Python API: Geometry

1. TiGL Workshop, September 11 / 12, Cologne

Martin Siggel + Jan Kleinert
German Aerospace Center
Motivation

- We had different problems with the surfaces created by OpenCASCADE. Reasons are:
  - We used OpenCASCADE as a black box (we had no background in B-splines back then)
  - The resulting surfaces have sometimes bad quality
  - Not all modelling algorithms available that we needed

- We started developing our own algorithms 😊

- Even if you don't use CPACS oder TiGL's aircraft models, you can now use the algorithms in your own applications

We show you now how to do it!
Big Picture
The shape creation process

- Named Shapes
  - Metadata: by TiGL

- Solids
  - Topology:
    - How do faces, edges belong together?
    - only OpenCASCADE

- Shells

- Surfaces
  - Geometry:
    - How do create curves and surfaces?
    - using new TiGL algorithms
    - or with OpenCASCADE

- Curves

- Points
Beizer / B-splines / NURBS
The basis for all geometry representations in OpenCASCADE

- **B-spline curve:**
  \[ c(u) = \sum_{i=0}^{n} P_i \ast N_i^d(u, t) \]
  
  with:
  - Control points \( P_i \)
  - B-spline basis functions \( N_i^d(u, t) \)
  - Knot vector \( t, t_i \leq t_{i+1} \)

- **B-spline surface:**
  \[ s(u, v) = \sum_{i=0}^{n} \sum_{j=0}^{m} P_{ij} \ast N_i^{d_u}(u, t_u) \ast N_j^{d_v}(v, t_v) \]
B-spline curve interpolation
Or: PointsToCurve

- Solve control points $P_i$, given data points $D_j$, such that:

$$ \sum_{i=0}^{n} P_i \times N_i^d(u_j, t) = D_j $$

$\Rightarrow Np \equiv d$

i.e. the curve passes though the data points
Creating curves with TiGL

Curve Factories

- The package `tigl3.curve_factories` provides functions to create B-spline curves

- B-spline Interpolation:

```python
import tigl3.curve_factories

# array of 3d points
points = [[0, 0, 0], [1, 0, 0], [1, 3, -1], [0, 0, 0]]

# create the curve
curve = tigl3.curve_factories.interpolate_points(points)

parameters = [0., 0.2, 0.7, 1.0]

# create the curve
curve = tigl3.curve_factories.interpolate_points(points, parameters, degree=2)
```

- Even better: control at which curve parameter each point is interpolated!
B-spline surface skinning
Or: CurvesToSurface

• Interpolates set of B-spline curves \( c_i(u) \) by B-spline surface \( s(u, v) \)

• Also involves solving multiple linear systems
Creating surfaces with TiGL
Surface Factories

• The module `tigl3.surface_factories` provides functions to create B-spline surfaces

• Skinning a set of curves:

```python
import tigl3.surface_factories

# create the surface
surface = tigl3.surface_factories.interpolate_curves([curve1, curve2, curve3, ... curveN])
```

• Similar to curve interpolation – define a set of parameters at which each curve should be interpolated:

```python
parameters = [0., 0.333, 1.0]

# create the surface
surface = tigl3.surface_factories.interpolate_curves([curve1, curve2, curve3], parameters)

# or control the degree. degree=1 is linear lofting. Default degree is 3
surface = tigl3.surface_factories.interpolate_curves([curve1, curve2, curve3], degree=2)
```
Gordon Surfaces
Or: Curve network interpolation

- Given network of profile and guide curves: Find surface that interpolates these curves

\[ G(u,v) = S_u(u,v) + S_v(u,v) - T(u,v) \]
Creating surfaces with TiGL

Surface Factories

• Interpolating a curve network (using the Merlin’s Gordon Surface technique):

```python
import tigl3.surface_factories

# interpolate the curve network of profiles and guides
surface = tigl3.surface_factories.interpolate_curve_network(
    [profile1, profile2, profile3, ...],
    [guide1, guide2, ...],
    spatialTol=3e-4
)
```

• In theory: Profiles and Guides must all intersect each other exactly!

• In TiGL:
  • Parameter spatialTol defines the maximum allowed distance between a guide and a profile
  • If they don’t intersect exactly, the surface will be somewhere between both curves!
Other Geometry Algorithms in TiGL

• B-spline approximation: Fit a curve to a set of points
  → tigl3.geometry.CTiglBSplineFit

• B-spline representation of arbitrary analytical functions
  (e.g. How does the B-spline representation of CST curves look like?)
  → tigl3.geometry.CFunctionToBSpline

• Hybrid B-spline approximation + interpolation of selected points
  → tigl3.geometry.CTiglBSplineApproxInterp

• Interpolate a grid of points with a surface
  → tigl3.geometry.BSplineAlgorithms_points_to_surface
Remarks on programming with the geometry module

- Geometric objects (curves and surfaces) are returned by a **Handle**

- OpenCASCADE Handles are **Smart Pointers** that automatically free memory, when not needed

- Unfortunately, this is still exposed in Python. We are working on it, to remove this from Python.

- Whenever you e.g. get a `Handle_Geom_BsplineCurve` and you need a `Geom_BsplineCurve`, call its `.GetObject()` method:

  ```python
  b_spline = b_spline_handle.GetObject()
  ```

- Whenever you have e.g. a `Geom_BsplineCurve` and you need its Handle, call its `.GetHandle()` method:

  ```python
  b_spline_handle = b_spline.GetHandle()
  ```
Excercise Geometry:

Goal:
Learn, how to use our geometry tools.

Tasks:
1. Create an airfoil by interpolating a list of points.
2. Use our geometry tools to skin multiple airfoils to create wing loft.
3. Use the curve network interpolation to create a wing loft with a custom leading edge.
Questions?