

Overview documentation, gotools core

Vibeke Skytt

June 17, 2010

1 Geometric Data Structures

Figure 1 shows the main geometric classes in GoTools and how they are divided between the modules. The current version of the isogeometric toolbox is to a large extent concerned with the entities in this figure, operations on and construction of these entities.

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

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

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

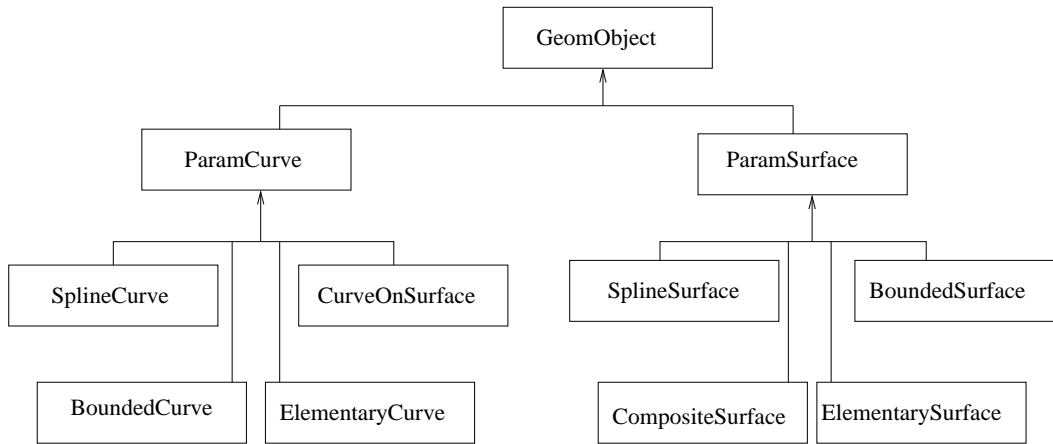


Figure 1: Simplified overview of the geometry class hierarchy

- closest point computations
- append another curve to the current curve
- check for degeneracy
- make a copy of itself

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

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

- parameter domain operations; fetch the domain and the rectangular surrounding domain, check if a parameter pair lies in the domain of a surface, turn the directions of the parameter domain
- evaluation
- fetch constant parameter curves
- get the boundary curves surrounding this surface
- compute bounding box

- compute the directional cone surrounding all surface normal directions of the current surface
- area computations
- fetch a sub surface of this surface
- check for degeneracy
- make a copy of itself

2 B-spline Curves

A B-spline curve is represented by `SplineCurve`.

The curve is defined by the formula

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

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

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

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

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

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

The complete representation of a B-spline curve consists of

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

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

k : The order of the B-splines.

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

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

We note that arrays in c start at index 0 which means, for example, that if the array t holds the knot vector, then $t[0] = t_1, \dots, t[n+k-1] = t_{n+k}$ and the parameter interval goes from $t[k-1]$ to $t[n]$. Similar considerations apply to the other arrays.

The data in the representation must satisfy certain conditions:

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

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

2.1 B-spline Basis Functions

The spline space is represented by the class `BsplineBasis`. A set of B-spline basis functions is determined by the order k and the knots. For example, to define a single basis function of degree one, we need three knots. In figure 2 the three knots are marked as dots. Knots can also be equal as shown in figure 3. By taking a linear combination of the three basis functions shown in figures 2 and 3 we can generate a linear spline function as shown in figure 4.

A quadratic B-spline basis function is a linear combination of two linear basis functions. Shown in figure 5 is a quadratic B-spline defined by four knots. A quadratic B-spline is the sum of two products, the first product between the linear B-spline on the left and a corresponding line from 0 to 1, the second product between the linear B-spline on the right and a corresponding line from 1 to 0; see figure 5. For higher degree B-splines there is a similar

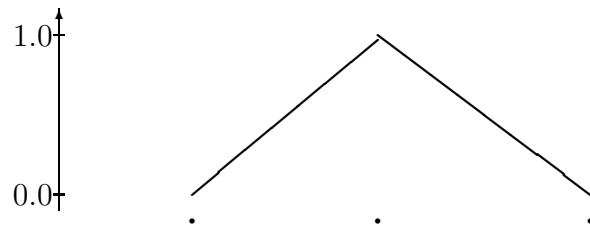


Figure 2: A linear B-spline (order 2) defined by three knots.

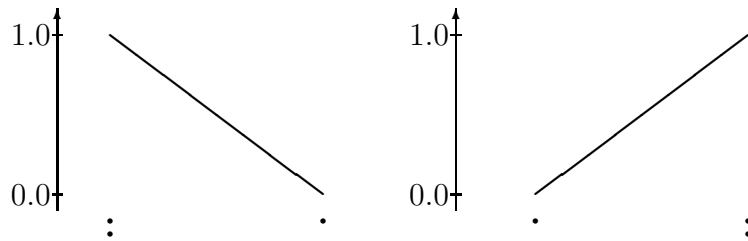


Figure 3: Linear B-splines of with multiple knots at one end.

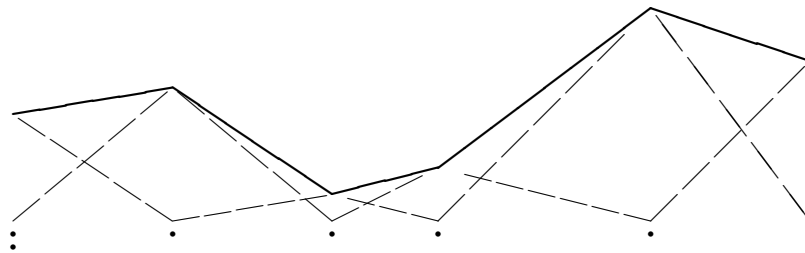


Figure 4: A B-spline curve of dimension 1 as a linear combination of a sequence of B-splines. Each B-spline (dashed) is scaled by a coefficient.

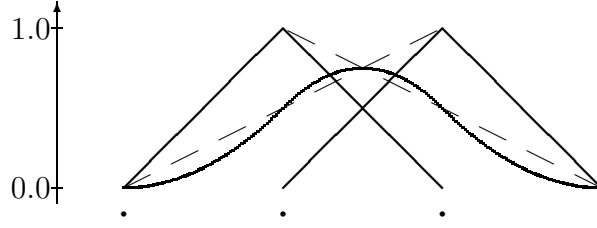


Figure 5: A quadratic B-spline, the two linear B-splines and the corresponding lines (dashed) in the quadratic B-spline definition.

definition. A B-spline of order k is the sum of two B-splines of order $k - 1$, each weighted with weights in the interval $[0,1]$. In fact we define B-splines of order 1 explicitly as box functions,

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

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

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

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

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

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

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

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

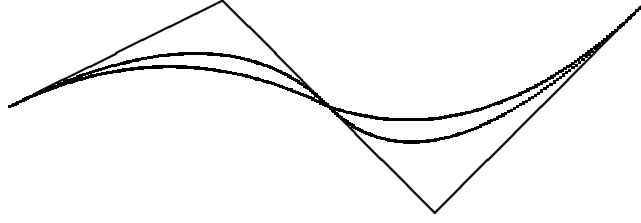


Figure 6: Linear, quadratic, and cubic B-spline curves sharing the same control polygon. The control polygon is equal to the linear B-spline curve. The curves are planar, i.e. the space dimension is two.

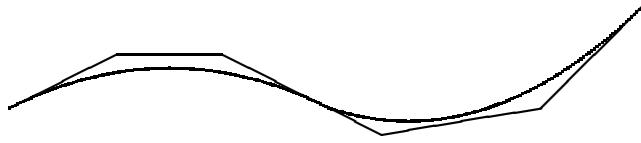


Figure 7: The cubic B-spline curve with a redefined knot vector.

2.2 The Control Polygon

The control points \mathbf{p}_i define the vertices. The *control polygon* of a B-spline curve is the polygonal arc formed by its control points, $\mathbf{p}_0, \mathbf{p}_1, \dots, \mathbf{p}_n$. This means that the control polygon, regarded as a parametric curve, is itself a piecewise linear B-spline curve (order two). If we increase the order, the distance between the control polygon and the curve increases (see figure 6). A higher order B-spline curve tends to smooth the control polygon and at the same time mimic its shape. For example, if the control polygon is convex, so is the B-spline curve.

Another property of the control polygon is that it will get closer to the curve if it is redefined by inserting knots into the curve and thereby increasing the number of vertices; see figure 7. If the refinement is infinite then the control polygon converges to the curve.

2.3 The Knot Vector

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

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

In figure 8 we have two curves with the same control polygon and order but with different parameterization.

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

The number of equal knots determines the degree of continuity. If k consecutive internal knots are equal, the curve is discontinuous. Similarly if $k - 1$ consecutive internal knots are equal, the curve is continuous but not in general differentiable. A continuously differentiable curve with a discontinuity in the second derivative can be modelled using $k - 2$ equal knots etc. (see figure 9). Normally, B-spline curves in SISL and GoTools are expected to be continuous. For many algorithms, curves should be continuously differentiable (C^1).

2.4 NURBS Curves

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

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

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

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

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

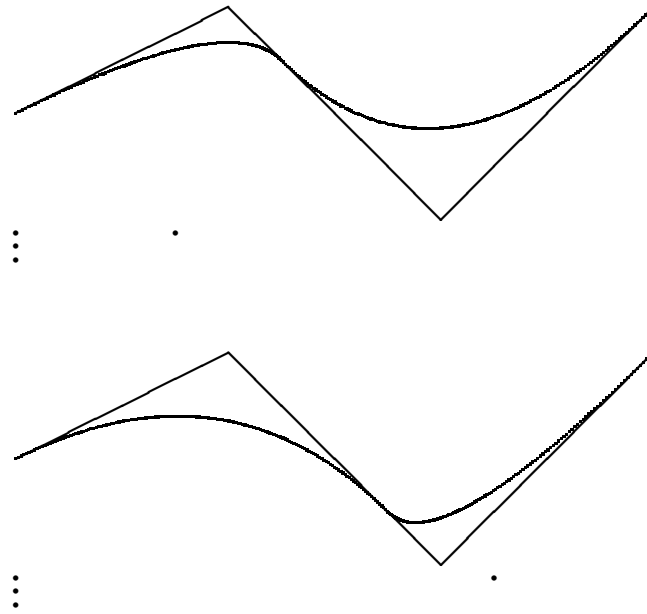


Figure 8: Two quadratic B-spline curves with the same control polygon but different knot vectors. The curves and the control polygons are two-dimensional.

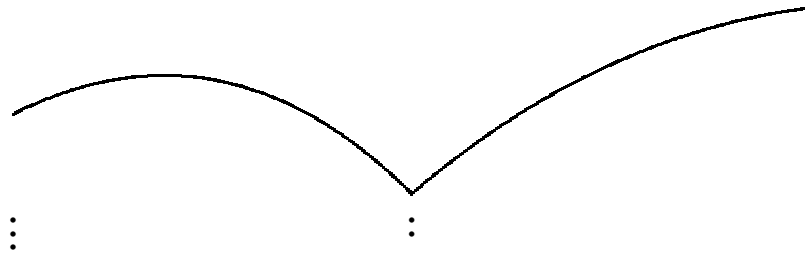


Figure 9: A quadratic B-spline curve with two equal internal knots.

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

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

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

2.5 Spline Curve Functionality

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

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

3 B-spline Surfaces

A tensor product B-spline surface is represented by the class SplineSurface. the B-spline surface is defined as

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

with control points $\mathbf{p}_{i,j}$ and two variables (or parameters) u and v . The formula shows that a basis function of a B-spline surface is a product of two

basis functions of B-spline curves (B-splines). This is why a B-spline surface is called a tensor-product surface. The following is a list of the components of the representation:

dim : The dimension of the underlying Euclidean space.

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

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

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

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

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

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

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

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

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

The properties of the representation of a B-spline surface are similar to the properties of the representation of a B-spline curve. The control points $\mathbf{p}_{i,j}$ form a *control net* as shown in figure 10. The control net has similar properties to the control polygon of a B-spline curve, described in section 2.2. A B-spline surface has two knot vectors, one

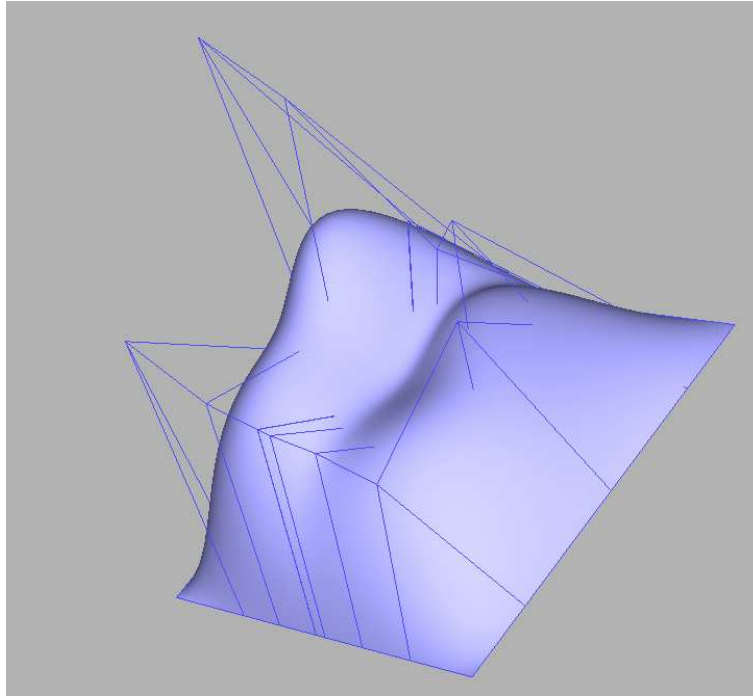


Figure 10: A B-spline surface and its control net.

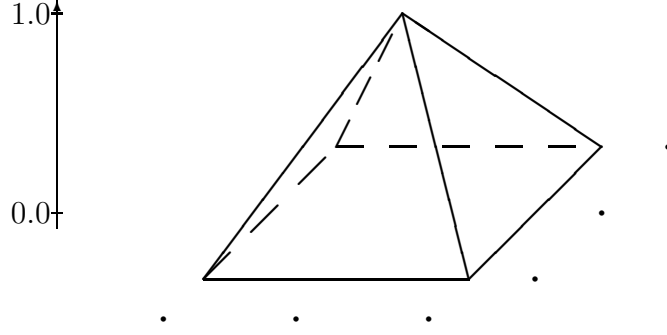


Figure 11: A basis function of degree one in both variables.

3.1 The Basis Functions

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

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

Its support is the rectangle $[u_i, u_{i+k_1}] \times [v_j, v_{j+k_2}]$. If the basis functions in both directions are of degree one and all knots have multiplicity one, then the surface basis functions are pyramid-shaped (see figure 11). For higher degrees, the surface basis functions are bell shaped.

3.2 NURBS Surfaces

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

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

In addition to the data of a B-spline surface, the NURBS surface has a weights $w_{i,j}$. NURBS surfaces can be used to exactly represent several common ‘analytic’ surfaces such as spheres, cylinders, tori, and cones. A disadvantage is that NURBS surfaces depend nonlinearly on their weights, making some calculations less efficient.

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

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

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

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

3.3 Spline Surface Functionality

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

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

4 Other Curve Types

As shown in figure 1, a spline curve is not the only parametric curve in `GoTools` although it is the most important one. There are also other curves that share parts of the public interface:

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

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

CurveOnSurface This class represents a curve lying on a surface.

4.1 Curve On Surface

The CurveOnSurface entity is often related to a bounded, or trimmed, surface. A bounded surface is limited by a curve loop where each curve in the loop most often is represented by a CurveOnSurface. However, a CurveOnSurface is an entity in its own right. It simply represents a curve lying on a surface. It can for instance originate from intersecting the surface with a plane.

A CurveOnSurface consists of a parametric surface and two corresponding curves, one curve in the parameter domain of the surface and one spatial curve. The master representation is specified. Some operations may require the existence of one such curve, in particular the parameter curve is requested. It exists functions that compute the missing curve from the existing one.

4.2 Elementary Curve

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

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

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

ellipse is parameterized in terms of the angle which gives a bounded parameterization.

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

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

5 Other Surface Types

Similar to the curve case, GoTools supports also other types of parametric surfaces than spline surfaces, see figure 1. The additional surface types are bounded surface, elementary surface and composite surface. The elementary surfaces are conic surfaces, plane and torus. A composite surface consists of a number of spline surfaces where the associated parameter domains are mapped into one composite domain. The class is not complete in the sense that it does not support all functions specified in the interface for a parametric surface.

5.1 Bounded Surface

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

5.2 Elementary Surface

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

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

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

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

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

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

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

6 Geometry Construction

The combination of SISL and GoTools provide a lot of methods for geometry construction. They partly overlap. The SISL geometry construction methods are described in the SISL manual.

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

6.1 Curve Construction

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

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

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

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

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

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

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

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

- Given a curve in the parameter domain of a surface, the geometry space curve is evaluated.
- Represent an offset curve from a given space curve. The direction of the offset is given as an expression of corresponding curves.

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

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

6.2 Surface Construction

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

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

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

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

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

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

7 Tessellation

The geometry entities are tessellated in gotools-core in the sub module tessellator. Depending on the type of the geometry object, the result of the tessellation is stored in GenericTriMesh, LineStrip, QuadMesh or RegularMesh that all inherit the abstract class GeneralMesh. The tessellators in this module relate to a given resolution in each parameter direction of the geometry object.

A curve is tessellated by the class CurveTessellator and the tessellated curve is returned by the function CurveTessellator::getMesh() as a shared pointer to a LineStrip object. A rectangular surface is tessellated in RectGridTessellator and the tessellator returns a shared pointer to a RegularMesh. The triangles produced during tessellation is organized in a set of triangle strips. This structure is taken advantage of to improve the drawing efficiency. A trimmed surface is tessellated by ParametricSurfaceTessellator and the tessellation is represented as an unorganized bunch of triangles. The output from the functionality in the tessellation module is taken as input to the GoTools viewers and the actual drawing is performed using OpenGL.