# TiGL Workshop

## Curve Network Interpolation with Gordon Surfaces

12.09.2018

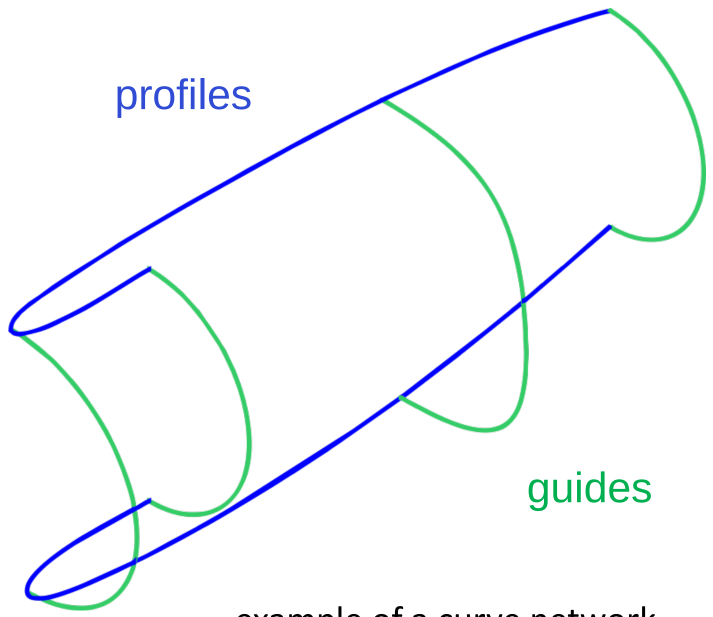
Merlin Pelz
System Dynamics and Control | SR-FLS
DLR Oberpfaffenhofen


DLR

# Curve Network Interpolation with Gordon Surfaces

# The Curve Network Interpolation Problem

- Definition of the problem:
  - Given a network of profile and guide curves, find surface(s) which interpolate(s) these curves




example of a curve network

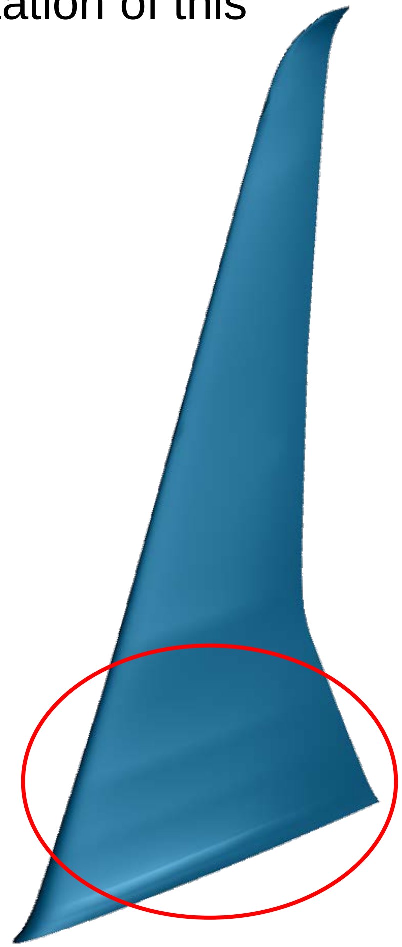
# Why Gordon Surfaces? Problems with Old Method

- In Digital-X, Open CASCADE was contracted for the implementation of this algorithm (based on Coons Patches)

- 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** on the surface

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



DLR-SC / tigl

Watch 16 | Star 46 | Fork 17

Code | Issues 35 | Pull requests 2 | Projects 2 | Wiki | Insights

Branch: cpacs_3 | tigl / src / geometry /

Create new file | Find file | History

rainman110 Allow to use geom_curves for surface creation ... Latest commit c61ed26 2 days ago

..

| | | |
|---|---|---|
| CCPACSACSystems.cpp | made CCPACSTransformation automatically update its internal matrix if... | 8 months ago |
| CCPACSACSystems.h | made CCPACSTransformation automatically update its internal matrix if... | 8 months ago |
| CCPACSFarField.cpp | Implemented new componentSegment coordsystem math | a year ago |
| CCPACSFarField.h | Added missing method overrides | 8 months ago |
| CCPACSGenericSystem.cpp | fixed reading transformation from wrong xpath | 4 months ago |
| CCPACSGenericSystem.h | Merge branch 'cpacs_3' into cpacs_3_parent_transformations | 8 months ago |
| CCPACSGenericSystems.cpp | made CCPACSTransformation automatically update its internal matrix if... | 8 months ago |
| CCPACSGenericSystems.h | made CCPACSTransformation automatically update its internal matrix if... | 8 months ago |
| CCPACSTransformation.cpp | various refactorings and minor changes | 4 months ago |
| CCPACSTransformation.h | made CCPACSTransformation automatically update its internal matrix if... | 8 months ago |
| CCSTCurveBuilder.cpp | Added missing method overrides | 8 months ago |
| CCSTCurveBuilder.h | Fixed compilation with occt 6 and 7 | 3 years ago |
| CFunctionToBspline.cpp | Added missing method overrides | 8 months ago |
| CFunctionToBspline.h | Fixed compilation with occt 6 and 7 | 3 years ago |
| CNamedShape.cpp | Minor refactoring | 6 days ago |
| CNamedShape.h | Minor refactoring | 6 days ago |
| CPointsToLinearBSpline.cpp | Applied upgrade tool to sources | 3 years ago |
| CPointsToLinearBSpline.h | Fixed compilation with occt 6 and 7 | 3 years ago |
| CTiglAbstractGeometricComponent.c... | removed svn $Id$ and $Revision$ comments | 5 months ago |
| CTiglAbstractGeometricComponent.h | removed svn $Id$ and $Revision$ comments | 5 months ago |
| CTiglAbstractSegment.h | removed svn $Id$ and $Revision$ comments | 5 months ago |
| CTiglApproximateBsplineWire.cpp | removed svn $Id$ and $Revision$ comments | 5 months ago |
| CTiglApproximateBsplineWire.h | removed svn $Id$ and $Revision$ comments | 5 months ago |
| CTiglArcLengthReparameterization.cpp | Added missing method overrides | 8 months ago |
| CTiglArcLengthReparameterization.h | Added more geometry algorithms to the python interface | 4 months ago |
| CTiglBSplineAlgorithms.cpp | Allow to use geom_curves for surface creation | 2 days ago |
| CTiglBSplineAlgorithms.h | Removed duplicate function GetCentripetalParameters | 3 days ago |
| CTiglBSplineApproxInterp.cpp | Fixed performance bottleneck in CTiglBSplineApproxInterp | 22 days ago |
| CTiglBSplineApproxInterp.h | Adaptions to compile with OCC 7.2.0 (#430) | 3 months ago |
| CTiglBSplineFit.cpp | Fixed issue #440 | 3 months ago |
| CTiglBSplineFit.h | Adaptions to compile with OCC 7.2.0 (#430) | 3 months ago |
| CTiglCurvesToSurface.cpp | Allow to use geom_curves for surface creation | 2 days ago |
| CTiglCurvesToSurface.h | Allow to use geom_curves for surface creation | 2 days ago |
| CTiglGordonSurfaceBuilder.cpp | Allow to use geom_curves for surface creation | 2 days ago |
| CTiglGordonSurfaceBuilder.h | Removed gordon surface from CTiglBSplineAlgorithms | 4 months ago |

DLR

# B-splines

# B-splines

**Definition:** A ***B-spline curve*** of order $k$ (and degree $k-1$) is defined as

$$C(t) = \sum_{i=0}^{n} B_i^k(t) P_i \quad , \ t_{min} < \mathrm{t} < \mathrm{t}_{\max}$$

with its ***control points*** $P_0, \dots, P_n$ and ***basis functions*** of order $k$ $B_i^k(t)$.

Specify a ***knot vector*** $T = (t_0, \dots, t_{n+k})$ where $t_i \le t_{i+1}$ for all $i$.

Then:

$$B_i^1(t) = \begin{cases} 1\,, & t_i < t < t_{i+1} \\ 0\,, & otherwise \end{cases}$$

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

# B-splines

**Definition:** A ***B-spline curve*** of order $k$ (and degree $k-1$) is defined as

$$C(t) = \sum_{i=0}^{n} B_i^k(t) P_i \quad , \; t_{min} < \mathrm{t} < \mathrm{t}_{\max}$$

with its ***control points*** $P_0, \dots, P_n$ and ***basis functions*** of order $k$ $B_i^k(t)$.

Specify a ***knot vector*** $T = (t_0, \dots, t_{n+k})$ where $t_i \leq t_{i+1}$ for all $i$.

Then:

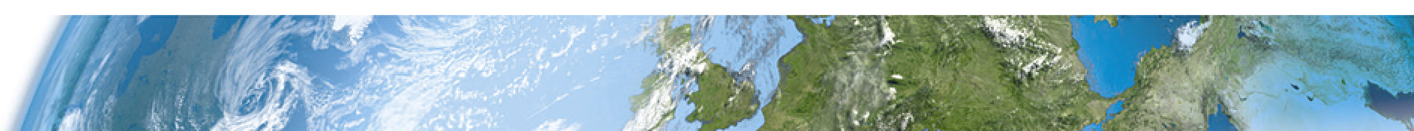
$$B_i^1(t) = \begin{cases} 1\,, & t_i < t < t_{i+1} \\ 0\,, & otherwise \end{cases}$$

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

**Definition:** - ***uniform*** knot vector: for all $i$: $t_{i+1} - t_i = const$, non-uniform otherwise

- ***clamped*** knot vector: for all $i \in \{0, \dots, k-1\}$: $t_i = t_0$

(endpoint interpolation) for all $i \in \{n+1 \dots, n+k\}$: $t_i = t_{n+k}$



# B-splines

**Some other properties:**

- $B_i^k(t) = 0$ for all $t \notin [t_i, t_{i+k})$ (locality)
- $C(t)$ is $C^{k-2}$-continuous
- $2 \leq k \leq n+1$
- convex hull property

# B-splines

**Some other properties:**

- $B_i^k(t) = 0$ for all $t \notin [t_i, t_{i+k})$ (locality)
- $C(t)$ is $C^{k-2}$-continuous
- $2 \leq k \leq n+1$
- convex hull property

**Changing the shape of a B-spline curve by:**

- moving the control points

# B-splines

**Some other properties:**

- $B_i^k(t) = 0$ for all $t \notin [t_i, t_{i+k})$ (locality)
- $C(t)$ is $C^{k-2}$-continuous
- $2 \leq k \leq n + 1$
- convex hull property

**Changing the shape of a B-spline curve by:**

- moving the control points
- adding/removing control points

# B-splines

**Some other properties:**

- $B_i^k(t) = 0$ for all $t \notin [t_i, t_{i+k})$ (locality)
- $C(t)$ is $C^{k-2}$-continuous
- $2 \leq k \leq n+1$
- convex hull property

**Changing the shape of a B-spline curve by:**

- moving the control points
- adding/removing control points

$\Rightarrow$ closer to/farther from the point

# B-splines

**Some other properties:**

- $B_i^k(t) = 0$ for all $t \notin [t_i, t_{i+k})$ (locality)
- $C(t)$ is $C^{k-2}$-continuous
- $2 \leq k \leq n+1$
- convex hull property

**Changing the shape of a B-spline curve by:**

- moving the control points
- adding/removing control points
  - $\Rightarrow$ closer to/farther from the point
- using multiple control points

# B-splines

**Some other properties:**

- $B_i^k(t) = 0$ for all $t \notin [t_i, t_{i+k})$ (locality)
- $C(t)$ is $C^{k-2}$-continuous
- $2 \leq k \leq n+1$
- convex hull property

**Changing the shape of a B-spline curve by:**

- moving the control points
- adding/removing control points
  - $\Rightarrow$ closer to/farther from the point
- using multiple control points
- changing the order of $k$

# B-splines

**Some other properties:**

- $B_i^k(t) = 0$ for all $t \notin [t_i, t_{i+k})$ (locality)
- $C(t)$ is $C^{k-2}$-continuous
- $2 \leq k \leq n+1$
- convex hull property

**Changing the shape of a B-spline curve by:**

- moving the control points
- adding/removing control points
  - ⇒ closer to/farther from the point
- using multiple control points
- changing the order of $k$
  - ⇒ smoother and less close to the ***control polygon*** for bigger $k$

# B-splines

**Some other properties:**

- $B_i^k(t) = 0$ for all $t \notin [t_i, t_{i+k})$ (locality)
- $C(t)$ is $C^{k-2}$-continuous
- $2 \leq k \leq n + 1$
- convex hull property

**Changing the shape of a B-spline curve by:**

- moving the control points
- adding/removing control points

  $\Rightarrow$ closer to/farther from the point
- using multiple control points
- changing the order of $k$

  $\Rightarrow$ smoother and less close to the ***control polygon*** for bigger $k$
- using multiple knots

# B-splines

**Some other properties:**

- $B_i^k(t) = 0$ for all $t \notin [t_i, t_{i+k})$ (locality)
- $C(t)$ is $C^{k-2}$-continuous
- $2 \leq k \leq n+1$
- convex hull property

**Changing the shape of a B-spline curve by:**

- moving the control points
- adding/removing control points
  - $\Rightarrow$ closer to/farther from the point
- using multiple control points
- changing the order of $k$
  - $\Rightarrow$ smoother and less close to the ***control polygon*** for bigger $k$
- using multiple knots
  - $\Rightarrow$ clamped curve
  - $\Rightarrow$ for knot multiplicity $p$: curve is $C^{k-p-1}$-continuous at the corresp. point

# B-splines

**Some other properties:**
- $B_i^k(t) = 0$ for all $t \notin [t_i, t_{i+k})$ (locality)
- $C(t)$ is $C^{k-2}$-continuous
- $2 \leq k \leq n+1$
- convex hull property

**Changing the shape of a B-spline curve by:**
- moving the control points
- adding/removing control points
  - ⇒ closer to/farther from the point
- using multiple control points
- changing the order of $k$
  - ⇒ smoother and less close to the ***control polygon*** for bigger $k$
- using multiple knots
  - ⇒ clamped curve
  - ⇒ for knot multiplicity $p$: curve is $C^{k-p-1}$-continuous at the corresp. point
- changing the relative spacing of the knots

# B-splines

**Some other properties:**

- $B_i^k(t) = 0$ for all $t \notin [t_i, t_{i+k})$ (locality)
- $C(t)$ is $C^{k-2}$-continuous
- $2 \leq k \leq n+1$
- convex hull property

**Changing the shape of a B-spline curve by:**

- moving the control points
- adding/removing control points
  - $\Rightarrow$ closer to/farther from the point
- using multiple control points
- changing the order of $k$
  - $\Rightarrow$ smoother and less close to the ***control polygon*** for bigger $k$
- using multiple knots
  - $\Rightarrow$ clamped curve
  - $\Rightarrow$ for knot multiplicity $p$: curve is $C^{k-p-1}$-continuous at the corresp. point
- changing the relative spacing of the knots
  - $\Rightarrow$ closer knots: curve moves closer to corresponding control point

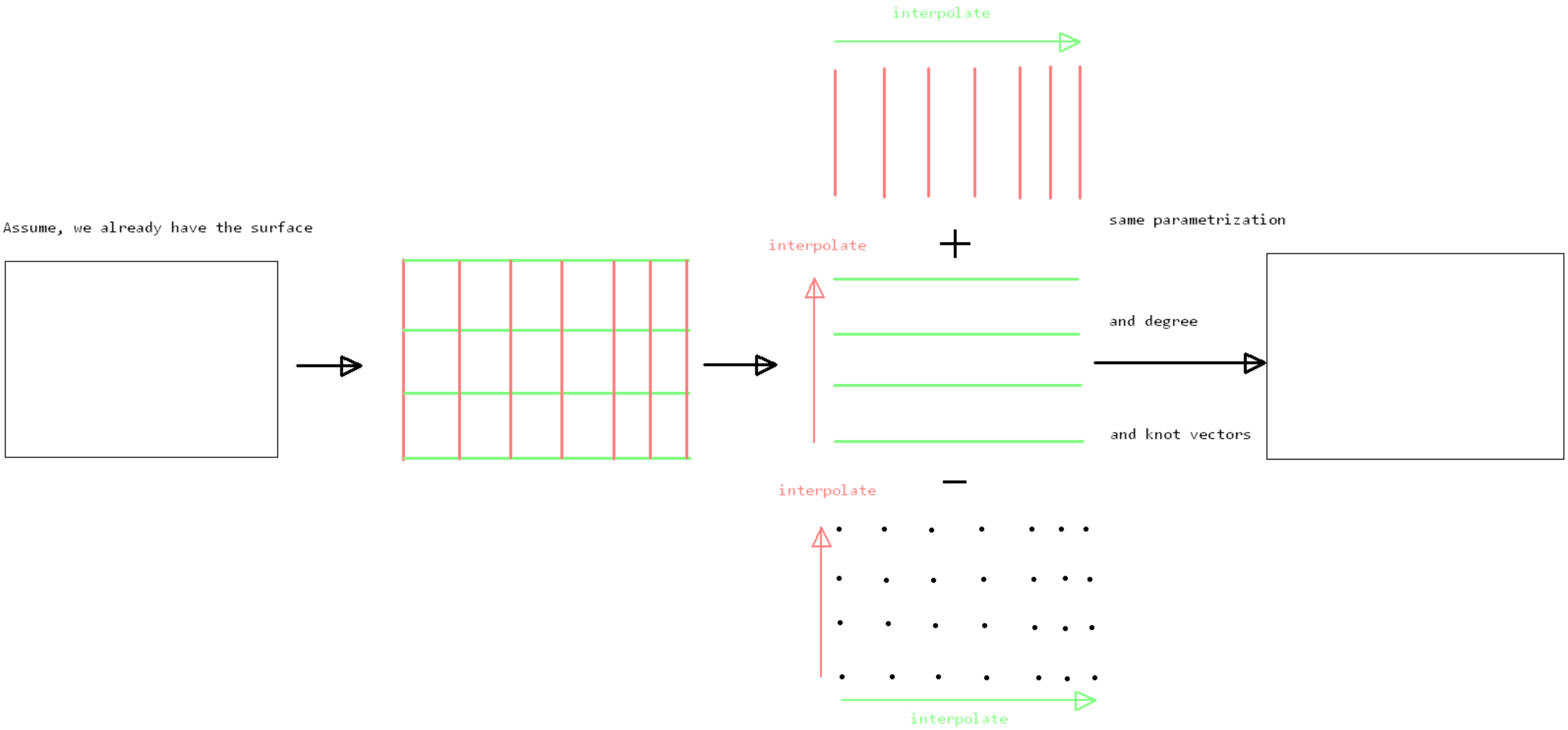
# B-splines

**Default setting:**
- cubic: $k = 4$
- no multiple control points

**Remarks:**
- Beziér curve for $k = n + 1$ and no internal knots
- closed curve defined by the points $P_1, \dots, P_M$:
  control points: $P_1, \dots, P_M, P_1, \dots, P_{k-1}$
- B-spline surface, also called ***patch***:
  $S(s,t) = \sum_{i=0}^{m} \sum_{j=0}^{n} B_i^k(s) B_j^l(t) P_{i,j}$ , $s_{min} \leq s < s_{max}$ , $t_{min} \leq t < t_{max}$

# Mathematical Theory of Gordon Surfaces

# Gordon Surfaces



# Steps for Creating the Gordon Surface in Detail

# Main Steps

- ***Reparametrization*** of B-spline curves
- Creating a ***common knot vector*** for B-spline curves and surfaces
  - ***Degree elevation*** for B-spline curves and surfaces
  - ***Knot insertion*** for B-spline curves and surfaces
- Finding the ***intersection points*** of the curve network
- Creating the two ***skinned surfaces***
- Creating ***tensor product surface***
- Creating the ***Gordon Surface***

# $C^1$-Continuous B-spline Curve Reparametrization

- Problem:
  - Make curve network compatible for the Gordon Surface theory
  - Therefore: reparametrize every curve $C$ by giving certain new parameter values for all $u$-directional and all $v$-directional curves at their intersection parameter values
- So, find $f(s)$ such that

$$C(u_i) = C(f(s_i))$$

# $C^1$-Continuous B-spline Curve Reparametrization

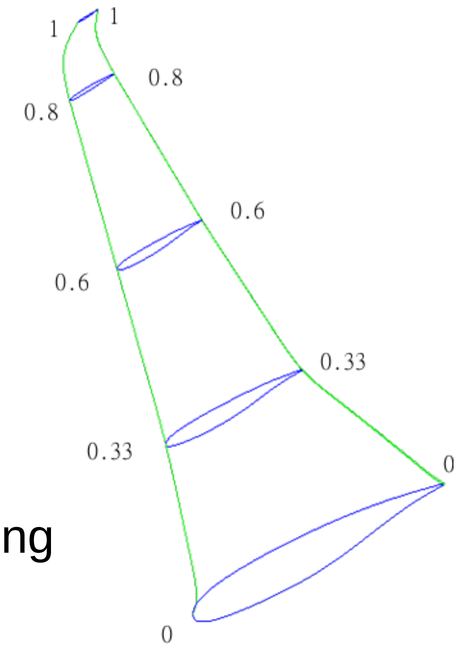
- Compute sample points on the curve $C$
- Interpolate these points at new parameter values by a B-spline curve $D$ such that:

$$D(s_i) = C(f(s_i))$$

# $C^1$-Continuous B-spline Curve Reparametrization

- Compute sample points on the curve $C$
- Interpolate these points at new parameter values by a B-spline curve $D$ such that:

$$D(s_i) = C(f(s_i))$$




- The approximation by $D(s)$ is done for not having to elevate the degree of the input curve
- Higher degree of B-spline curve ~ lower efficiency for computations
- Picture: example of a compatible curve network for a wing of an airplane after reparametrization

# Creating a Common Knot Vector

- Problem: all B-spline curves and surfaces shall have the same knot vector
- In case of curves:
    - Create a vector of all unique knots of all curves $W$
    - Create a vector of the multiplicities of each knot in $W$ $M$
    - Insert as many knots as needed in all knot vectors to get the vector $W$ with multiplicities $M$
- For surfaces, do this with both knot vectors $U$ and $V$

# Creating the Skinned Surfaces

- Problem: *skin* all $u$-directional curves of the curve network
- Write all control points of all B-spline curves $\{P_{ij}\}$ in a matrix
- Interpolate control points by $v$-directional B-spline curves $C_i$ at certain parameter values $\{v_k\}$ to get $\{Q_{ij}\}$:

$$P_{ik} = C_i(v_k) = \sum_{j=0}^{m} B_j^l(v_k) Q_{ij}, \qquad i \in \{1, \dots, n\}, j \in \{1, \dots, m\}$$

- Create skinned surface with control points $\{Q_{ij}\}$, knot vectors and degrees of $u$-directional and $v$-directional B-spline curves

# Creating the Skinned Surfaces

- Problem: *skin* all the $v$-directional B-spline curves

- Use the same method as before and flip the parameters $u$ and $v$ of the surface $S(u, v)$ to get $S(v, u)$

- Create the skinned surface by transposing the control point matrix, using the knot vector $V$ for $U$ and vice versa, and the $v$-directional degree for the $u$-directional degree and vice versa

# Creating the Tensor Product Surface

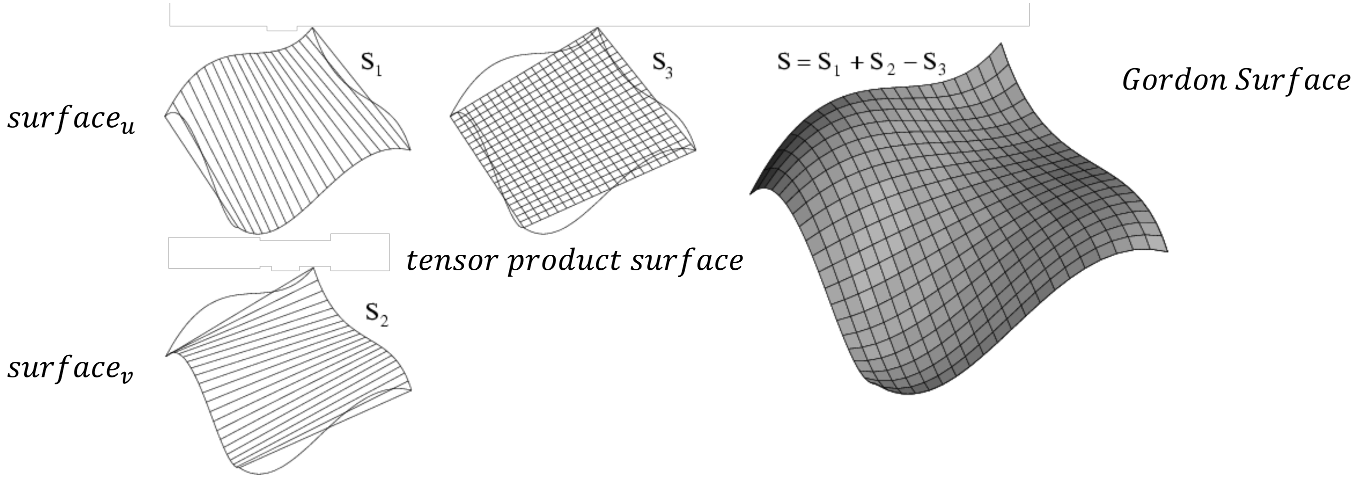
- Problem: create intersection points interpolating tensor product surface
- Find the intersection points $\{X_{ij}\}$ of the curve network
- Interpolate all the points by $u$-directional B-spline curves at certain parameter values $\{u_l\}$ :

$$X_{jl} = \sum_{i=0}^{n} B_i^p(u_l) P_{ij}\,, \qquad i, l \in \{1, \dots, n\}, j \in \{1, \dots, m\}$$

- Create a skinned surface with these curves

# Finally: Creating the Gordon Surface

1. Making the curve network **compatible by reparametrization**
2. Get **common degree** and **knot vector** of the curves in the two directions
3. Find the **intersection points** and their **parameters**
4. Create the **two skinned surfaces** with these parameters
5. Create the **tensor product surface** with these parameters
6. Create **common knot vector** for these three surfaces
7. Create B-spline **Gordon Surface** by *superposing the corresponding control points* of the three surfaces (this way B-spline surfaces are superposed)

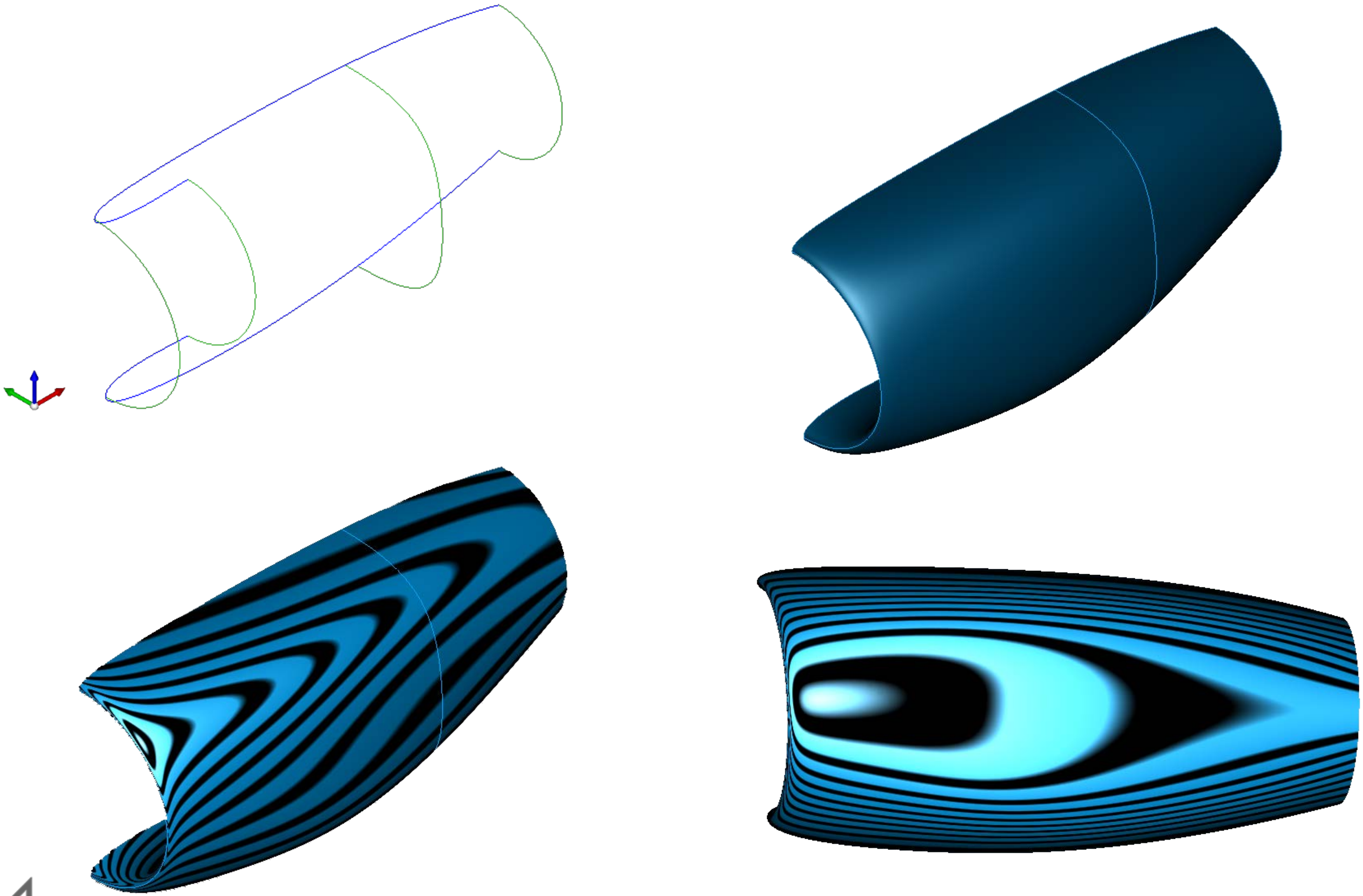


# Results

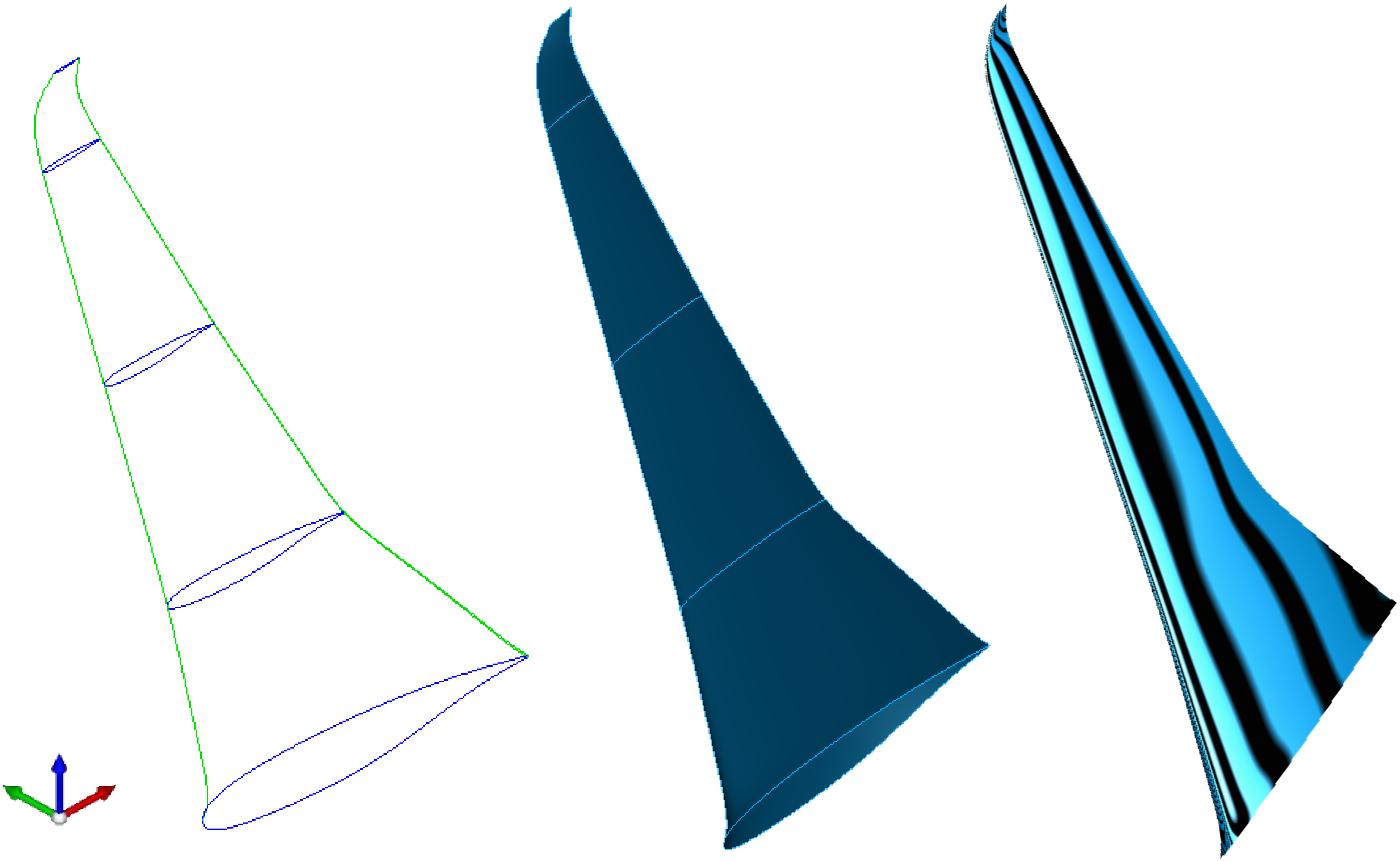
# Zebra Stripe Plot

- Surface quality analysis with zebra stripe plot (in TiGL Viewer 3)

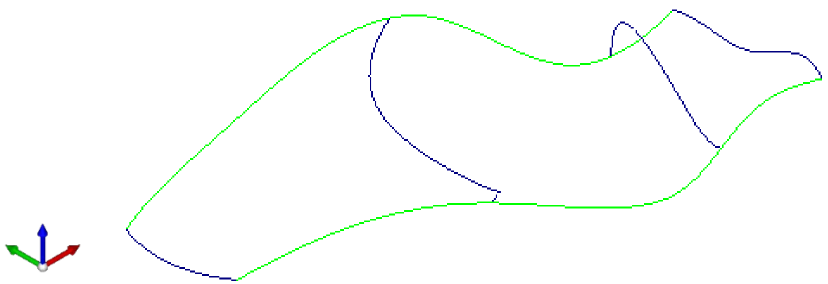
# Gordon Surfaces: Results for a Nacelle

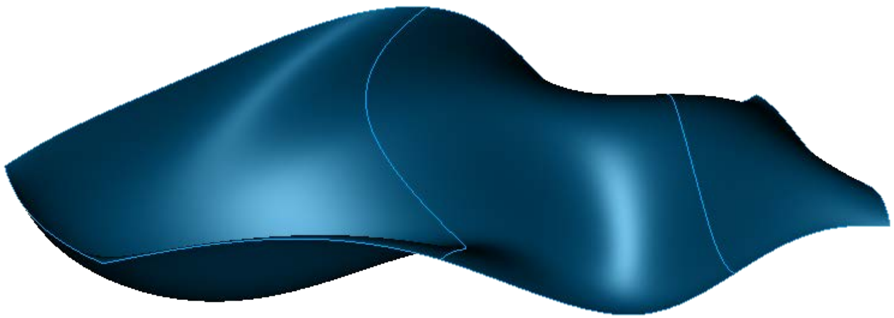


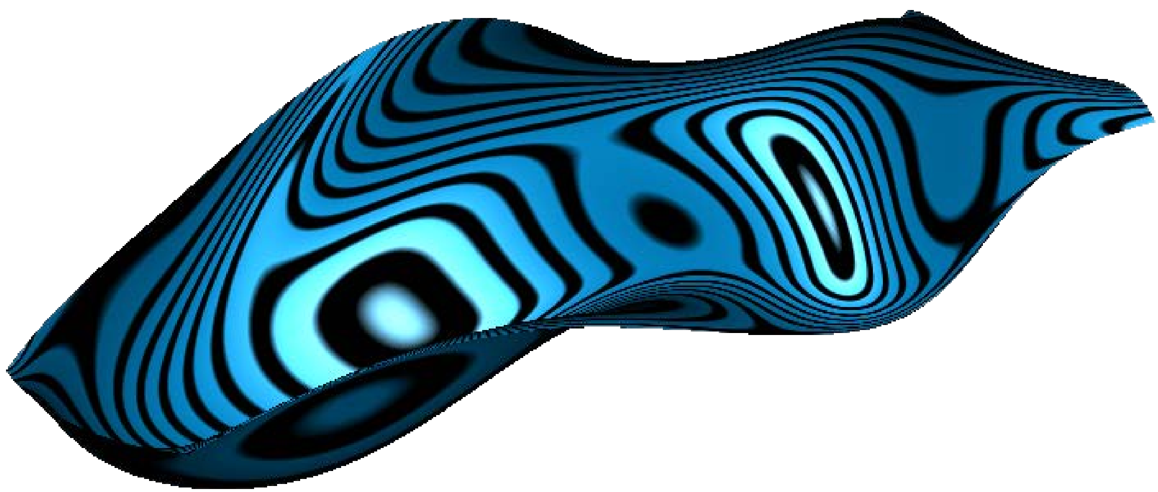
# Gordon Surfaces: Results for a Wing



# Gordon Surfaces: Results for a General Surface

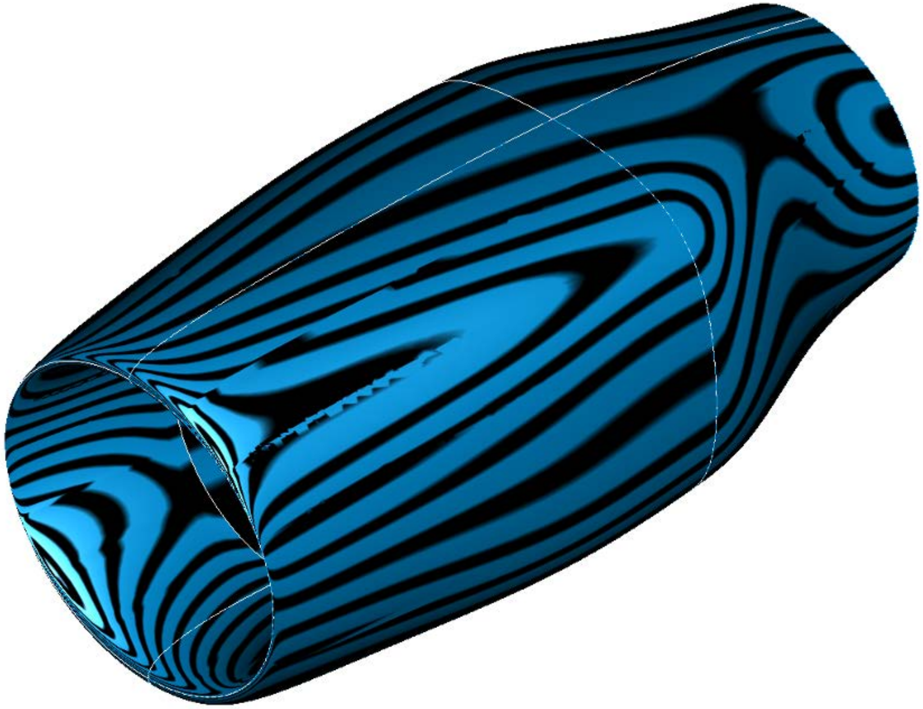






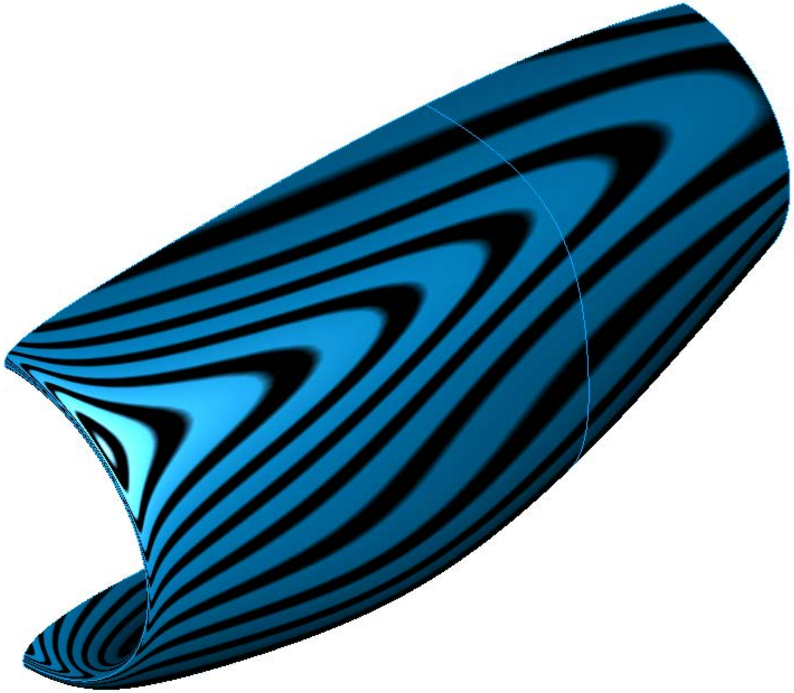
# Comparison of Coons and Gordon: Nacelle

## Coons Patches



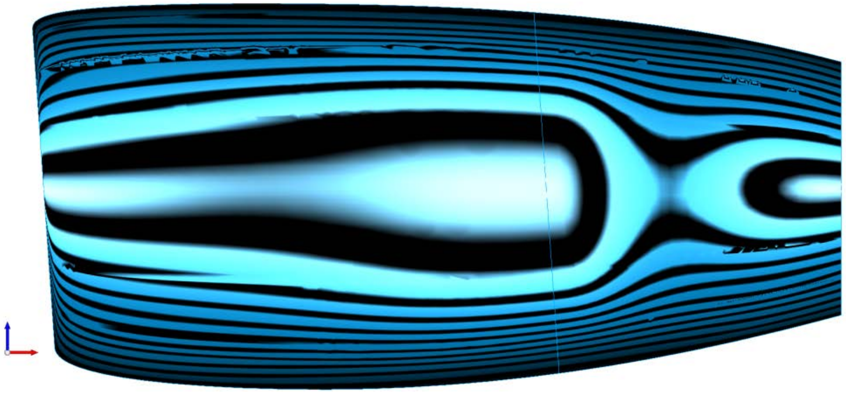
## Gordon Surface

(one half of the same nacelle)

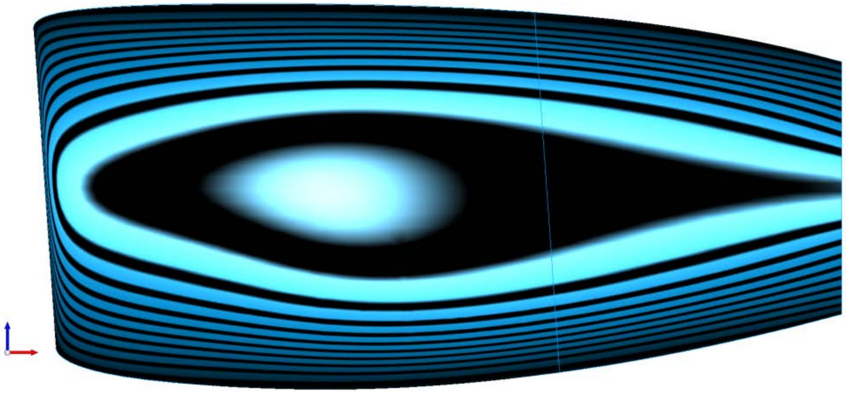


# Comparison of Coons and Gordon: Nacelle

## Coons Patches

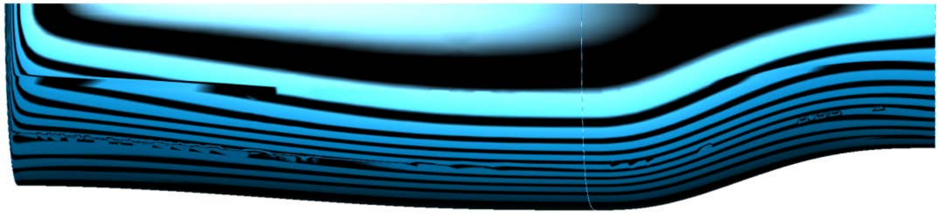


## Gordon Surface

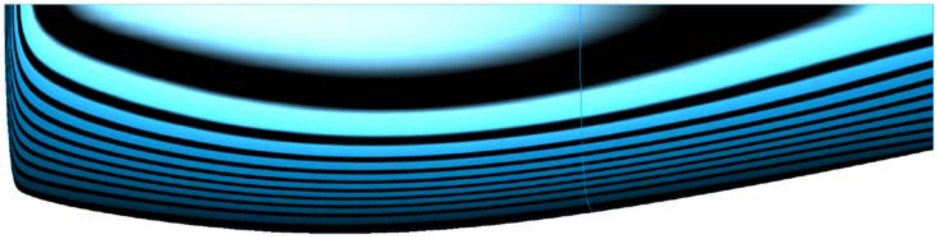


# Comparison of Coons and Gordon: Nacelle

## Coons Patches

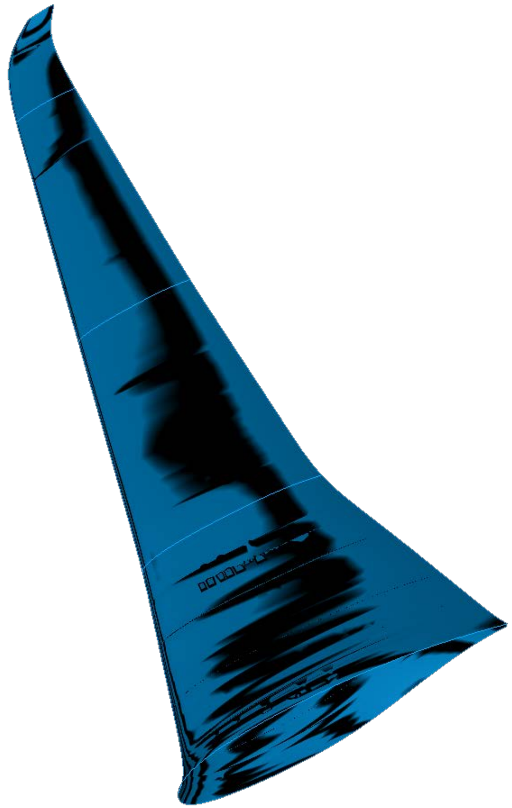


## Gordon Surface

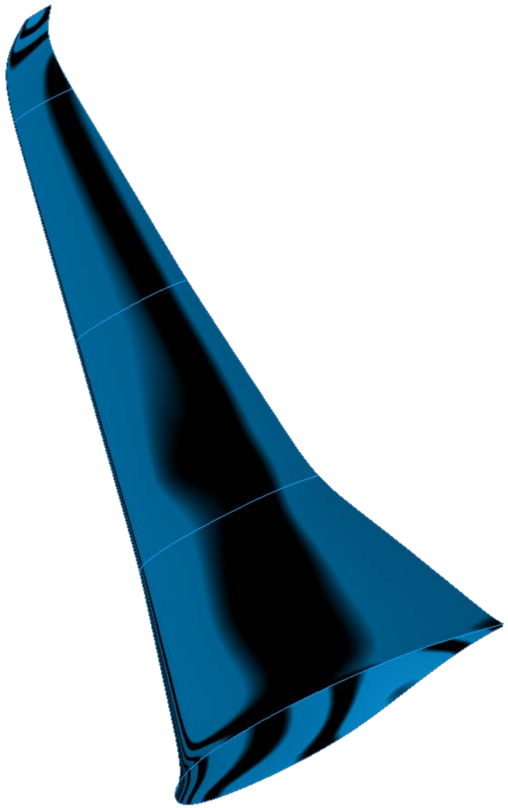


# Comparison of Coons and Gordon: Wing

## Coons Patches

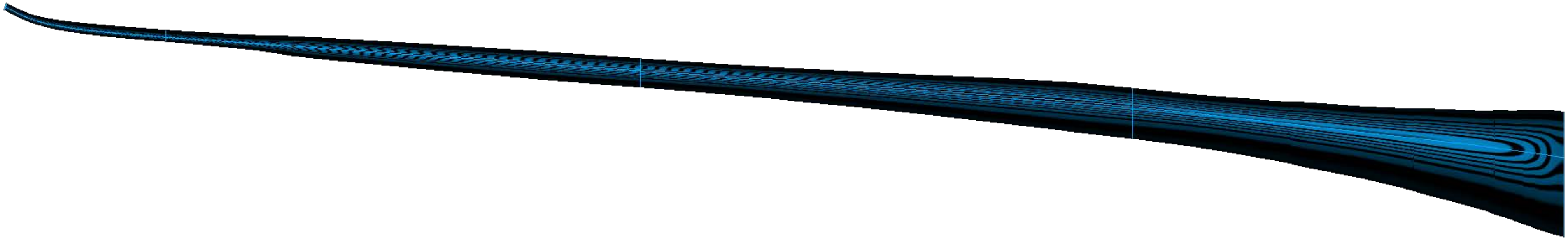


## Gordon Surface



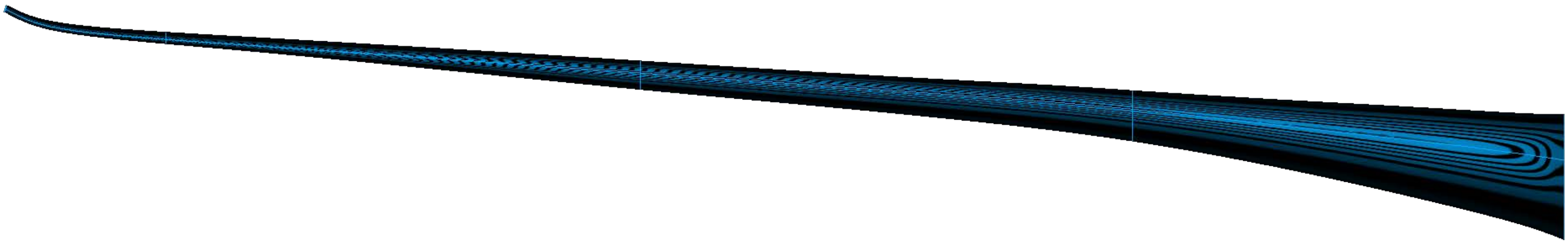
# Comparison of Coons and Gordon: Wing

## Coons Patches



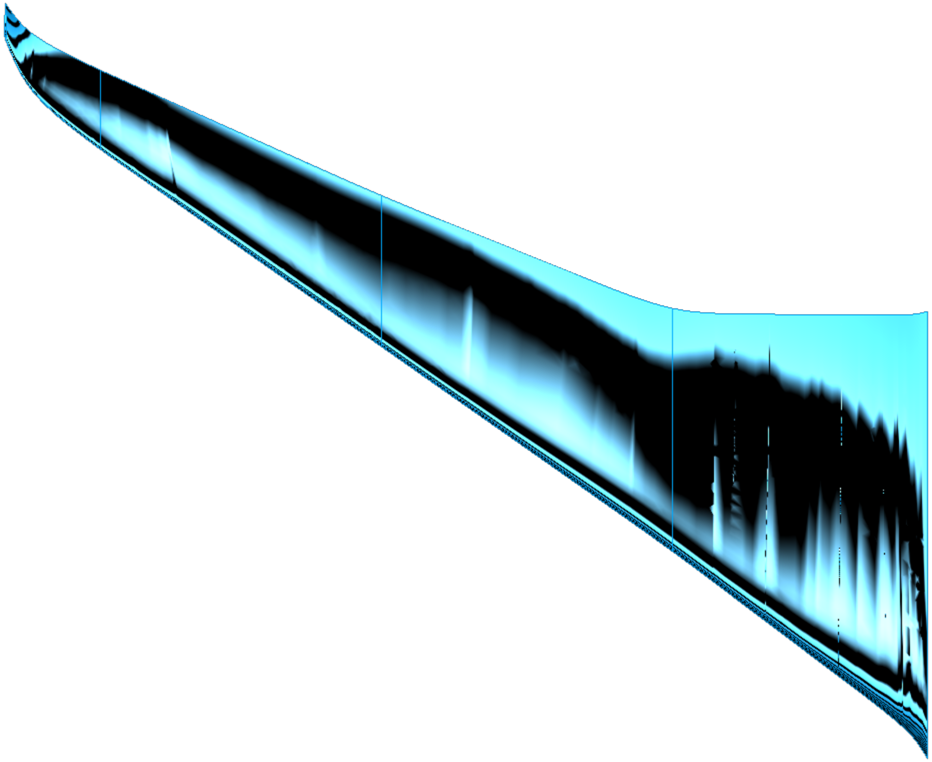
# Comparison of Coons and Gordon: Wing

## Gordon Surface

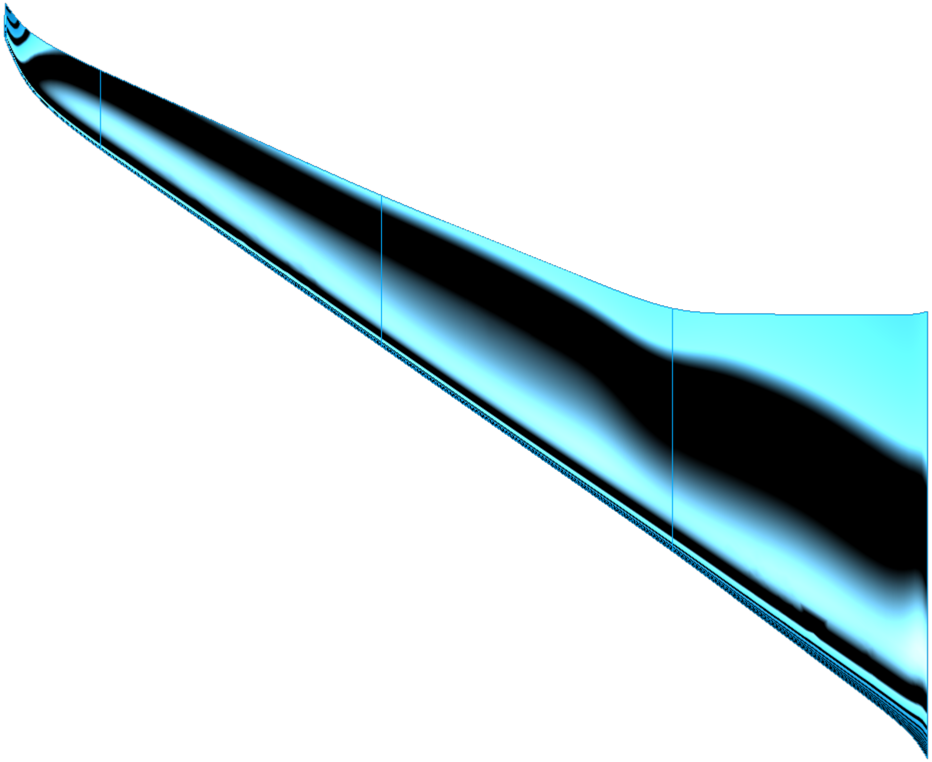


# Comparison of Coons and Gordon: Wing

## Coons Patches

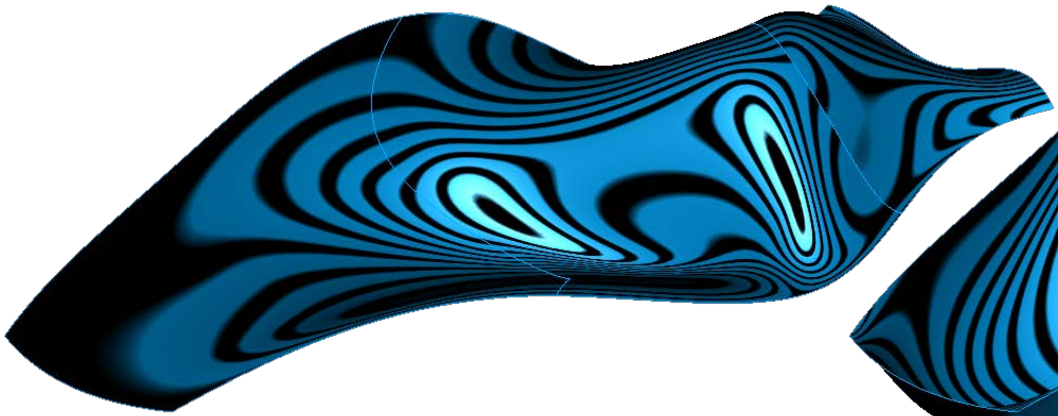


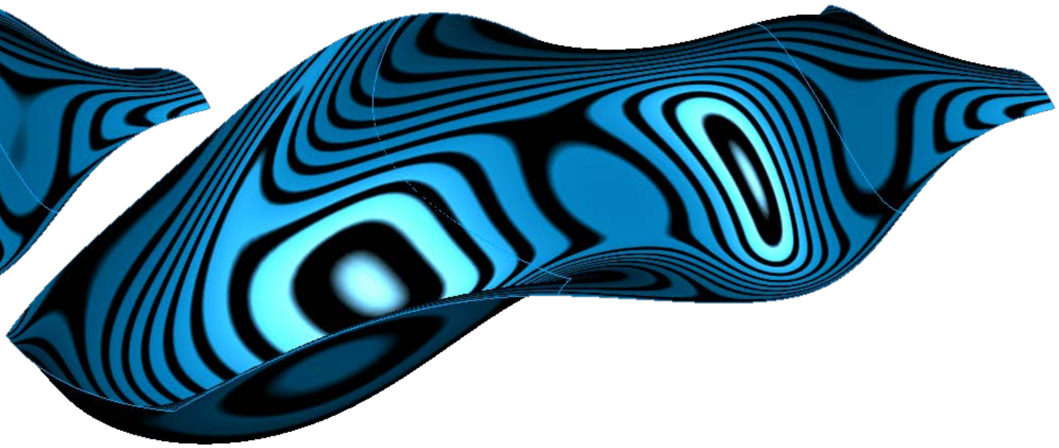
## Gordon Surface



# Comparison of Coons and Gordon: General Surface

**Coons Patches** **Gordon Surface**

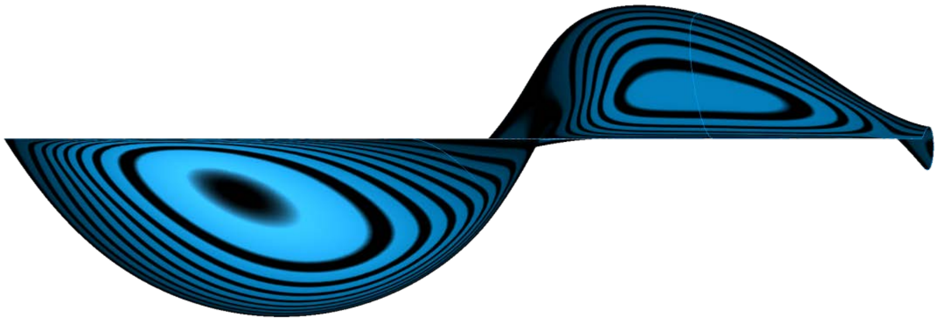




# Comparison of Coons and Gordon: General Surface

## Coons Patches

## Gordon Surface



# Summary

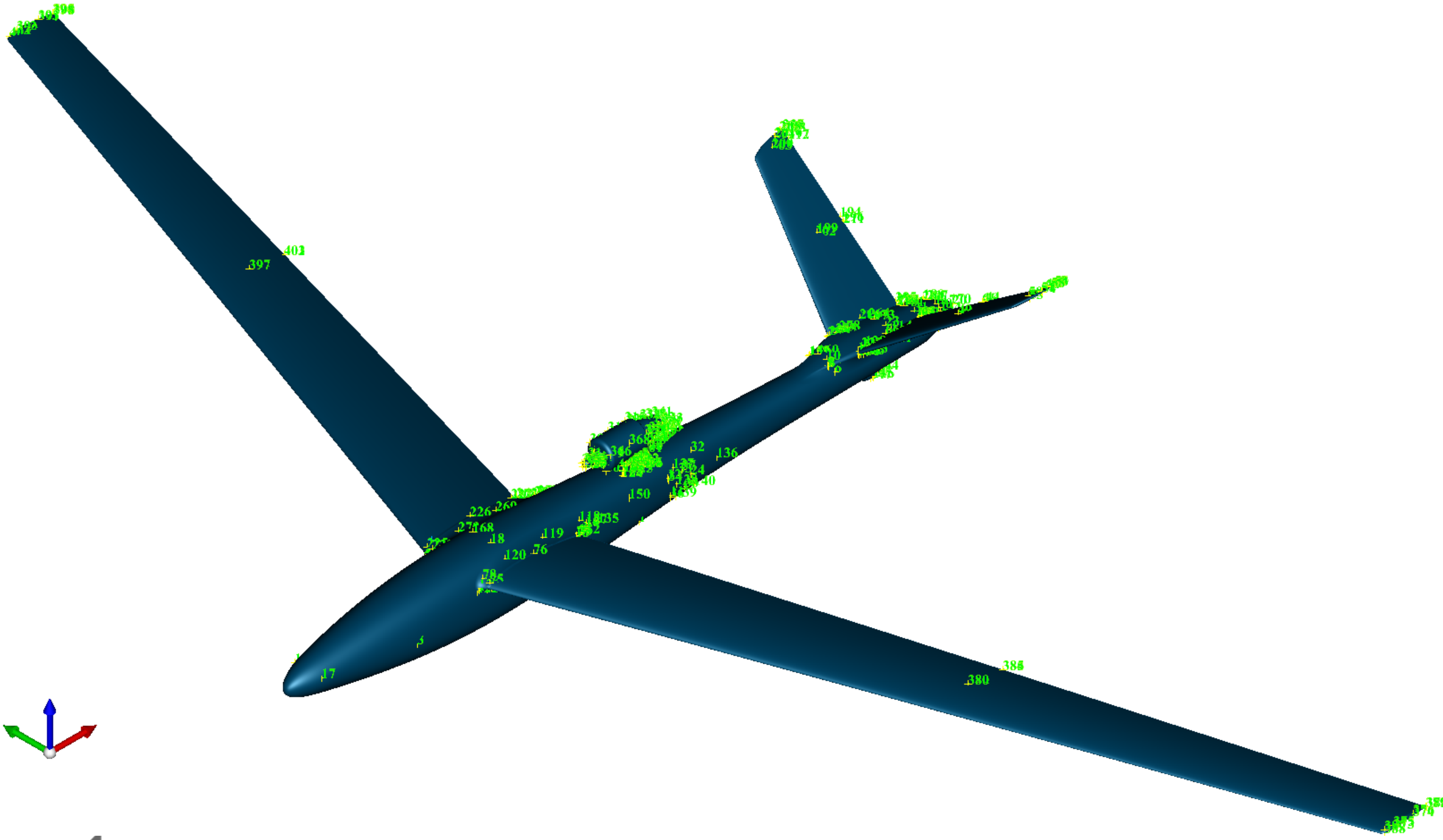
# Summary

- Gordon Surface method was presented
- Curve reparametrization is crucial→ compatible curve network

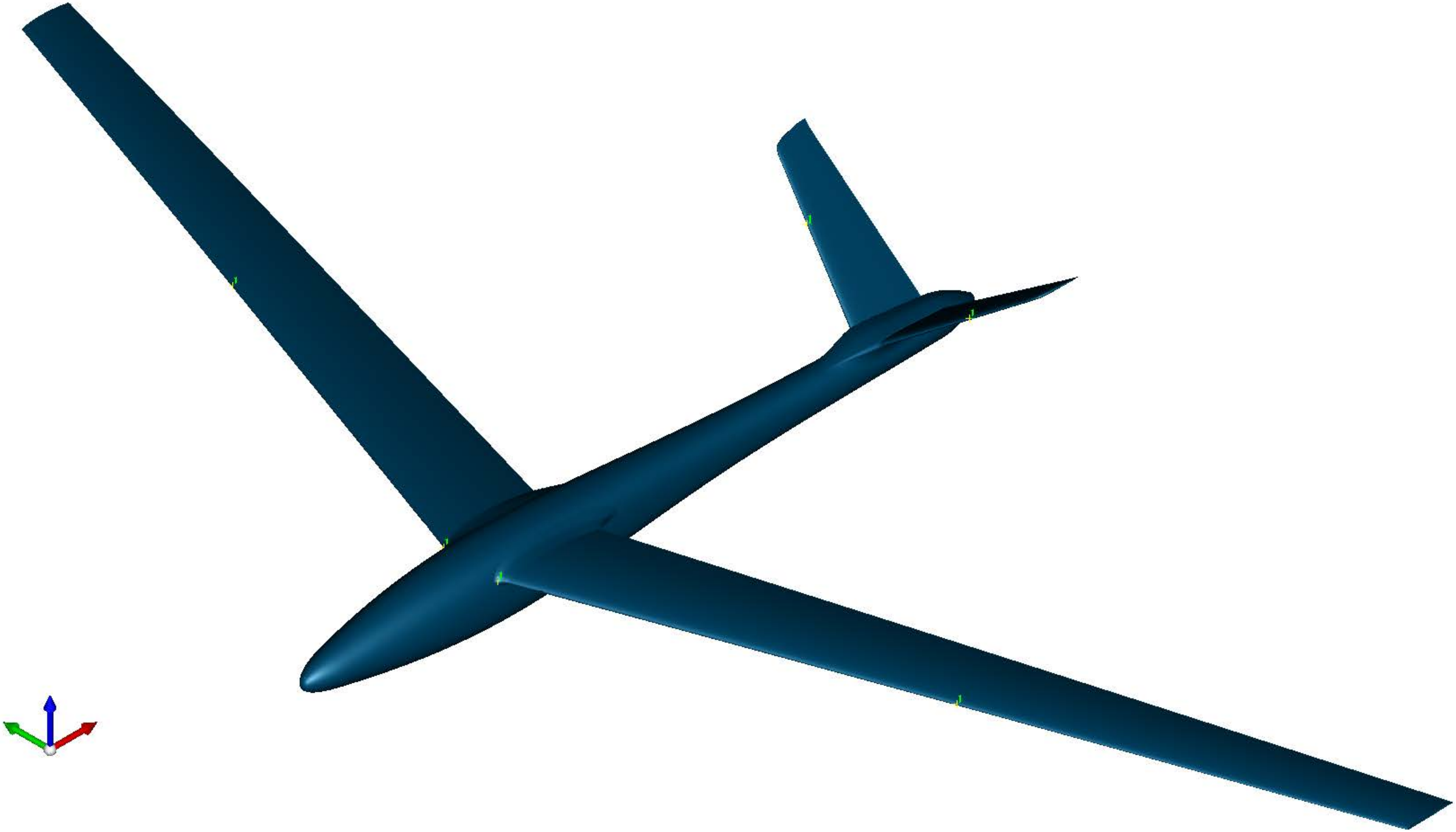
- Results:

  - ➢ all the previous problems are eliminated now
  - ➢ but: because of global interpolation oscillations might occur
  - ➢ solution: different reparametrization

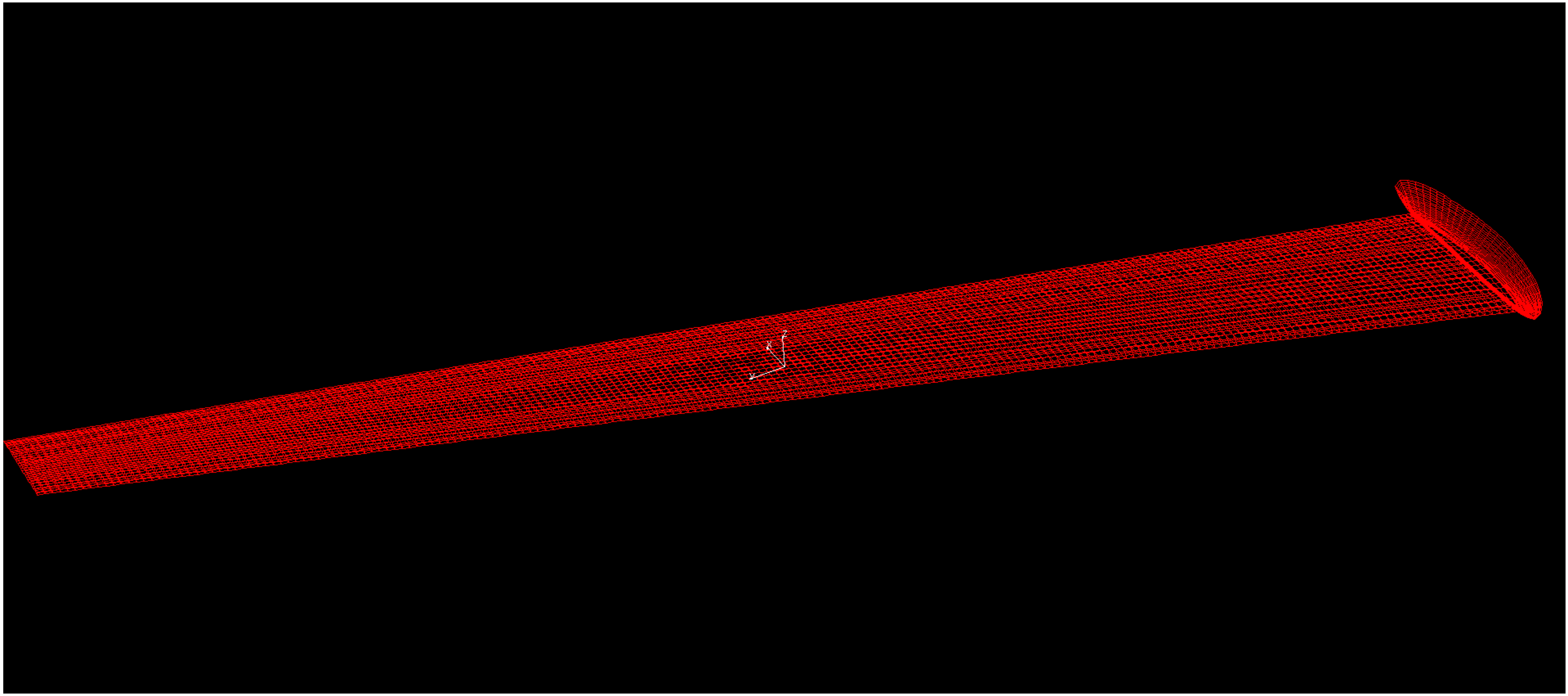
# My Current Work at SR-FLS



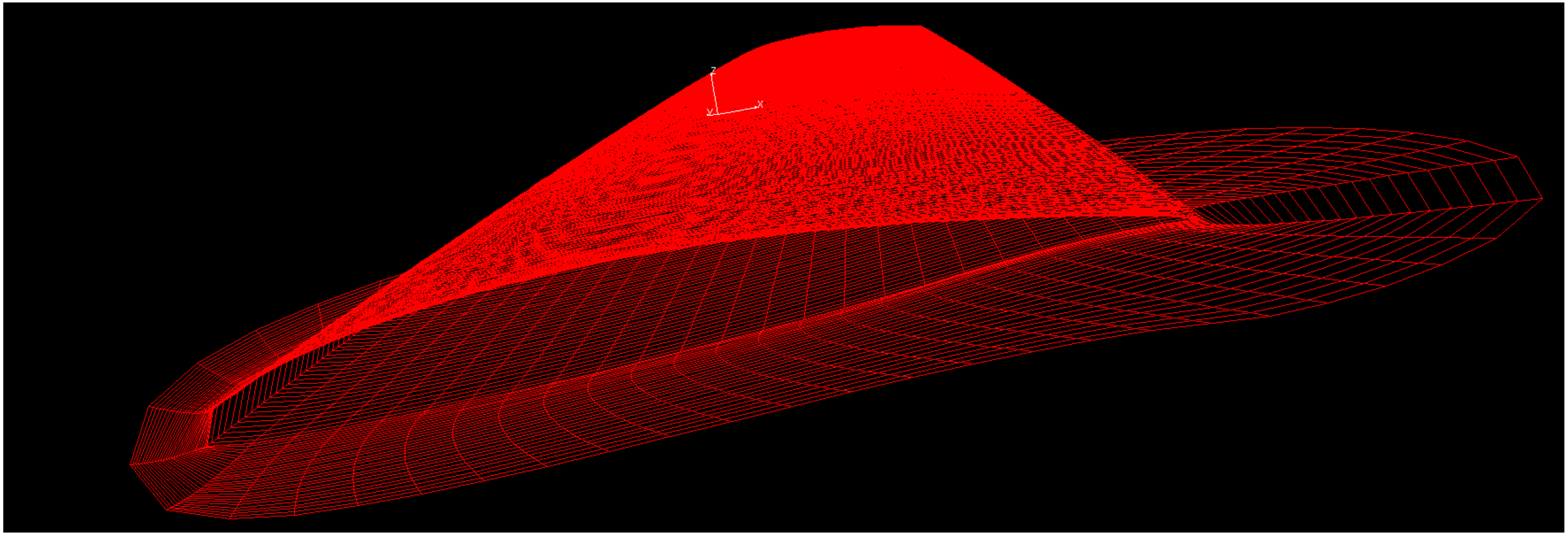
# My Current Work at SR-FLS



# My Current Work at SR-FLS



# My Current Work at SR-FLS



# References


- University of Cambridge, Department of Computer Science and Technology, https://www.cl.cam.ac.uk/teaching/2000/AGraphHCI/SMEG/

- Les Piegl and Wayne Tiller: **The NURBS book**, Springer Science & Business Media, 2012

- opencascade.com

Mathematical Theory of Gordon Surfaces:

- William J Gordon: **Spline-blended surface interpolation through curve networks**, Journal of Mathematics and Mechanics, pages 931–952, 1969.

# Thank you! ☺

# Degree Elevation

- Problem: the B-spline curve or surface shall have a higher degree
- Knot vector before:

$$U = \{\underbrace{u_0, \dots, u_0}_{k\ times}, \underbrace{u_1, \dots, u_1}_{m_1\ times}, \dots, \underbrace{u_s, \dots, u_s}_{m_s\ times}, \underbrace{u_n, \dots, u_n}_{k\ times}\}$$

- Knot vector afterwards:

$$\hat{U} = \{\underbrace{u_0, \dots, u_0}_{k+1\ times}, \underbrace{u_1, \dots, u_1}_{m_1+1\ times}, \dots, \underbrace{u_s, \dots, u_s}_{m_s+1\ times}, \underbrace{u_n, \dots, u_n}_{k+1\ times}\}$$

- The new control points are computed by evaluating the basis functions at certain $\hat{n}+1$ parameter values and solving:

$$\sum_{i=0}^{\hat{n}} B_i^{k+1}(u) Q_i = \sum_{i=0}^{n} B_i^{k}(u) P_i$$

DLR

# Knot Insertion

- Problem: insert knot $\bar{u}$ in the knot vector after the knot with index $l$

- New formula of the curve:

$$C(u) = \sum_{i=0}^{n+1} \bar{B}_i^k(u) Q_i$$

- The new control points $Q_i$ are computed by:

$$Q_i = \alpha_i P_i + (1-\alpha_i) P_{i-1}, \qquad where\ \alpha_i = \begin{cases} 1, & i \le l-k \\ \dfrac{\bar{u}-u_i}{u_{i+k}-u_i}, & l-k+1 \le i \le l \\ 0, & i \ge l+1 \end{cases}$$

- Knot insertion for B-spline surfaces analogue for both knot vectors $U$ and $V$