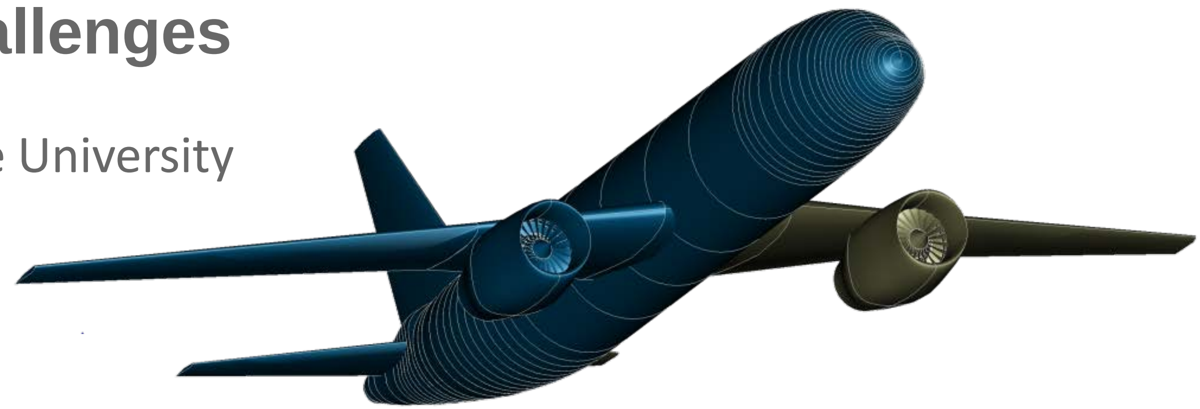
The TiGL Geometry Library and its Current Mathematical Challenges



Workshop DLR / Cologne University

06.10.2017

Dr. Martin Siggel

Merlin Pelz

Konstantin Rusch

Dr. Jan Kleinert

Simulation and Software Technology

High Performance Computing

DLR Cologne

Knowledge for Tomorrow

DLR

Outline

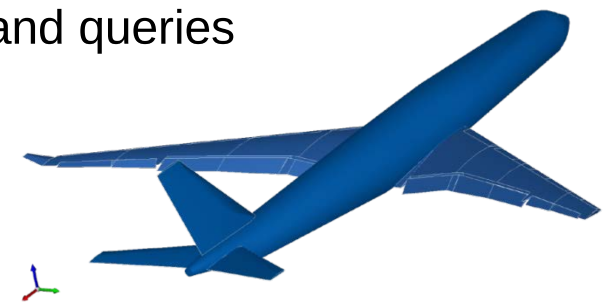
- What is the TiGL Library
- B-spline basics
- Surface Modelling: Gordon surfaces
    - Alternative to Coons Patches to interpolate curve networks
    - Solves problems with surface discontinuities
    - Resulting geometries
- Fitting aircraft engines with Free Form Deformation (FFD)
    - FFD Parametrization of aircraft engine cover
    - The minimization problem + regularization
    - Results
- Conclusion and Outlook

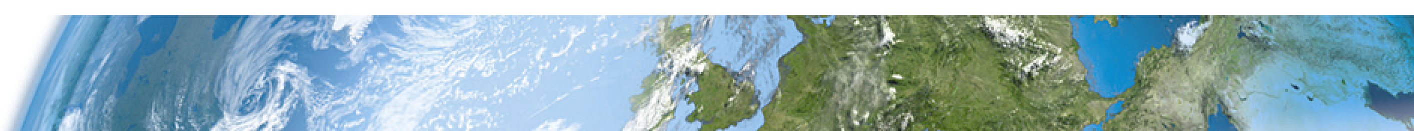
TiGL
What it does

- **CPACS** *(Common Parametric Aircraft Configuration Schema)* is an xml schema that describes the characteristics of an aircraft
  - includes a **parametric description** for airplane or rotor-craft geometries

- **TiGL**
  - generates geometries from CPACS files
  - provides an interface for geometry manipulation and queries

```xml
<positioning uID = "D150_wing_1Positioning3ID">
    <name>D150_wing_1Positioning3</name>
    <length>5.0495526583</length>
    <sweepAngle>27.418</sweepAngle>
    <dihedralAngle>5.0</dihedralAngle>
    <fromSectionUID>D150_wing_1Section2ID</fromSectionUID>
    <toSectionUID>D150_wing_1Section3ID</toSectionUID>
</positioning>
```

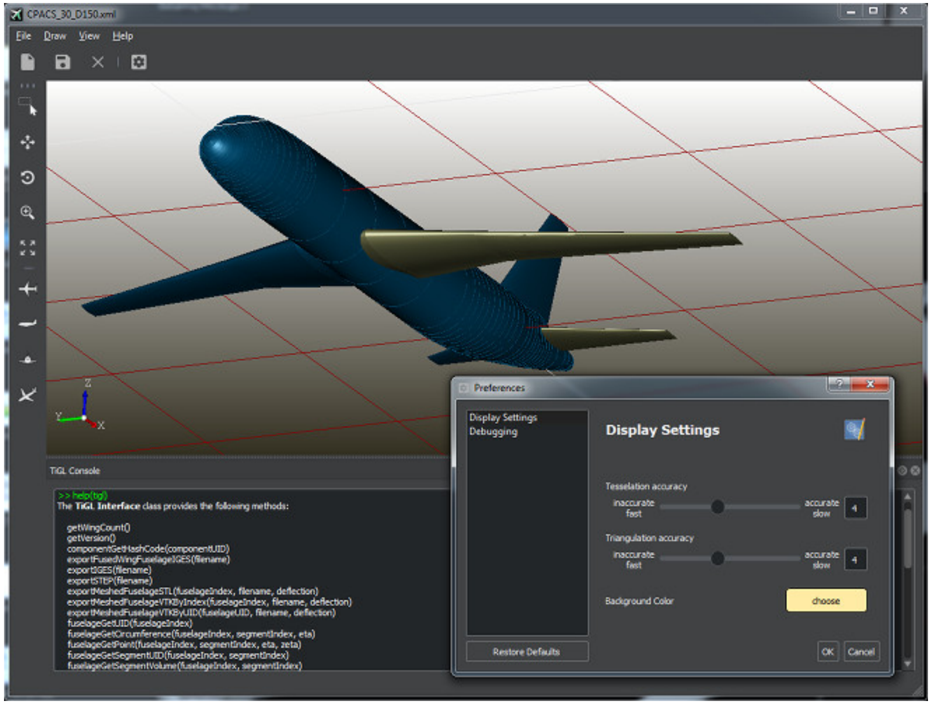




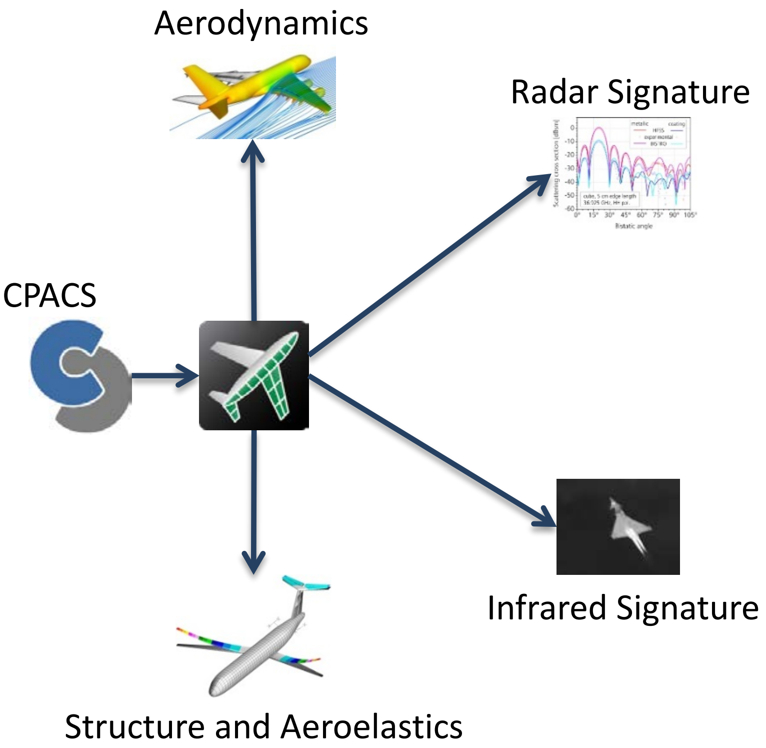
TiGL
What it is

- TiGL is a **C++ library** with interfaces to C, Java, Python, Matlab and Fortran
- Uses **Open CASCADE** CAD kernel to represent airplane geometries as B-spline / NURBS surfaces
- Ships **TiGL Viewer** to visualize aircraft geometries or other CAD files.
- Is **Open Source**, Apache 2.0

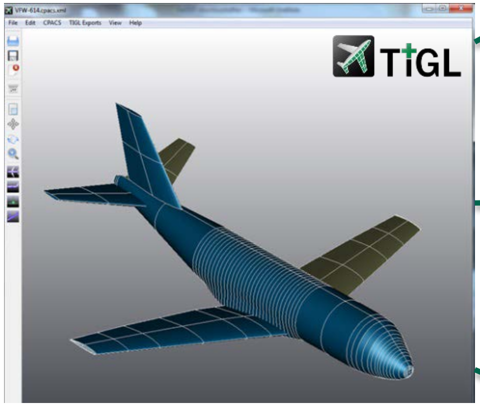
  https://github.com/DLR-SC/tigl
- Joint development



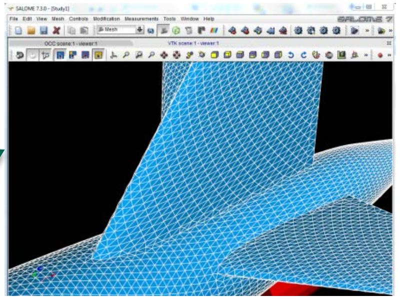
TiGL
How it is used



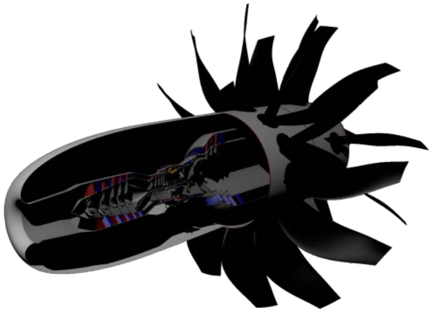
TiGL
How it is used



VTK, BRep, IGES



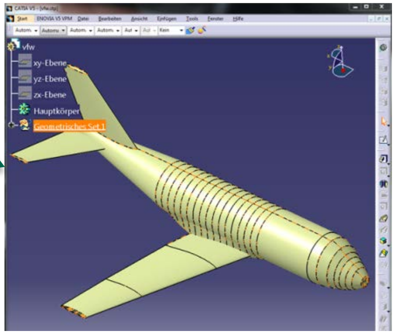
Mesh Generation for CFD Simulations



Collada, STL

Rendering, Visualization, 3D Printing

Step, IGES

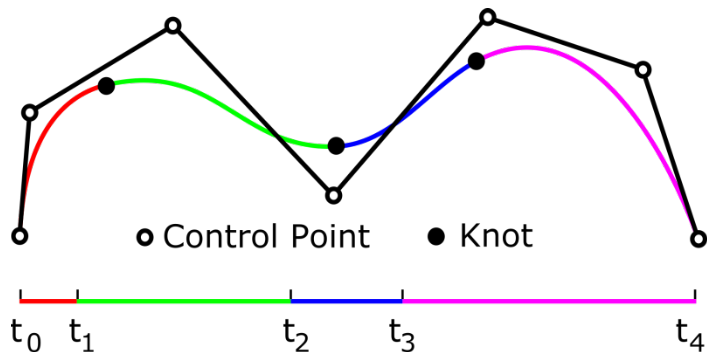


Modeling with CAD Systems

DLR

B-spline basics

B-spline basics
B-spline curves



Definition:

$$\boldsymbol{c}(u) = \sum_{i=0}^{n} \boldsymbol{P}_i^c * N_i^d(u, \boldsymbol{t})$$

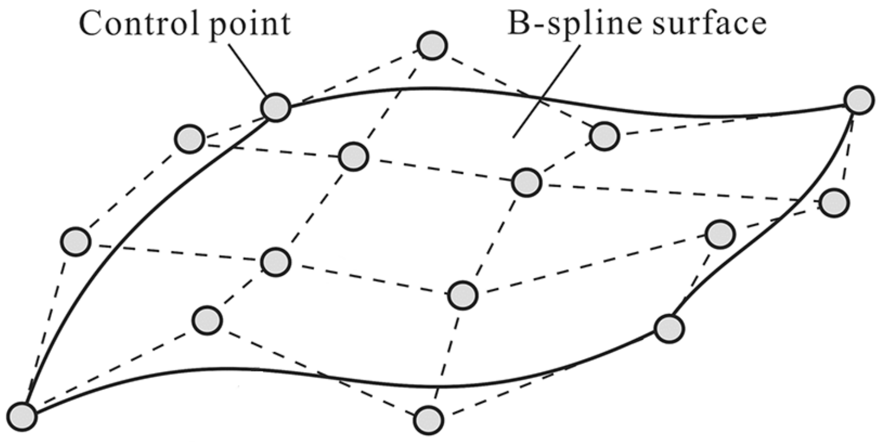
Linear in P!

with:

- Control points $\{P_i^c\}$
- B-spline basis functions $N_i^d(u, \boldsymbol{t})$
- Knot vector $\boldsymbol{t}$, $t_i \leq t_{i+1}$

DLR

B-spline basics
B-spline surfaces



Definition:

$$\boldsymbol{s}(u,v) = \sum_{i=0}^{n}\sum_{j=0}^{m} \boldsymbol{P}_{ij}^{s} * N_i^{d_u}(u,\boldsymbol{t}_u) * N_j^{d_v}(v,\boldsymbol{t}_v)$$

Linear in P!

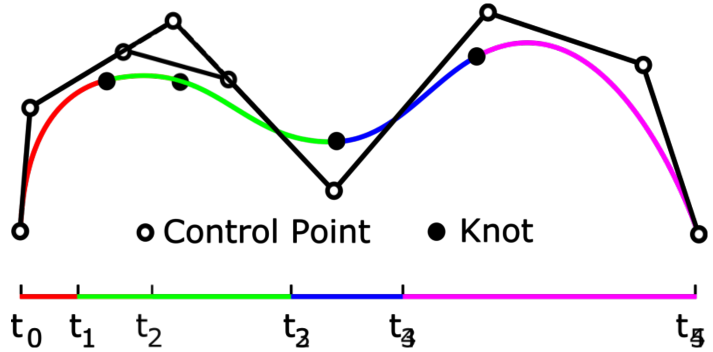
with:

- Control points $\{P_{ij}^{s}\}$
- B-spline basis functions $N_i^d(u,\boldsymbol{t})$
- Knot vectors $\boldsymbol{t_u}$, $\boldsymbol{t_v}$

B-spline basics

Algorithm: Knot Insertion

- Adds an entry to the knot vector *t*, without changing the curve shape



- Modifes control points and adds one control point!

- Knot insertion basis for many higher level algorithms

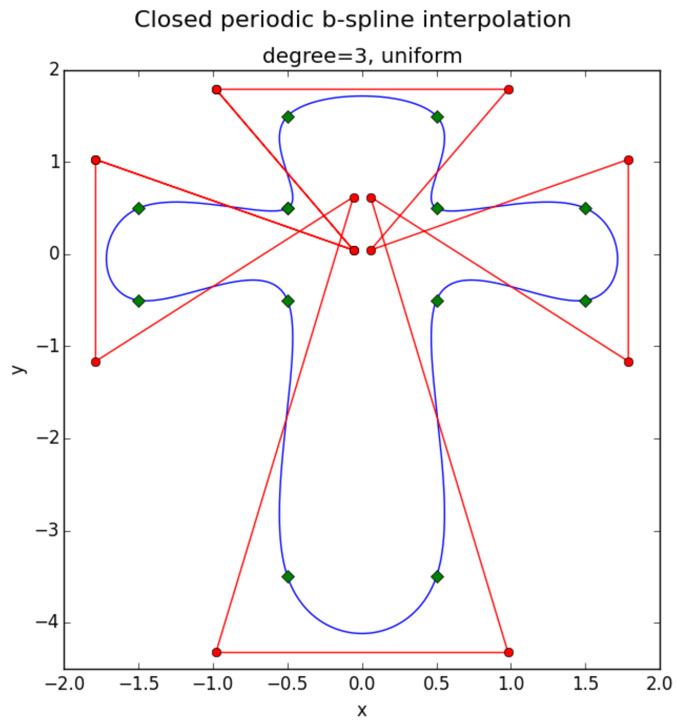
B-spline basics
Algorithm: B-Spline Curve Interpolation

- Given data points Pj, compute control points Ci, such that:

$$\sum_{i=0}^{n} \boldsymbol{C}_i * N_i^d(u_j, \boldsymbol{t}) = P_j$$

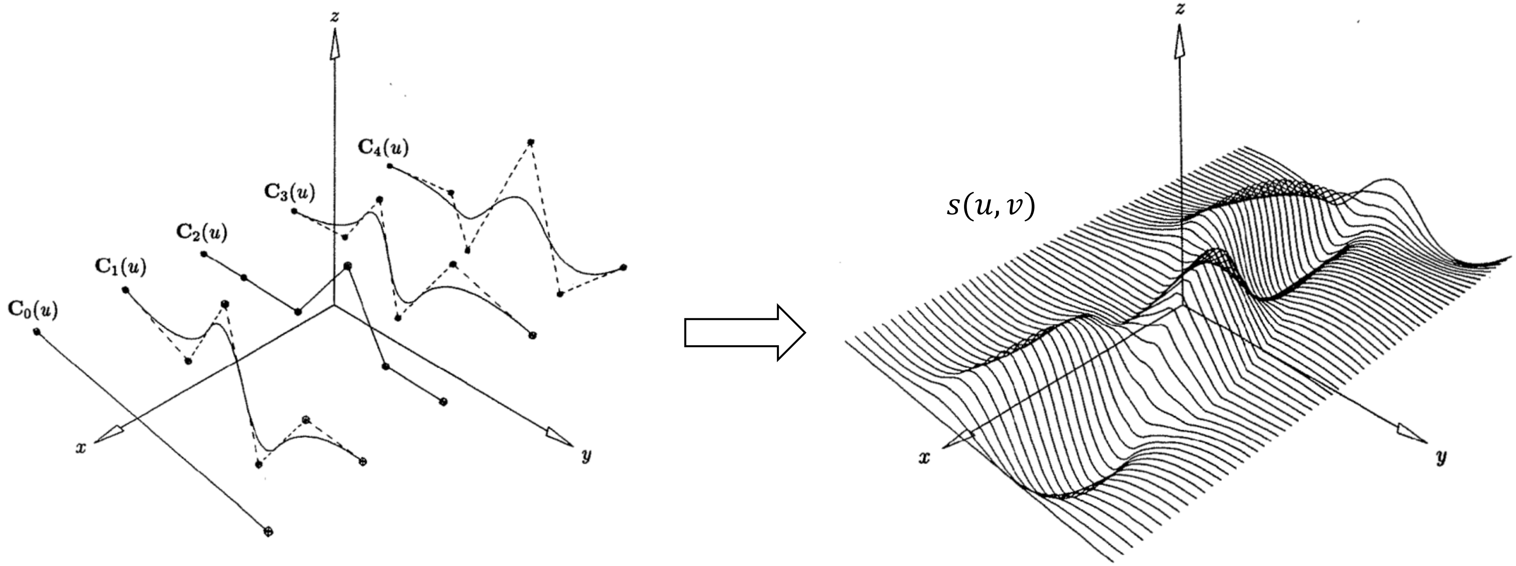
$$\Rightarrow \boldsymbol{N}\boldsymbol{C} \equiv \boldsymbol{P}$$

➢ Solve a linear equation



B-spline basics
Algorithm: Surface Skinning

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



- Similar to curve interpolation

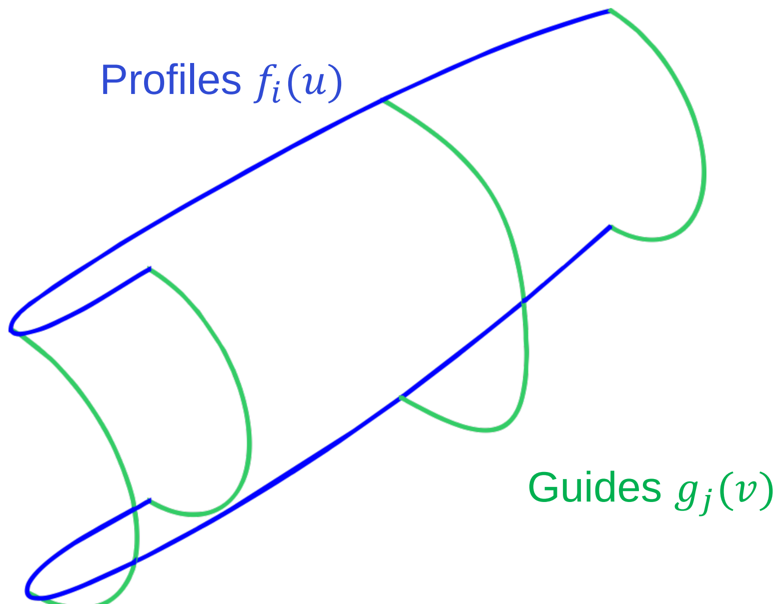
- Requires knot insertion to make input curves compatible

DLR

Curve Network Interpolation with Gordon Surfaces

Gordon Surfaces
The curve network interpolation problem

- Definition of the problem:
  - Given a network of profile and guide curves:
    Find surface that connects (interpolates) these curves



Curve network example

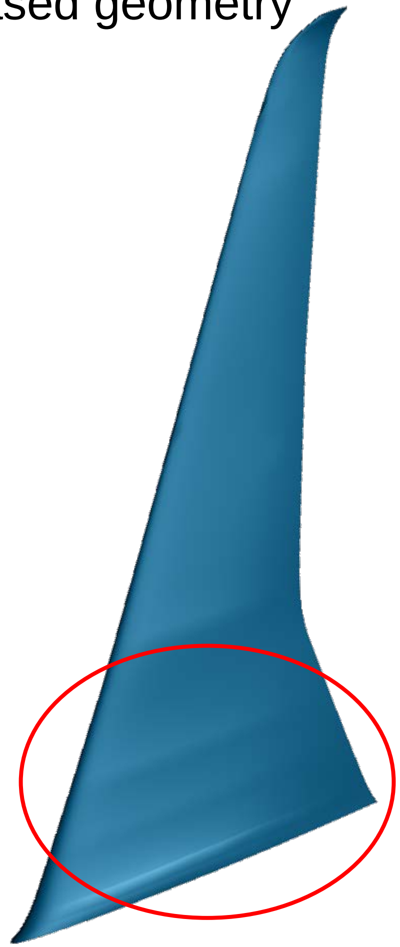
Gordon Surfaces
Motivation

- Before: Used OpenCASCADE implemention of Coons-patch based geometry generation

➢ Problems from aero simulations:
  - **Pressure oscillations** on the wing
  - Surface modelling probably the issue!

- Analysis:
  - Generated surfaces have small **bumps**, **waves** and **kinks**
  - The latter is caused by C1 or C2 **discontinuities**

- Conclusion:
  - Must implement algorithm by our own: **Gordon Surfaces**



Gordon Surfaces
Algorithm overview

Create **three different surfaces**:

1) Create skinning surface **interpolating the curves** $\boldsymbol{f_i(u), \forall i}$:

$$S_u(u, v)$$

2) Create skinning surface **interpolating the curves** $\boldsymbol{g_j(v), \forall j}$:

$$S_v(u, v)$$

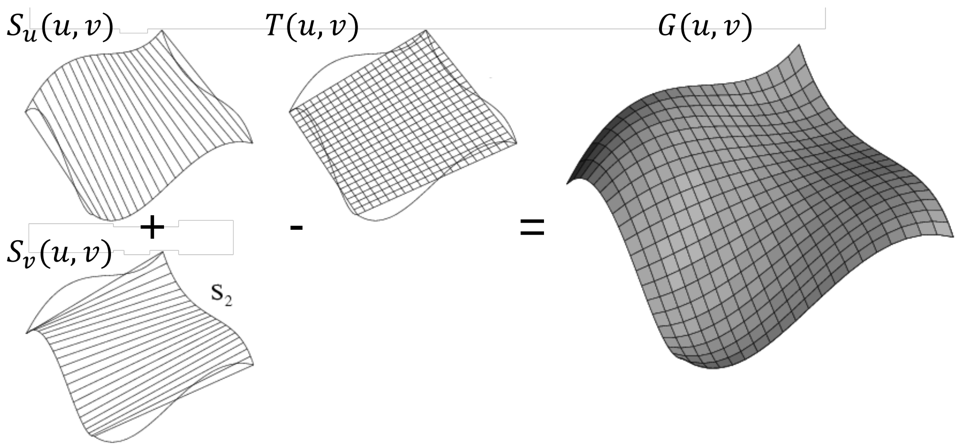
3) Create surface **interpolating the intersection points of the curve network:**

$$T(u, v)$$

Gordon Surfaces
Algorithm overview

→ Gordon Surface is superposition of these three surfaces:

$$G(u,v) = S_u(u,v) + S_v(u,v) - T(u,v)$$



→ Convert to Gordon surface B-Spline / NURBS for the use in TiGL

Gordon Surfaces
Compatibility Conditions

- Condition for the algorithm: Curves must be compatible

Compatibility condition

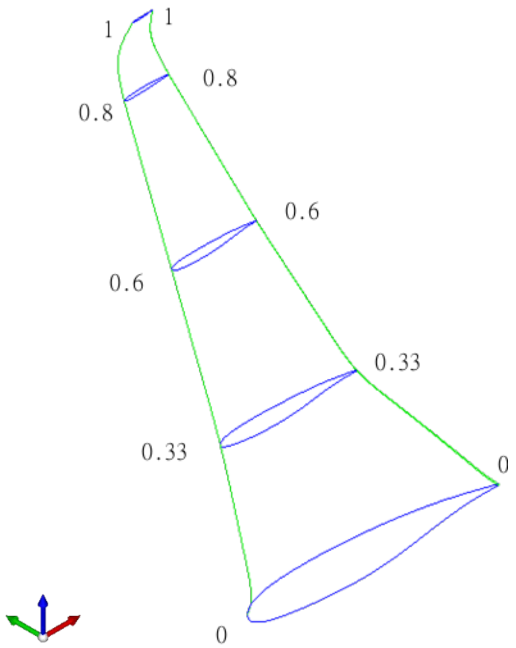
- All profile curves $f_i(u)$ must intersect with a guide curve $g_j(v)$at exactly the same parameter value u_j and vice versa:

$$f_i(u_j) = \mathrm{g_j}(\mathrm{v_i}), \forall i, j$$

- Interpretation: The input curves $f_i(u)$, $g_j(v)$ must be iso-parametric curves of the resulting surface

Gordon Surfaces
Compatibility Conditions, Example

Compatible intersections of the guides with the profiles:



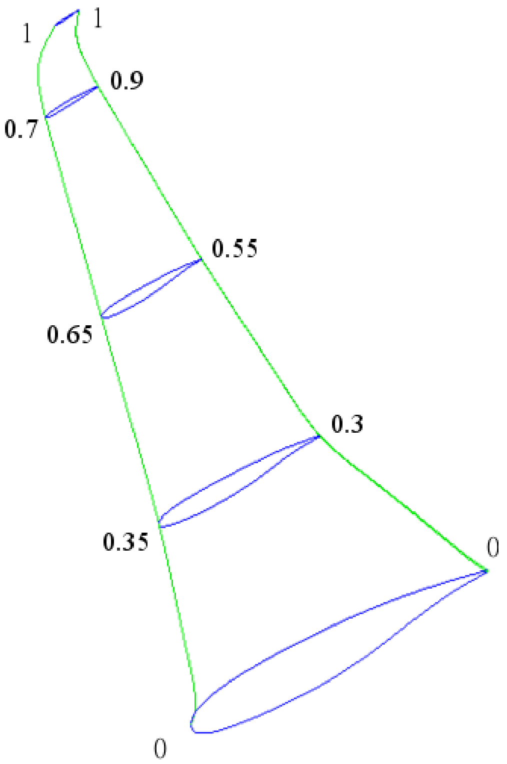
Gordon Surfaces
Reparameterization

- Compatibility condition very restrictive

- Compatibility practically never the case!

→ All curves have to be **reparametrized**

→ **Reparameterization is the main issue!**



Gordon Surfaces
Reparameterization

Rough Idea:

- Find function r_i such that

$$r_i(u_j^i) = \widehat{u_j^i}$$

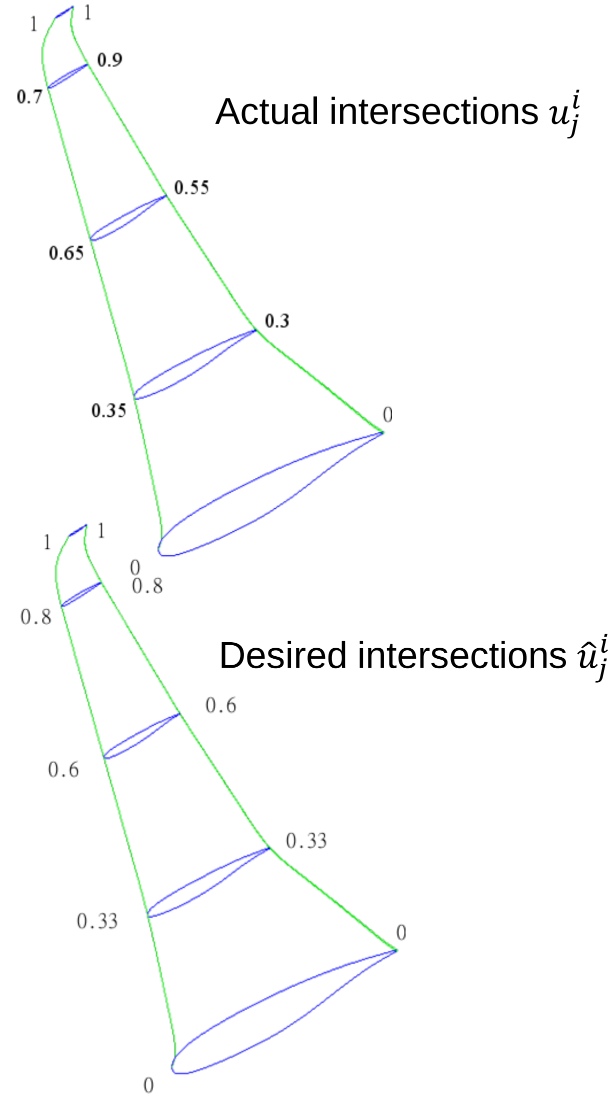
with:

  - u_j^i the parameter of the intersection of curve $f_i(u)$ with curve $g_j(v)$
  - $\widehat{u_j^i}$ the desired intersection parameter to achieve compatibility

- Reparametrized function is given by

$$\widehat{f_i}(u) \coloneqq f_i(r_i(u))$$

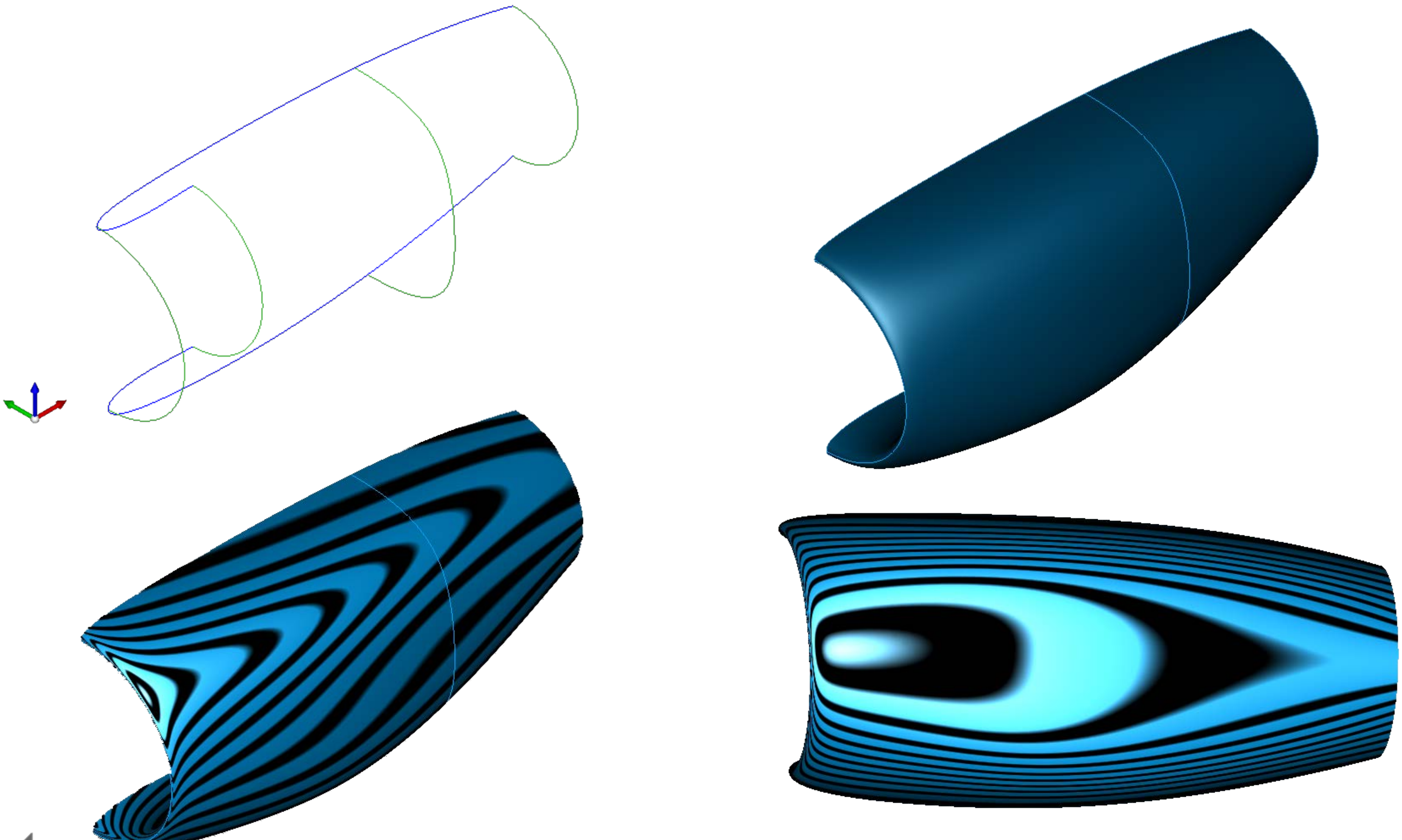
- Challenge: Find B-spline parameterization of $\widehat{f_i}(u)$



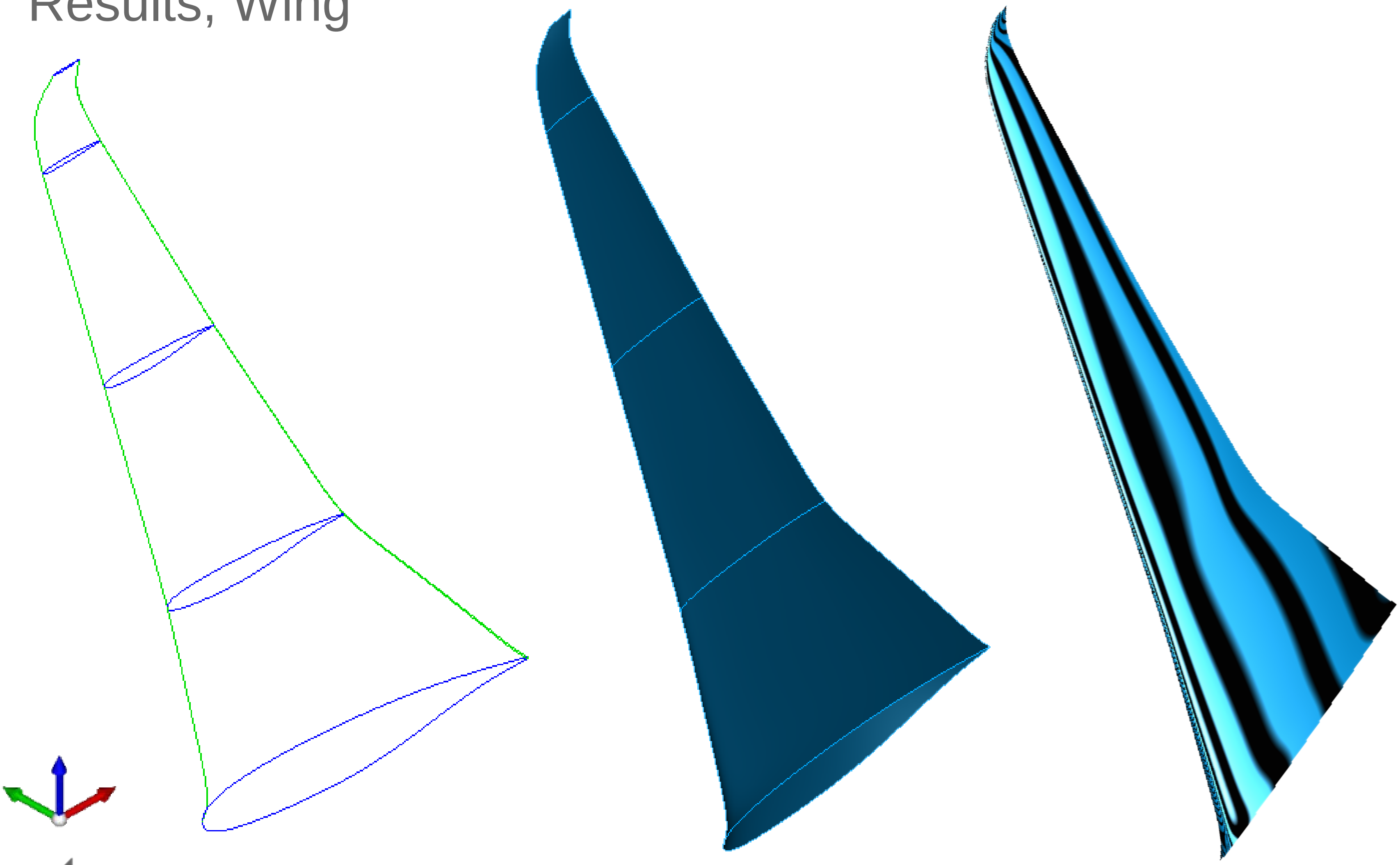
Gordon Surfaces
Quality Analysis

- Surface quality analysis with zebra stripe plot

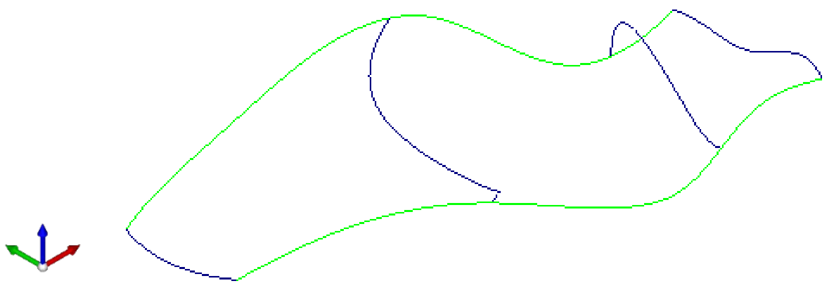
Gordon Surfaces
Results, Nacelle (engine cover)

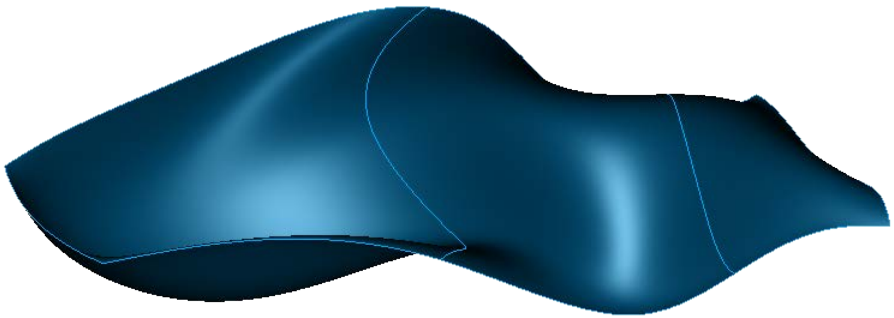


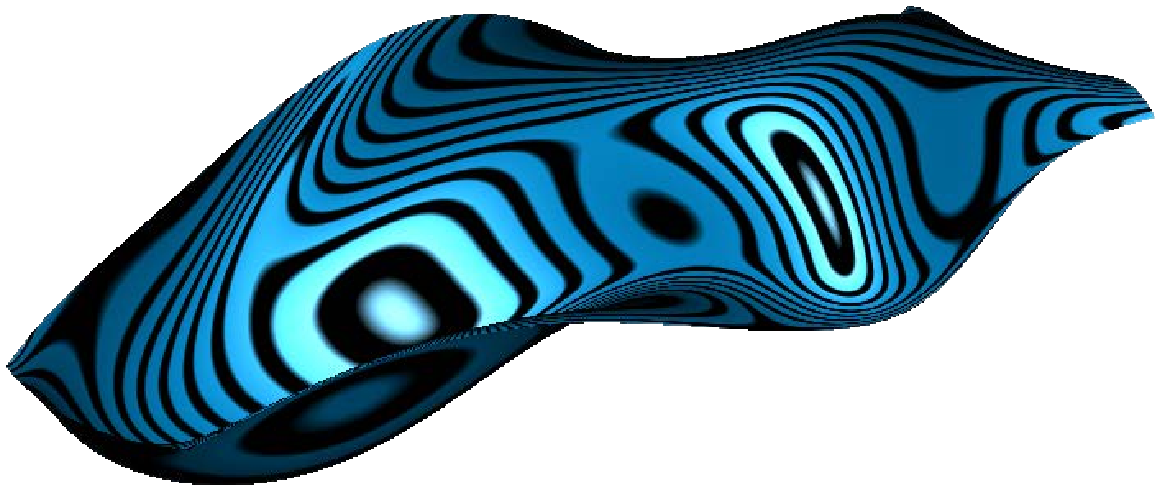
Gordon Surfaces
Results, Wing



Gordon Surfaces
Results, Arbitrary surface

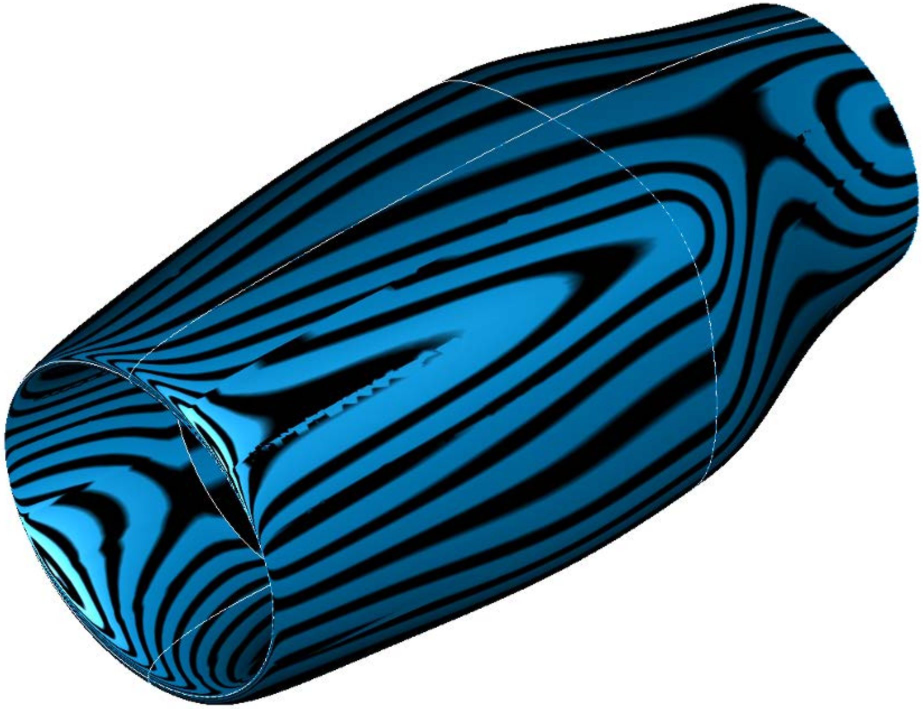






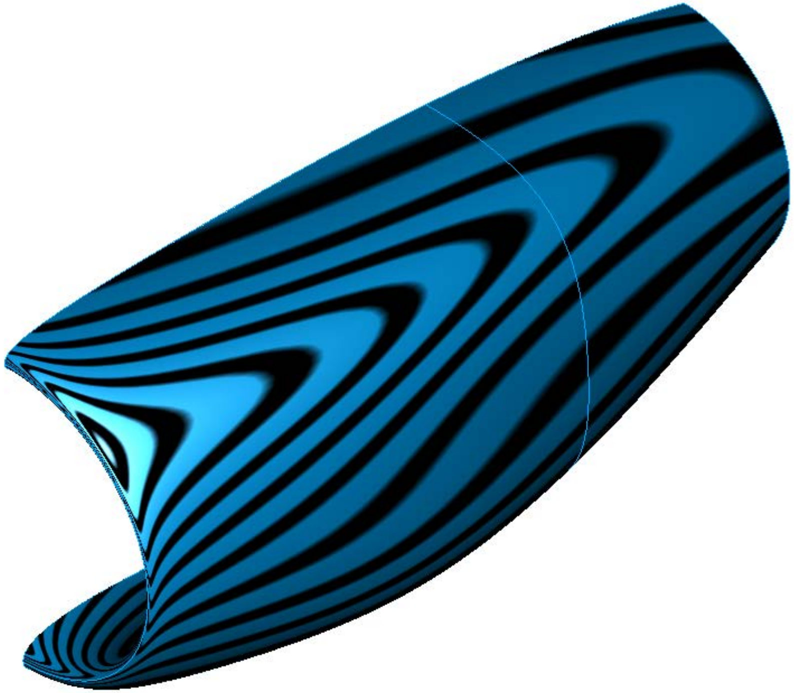
Gordon Surfaces
Results, Coons vs. Gordon, Nacelle

Coons Patches



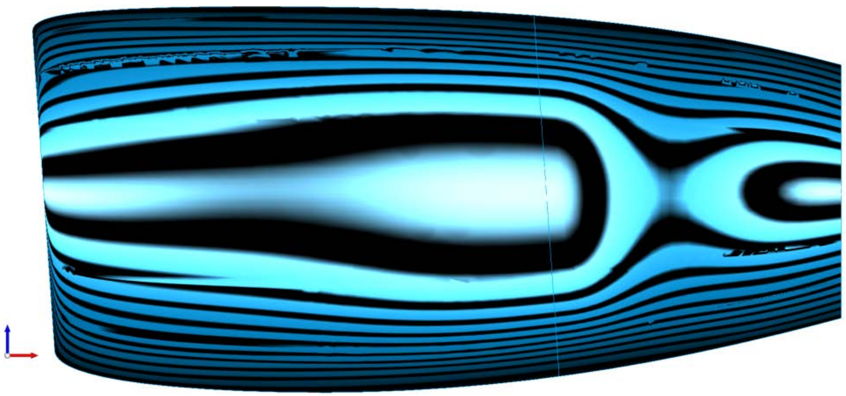
Gordon Surface

(one half of the same nacelle)

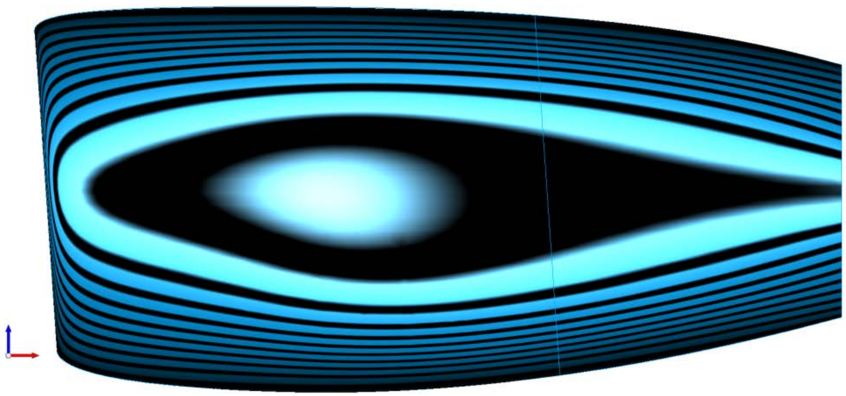


Gordon Surfaces
Results, Coons vs. Gordon, Nacelle

Coons Patches

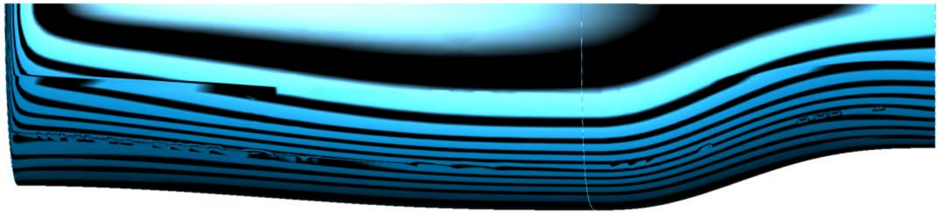


Gordon Surface

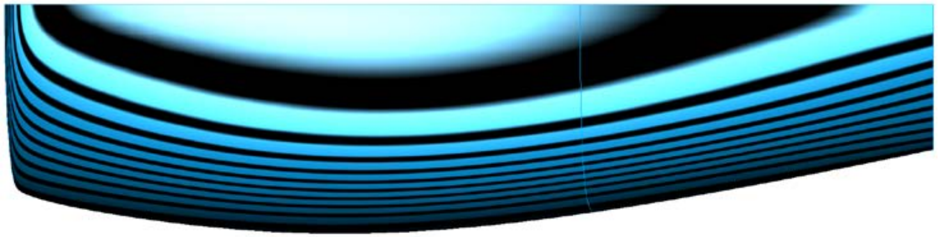


Gordon Surfaces
Results, Coons vs. Gordon, Nacelle

Coons Patches

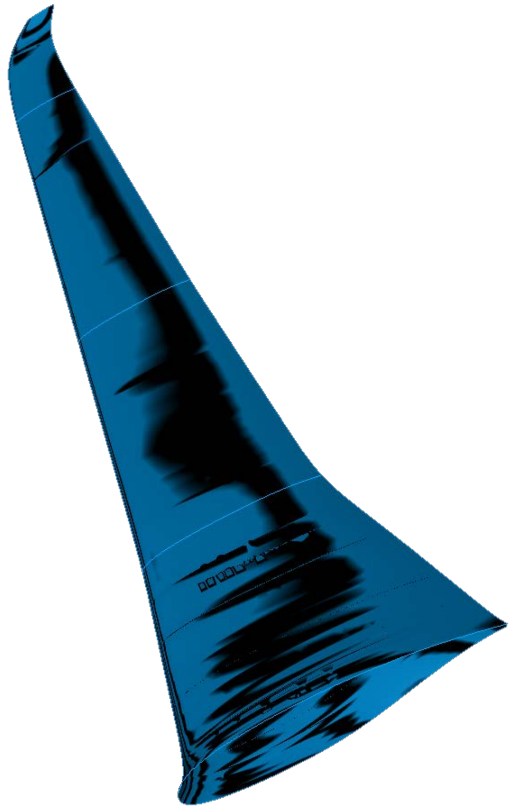


Gordon Surface

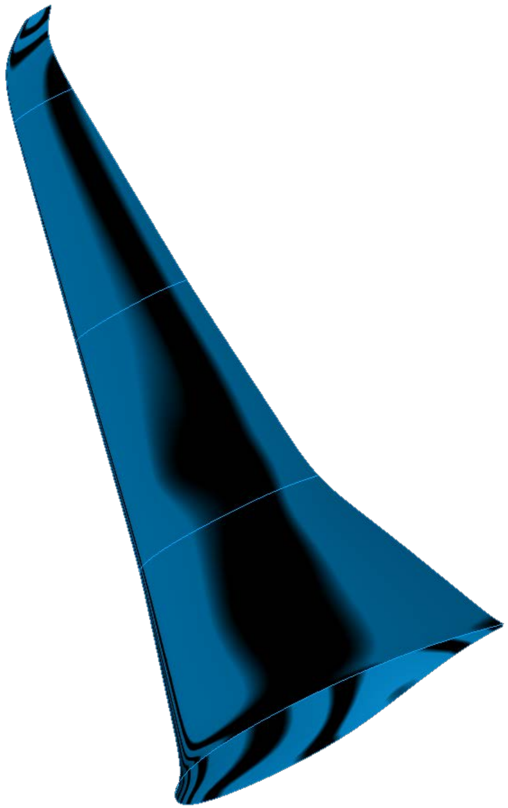


Gordon Surfaces
Results, Coons vs. Gordon, Wing

Coons Patches

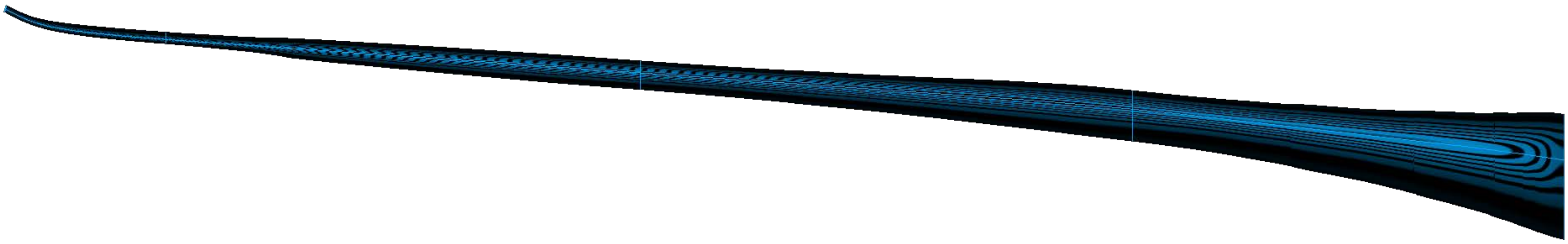


Gordon Surface



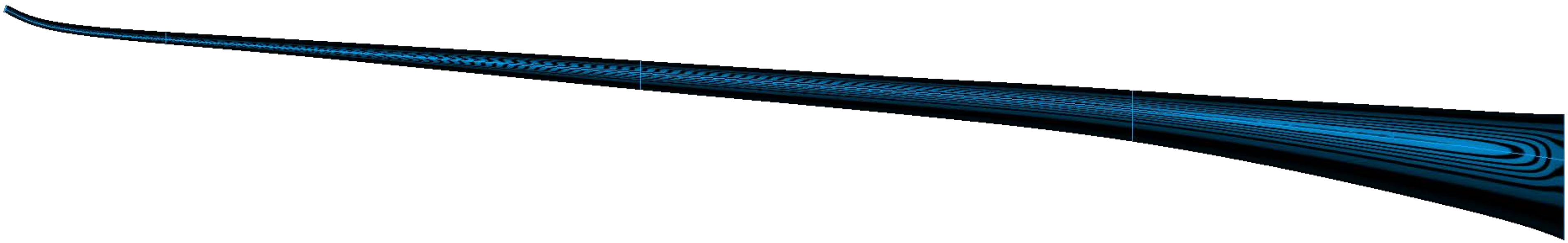
Gordon Surfaces
Results, Coons vs. Gordon, Wing

Coons Patches



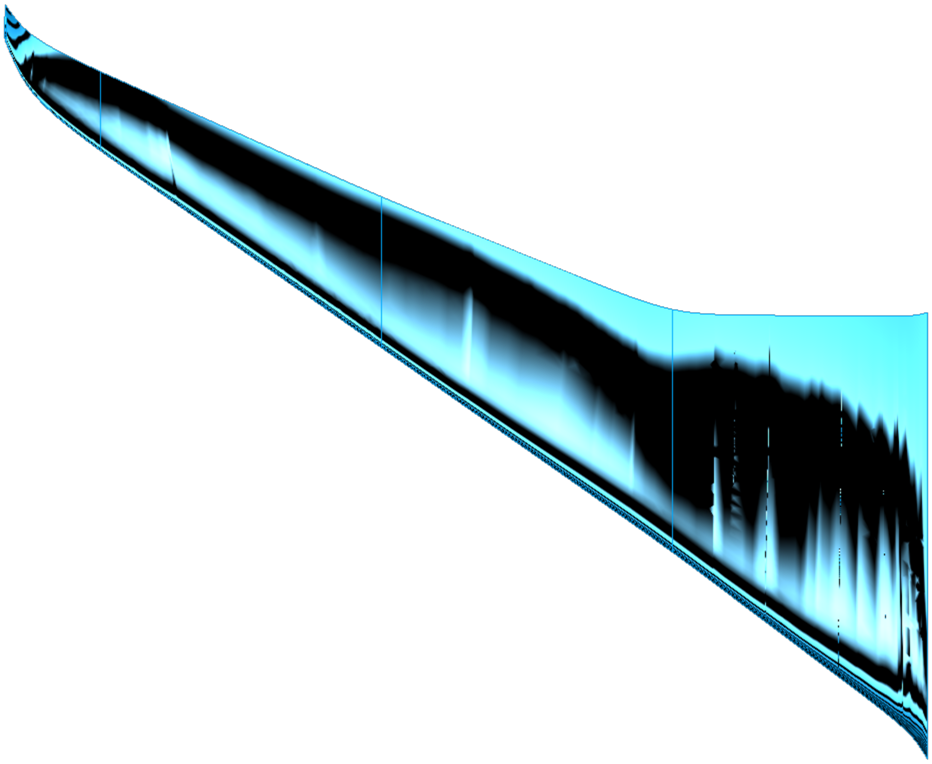
Gordon Surfaces
Results, Coons vs. Gordon, Wing

Gordon Surface

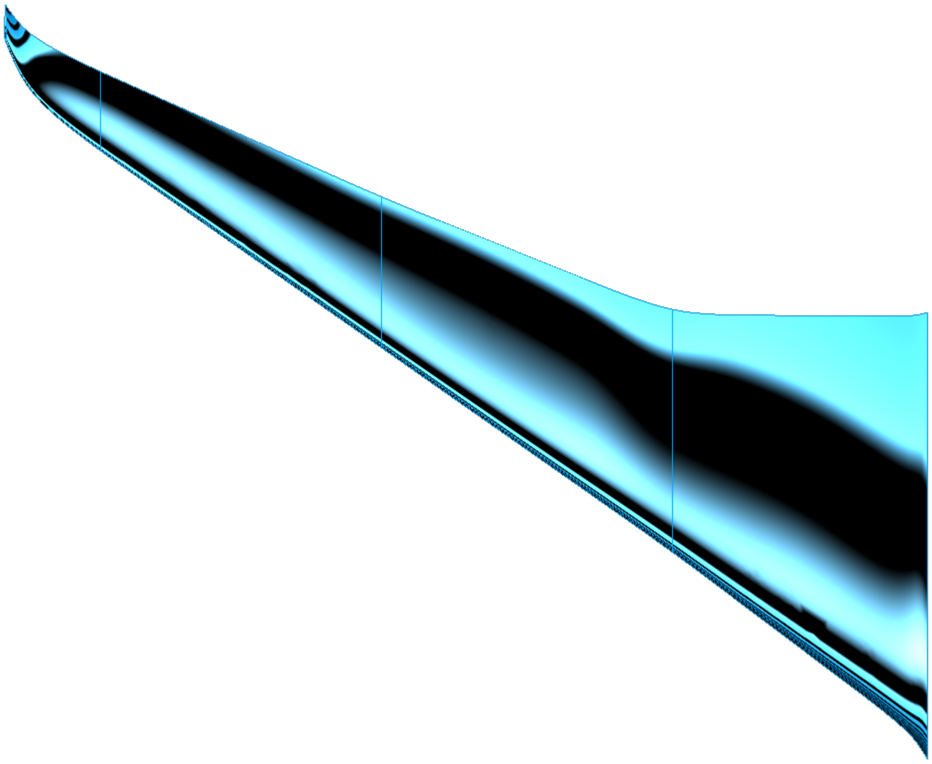


Gordon Surfaces
Results, Coons vs. Gordon, Wing

Coons Patches

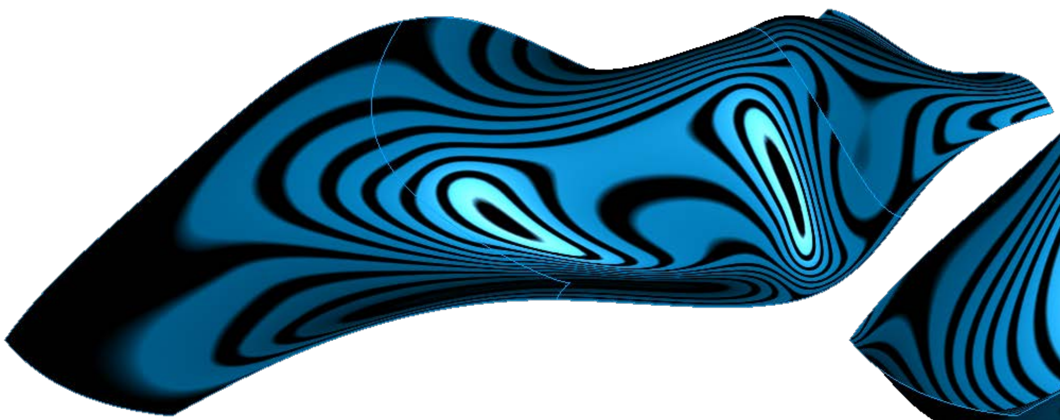


Gordon Surface

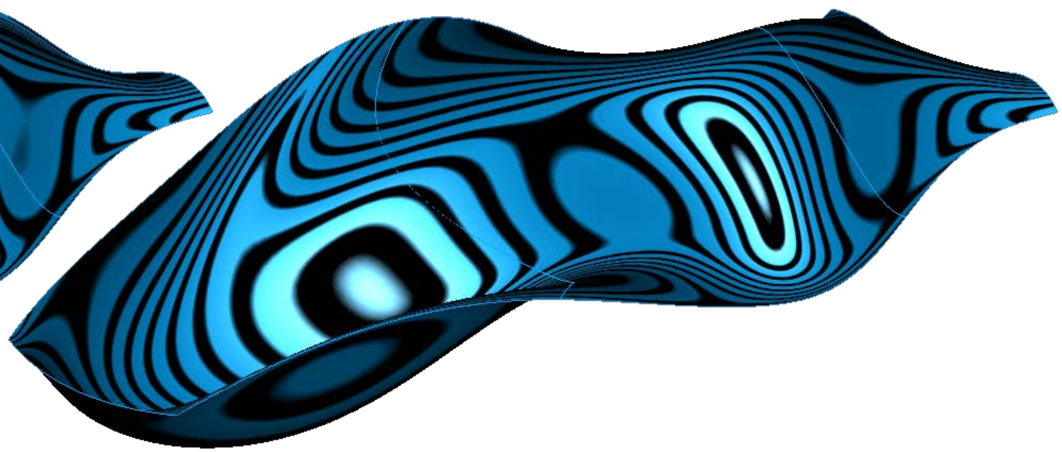


Gordon Surfaces
Results, Coons vs. Gordon, Arbitrary Surface

Coons Patches



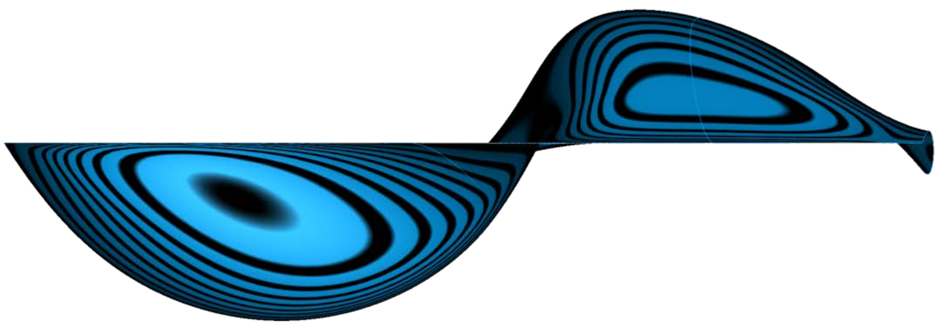
Gordon Surface



Gordon Surfaces
Results, Coons vs. Gordon, Arbitrary Surface

Coons Patches

Gordon Surface



Gordon Surfaces
Results, Summary

- Gordon Surface method works!
- Better quality than previous Coons patch implementation
- Global interpolation:
  - Smooth and natural interpolation of the curve network with a single surface
  - But, Oscillation might occur!

Free-Form Deformation of Aircraft Nacelles

Free Form Deformation of Nacelles
Motivation

- Different possibilities to parametrize engine nacelles, such as
  - Interpolation of curve network
  - Free-Form Deformation

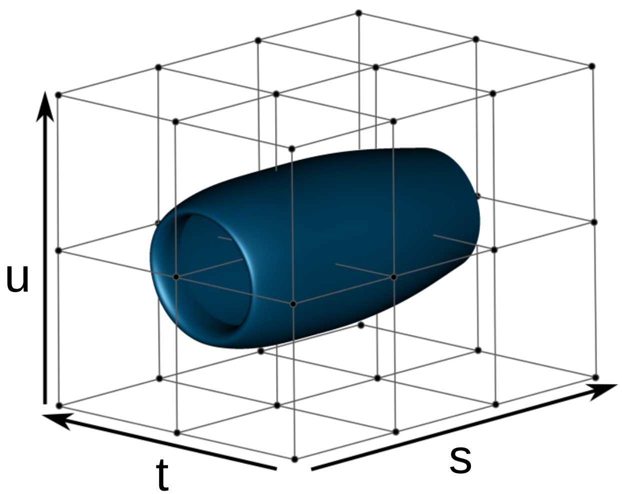
- Proof of concept:
  Show, that we can rebuild existing geometries from real aircraft with the FFD parameterization approach

Free Form Deformation of Nacelles

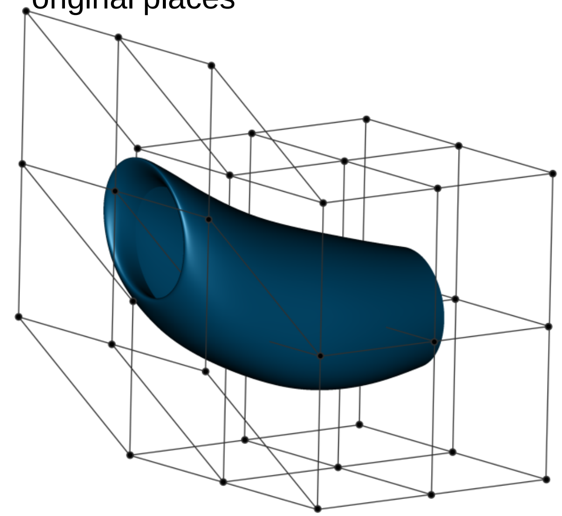
Concept

- **Free-Form-Deformation** (FFD)
  - start with simple, rotationally symmetric geometry
  - deform the geometry using FFD (parametrization = grid point displacements)

1. Define a so-called FFD-grid around the geometry:



2. Deform your geometry by moving the FFD points from their original places



Free Form Deformation of Nacelles
Definition

- Each point on the engine surface s has local grid coordinates (s,t,u)

- Deformation of the point (s,t,u) using FFD:

$$F_{ffd}(s,t,u) = \sum_{i}^{l} \sum_{j}^{m} \sum_{k}^{n} B_{i,l}(s) B_{j,m}(t) B_{k,n}(u) \cdot x_{ijk}$$

Linear in x!

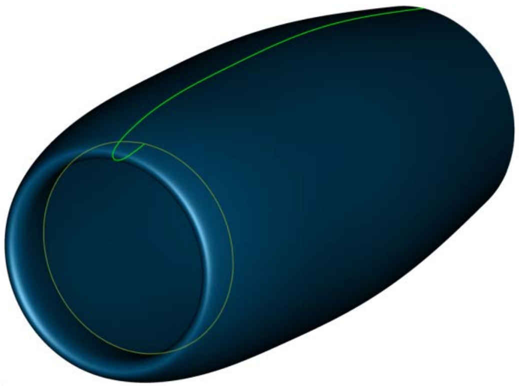
with:

- FFD grid points $\{x_{ijk}\}$
- Bernstein basis functions $B_{i,n}(u)$

Free Form Deformation of Nacelles
Initial Surface Generation

- Rotation of iso-parametric curve of reference surface around central axis

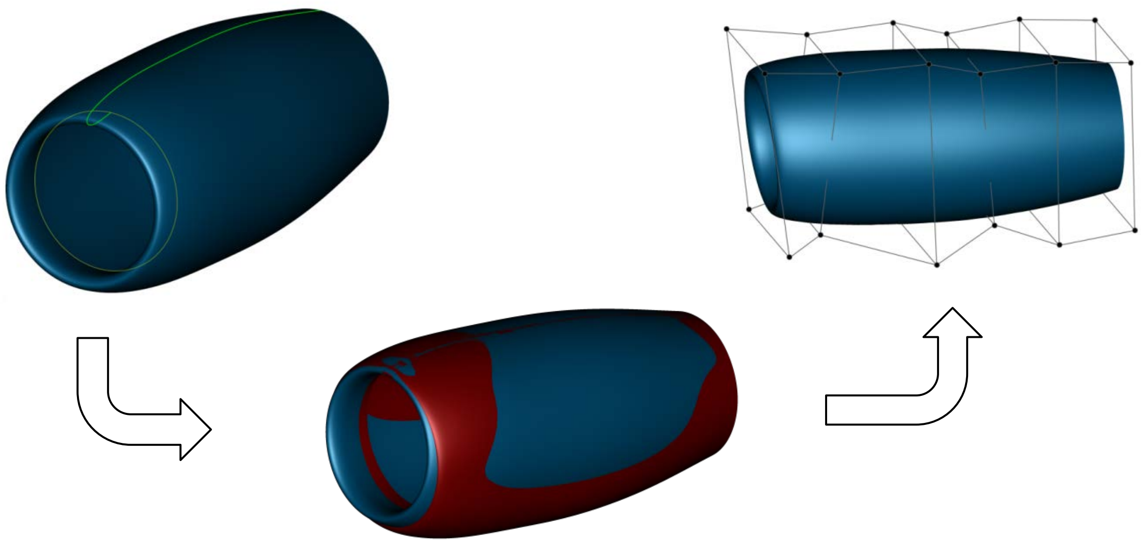
→ Identical parametrization of reference geometry and created geometry



Initial surface with iso-parametric curve in axial direction
and rotational curve

Free Form Deformation of Nacelles
Basic Principle

Modify FFD grid points $\{x_{ijk}\}$, such that initial surface fits reference surface :



Free Form Deformation of Nacelles
The Minimization Problem

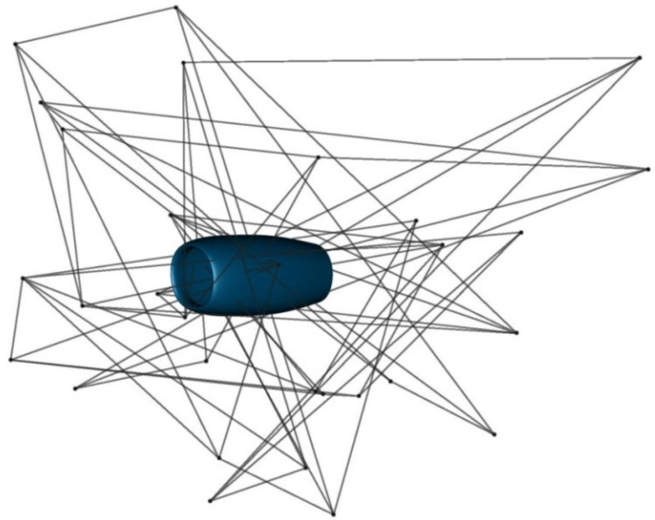
- The discretized surface z depends linearly on the control points x.

$$z = B \cdot x$$

- Given the discretized reference surface as r, the fit is given by the solution of

$$\|B \cdot x - r\|_2^2 \xrightarrow{x} min$$

Free Form Deformation of Nacelles
Regularization



- Problem: Matrix B has bad condition, resulting FFD grid x distorted
- Remedy: Use modified L2 regularization

$$\|B \cdot x - r\|_2^2 + \frac{\lambda}{n}\|x - x_0\|_2^2 \xrightarrow{x} min$$

with:

- Regularization parameter $\lambda \in R^+$
- Number of FFD grid points n

Free Form Deformation of Nacelles
Solving the Minimization Problem

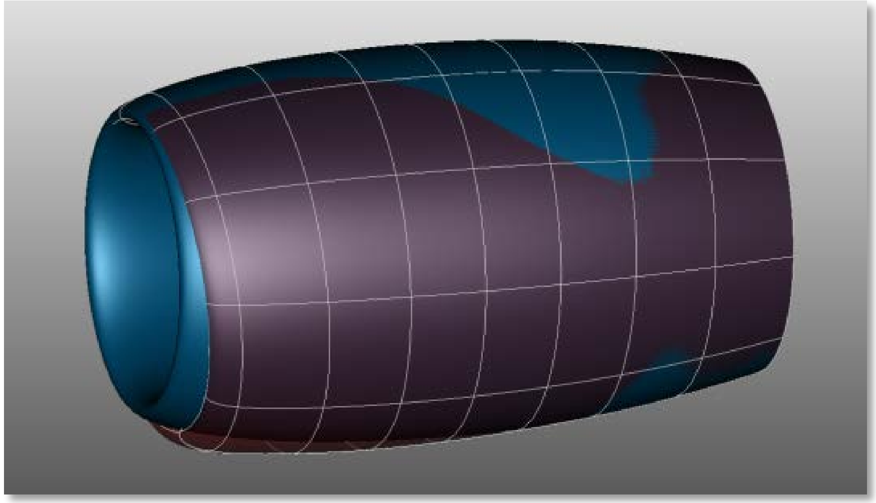
- Re-arrange minimization problem:

$$\|B \cdot x - r\|_2^2 + \frac{\lambda}{n} \Big\| \underbrace{x - x_0}_{\Delta x} \Big\|_2^2$$
$$= \|B \cdot \Delta x - (r - B \cdot x_0)\|_2^2 + \frac{\lambda}{n} \|\Delta x\|_2^2$$
$$= \|B \cdot \Delta x - \hat{r}\|_2^2 + \frac{\lambda}{n} \|\Delta x\|_2^2 \xrightarrow{\Delta x} min$$

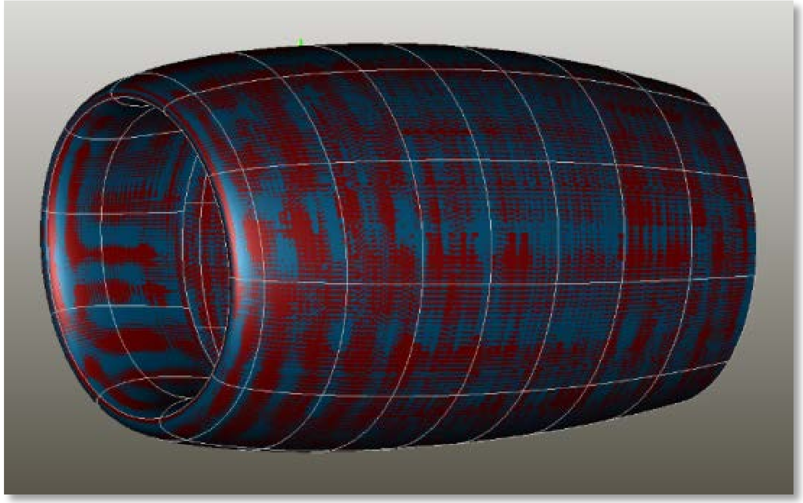
- Solve using Singular Value Decomposition (SVD):
  - Let $B = U\Sigma V^T$ its SVD, with diagonal Matrix Σ, containing singular values Σ_{ii}, then

$$\Delta x = V\Sigma^{+}U^T\hat{r} \text{ , with } \Sigma_{ii}^{+} = \frac{\Sigma_{ii}}{\Sigma_{ii}^2 + \lambda/n}$$

Free Form Deformation of Nacelles
Results, best Fit

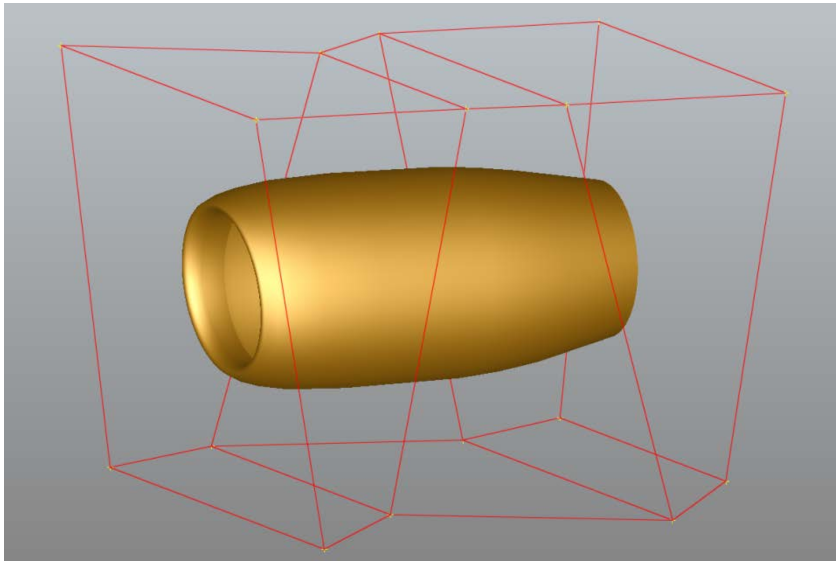


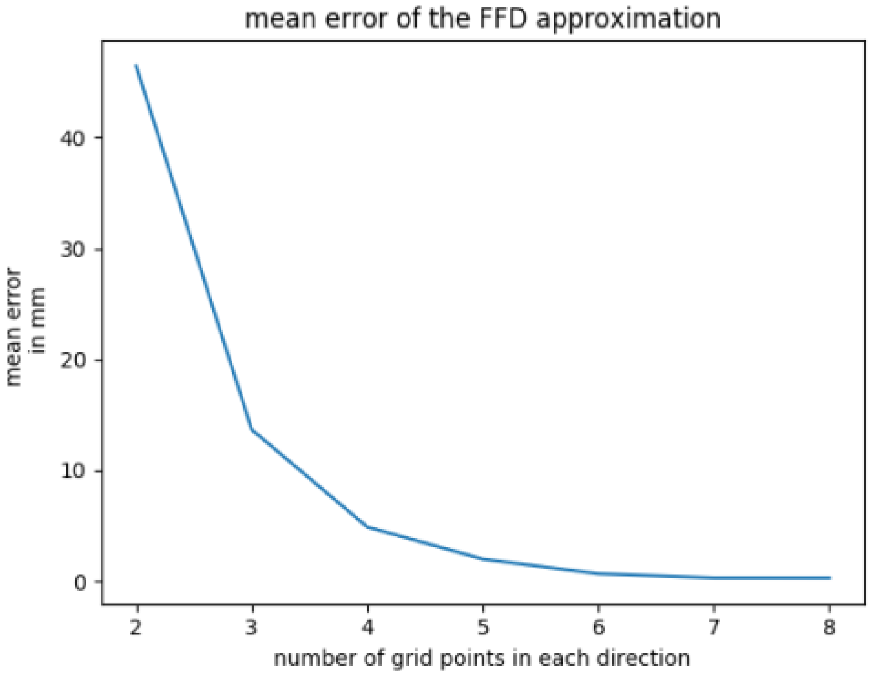
Reference surface and initial surface



Reference surface and deformed surface using 10x10x10 FFD grid points

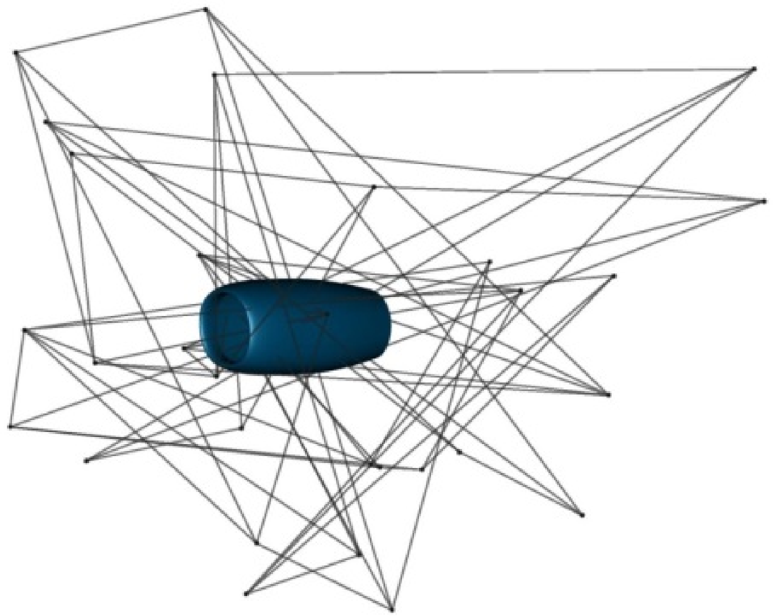
Free Form Deformation of Nacelles
Results, Error of the Fit



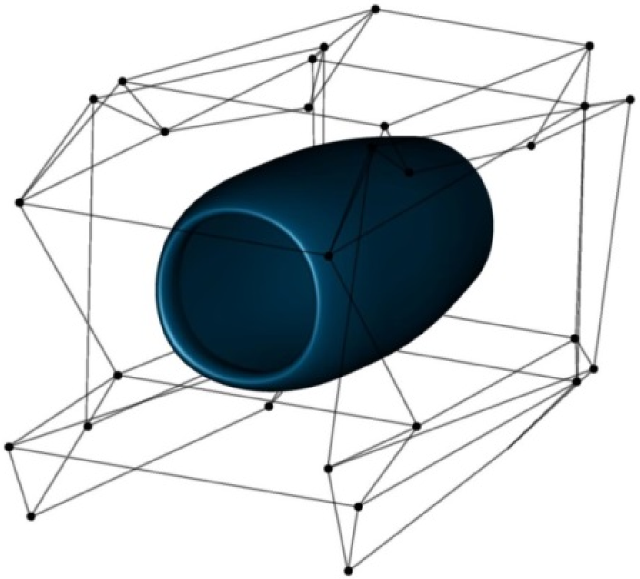


Optimiertes FFD Gitter mit 4 / 2 / 2
Punkten und deformierte Gondel

Free Form Deformation of Nacelles
Results, Regularization



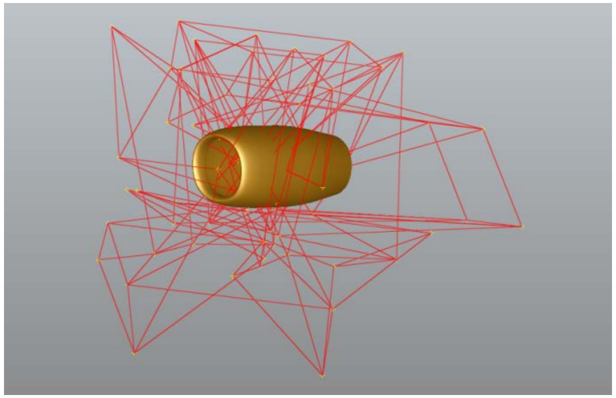
Distorted Grid (unregularized)



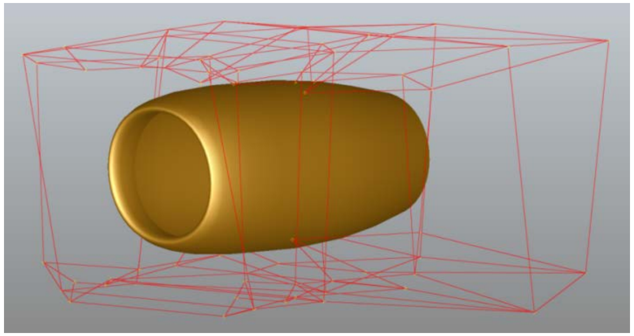
Same FFD grid, regularization, with $\lambda = 0.5$

Free Form Deformation of Nacelles

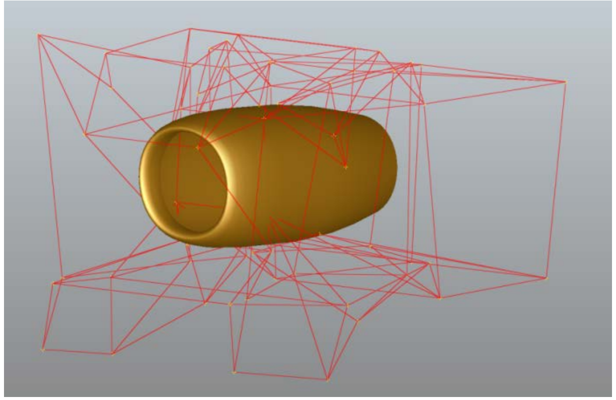
Results, Regularization



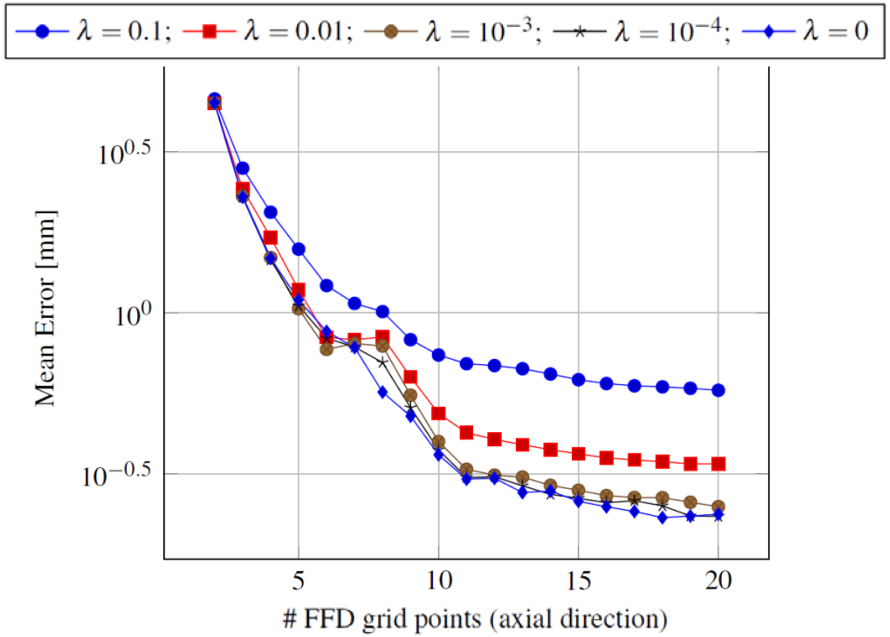
$\lambda = 0.01$



$\lambda = 1.0$



$\lambda = 0.1$



Free Form Deformation of Nacelles
Summary and Outlook

- Approximation engine surfaces with FFD method
- Method converges with rising number of FFD grid points
- Regularization is required due to bad conditioning
- Regularization has small influence on fit error, but large influence on the resulting grid quality

Outlook

- Use B-spline basis function in FFD function → also local deformation possible (already implemented but not presented here)
- Hierarchical optimization → could replace regularization

Questions?

martin.siggel@dlr.de
http://github.com/DLR-SC/tigl