumes.

- Compute position in a regular grid.
- Compute position and 1. derivative in a regular grid.
- Compute position in a grid specified by parameter values in the parameter directions.
- Compute position and 1. derivative in a grid specified by parameter values in the parameter directions.
- Compute the value of all basis functions in a grid specified by parameter values in all parameter directions. In the rational case, the weights are included in the basis functions.
- Compute the value and 1. derivatives of all basis functions in a grid specified by parameter values in all parameter directions. In the rational case, the weights are included in the basis functions.

The grid evaluators for B-spline basis functions at specified parameter values are appropriate in numerical integration where the functions are evaluated in the Gauss quadrature points.

6.2 Refinement and Degree Elevation

Refinement is a standard spline functionality that is crucial for isogeometric analysis. A number of algorithms exists, the best known are

- The Oslo-algorithm where many new knots are inserted simultaneously
- Böhms algorithm which inserts one single knot

Let the knot vector t be a refinement of the knot vector τ. Then the spline space spanned by τ is a subset of the spline space spanned by t and the basis functions related to the larger space can be expressed by the basis functions of the smaller space as

$$B_{j,k,\tau}(u) = \sum_{i=1}^{m} \alpha_{j,k}(i) B_{j,k,t}(x), \quad j = 1, \ldots, n$$

$\alpha_{j,k}(i)$ are the discrete B-splines corresponding to the two spline spaces. They are computed recursively using an algorithm similar to the one used for evaluating B-spline basis functions.

The possibility for degree elevation is also a fundamental property of spline spaces that are important in this context. Also in this case, the new coefficients are expressed as combinations of the old ones.

Common to refinement and degree elevation is that we increases the degrees of freedom related to a curve, surface or volume without altering the geometric object at all. If subsequent operations change the coefficients corresponding to the basis functions, the object will change, but not by applying refinement or degree elevation.

6.3 Numerical Integration

The solution field of a PDE lives on a geometric entity consisting of curves, surfaces or volumes. In the isogeometric setting both the geometric entity and the solution field is represented in the same spline space. In order to solve the PDE, one builds a linear equation system where the elements in the matrix consists of integrals of expressions involving derivatives of the basis functions of the solution space. The domain for the integral is the geometric entity on which the solution lives. In order to integrate over the parameter domain of this geometric entity, we have to include the Jacobi matrix of the geometry description into the expression of the integral.

The expression to be integrated may be polynomial if the geometric entity is described by non-rational B-splines and the geometry is simple. In most cases, this will not be the case.

Integration is performed using Gauss quadrature. The function is evaluated in specific points lying in a grid and assigned pre-defined weights. This method reproduces polynomials up to a certain degree depending on the number of evaluations. The function is build from derivatives of the basis functions and the Jacobi matrix. For both contributions, the use of grid evaluators will improve the performance drastically.

6.4 Parameterization of Computational Domains

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 non-trimmed CAD object(such as B-spline curve or B-spline surface) is given, we need to construct the same CAD representation of the whole computational domain that surrounds the CAD object. A discussion on how to determine the surrounding objects can be found in section 7. For a 2D problem(the CAD object consists of B-spline curves), the computational domain is represented as a 2D B-spline surface on a plane; for a 3D problem (the CAD object consists of B-spline surfaces), the corresponding computational domain is represented as a 3D B-spline volume. Hence, how to construct the computational domains(2D B-spline surface or 3D B-spline volume) from the given CAD objects(two boundary B-spline curves or two boundary B-spline surfaces), is a key issue for isogeometric analysis.

- Determination of the inner control points of B-spline surfaces from two given boundary curves. We have proposed a method to produce them based on the minimization of linear square mean curvature energy. See Fig.14. The red curves and points are

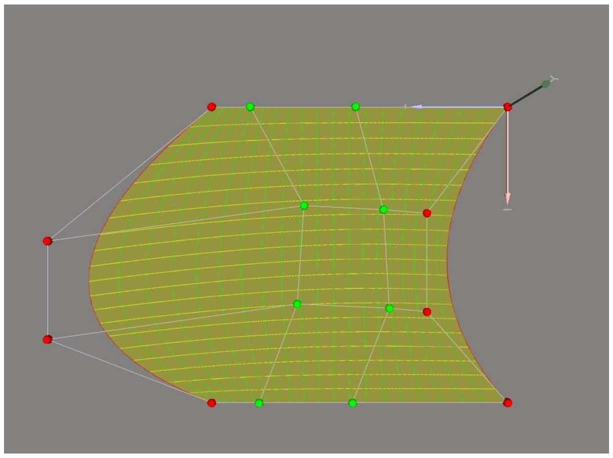


Figure 14: Generating a harmonic-like patch (green control points) from boundary curves (red control points)

the two given boundary B-spline curves and corresponding control points, the green points are the inner control points constructed by our method, the yellow curves and the green curves are the isoparametric curves on the resulted B-spline surface. How to find a better method to produce reasonable 2D computational domain is planned in our future work.

- Determination of the interior control points of B-spline volume for two given boundary surfaces. The key problem is how to find a measurement to optimize the interior control points.
- Determination of other control points of B-spline volume for given twelve boundary curves. This is a similar version with two given boundary surfaces, but it provides more freedom variables and is friendly to the users.
- Generate the B-spline volume from the closed 3D-boundary surfaces. This problem also leads to a problem of fitting(approximating) 3D boundary object with B-spline volumes. Offset method maybe a good candidate, but how to avoid the self-intersection is an open problem.

6.5 Volume Creation

The GoTools class SplineVolume has the following methods to create volumes from curves and surfaces:

- Linear sweep, i.e. sweep a surface along a curve
- Rotational sweep, sweep a surface along a circle to create a rotational volume. This is a specialization of linear sweep.
- Lofting, i.e. interpolate a set of non-rational surfaces with a volume
- Set the spline coefficients directly for simple volumes

A linear swept volume is constructed as follows. Let

$$S(u,v) = \frac{\sum_{i=1}^{n_1}\sum_{j=1}^{n_2} h_{i,j} c_{i,j} B_{i,o_1,\tau_1}(u) B_{j,o_2,\tau_2}(v)}{\sum_{i=1}^{n_1}\sum_{j=1}^{n_2} h_{i,j} B_{i,o_1,\tau_1}(u) B_{j,o_2,\tau_2}(v)}$$

be the surface that is to be swept along the curve

$$f(w) = \frac{\sum_{k=1}^{n_3} g_k d_k B_{k,o_3,\tau_3}(w)}{\sum_{k=1}^{n_3} g_k B_{k,o_3,\tau_3}(w)}$$

with respect to the point P. Then the formula for the volume is

$$V(u,v,w) = \frac{\sum_{i=1}^{n_1}\sum_{j=1}^{n_2}\sum_{k=1}^{n_3} h_{i,j} g_k (c_{i,j} + d_k - P) B_{i,o_1,\tau_1}(u) B_{j,o_2,\tau_2}(v) B_{k,o_3,\tau_3}(w)}{\sum_{i=1}^{n_1}\sum_{j=1}^{n_2}\sum_{k=1}^{n_3} h_{i,j} g_k B_{i,o_1,\tau_1}(u) B_{j,o_2,\tau_2}(v) B_{k,o_3,\tau_3}}$$

$h_{i,j}$ where $i = 1, \cdots n_1$ and $j = 1, \cdots n_2$.

Planned new functionality is

- Make a volume interpolating its boundary surfaces using a Coons patch approach
- Sweep a profile surface along a spine curve.
- Smoothing of an already defined spline volume

The swept volume is defined by:

$$V(u,v,w) = f(w) + \tilde{S}_1(u,v)t(w) + \tilde{S}_2(u,v)n(w) + \tilde{S}_3(u,v)b(w)$$

where $S(u,v)$ is the profile surface and $f(w)$ is the spine curve. $(\tilde{S}(u,v) = (\tilde{S}_1(u,v), \tilde{S}_2(u,v), \tilde{S}_3(u,v))$ is the surface S described in the coordinate system of $(t(w), n(w), b(w))$ in the start point of the curve f. This curve triple is the approximation of the Frenet frame curves corresponding to the profile curve f described in a spline space that gives appropriate accuracy. Special treatment is required in inflection points as the second derivative of f vanishes in these points.

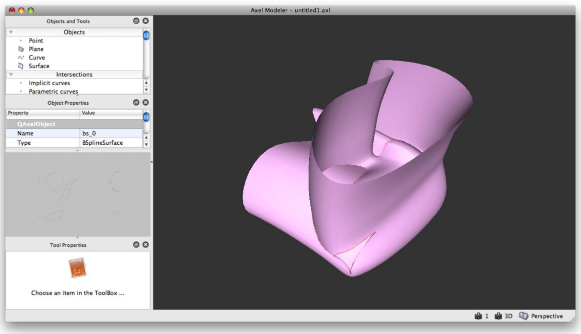
Volume smoothing is a generalization of surface smoothing.

6.6 Intersection, Self-intersection, Topology and Arrangements

To assemble different pieces needed in a whole simulation, several ingredients related to geometric modeling might be necessary. In particular, tools to perform the following tasks should be interesting:

- Detection and approximation of (self)-intersection curves of surfaces.
- Detection and approximation of (self)-intersection volumes.

As an extension, tools to ensure topology and arrangement of parametric surfaces and volumes are needed to manipulate collection of B-spline primitives.



Property	Value
QAxelObject	
Name	bs_0
Type	BSplineSurface

Figure 15: Computing a self-intersection curve of parametric surfaces in Axel.

6.7 Curve and Surface Fitting

Appendix A gives a broad overview of interpolation and approximation methods in SISL and GoTools with emphasis on curve and surface fitting. Also some information about the approximation methods is given. In short, we define a functional that we want to minimize. The functional is denoted the smoothness functional and has one approximation term and one smoothing term. The application can weigh between the terms. The functional is differentiated, and the problem of minimizing the functional with respect to the coefficients of the actual curve or surface, is transformed to a linear equation system.

Currently, the approximation methods handle only non-rational curves and surfaces. They will be extended to handle also the rational case. The initial weights will be kept fixed.

6.8 Data Reduction

In work package 3 two different solvers relate to a moving surface between a structural part and a fluid domain. The solvers use the same spline space, but does not necessarily need the same resolution. Thus, it is highly relevant not only to be able to increase the resolution, but also to decrease it. This is especially important if the solution field which is expressed in the refined spline space actually is different from the field in the reduced spline space only in the inner of the domain. In that case, the interface between the two fields can be described in the reduced spline space without loosing any accuracy.

The question is: Can a field or a surface represented in a refined spline space be represented in the reduced space with no error or an error that is small enough? It is also interesting to check if the larger field can be represented in a smaller spline space even if it is still larger than the reduced space.

Let $S(u, v)$ be the moving surface between the solid and the fluid. S is represented in a spline space corresponding to the knot vectors τ_1 and τ_2 in the u and v direction, respectively. Assume now that the first solver leaves the spline space as it is while the second refines it to use the knot vectors t_1 and t_2. $S(u, v)$ may also be modified during this process. The updated surface is $\tilde{S}(u, v)$. Can $\tilde{S}(u, v)$ be represented in the space corresponding to τ_1 and τ_2 with sufficient accuracy? Otherwise, also the spline spaces used in the first solver must be refined even if it is not necessary for the analysis in this case. Alternatively, can $\tilde{S}(u, v)$ be expressed using the knot vectors λ_1 and λ_2 where $\tau_1 \subset \lambda_1 \subset t_1$ and $\tau_2 \subset \lambda_2 \subset t_2$? In that case the spline space used in the first solver must be refined, but not as much as if we were using the full spline space of the second solver.

SISL has got functionality for data reduction of curves and surface. Thus, it is possible to use SISL to find the answers to the questions stated above. Data reduction for volumes may be implemented. The essence of the method is

- The importance of each knot in the knot vector is estimated to be able to decide which knots may be removed without any large consequences for the shape of the corresponding curve or surface
- Express a spline curve in a reduced spline space using a least squares approach
- An error estimate where the size of the modification of the curve or surface is measured in terms of coefficients
- An iterative approach to remove as many knots as possible

The method is guaranteed not to change the geometry more than a given tolerance. Degree reduction functionality does also exist. Note that in both cases, the curves and surfaces are expected to be non-rational.

6.9 Merging of Surface Patches

Figure 11 shows a ship hull consisting of a large amount of surfaces. This is not necessarily the most appropriate format for subsequent operations. Assume that a set of surface patches that is to be merged into one surface and the related accuracy requirements, are given. From the boundaries of the surface set, a spline space is chosen. A spline surface in this spline space is generated approximating a set of sampling points in the set of surface patches. A good initial parameterization of the sampling points is crucial for a good result. An iteration on the parameterization of the points, the spline space, and the weights of the styling functional is performed if the first version of the surface does not meet the given quality requirements. The iteration is stopped if the accuracy is not improved even if the accuracy is not met.

A set of surface patches can consist of one surface. The merging functionality will then perform smoothing on the surface. It is also possible to use this function to approximate a trimmed surface by a non-trimmed one provided that the trimmed surface is 4-sided, or 3-sided and allowed to be degenerate.

Merging of surface patches is performed in the GoTools module model_toolbox. The surface generation itself uses approximation methods presented in section 6.7 and appendix A.

However, even if a model can be represented by fewer surface than it is done initially, more than one surface is normally required. Thus, we need to enforce appropriate continuity between the surfaces. This topic is treated in the next section.

6.10 Tangent Plane Continuity between Surfaces

Assume that a set of surfaces is given. It reflects the shape and the structure of the model sufficiently well, but the continuity requirements between the surfaces are not satisfactorily met. The task is to modify these surfaces in such a way that they meet their neighboring surfaces with a given continuity. Continuity across surface boundaries may be defined to be within a tolerance.

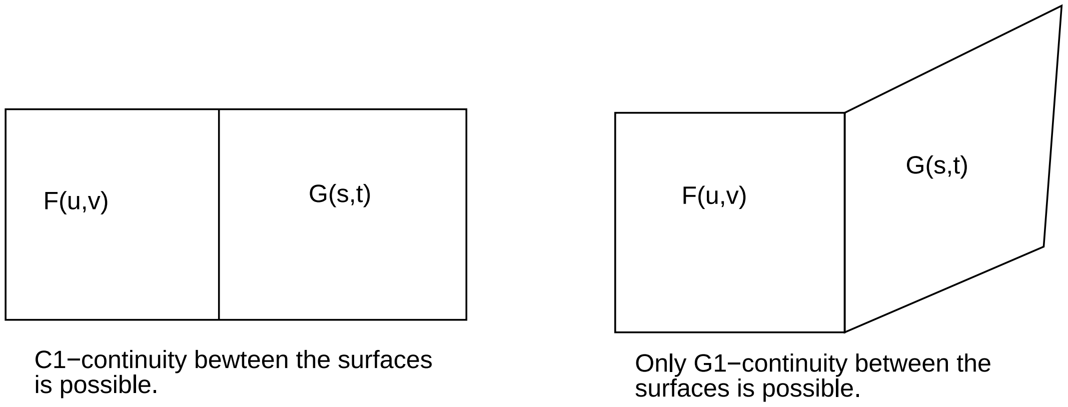

F(u,v)
G(s,t)
C1−continuity bewteen the surfaces is possible.
F(u,v)
G(s,t)
Only G1−continuity between the surfaces is possible.

Figure 16: Configurations between neighboring surfaces.

Assume that F and G are the two adjacent tensor-product spline surfaces and that $F(0, v) \approx G(s, t_q)$. F has knot vector τ_1 with degree k_1 and m coefficients in the direction along the common boundary. G has knot vector τ_2 with degree k_2 and p coefficients. $c_{0,i}$ for $i = 1, \ldots, m$ and $d_{j,q}$ for $j = 1, \ldots, p$ are the coefficients along the common boundary of F and G, respectively. Let $k_1 = k_2$ and $\tau_1 = \tau_2$, i.e. the two surfaces have the same knot vector along their common boundary curve. This implies $m = p$. Equality along the boundary curve is now ensured by requiring $c_{0,j} = d_{j,q}, \quad j = 1, ..., m$. Assume next that $\tau_1 \subset \tau_2$, i.e. τ_1 is a subset of τ_2. Then the necessary relation between the coefficients of the two surfaces in order to ensure C^0 continuity is less simple, but still linear since $c_{0,i}$ can be written as a linear combination of $d_{j,q}$ using discrete B-splines. Similar relations exist if the relation between τ_1 and τ_2 is more complex. Thus, if the degrees of freedom in the spline space is sufficient, then C^0 continuity between surfaces can be achieved by modifying the relevant surfaces at their common boundaries using linear conditions between the unknown coefficients to ensure positional continuity between the surfaces. The modification is performed by least squares approximation with a smoothing term, see section 6.7. A point set is sampled from the initial surfaces to maintain control over the surface shape. Lagrange multipliers are used to add linear side constraints to the approximation.

C^1 continuity across a common boundary between two surfaces can in some configurations be expressed by linear conditions in the coefficients. Now two rows of coefficients at the boundary are involved in the expressions. C^1 continuity is possible only if the boundary curves of the two surfaces lying adjacent to the common boundary meet with C^1 continuity, see figure 16. In most cases C^1 continuity between neighboring surfaces is considered to be a too strong requirement. G^1 continuity is more relevant.

G^1 continuity between two surfaces is the same as tangent plane continuity, i.e. the surface normals of the two surfaces are parallel along the common boundary. Alternatively, the partial derivatives of one surface along the boundary must lie in the tangent plane of the other surface. Thus, tangent plane continuity along the common boundary is ensured if there is C^0 continuity and in the current case

$$\frac{\partial}{\partial u}F(0, v) = \alpha_1(v)\frac{\partial}{\partial v}G(v, t_q) + \beta_1(v)\frac{\partial}{\partial t}G(v, t_q), \quad \forall v \in [0, v_m]$$

and

$$\alpha_2(v)\frac{\partial}{\partial u}F(0,v) + \beta_2(v)\frac{\partial}{\partial v}F(0,v) = \frac{\partial}{\partial t}G(v,t_q), \quad \forall v \in [0,v_m]$$

Here the two surfaces are assumed to have the same parameterization, but not necessarily the same knot vector along the common boundary. $\frac{\partial}{\partial v}F(0,v)$ already lies in the tangent plane of G since this vector points along the common boundary. The blending functions α_1, α_2, β_1 and β_2 are scalar functions. These functions are unknown, but can be set from information in the surface corners. Due to the variation diminishing property of spline curves, the tangent plane continuity is achieved along the entire common boundary if it is achieved in R points where the number R depends on the polynomial degree and the number of polynomial pieces in the spline space defined from the union of the spline spaces of the adjacent surfaces in the relevant parameter directions. Exact G^1 continuity can only be achieved if there is enough degrees of freedom to enforce it in the two surfaces. In particular the partial derivatives of both surfaces must be able to span the same tangent planes along the common boundary curve.

Approximation of the conditions on G^1 continuity can be achieved by applying the following functionals:

$$\min_{\vec{c},\vec{d}} \int_0^{v_m} [\frac{\partial}{\partial u}F(0,v) - \alpha_1(v)\frac{\partial}{\partial v}G(v,t_q) - \beta_2(v)\frac{\partial}{\partial t}G(v,t_q)]^2 dv$$

and

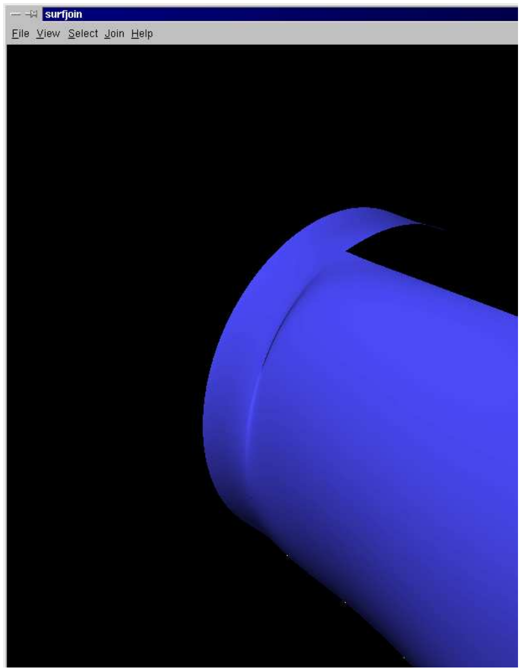
$$\min_{\vec{c},\vec{d}} \int_0^{v_m} [\alpha_2(v)\frac{\partial}{\partial u}F(0,v) + \beta_2(v)\frac{\partial}{\partial v}F(0,v) - \frac{\partial}{\partial t}G(v,t_q)]^2 dv$$

The coefficients of the two surfaces that are allowed to be changed are denoted $\vec{c}$ and $\vec{d}$. The terms above are included in the smoothing functional, and the surface modification is performed. The free coefficients of the surfaces will be close to the common boundary. If the accuracy regarding tangent plane continuity is too poor, there is a possibility of increasing the spline spaces of the surfaces or changing the relation between the weights in the extended styling functional.

Figure 17 shows a case where, in the first picture, a requirement on exact G^1 continuity between the two surfaces, is too strict. Using side constraints, a problem in satisfying all the various requirements to the surface modification, occurs. The smoothness of the surface set is sacrificed in order to achieve G^1 continuity, and the surfaces behave strangely close to their common boundary. When the requirements are released somewhat, the overall result is improved as can be seen in the second picture of figure 17. Here the minimization terms are added to the smoothing functional.

If the surface set consists of many surfaces, simultaneous modification of the complete surface set becomes very complicated and gives rise to an equation system with a huge number of unknown coefficients. Thus, the modification process treats only a limited number of inner boundaries at the time, and it is possible to traverse the boundaries more than one time. For the same reason, smoothing in the inner of the surfaces and at the boundary is kept separate.

As an alternative to the symmetric enforcement of approximate tangent plane continuity that is described here, one surface can inherit the tangent plane of its adjacent surface. Depending on the surface configuration, see figure 16, it may be required to increase the



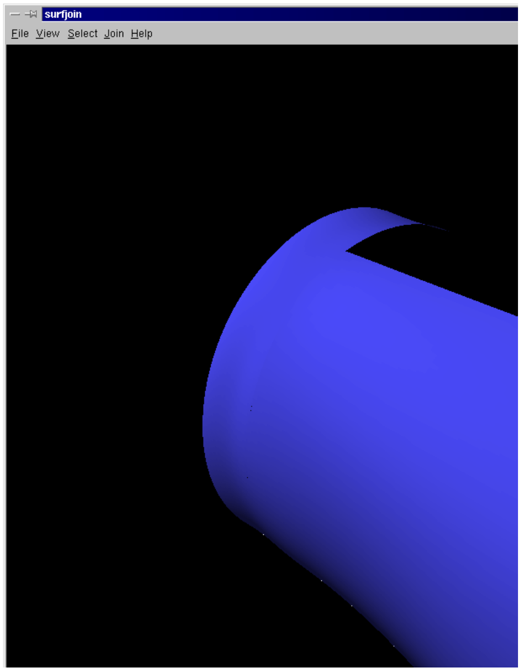


Figure 17: Symmetric enforcement of tangent plane continuity using side constraints and by minimizing the error

degree of the surface to adapt exactly to the specified tangent plane. Instead of applying degree elevation, the tangent plane information is approximated to lie in the spline space of the current surface.

The module model_toolbox offers the functionality for enforcing and maintaining up to G^1 continuity between surfaces. Only non-rational surfaces are supported.

6.11 Modification of Sets of Surfaces

Assume that we have a number of surfaces describing a model and we want to adapt these surface to a given curve. This is a relevant setting in the context of WP2 and WP5. This may be done using the methods of the previous section. The curve is translated to a sequence of points that we want to approximate and the surface set is modified while maintaining the continuity between the surfaces. However, we need to divide the modification into phases to avoid a too resource demanding process.

6.12 Graphical rendering of surfaces and volume object

For vizualisation purposes, we have also to consider the following questions:

- Render volume by points.
- Render volume by hexahedron grid.
- Render volume by tetrahedron grid.
- Render surfaces and volume with color information.

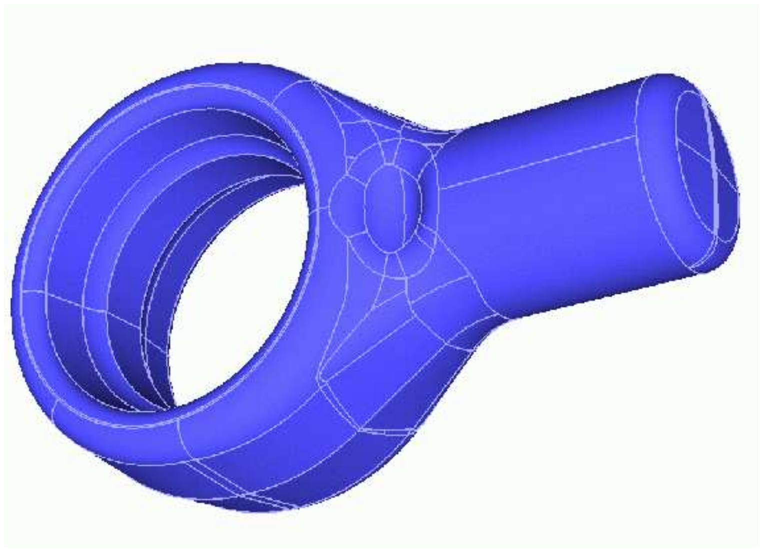


Figure 18: A boundary represented CAD model

7 From a Brep CAD Model to a Multi-block NURBS Volume Model

One goal for the Isogeometric Toolbox is to be able to build isogeometric volumes from a well behaved surface model. This can be generalized to generate a NURBS volume mesh from a boundary represented CAD model. This task can be divided into two parts, finding the block structure which suitably reflects the model and to create each block that in our context is represented as a trivariate NURBS volume. The block structuring part is similar to block structuring in multi-block mesh generation which is highly relevant in solving PDEs by difference method. In this field, no method for completely automatic block structuring for general models have been found. However, several approaches exist, and we should consider them for our purpose.

Given a suitable block structure, GoTools contains much functionality to produce the actual blocks, i.e. NURBS volumes. The boundary surfaces of the NURBS volumes that also belong to the boundary of the total model must adapt to the initial CAD model. In most cases this will imply an approximation as the structure of the CAD model and the volume model do not necessarily correspond. Thus, the methods described in section 6.7 will be particularly useful. Also the sections 6.9, 6.10 and 6.11 are relevant.

Consider the model in figure 18. The object is relatively simple, but the CAD representation is very complex. The lines in the model shows the boundaries between surfaces in the outer shell. All the surfaces are represented as trimmed in the CAD model, but about half of them may be represented as non-trimmed B-spline surfaces without approximation. Most surfaces are of B-spline type, but there are also elementary surfaces included in the model. It is, in this case, no correspondence between the CAD model and the volumetric model one could want for isogeometric analysis. An automatic creation of the volumetric model would be out of scope for Exciting.

Let us now consider the CAD models provided by WP3. The initial CAD models are much better behaved and the divisions of the models into surfaces give some information about how the volumetric blocks may be structured. The two models in figure 19 are swept

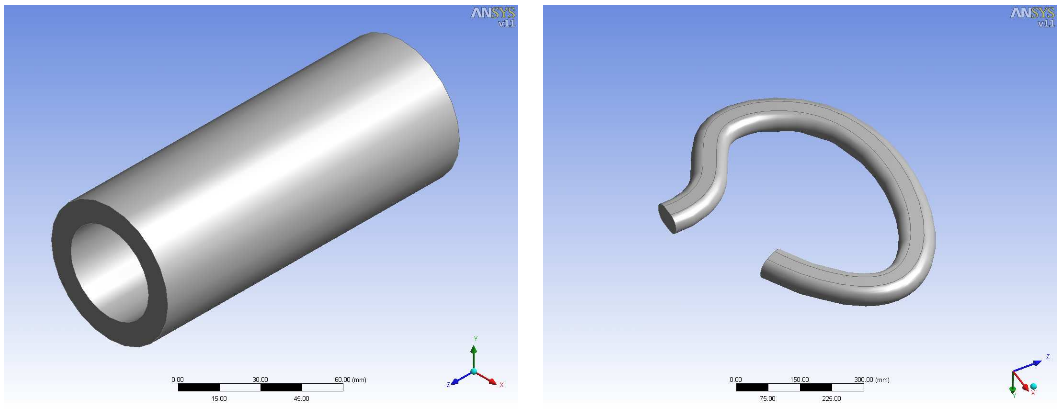


Figure 19: CAD models representable as a swept volumes

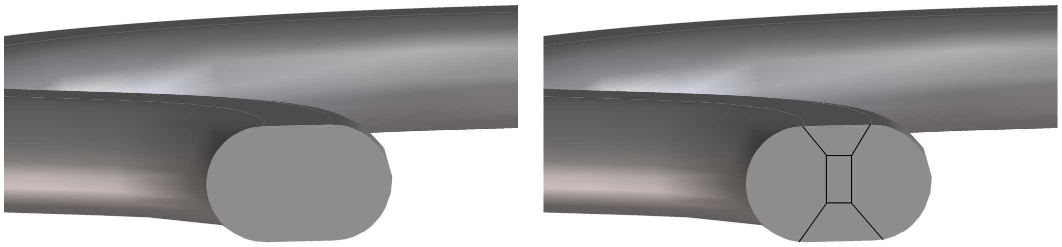


Figure 20: End surface of CAD model to be swept

configurations. Provided that the sweep information is recognized from the CAD file, the models may be represented as one swept NURBS volume. Alternatively, the volume may be created using the surfaces in the CAD file as boundary surfaces and use a Coons type approach. In both cases elementary surfaces must be translated to NURBS surfaces and in the first case, surfaces at the seam of the cylinder must be defined. For volume creation methods, see section 6.5.

Still, even in these simple one-block models ambiguities exist. The first picture in figure 20 shows one end surface of the second model in figure 19. In the CAD model it is represented as a trimmed plane. The adjacent surfaces are two trimmed planes and two NURBS surfaces going along the model in the sweep direction. Using the surfaces of the CAD model as boundary surfaces for the NURBS volume, we get one volume where all corners in the swept direction are singular, i.e. the partial derivatives of the volume in the actual directions are parallel. Alternatively, one could let the complete outer boundary of this trimmed plane represent one boundary of a surface, the midpoint can represent the opposite boundary while two boundaries are defined along a seam. This profile surface can then be swept to create a volume with the same shape as the previous one, but with completely different properties. This volume will be closed and have one boundary surface which degenerates to a curve and one boundary surface that is constructed from the two planar surfaces and the two NURBS surfaces. Finally, the model can be represented by four non-degenerate volumes as indicated in the second picture of figure 20. The three solutions might not all be of the same quality. Still, it is not for the library software alone

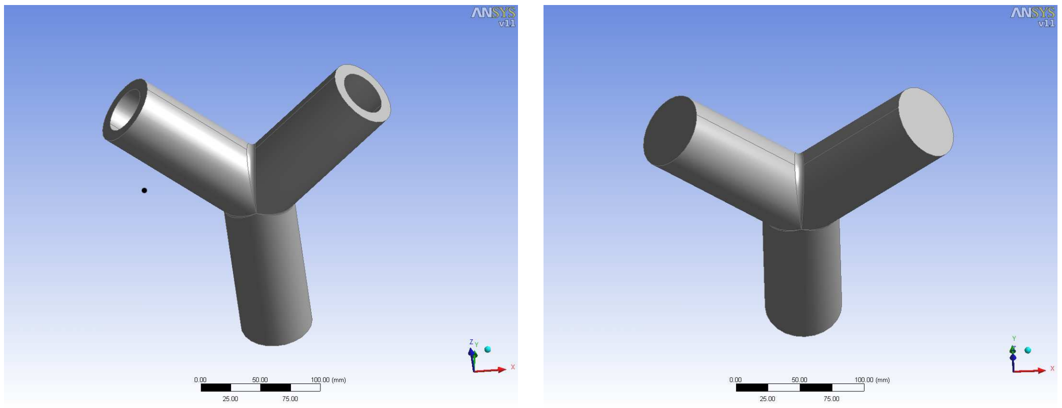


Figure 21: Multi-block models

to decide which of them should be chosen.

Figure 21 shows two multi-block models of relatively modest complexity. Still, they point at some of the problems in translating a Brep CAD model to a multi-block NURBS volume model. Given the CAD file, the following steps must be addressed:

- Recognize the swept parts of the models
- Decide on how to handle the singularity in the second model. This is similar to the model in figure 20.
- Recognize the blends
- Create the swept volumes for each part of the model
- Join the initial volumes taking the conditions for an isogeometric represented volume model into account, i.e. the individual NURBS volume must meet with exact C^0 continuity. There should be no gaps and no overlaps
- Modify the model with respect to the blends

As explained earlier, translating a general Brep CAD model to a isogeometric model is extremely challenging and out of scope for Exciting. The models in figure 21, however, is at the right level.

Figure 22 shows a more complex model. Even if the surface structure appropriately models the outer shape and the model contains no hole, the CAD model surfaces cannot be used directly as boundary surfaces for NURBS volume blocks. A block structuring will require much analysis of the model and user interaction. It is uncertain whether this model can be handled during Exciting.

8 Conclusion

The content of this document is related to GoTools and Axel and the current data structure and functionality of this software. Some modifications and extensions are described. Major new development is not planned. However, Exciting as a project will do new development and some of the produced code are likely to fit into the framework of the toolbox.

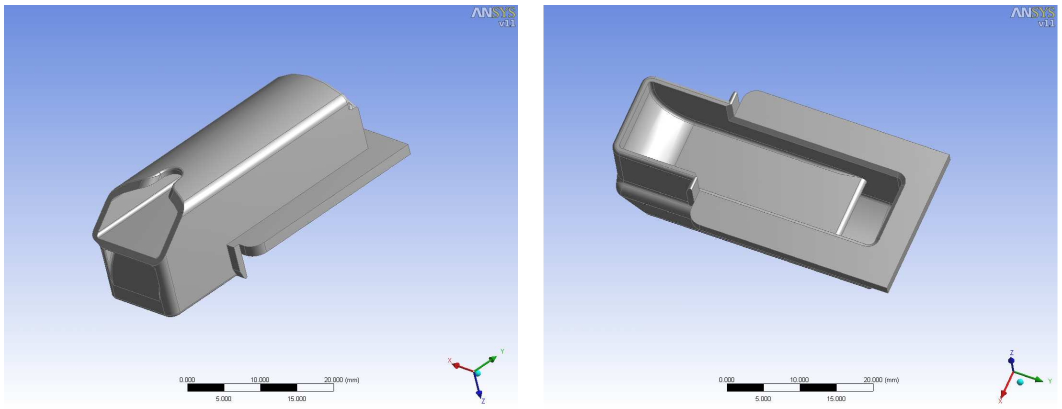


Figure 22: A more complex model

Much of the efforts in the Toolbox work package will be used for ensuring that we after Exciting has an as complete as possible isogeometric toolbox that will gain wide acceptance among researcher within and developers and users of isogeometric analysis tools.

The restructuring of a CAD model to an isogeometric NURBS volume model is the most demanding part of the Toolbox. We expect to develop some sideground software in this field during the project period. However, it is open how far we will get in this direction. The aim is to handle simple models that are of interest for the project. Complex CAD models are out of scope.

We also expect that the other partners will give feedback on the focus and structure of this document and plan to make an update version of this document after the meeting in Athens at the end of April 2009.

References

[1] Vibeke Skytt. Technical Report of Fairing and Surface Representation. Fantastic, Functional Design and Optimization of Ship Hull Forms (2000).

[2] Vibeke Skytt. Technical report on the fairing toolbox for complex surfaces. Fantastic, Functional Design and Optimization of Ship Hull Forms (2003).

A WP6

Methods for curve and surface fitting in GoTools

Vibeke Skytt

SINTEF

A.1 Introduction

The intention of this document is to given an overview over methods for curve and surface creation in GoTools. Some information about methods which exist in SISL, but not in GoTools is included. The emphasis of the document is on fitting methods, i.e. approximation.

The SISL library and the GoTools libraries are background software from SINTEF that will be a part of the isogeometric toolkit. SISL and some parts of GoTools are open software with a GPL license. The software can be found at *http://www.sintef.no/math_software*.

The software is documented. However, the information in this document is aimed directly at the geometry creation process and should give an overview over the related functionality before going into details in each module and class. Moreover, information about existing possibilities can trigger ideas about even more useful tools or modifications to the current ones. We will be happy to know about such ideas.

A.2 Exact methods for curve and surface generation

A.2.1 Curves

GoTools provides the following methods for exact curve creation:

- Interpolate a set of points
- Interpolate a set of points and tangents, Hermite interpolation
- Interpolate a set of points where some points may be assigned a tangent

SISL also possesses similar methods for curve creation. In addition a number of methods to create NURBS curves representing conic sections exists as well as blending functionality.

A.2.2 Surfaces

GoTools provides the following method for exact surface creation. The input curves are expected to be non-rational.

- Create a lofted surface by interpolating a set of curves
- Create a lofted surface interpolating a set of curves where some of them may be assigned a cross tangent curve

- Create a Coons patch interpolating 4 boundary curves, some of them may be assigned a cross tangent curve
- Create a surface interpolating a set of curves in both parameter direction
- Create a surface interpolating a set of curves in both parameter direction where some curves may be assigned a cross tangent curve
- Create n surfaces filling an n-sided hole with G^1 continuity in the inner. The hole is described by positional and cross tangent curves at the boundaries of the hole

The skinning type functionality, i.e. functionality where curves in both parameter directions are given, depends on corner consistency in the input geometry. Inconsistent information in corners will violate the interpolation.

SISL also contains lofting functionality and can, in contrast to GoTools, also handle rational input curves. This is, however, done in a special function that does not accept tangent curves. The resulting surface will be cubic and C^1. SISL has functionality to represent conic surfaces as NURBS and methods for linear and rotational sweep.

A.3 SmoothCurve and ApproxCurve

The class SmoothCurve offers a method for least squares approximation of a non-rational B-spline curve. A smoothing term may be included in the computation. Let the curve be

$$f(t) = \sum_{i=1}^{n} c_i B_{i,d,\tau}(t)$$

The curve lies in the spline space described by the knot vector τ, and the polynomial degree d.

The curve is defined to minimize the following expression with respect to the unknown coefficients $\vec{c}$.

$$\min_{\vec{c}}[\int_a^b (\omega_1 f'(t)^2 + \omega_2 f''(t)^2 + \omega_3 f'''(t)^2)dt + \omega_4 \sum_{r=1}^{n} (f(t_r) - a_r)^2$$

The parameter interval of the curve is $[a, b]$. The first part of the functional is related to smoothing while the last term performs least squares approximation. The curve will approximate the points a_r, $r = 1, \cdots, n$. The points must be parameterized with respect to the parameter interval of the curve prior to curve generation.

The curve entity must be defined before the class SmoothCurve is invoked, but the coefficients may all be set to zero. The functionality may also be used to smooth an existing curve. In that case, it is not required to approximate any data points. If a sufficiently dense set of points to approximate is provided, smoothing may be omitted. The curve quality, however, is expected to benefit from a small smoothing contribution. If only least squares approximation of a sequence of parameterized point is to be performed, GoTools contains an alternative to SmoothCurve, namely the class SplineApproximator. The coefficients of a cubic curve is created.

The minimization functional is parameter dependent. Minimization of the first derivative approximates minimization of curve length, second derivative approximates minimization of curvature, and the third derivative corresponds to variation of curvature. Minimizing the functional gives rise to a linear equation system. The result will be closer to a parameter independent approach if the parameterization of the curve or the data points is close to iso-metric. Minimization of first derivative is normally not recommended. Minimization of second and third derivative give good results, but require C^1 or C^2 continuity of the spline space, respectively.

A number of coefficients can be kept fixed at the endpoints of the given curve during the smoothing or least squares operation. As this operation tends to shrink the curve slightly, it is recommended to set the two end coefficients before applying the functionality even if it is used for curve creation.

The curve can be defined to be closed and have up to C^1 continuity across the seam. A number of linear side constraints can be added. These are handled using Lagrange multipliers.

SmoothCurve can be extended to handle rational curves if required. In that case the weights must be provided initially and will not be changed during the smoothing operation. Also curve generation may be performed to create a rational curve with prescribed weights, but I can see no particular reason for doing so.

ApproxCurve is intended for approximation of points and provides a wrapper around SmoothCurve. The class generates an initial curve as input to SmoothCurve. Moreover, it iteratively performs a number of least squares approximations where it for each step does parameter iteration and increases the spline space in order to approximate a set of data points within a given tolerance. The points must be provided with an initial parameterization.

A.4 SmoothSurf and ApproxSurf

SmoothSurf is the surface equivalent of SmoothCurve. It works with a non-rational B-spline surface

$$f(u,v) = \sum_{i=1}^{n}\sum_{j=1}^{m} c_{i,j} B_{i,d_1,\tau_1}(u) B_{j,d_2,\tau_2}(v)$$

with control points $c_{i,j}$ in R^3 defined over the knot vectors τ_1 and τ_2. f is the solution to the minimization problem

$$\min_{\vec{c}} \Big[\int\int_{\Omega}\int_0^{\pi} \sum_{l=1}^{3} \omega_l \Big(\frac{\partial^l f(u_0 + r\cos\phi, v_0 + r\sin\phi)}{\partial r^l}\Big|_{r=0}\Big)^2 d\phi du_0 dv_0 \\ + \omega_4 \sum_{r=1}^{R} w_r (f(u_r, v_r) - a_r)^2 \Big]$$

Ω is the parameter domain of the surface f. (u_0, v_0) is a point in Ω and ϕ is the angle between a current direction in the parameter plane and the u-axis.

It is also possible to approximate another surface in the same spline space, or approximate the direction of a set of surface normals. The surface boundary and some derivatives across the boundary may be kept fixed.

The least squares approximation term and the three terms in the smoothness part have individual weights. Thus, there is a choice of what to emphasize: approximation order or smoothness. In addition, each data point may be attached an individual weight. If we approximate a data set without keeping the surface boundary fixed, data points close to the boundary can be assigned larger weights than the remaining points in order to avoid shrinkage of the surface.

A dense set of data may be approximated without applying any smoothing. The smoothness term is included to be able to handle data sets with holes and data sets with areas where the point set is sparse. In addition it tends to minimize oscillations and the influence of bad data points and in general create a surface of better quality.

The first order term of the smoothness functional implies minimization of area. It is normally not feasible to include this term. It has a sometimes unexpected influence on the surface shape, and if the surface boundary is not kept fixed, minimization of area must not be performed. That would lead to a drastic shrinkage of the surface. However, if we want to approximate a surface that is close to planar, the term should be included.

The second order term is an approximation to minimization of curvature, and the third order term is an approximation to variation in curvature. Approximating curvature by an expression in terms of second derivatives, and variation in curvature by an expression in terms of third derivatives, make sense only if the surface parameterization is close to isometric. The derivative expression is integrated around the circle in order not to emphasize smoothness only in certain parameter directions. Then this expression is integrated over the parameter domain of the surface to get the final smoothness term.

The minimization functional gives rise to a linear equation system which is solved iteratively. Thus, reasonable start values for the coefficients can improve the performance of the function.

Also SmoothSurf can handle linear side constraints, but this option must not be combined with approximation of surface normals. Continuity up to and including C^2 continuity may be enforced across the seam for a closed surface.

We have previously, outside of GoTools, worked with smoothing of a rational surface. The weights are kept fixed. Rational smoothing can not handle as large surfaces as non-rational smoothing because it takes more time to build the equation system.

ApproxSurf is intended for point approximation and provides a wrapper around SmoothSurf in a similar way to ApproxCurve and SmoothCurve. It creates an initial surface for use in SmoothSurf and performs parameter iteration in a series of least squares approximation. However, the spline space is currently **not** increased. That is required if we aim at satisfying some approximation tolerance and it is planned to add knot insertion dependent on the approximation error to this class. Note that approximation of a set of scattered data points within a tight tolerance is not always possible. In particular, noisy points will create problems. Thus, the function needs to stop iterating when the accuracy is not improved even if the accuracy requirement is not met.

ApproxSurf may also be extended with functionality to create surface boundaries prior to approximation of the points lying in the inner of the surface. This functionality will counteract the shrinking tendency with respect to the points, that is typical for the kind of operations performed within SmoothSurf.

A.5 The parameterization module

The input points must be parameterized prior to the approximation performed in SmoothCurve and SmoothSurf. In the curve case, a chord length parameterization is a good choice. The surface case is more complex. If we have or can create an initial surface with a shape resembling the final one, the points can be parameterized by projecting them onto the initial surface. If no good initial surface exist, GoTools provides another possibility.

In GoTools, there is a module called parameterization. The module contain a number of methods for parameterization of 3D points. The points may be organized, for instance in a triangulation, or unorganized. In the latter case, the boundary points must be identified and ordered. The boundary points are parameterized first, then the points in the inner. In both cases, it is a number of methods to choose from. The methods in this module guarantees a legal parameterization, i.e. there is no crossing of edges between parameter points in the domain.

The GoTools documentation provides substantial documentation on how to use this module.

A.6 Surface creation from point sequences

The problem is as follows: *Given a number of ordered point sequences, create a surface where constant parameter curves in the surface approximates the points.* The number and distribution of points in the sequences may vary and it is no fixed distance between the point sequences.

The creation method is as follows:

- The points in each point sequence is parameterized independently
- The parameterizations of the point sequences are modified with respect to each other
- A common spline space for a set of curves approximating the point sequences is chosen
- The point sequences are approximated by ApproxCurve
- A parameterization of the curve set is computed
- The final surface is computed by lofting

Each point sequence is thus approximated by a curve while the resulting curves are fitted exactly by a surface.

This functionality is implemented in a class called ftOffsetSurfGen. It is a part of GoTools, but does currently not belong to the open source part of it. The function will be added to the iso-geometric toolbox, but may need some cleanup and testing before it is made available.

SmoothSurf or ApproxSurf may be used to improve the smoothness of surfaces created by ftOffsetSurfGen. The initial point set, parameterized as in ftOffsetSurfGen, is used in an additional smoothing and approximation process. ApproxSurf is able to perform parameter iteration in one parameter direction only. Thus, the initial points will still lie at constant parameter curve in the surface after a post process.

A.7 Approximation of evaluator based curves

GoTools contains methods to approximate one curve or a set of curves with C^1 continues polynomial spline curves. The only requirements on the input curves is that they can be evaluated and that they can be accessed through an interface specified in the classes EvalCurve or EvalCurveSet for one curve or a set of curves, respectively. The classes are abstract and the evaluator based curve(s) to approximate are represented as subclasses of these classes. The classes HermiteAppC and HermiteAppS refine the spline space of the output curve(s) until a requested accuracy is reached and generate the curve(s) by Hermite interpolation.

Examples of use of this functionality is

- Approximate an intersection curve
- Create a curve in geometry space from a curve in the parameter domain of a surface
- Compute the curve in the parameter domain of a surface corresponding to a curve lying in the surface
- Projection of a curve onto a surface
- Approximate a rational curve by a non-rational one
- Create a number of curves and cross tangent curves lying in the same spline space based on some input constraints to be used in operations like lofting or skinning

A.8 Modification of a surface set

Most models may not be represented by one spline surface only. Assume that we have a set of surfaces and adjacency information related to the surfaces. Assume also that these surfaces need to be smoothed or modified to approximate a set of data points. Smoothing in this context may also mean that two or more surfaces in the set does not meet with the expected continuity along common boundaries.

In GoTools the combination of the classes ftSmootSurfSet and SmoothSurfSet is intended to solve the problem described above. SmoothSurfSet corresponds to SmoothSurf for a set of surfaces. ftSmoothSurfSet performs parameter iteration on the points to approximate, and generates linear constraints between coefficients in the relevant surfaces in order to ensure a certain continuity between surfaces being modified by SmoothSurfSet. The input surfaces are expected to have a certain similarity to the expected final surfaces. In particular C^0 continuity between adjacent surfaces should be present when we want to enforce G^1 continuity. Each of the points we want to approximate should be coupled with one surface in the set and parameterized with respect to this surface. ftSmoothSurfSet increases the spline spaces of the surfaces in order to approximate the point set within a prescribed tolerance, but stops the iteration if the improvement in accuracy stops.

The classes, especially ftSmoothSurfSet, are in a prototype stage. The use so far has been to modify surfaces close to a common boundary to improve the continuity between the surfaces. The point set we want to approximate lies close to the initial surface set. In this context, we have seen that some effort is required to ensure that the spline space is rich enough to support the wanted continuity between surfaces. C^0 continuity is expected to be exactly satisfied while G^1 continuity is aimed at being satisfied within a given tolerance.

The functionality is demanding. Thus, it makes sense to modify only a few surfaces or sub surfaces simultaneously. A combination of modification of single surfaces while maintaining the boundaries fixed and modification close to and across surface boundaries is recommended.

The functionality described in this section need some extension and quality improvements before it can be a part of the toolkit. However, it may be useful for instance in the context of moving surfaces and shape optimization.

B WP2

Technical Report No BRINO917GES081114-1

Ship Types and Design Parameters Used in WP2

Geir Skeie

DNV

TECHNICAL REPORT

EXCITING

SHIP TYPES AND DESIGN PARAMETERS USED IN WP2

REPORT NO BRINO917GES081114-1

REVISION NO <REV>

DET NORSKE VERITAS

TECHNICAL REPORT

Date of first issue: 14.11.2008	Project No: 91720320	DET NORSKE VERITAS AS 1322 Høvik Norway Tel: Fax: http://www.dnv.com NO 945 748 931 MVA
Approved by: Geir Skeie	Organisational unit: BRINO917	
Client: Exciting	Client ref.:	

Summary:

Report No: BRINO917GES081114-1	Subject Group:
Report title: Ship types and design parameters used in WP2	
Work carried out by: Geir Skeie	
Work verified by:	

Date of this revision:	Revision No: <rev>	Number of pages: 8

Indexing terms

Keywords	Service Area
	Market Sector

[✓ Unrestricted distribution (internal and external)]

☐ Unrestricted distribution within DNV

☐ Limited distribution within DNV after 3 years

☐ No distribution (confidential)

Report No: BRINO917GES081114-1, rev. <rev>

TECHNICAL REPORT

Table of Contents ***Page***

1 EXECUTIVE SUMMARY 1

2 INTRODUCTION 1

3 DESIGN PARAMETERS 1

4 THE KRISO CONTAINER SHIP (KCS) 7

5 MS EXCITING 7

6 REFERENCES 8

TECHNICAL REPORT

1 EXECUTIVE SUMMARY

The document specifies the ship types, the main free design parameters and their constraints used in work package 2 of the Exciting project. The two ship hulls, both container ships, represent commercially interesting ship hulls while at the same time being slender enough to ensure that the Neumann-Kelvin model, with fixed trim and sink, will provide a satisfactory approximation of the ship wave resistance.

2 INTRODUCTION

The current document defines the ship hull types used in WP2 in the Exciting project. Both the ships are container ships. They represent interesting commercially hull shapes while at the same time being slender enough to ensure that the Neumann-Kelvin model, with fixed trim and sink, will provide a satisfactory approximation of the ship wave resistance.

The free parameters together with the constraints define the appropriate design space to be explored for the different container ships.

The overall aim is to use the Isogeometric Analysis concept to allow direct manipulation on hull shape parameters in an optimization loop. However, the intent of this document is to describe the hull characteristics at a higher level and not explore the spline/NURBS control point level. The hull characteristics are included to describe the intended shapes and their possible variations in the optimization process.

The hull surface shape is an instance of functional surfaces and simulation driven design means that simulations produce shapes rather than just evaluating them. (It must be remembered that each mathematical formulation is a restriction of the freedom of the system. Gert Kuiper 1970.)

The overall aim in the project is a proof-of-concept of Isogeometric Analysis. The optimization problem is considered as an improvement of an already established (or perturbed) baseline ship hull shape design. Consequently, optimization of a completely new design is ruled out of the current scope.

Reference is made to the EC funded project: Functional design and optimisation of ship hull forms (FANTASTIC).

IGES CAD models are provided together with their associated body lines.

3 DESIGN PARAMETERS

The ship is divided into three portions in the longitudinal direction:

- Bow,
- midsection and
- stern.

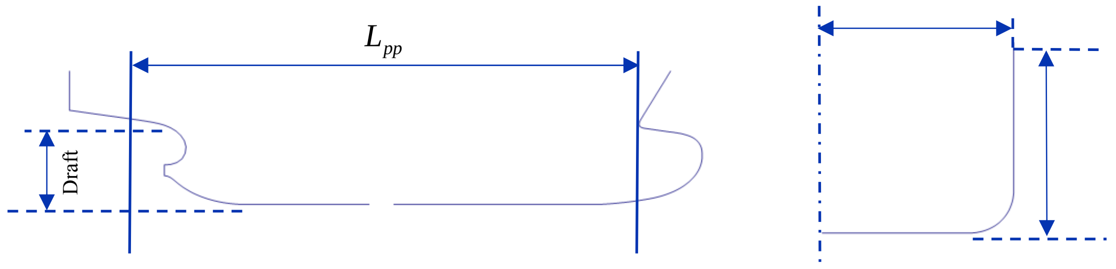
Design parameters are associated with both the global dimensions and the three main partitions. In addition, other non parametric quantities may be addressed to manipulate the global and local characteristics of the hull shape (like the transition lines/surfaces). These are associated with the control points in a NURBS representation.

Design parameters that enter into the four classes are:

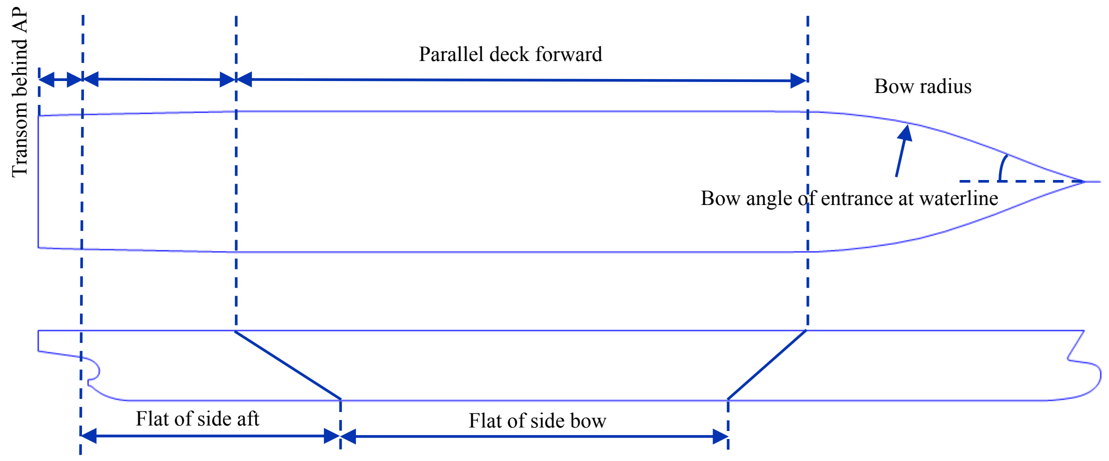
- Free global parameters: length (length between perpendiculars), beam (breadth), depth, draft.

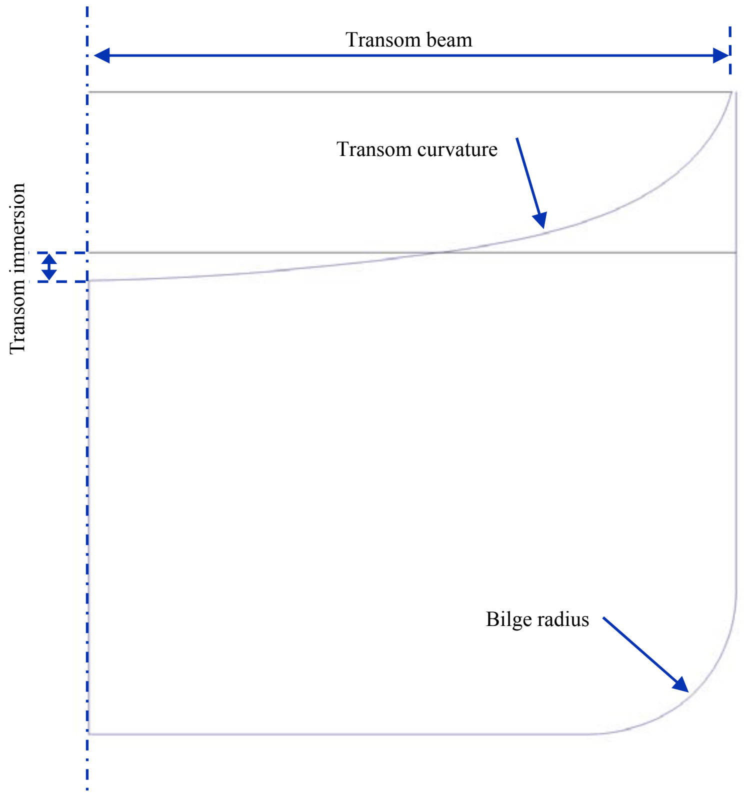
Beam/2

Depth

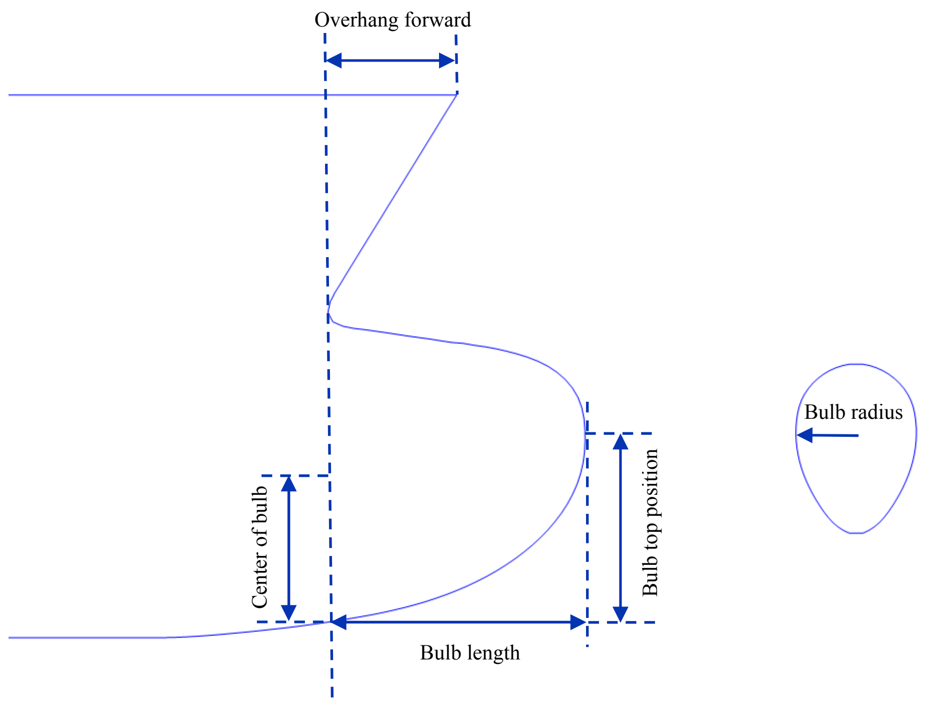


- Mid hull parameters: Extent of parallel deck and middle body, flat of side aft and bow, radius of bilge, bow radius and bow angle of entrance at the waterline.

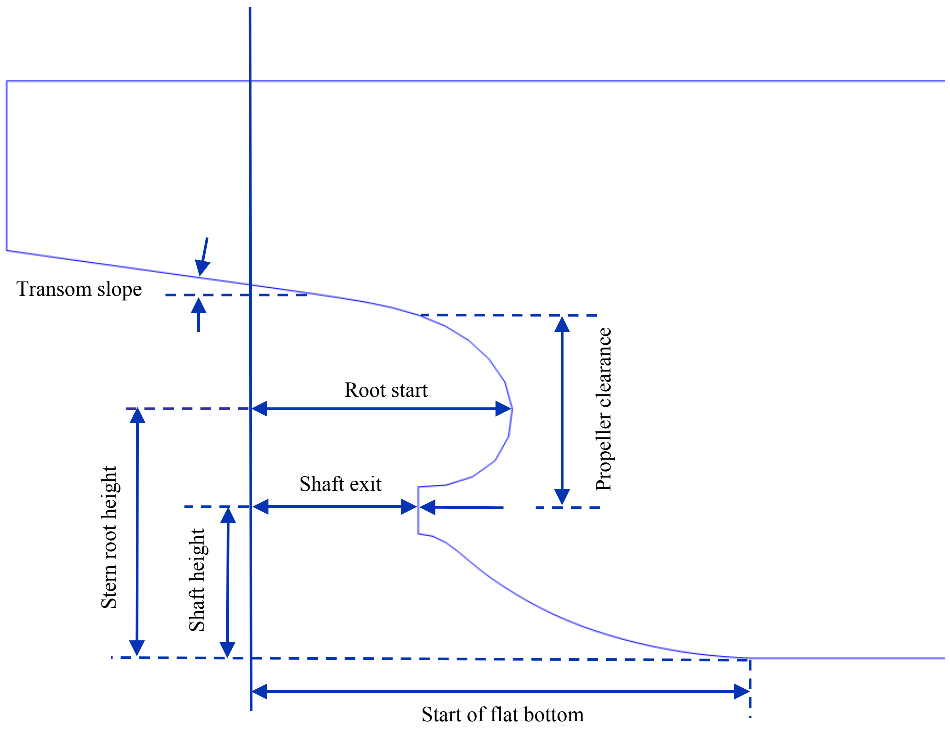




- Bow parameters: Distance between the forward perpendicular and the extreme forward end of the hull, bulb length, bulb radius, bulb centre and top position.



- Stern parameters: Transom behind aft perpendicular, breadth of the transom at the deck, curvature of transom, height of immersion of transom, transom slope, .



The design parameters have to be constrained in order to yield allowable shapes and properties:

- Displacement.
- Hydrostatic stability (seakeeping, but this may become too complicated).
- Operational constraints.

In general, a ship operates at different speed regimes, thus the optimized ship hulls have to take this into account. The cost-function with respect to wave resistance should thus include a speed range:

Speed (knots)	20	25	27
Weight	0.35	0.50	0.15

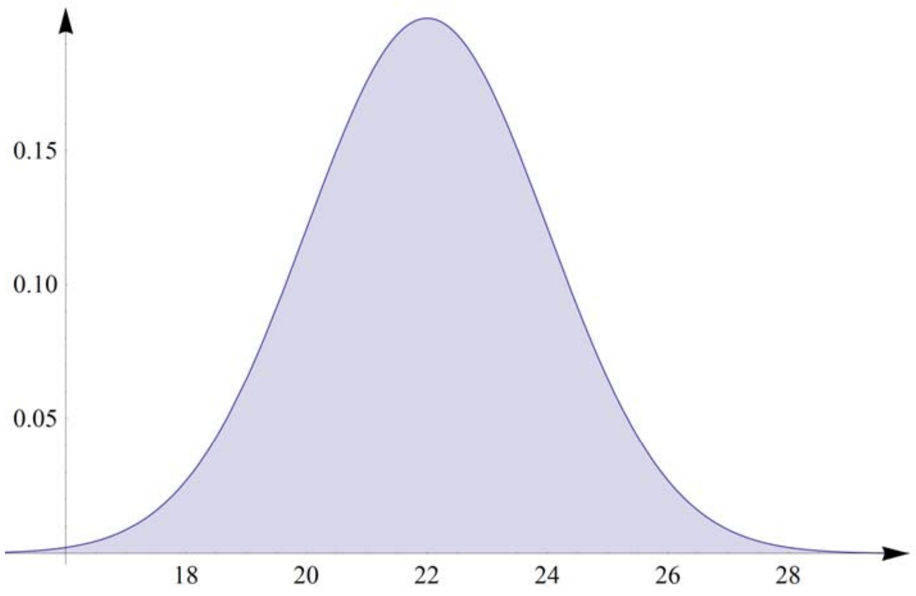


Figure 1 Velocity profile used in optimal design.

Several draft configurations should also be studied together with different trim positions and are included in the multi-objective optimization where each position has a different weight. Thus, the objective function is to consider wave resistance at different displacements for a range of ship speeds:

- Wave resistance according to the linearized Neumann-Kelvin model.
- Speed range:

Speed (knots)	20	25	27
Weight	0.35	0.5	0.15

- Displacement range:

Displacement	-10 %	0%	10%
Weights	0.25	0.5	0.25

This may be considered as minimizing fuel consumption during service life that are expanded into operational conditions. Thus,

$$F = \int_T R dt \qquad \textbf{(1)}$$

and the operational conditions are visualized in Figure 2.

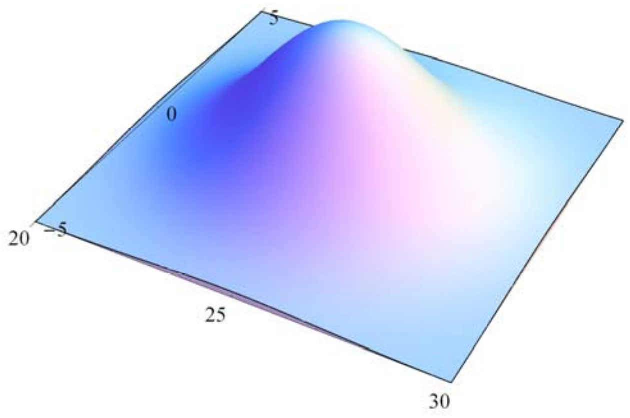


Figure 2 The operational conditions as a continuous distribution of displacement and speed ranges.

The free parameters and constraints are grouped into *Global*, *Bow* and *Stern* data. All the data is given a number according to priority.

The Global free parameters are given below:

No	Name	Free	Constraint
G1	Length (L_{pp})	Free	
G2	Over all length	+/- 5 %	
G3	Length of waterline	Free	
G4	Beam		Constant
G5	Draft	+/- 10%	
G6	Trim	Max 2 m down by the stern, Max 1 m down by the bow	
G7	Displacement		+/- 1%
G8	Longitudinal Centre of Buoyancy	Is here to specify change in volume distribution	
G9	Length of parallel midship	+/- 10%	
G10	Radius of bilge		Fixed at midship
G11	Transom behind AP	+/- 10 %	
G12	Start of flat bottom	+/- 10%	

The bow and bulb parameters are given below:

No	Name	Free	Constraint
B1	Bow radius at waterline	Free	

B2	Bow angle of entrance at waterline	Free	
B3	Overhang forward	Free	
B4	Bulb length	+ 30 %	
B5	Bulb center	Free	
B6	Bulb radius	Free	
B7	Bulb top position	Free	

The stern and transom parameters are given below:

No	Name	Free	Constraint
S1	Transom behind AP	+/- 10 %	
S2	Transom beam	Free	
S3	Transom curvature	Free	
S4	Transom immersion	Free	
S5	Transom slope	Free	
S6	Propeller clearance (transom immersion)		Fixed
S7	Shaft height		Fixed
S8	Shaft exit		Fixed
S9	Stern root height	+/- 10 %	
S10	Stern root start	+/- 10 %	

4 THE KRISO CONTAINER SHIP (KCS)

The KRISO Container Ship (KCS) is part of the CFD Workshop benchmark suit of problems. A complete reference may be found at the home page of the CFD Worshop in Tokyo 2005.

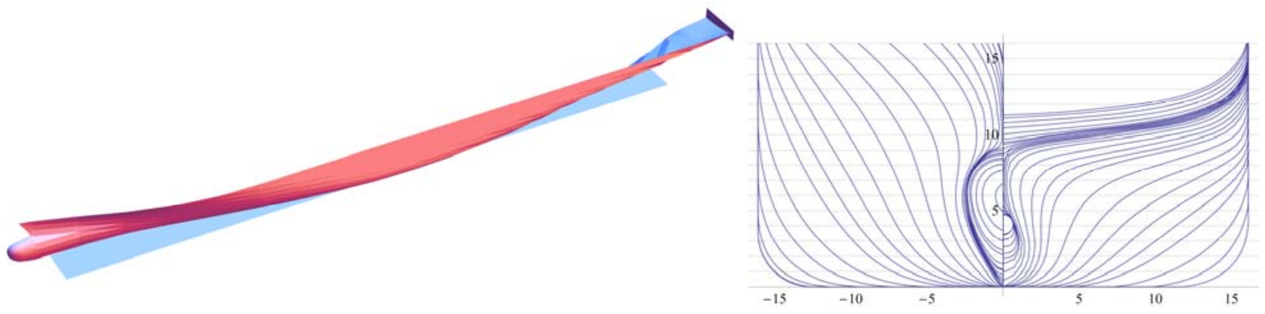


Figure 3 The KRISO Container Ship (KCS).

The main characteristics of the ship model are given as:

- Model length between perpendiculars, L_{pp}=7.2786 m.
- Model draft, d=0.3418 m.
- Model wetted surface area, S_0=9.4379m^2.
- Model speed, V_m=2.196 m/s.

5 MS EXCITING

The MS Exciting is a modified ship of commercial interest. The main characteristics of the ship model are given as:

- Length between perpendiculars, L_{pp}=277 m.
- Breath, B = 32.2 m.
- Draft, d=13 m.
- Displacement, Δ = 72993 m^3.
- LCB = 130.35 m.

The model is depicted in **Figure 4**.

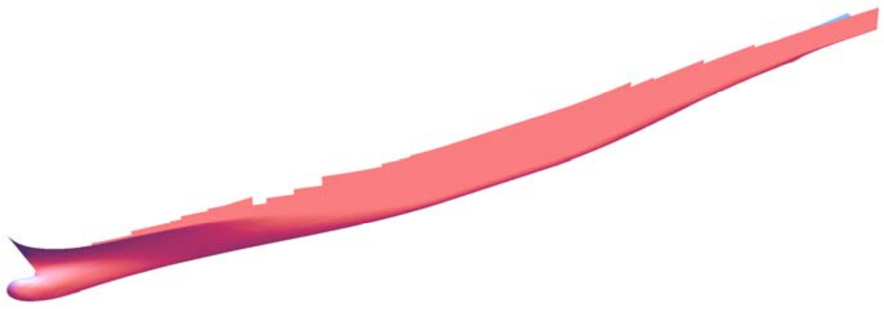


Figure 4 The MS Exciting container ship.

6 REFERENCES

/1/

- o0o -

C WP3

Contribution to D6.1

Workflow of the Structural Solver in WP3

A.-V. Vuong, B Simeon

Technische Universität München

Contribution to D 6.1: Workflow of the Structural Solver in WP3

A.-V. Vuong, B. Simeon

March 19, 2009
Technische Universität München

In this report we state the main workflow of the structural solver to be generated in WP3 of the EU-Project EXCITING and state (geometrical) requirements and interfaces, especially for integration and coordination of WP6 Isogeometric Toolbox.

1 Workflow of structural solver

We devide the workflow into three main parts: Preprocessing, simulation and postprocessing. A graphical overview is given in Fig. 1.

1.1 Preprocessing

The preprocessing for the structural solver represents one of the main interfaces between Computer Aided Geometric Design and Numerical Simulation.

In the current state of the project, there are several possibilites for us to obtain geometric models. Some were provided by JKU, which were generated as swept volumes. Ideal would be the possibility to use models provided by SIEMENS in a *common CAD format* [1]. The models typcally represent mechanical components with different geometrical complexity. In this special case the models by SIEMENS were provided in the IGES and STEP format. These files and other cases can be found in the eRoom
(`Work packages > WP3 Car Components and Frames > Siemens_cad.zip`)
Currently the generation of volume meshes from these surface models is not achieved. We find this step to be crucial for application purposes.

Another possibility favored by us would be a relatively simple method to generate geometries by ourselves for extended testing. Of course we are open-minded for alternatives made available by WP6.

After having obtained a geometrical model, it will be transformed in a geometric format (which is not necessarily a typical CAD format) that can serve as an input for the structural

[1] Which one is most suitable for the cause of EXCITING is currently unknown to us

solver. Additional information about the problem of the partial differential equation as for example boundary values and parameters for the numerical calculation are provided in a seperate input file or is merged or attached to the geometric information.

Together with JKU we have gained experience with this kind of problems by using the Code by the workgroup in Austin. Still all the geometric volumes in use were generated out of the "raw" geometric data, calculated by hand or provided by JKU. These data (degrees, knot vectors, control points and weights) were transformed by us into the Austin input format[2] expanded with additional parameters for simulation.

1.2 Simulation

The simulation follows the concept of Isogeometric Analysis by using NURBS basis function as the ansatz space for the Galerkin projection. For numerical integration the evaluation of the basis functions and their derivates (because of differential operators and the transformation of integrals) has to be performed. While the evaluation is also a typical CAD routine, an efficient execution should be achieved. We are also open for quadrature based on this, but it may also be necessary to modify the quadrature rules due to numerical reasons.

Another typical step during the simulation is the refinement process either for providing enough degrees of freedom on uniform grid (eventually needed if the geometric description is not very "complicated") or for local refinement. This can either be done by knot insertion (one single knot or division of all knot spans) and degree elevation. These should be standard procedures, and we assume that an efficient implementation is available or expandable to volumes. This task also influences the numerical parameters due to e.g. the change of the number of intervalls and therefore has also to be linked with the numerical PDE part.

The core of the structural solver is based on numerical routines for assembling the stiffness matrix and the right hand side and the solution of the resulting linear equation or eigenvalue problems. This is the responsibility of WP3, and there is just a need to coordinate which numerical libraries should be included into the toolbox.

1.3 Postprocessing

After having obtained the numerical solution it is important to visualise the results. It should be kept in mind that the data calculated by the solver is not directly the solution as in "Standard"-FEM because we do not deal with Lagrange polynomials anymore. (In the FEM context, functions b_i with the property $b_i(x_j) = \delta_{ij}$ with nodes x_j are called "shape functions".) Therefore the solution has to be evaluated from of the coefficients (as well as the geometry has to be evaluated with knots, control points and weights).

We are unaware of a possibility to directly visualise scalar or vector fields on a CAD model, both given in a NURBS or B-Spline basis coefficients. Even if this task may not be easily overcome we nevertheless are open for ideas and experience which software to use for visualisation.

Currently the MATLAB routines within the Austin solver are used, but we also experience the limits of this alternative. The performance is not very satisfying due to the fact that MATLAB is generally not designed for plotting polygons (which are used for approximating the NURBS volume for visualisation).

[2]Although the structural solver code by the group in Austin uses its own input format it may be beneficial to create a own format which better suits the requirements of EXCITING.

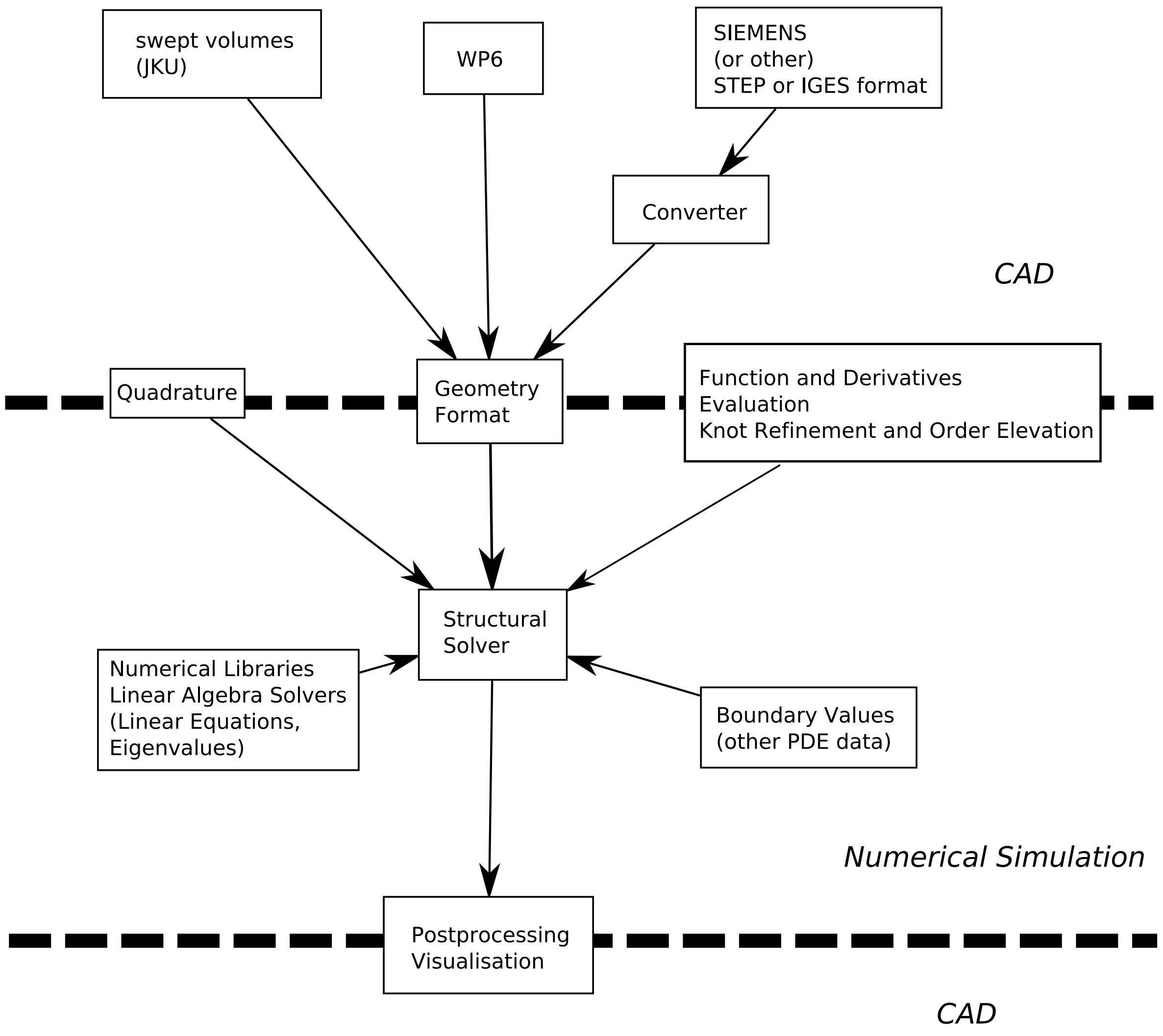


Figure 1: The workflow for the structural solver of EXCITING

2 Summary of Functionalities

Here we want to shortly summarise functionalities that could either already be available or could also be useful for other workpackages but are also out of the scope of our field of activity:

- simple geometry generation
- evaluation of NURBS/B-Splines and derivatives in three dimensions
- knot insertion (one knot or multiple knot over the whole knot vector) for NURBS volumes
- degree elevation for NURBS volumes
- visualisation of NURBS-Volumes
- visualisation of a scalar or vector field on a NURBS-Volume (both given by NURBS basis coefficients)
- generation of a volume mesh from its surfaces (or a standard CAD format)

We are quite optimistic that these requirements can be fulfiled within WP6.

3 Conclusions

As already stated we have the impression that our requirements for WP6 focus on algorithms and routines already well covered by accessable software. The main difference would be the usage for trivariate NURBS and the embedding into the context of numerical simulation. Therefore we come to the conclusion that the main challenge from our point of view would be the integration of several routines of CAD and numerical simulation to work together.

A collection of problems and benchmark examples will be provided in D 3.1.

Of course other requirements may arise during the project and may be specified, after we have gained more experience with advanced concepts as, for example, multipatch simulation.

D WP4

Contribution to D6.1

Requirements for fluid-structure interaction algorithms

Christoph Heinrich and Dr. Stefan Boschert

Siemens

Requirements for fluid-structure interaction algorithms

EXCITING, Contribution to D6.1

February 26, 2009

Work package 4: Fluid-structure interaction (FSI)
Siemens CT PP 2
Christoph Heinrich
Dr. Stefan Boschert

1 A short outline of fluid-structure interaction (FSI)

1.1 The problem

In fluid-structure interaction, as the name implies, two different physical fields, fluid and structure, interact with each other. In order to illustrate some basics of FSI, let us assume a very simple geometry (see figure 1). The two fields are non-overlapping and they contact each other at a common boundary, the so-called 'interface'.

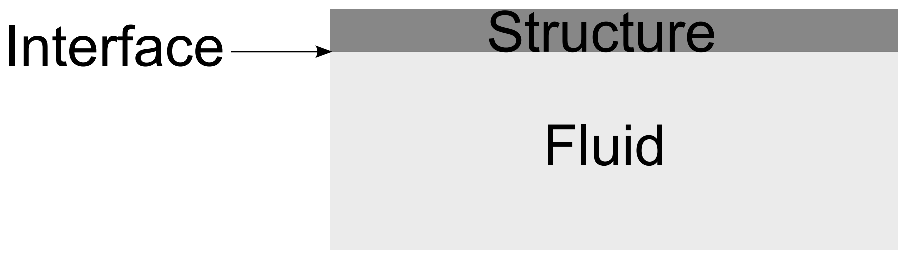


Figure 1: FSI schematically

The numerical simulation of such problems is a challenging task, which gets clear in the following. In the EXCITING project we aim to develop a so-called 'partitioned' solver, meaning that both fields, fluid and structure, are solved separately with single field solvers (These solvers will be developed in workpackages 1 and 3, respectively). The 'interaction' is integrated by the help of boundary conditions, which are transferred from one field to the other. This is shown in detail in subsection 1.2, where two different groups of partitioned solvers are presented, weak coupling and strong coupling. The latter should be the goal at the end of the project. But we think, that even a weak coupling scheme already contains the major difficulties. So if all problems in connection with a weak coupling scheme are solved, a strong coupling scheme should not be so far away. Nevertheless some additional requirements are necessary (see the last item in 'Ingredients of FSI algorithms' below).

1.2 Coupling schemes

Coupling algorithms in connection with FSI can be divided in monolithic and partitioned methods. We concentrate on the latter ones, which can be further subdivided in weak and strong coupling schemes:

- **Weak coupling**

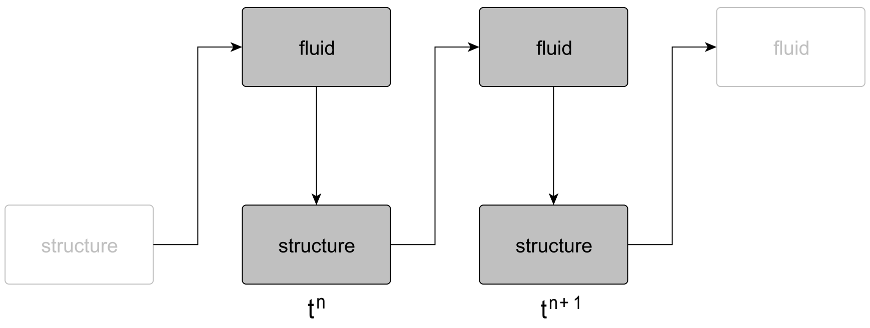


Figure 2: Weak coupling scheme

In every timestep the following steps have to be performed (cf. fig. 2):

1. Solve the fluid equations. (Fluid solver of workpackage 1)
2. With the help of step 1 compute the fluid forces at the interface and transfer them as a Neumann boundary condition to the structure.
3. Solve the structural equations. (Solid solver of workpackage 3)
4. Move the fluid mesh according to the displacement of the interface.

This is called an 'FSI cycle'. Of course step 4 can also be written before step 1. As a mesh moving algorithm is necessary to move the fluid mesh, the physical two-field problem gets an algorithmic three-field problem. Because of the fact that certain coupling conditions are not fulfilled in a weak coupling scheme leading to instability problems, a strong coupling scheme has to be preferred.

- **Strong coupling**

In a strong coupling scheme the weak coupling algorithm described above is extended by a loop over the fields in every timestep. As a consequence the single field

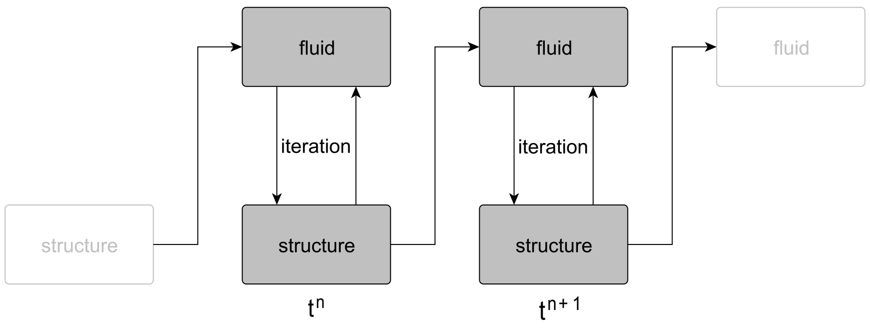


Figure 3: Strong coupling scheme

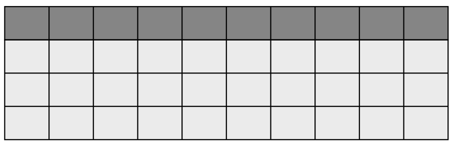
solvers must have the ability to run the same timestep several times. In combination with a fixed-point algorithm or Newton's method the coupling conditions at the interface can be fulfilled up to a certain tolerance and thus more stable results can be achieved.

Ingredients of FSI algorithms

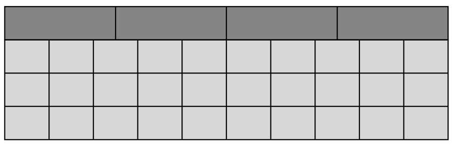
- Structural solver: both stationary and instationary, large displacements should be possible at the end
- Fluid solver: both stationary and instationary, ALE-description
- Solver for the mesh movement: e.g. with the help of a Laplace equation for the mesh velocity u^g: $\nabla \cdot (\kappa \nabla u^g) = 0$ with diffusivity κ; should be included in the fluid solver.
- easy access to the interface variables (displacements); easy computation of forces at the interface. These things are essential for all partitioned algorithms!
- for a strong coupling scheme: both solvers must be able to run the same timestep several times. It should be possible to increase the timestep manually from within the FSI coupling algorithm!

2 FSI and isogeometric analysis

In order to demonstrate additional aspects of fluid-structure interaction in an isogeometric framework, we assume for simplicity a 2D geometry together with grids for both physical fields. We distinguish between 'matching' and 'non-matching' grids (see figures 4a and 4b, respectively).



(a) matching grids



(b) non-matching grids

Figure 4: Initial fluid and structural grids

The fluid grid and the fluid domain are given by the knot vectors $\mathbf{u}^f$ and $\mathbf{v}^f$ together with control points $\mathbf{P}^f$ and weights. The corresponding expressions for the structural field are $\mathbf{u}^s$, $\mathbf{v}^s$ and $\mathbf{P}^s$. We assume quadratic NURBS basis functions and open knot vectors, so that all knot vectors have the following form

$$[0,0,0,\ldots,1,1,1]\,.$$

The geometry functions, that map the parameter space to the physical space, can now be formulated as

$$\mathbf{x}^f\left(\xi^f,\eta^f\right) = \mathbf{R}^f \cdot \mathbf{P}^f \qquad \text{for the fluid domain and}$$
$$\mathbf{x}^s\left(\xi^s,\eta^s\right) = \mathbf{R}^s \cdot \mathbf{P}^s \qquad \text{for the structural domain,}$$

where $\mathbf{R}^f$ and $\mathbf{R}^s$ represent matrices, that contain the corresponding rational basis functions. Now one FSI cycle (cf. subsection 1.2) reads in detail:

1. Solve the fluid equations, i.e. we get a pressure distribution

$$\mathbf{p}^f\left(\xi^f,\eta^f\right) = \mathbf{R}^f \cdot \hat{\mathbf{p}}$$

and a velocity distribution

$$\mathbf{u}^f\left(\xi^f,\eta^f\right) = \mathbf{R}^f \cdot \hat{\mathbf{u}}. \tag{1}$$

2. Transfer of fluid forces at the interface as a Neumann boundary condition to the structure. If we disregard shear forces, the fluid forces can be obtained from the pressure distribution at the interface $\mathbf{p}^f\left(\xi^f,1\right)$.

3. Solve the structural equations with the help of step 2. The structural displacement can be written as

$$\mathbf{d}^s\left(\xi^s,\eta^s\right) = \mathbf{R}^s \cdot \hat{\mathbf{d}}.$$

Thus the displacement at the interface is $\mathbf{d}^s\left(\xi^s,0\right)$.

4. Move the fluid mesh according to the displacement of the interface. Therefore solve the fluid mesh equations for the fluid mesh velocity $\mathbf{u}^g$, e.g. solve a Laplace

equation $\nabla\cdot(\kappa\nabla\mathbf{u}^g)$ with an appropriate diffusivity κ. At the interface the Dirichlet boundary condition $\mathbf{u}^g = \mathbf{d}^s\left(\xi^s, 0\right)$ is imposed and we get

$$\mathbf{u}^g\left(\xi^f, \eta^f\right) = \mathbf{R}^f \cdot \hat{\mathbf{u}}^g.$$

Now the new shape of the fluid domain $\bar{\mathbf{x}}^f$ can be obtained easily:

$$\bar{\mathbf{x}}^f\left(\xi^f, \eta^f\right) = \mathbf{x}^f\left(\xi^f, \eta^f\right) + \Delta t \cdot \mathbf{u}^g\left(\xi^f, \eta^f\right) = \mathbf{R}^f \cdot \mathbf{P}^f + \Delta t \cdot \mathbf{R}^f \cdot \hat{\mathbf{u}}^g = \mathbf{R}^f \underbrace{\left(\mathbf{P}^f + \Delta t \cdot \hat{\mathbf{u}}^g\right)}_{:=\bar{\mathbf{P}}^f}.$$

Conclusion

From a present-day perspective we don't see any fundamental problems in addition to the classical FSI requirements mentioned above (cf. 'Ingredients of FSI algorithms') for matching grids. The more general and the desirable case is 'non-matching' grids. Here we face the difficulty at the interface that fluid forces or displacements possibly cannot be represented in the basis of the other field. As a consequence one must be careful with surface integrals at the interface that arise from the boundary conditions there.

3 Cases

3.1 2D example 'Driven cavity with flexible bottom'

The 'Driven cavity' example is one of the most popular benchmarks in FSI. As a consequence this will be a major test case in EXCITING. Figure 5 shows the geometry and some of the boundary conditions (dark grey: structure, light grey: fluid). At the top of the fluid partition a time-dependent oscillating velocity u_x is imposed as Dirichlet boundary condition. Therefore an instationary FSI-solver is essential. A possible shape

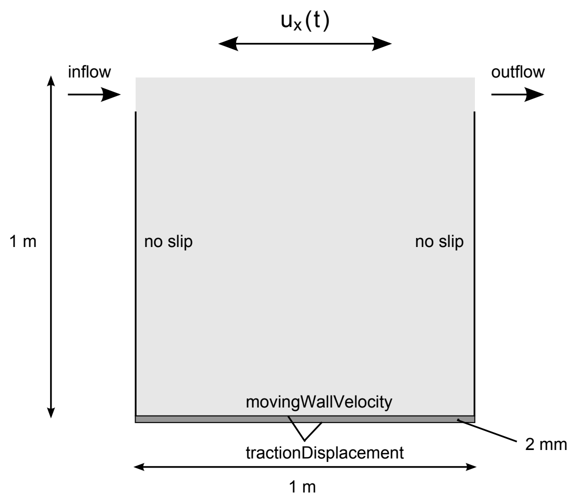


Figure 5: Geometry and boundary conditions of 'Driven cavity with flexible bottom'

of the fluid domain and the corresponding distribution of the total velocity together with streamlines can be seen in figure 6.

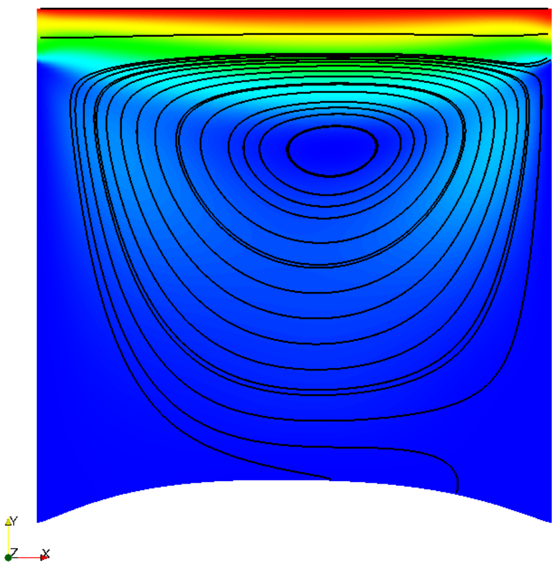


Figure 6: Possible total velocity distribution and streamlines at a certain timestep of 'Driven cavity with flexible bottom'

3.2 2D example 'Obstacle in flow'

Another 2D example is called 'Obstacle in flow', also widely used as benchmark in literature. As before the fluid and the structural domain are coloured light grey and dark grey, respectively (see figure 7). On the left-hand side of the fluid region an 'inflow'-boundary condition is imposed with a parabolic profile. For this case, also a stationary analysis is thinkable. Figure 8 shows a possible (total) velocity distribution at the stationary point.

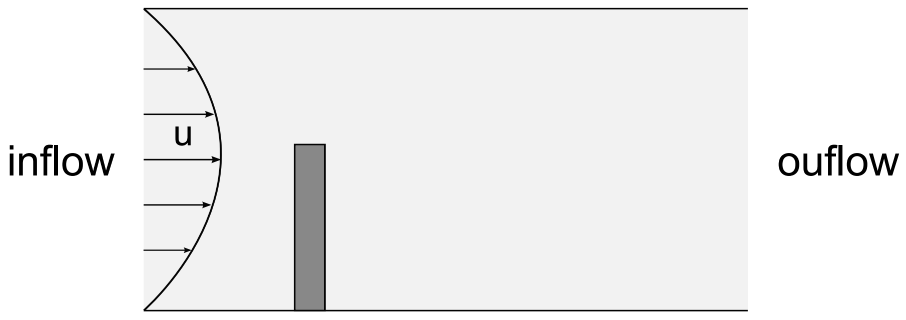


Figure 7: Geometry of 'Obstacle in flow'; structure (dark grey) and fluid (light grey)

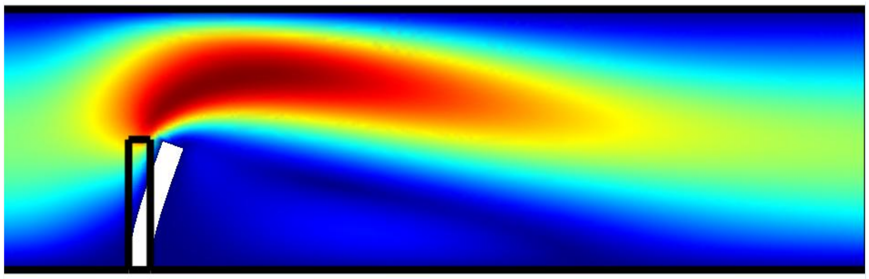


Figure 8: Total velocity distribution of ‘Obstacle in flow’ at the stationary point

3.3 3D example ‘Pipe with deformable wall’

One of the easiest 3D examples is the ‘Pipe with deformable wall’ test case, another benchmark. This should be our first 3D-FSI-example in EXCITING. The structural part consists of a hollow cylinder and the fluid domain is everything inside of it (see figure 9). Once again at the left hand side of the fluid domain an ‘inlet’-boundary condition is imposed. Figure 10 shows two slices of the velocity distribution of the fluid together with the total displacement of the structural part at the stationary point. Of course the 2D case should also be used as benchmark.

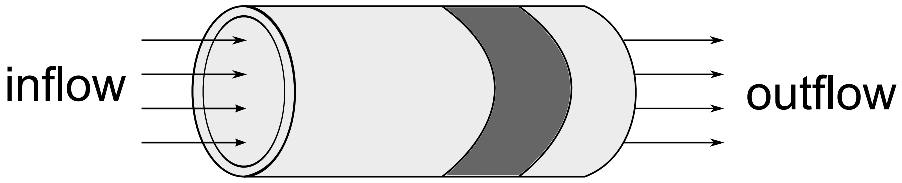


Figure 9: Geometry of ‘Pipe with deformable wall’

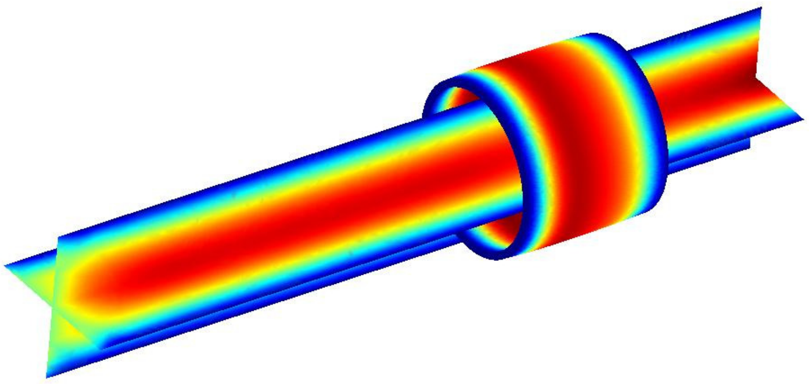


Figure 10: ‘Pipe with deformable wall (3D)’: Velocity distribution of the fluid (two slices) and total displacement of the structural part at the stationary point