Master's thesis in
Applied Computer Science

CoolingGen

A parametric 3D-modeling software for turbine blade cooling geometries using NURBS

August 9, 2022

Institute for Numerical and Applied Mathematics
at the Georg-August-University Göttingen

Institute for Propulsion Technology at the
German Aerospace Center in Göttingen

Bachelor's and master's theses at the Center for
Computational Sciences at the
Georg-August-University Göttingen

Julian Lüken
julian.lueken@dlr.de

Georg-August-University Göttingen
Institute of Computer Science

☎ +49 (551) 39-172000
FAX +49 (551) 39-14403
✉ office@cs.uni-goettingen.de

www.informatik.uni-goettingen.de

I hereby declare that this thesis has been written by myself and no other resources than those mentioned have been used.

Göttingen, August 9, 2022

Abstract

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris.

Nulla malesuada porttitor diam. Donec felis erat, congue non, volutpat at, tincidunt tristique, libero. Vivamus viverra fermentum felis. Donec nonummy pellentesque ante. Phasellus adipiscing semper elit. Proin fermentum massa ac quam. Sed diam turpis, molestie vitae, placerat a, molestie nec, leo. Maecenas lacinia. Nam ipsum ligula, eleifend at, accumsan nec, suscipit a, ipsum. Morbi blandit ligula feugiat magna. Nunc eleifend consequat lorem. Sed lacinia nulla vitae enim. Pellentesque tincidunt purus vel magna. Integer non enim. Praesent euismod nunc eu purus. Donec bibendum quam in tellus. Nullam cursus pulvinar lectus. Donec et mi. Nam vulputate metus eu enim. Vestibulum pellentesque felis eu massa.

Zusammenfassung

Quisque ullamcorper placerat ipsum. Cras nibh. Morbi vel justo vitae lacus tincidunt ultrices. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. In hac habitasse platea dictumst. Integer tempus convallis augue. Etiam facilisis. Nunc elementum fermentum wisi. Aenean placerat. Ut imperdiet, enim sed gravida sollicitudin, felis odio placerat quam, ac pulvinar elit purus eget enim. Nunc vitae tortor. Proin tempus nibh sit amet nisl. Vivamus quis tortor vitae risus porta vehicula.

Fusce mauris. Vestibulum luctus nibh at lectus. Sed bibendum, nulla a faucibus semper, leo velit ultricies tellus, ac venenatis arcu wisi vel nisl. Vestibulum diam. Aliquam pellentesque, augue quis sagittis posuere, turpis lacus congue quam, in hendrerit risus eros eget felis. Maecenas eget erat in sapien mattis porttitor. Vestibulum porttitor. Nulla facilisi. Sed a turpis eu lacus commodo facilisis. Morbi fringilla, wisi in dignissim interdum, justo lectus sagittis dui, et vehicula libero dui cursus dui. Mauris tempor ligula sed lacus. Duis cursus enim ut augue. Cras ac magna. Cras nulla. Nulla egestas. Curabitur a leo. Quisque egestas wisi eget nunc. Nam feugiat lacus vel est. Curabitur consectetuer.

Suspendisse vel felis. Ut lorem lorem, interdum eu, tincidunt sit amet, laoreet vitae, arcu. Aenean faucibus pede eu ante. Praesent enim elit, rutrum at, molestie non, nonummy vel, nisl. Ut lectus eros, malesuada sit amet, fermentum eu, sodales cursus, magna. Donec eu purus. Quisque vehicula, urna sed ultricies auctor, pede lorem egestas dui, et convallis elit erat sed nulla. Donec luctus. Curabitur et nunc. Aliquam dolor odio, commodo pretium, ultricies non, pharetra in, velit. Integer arcu est, nonummy in, fermentum faucibus, egestas vel, odio.

Contents

1 Introduction

1.1 Motivation

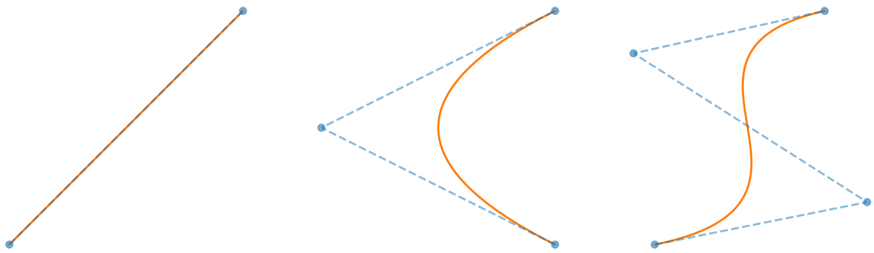
1.2 State of the Art

1.3 Problem Statement

2 Methods

2.1 Bézier Curves

Bézier curves are named after the French engineer Pierre Bézier, who famously utilized them in the 1960s to design car bodies for the automobile manufacturer Renault [Béz68]. Today, they are used in a wide variety of vector graphics applications (i.e. in font representation on computers). At first glance, the definition of the Bézier curve might seem cumbersome, but given the mathematical foundation and a few graphical representations, it becomes apparent why they are such a powerful tool in computer-aided design.



(a) Degree 1. (b) Degree 2. (c) Degree 3.

Figure 2.1: Beziér curves of different degrees (orange) and their control points (blue).

2.1.1 Definition

Definition 2.1.1. The *Bernstein basis polynomials* of degree n on the interval $[t_0, t_1]$ are defined as

$$b_{n,k,[t_0,t_1]}(t) := \frac{\binom{n}{k}(t_1 - t)^{n-k}(t - t_0)^k}{(t_1 - t_0)^n}, \tag{2.1}$$

for $k \in \{0, \ldots, n\}$.

Definition 2.1.2. Let V a vector space. A *Bézier curve* of degree n is a parametric curve $B_{P,[t_0,t_1]} : [t_0, t_1] \to V$ that has a representation

$$B_{P,[t_0,t_1]}(t) = \sum_{k=0}^{n} b_{n,k,[t_0,t_1]}(t) P_k = \sum_{k=0}^{n} \frac{\binom{n}{k}(t_1 - t)^{n-k}(t - t_0)^k}{(t_1 - t_0)^n} P_k. \tag{2.2}$$

We call the elements of the set $P = \{P_0, P_1, \ldots, P_n\} \subset V$ the *control points* of B_P.

Remark. Let $t_0 = 0$ and $t_1 = 1$. Then 2.2 simplifies to

$$b_{n,k}(t) := b_{n,k,[0,1]}(t) = \binom{n}{k}(1 - t)^{n-k} t^k \tag{2.3}$$

and 2.1 simplifies to

$$B_P(t) := B_{P,[0,1]}(t) = \sum_{k=0}^{n} \binom{n}{k}(1 - t)^{n-k} t^k P_k. \tag{2.4}$$

This case is the only case considered in this thesis. We also set the vector space V to be $\mathbb{R}^3$ unless specified otherwise.

2.1.2 De Casteljau's Algorithm

The computation of equation 2.4 is usually performed using de Casteljau's algorithm. This is because the algorithm yields a simple implementation and lower complexity than straightforwardly computing equation 2.4. The algorithm was proposed by Paul de Faget de Casteljau for the automobile manufacturer Citroën in the 1960s.

Algorithm 1 de Casteljau's algorithm

Input
    $P = \{P_0, P_1, ..., P_n\}$ set of control points
    t real number
Output
    $P_0^{(n)} = B_P(t)$ the point on the Beziér curve w.r.t. to t
procedure deCasteljau(P, t)
    $P^{(0)} \leftarrow P$
    for $i = 1, 2, ..., n$ **do**
        for $j = 0, 1, ..., n - i$ **do**
            $P_j^{(i)} \leftarrow (1-t) \cdot P_j^{(i-1)} + t \cdot P_{j+1}^{(i-1)}$
    return $P_0^{(n)}$

Theorem 2.1.3. Algorithm 1 computes $B_P(t)$.

Proof. By induction. Let $n = 1$. Then

$$P_0^{(1)} = (1-t) \cdot P_0 + t \cdot P_1.$$

By employing the induction hypothesis

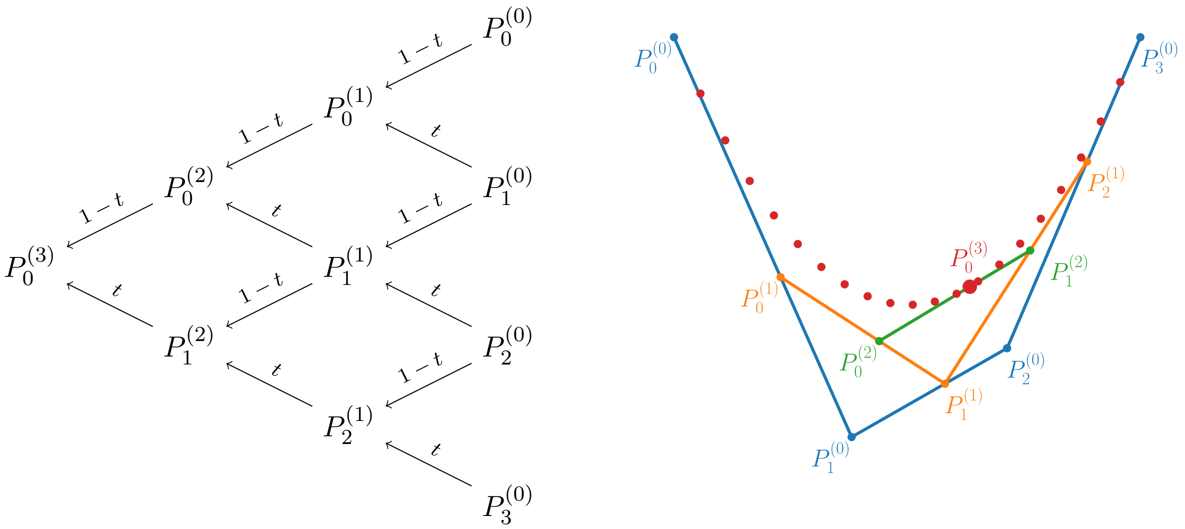
$$P_j^{(n)} = \sum_{k=j}^{n+j} \binom{n}{k} (1-t)^{n-k} t^k P_{j+k}$$

for some $n \in \mathbb{N}$, we can infer that

$$\begin{aligned} P_0^{(n+1)} &= (1-t) \cdot P_0^{(n)} + t \cdot P_1^{(n)} \\ &= (1-t) \cdot \sum_{k=0}^{n} \binom{n}{k} (1-t)^{n-k} t^k P_k + t \cdot \sum_{k=1}^{n+1} \binom{n}{k} (1-t)^{n-k} t^k P_{k+1} \\ &= \sum_{k=0}^{n+1} \binom{n+1}{k} (1-t)^{n+1-k} t^k P_k, \end{aligned}$$

which is equal to $B_P(t)$ for degree $n+1$. $\square$

A visual representation of Algorithm 1 yields a triangular scheme. To compute one point on a Beziér curve B_P with degree n, one has to perform $\frac{n^2-n}{2}$ additions and $n^2 - n$ scalar multiplications.



(a) Algorithmic visualization.

(b) Geometric visualization of $B_P(\frac{3}{5})$.

Figure 2.2: Visual representations of de Casteljau's algorithm.

Interestingly, the representation of the algorithm in Figure 2.2 also gives rise to an intuitive visualization of the geometric shape of the Beziér curve B_P. For all $i \in \{0, ..., n\}$ and all $j \in \{0, ..., n-i\}$, the point $P_j^{(i+1)}$ is the convex combination (always w.r.t. t) of $P_j^{(i)}$ and $P_{j+1}^{(i)}$. Thus $P_j^{(i+1)}$ always lies on the line segment between $P_j^{(i)}$ and $P_{j+1}^{(i)}$, as can be observed in the example in Figure 2.2.

2.1.3 Properties

Other than being remarkably intuitive, Beziér curves have a lot of properties which make them convenient. In computer-aided design software, most graphical user interfaces rely on the principle of letting the user interactively drag and drop the control points with a mouse, granting them control over the shape of Beziér curve. The following theorems further illustrate why this is a good idea.

Theorem 2.1.4. $B_P(0) = P_0$ and $B_P(1) = P_n$.

Proof. Explicit computation yields

$$B_P(0) = \sum_{k=0}^{n} \binom{n}{k} t^k P_k = \binom{n}{0} P_0 = P_0$$

and

$$B_P(1) = \sum_{k=0}^{n} \binom{n}{k} (1-t)^{n-k} P_k = \binom{n}{n} P_n = P_n.$$

□

Theorem 2.1.5. Let $T \in \mathbb{R}^{3\times 3}$. Then $B_{TP}(t) = TB_P(t)$ where $TP := \{TP_0, TP_1, ..., TP_n\}$.

Proof. For all $t \in [0,1]$ we can directly compute

$$TB_P(t) = \sum_{k=0}^{n} \binom{n}{k} (1-t)^{n-k} t^k TP_k = B_{TP}(t).$$

$\square$

Theorem 2.1.6. $B_P(t)$ lies in the convex hull of P for all $t \in [0,1]$.

Proof. By the algorithm of de Casteljau (1), we know that $P_j^{(i)} = (1-t) \cdot P_j^{(i-1)} + t \cdot P_{j+1}^{(i-1)}$ for all $t \in [0,1]$. Therefore, $P^{(i)}$ lie in the convex hull of $P^{(i-1)}$. But then $B_P(t) = P_0^{(n)}$ always lies in the convex hull of $P^{(0)} = P$ by induction. $\square$

Simple as their appearance may be, Beziér curves fall short of representing some of the most common geometric shapes. Given a finite number of control points, we can never make $B_P(t)$ a circular arc, although a circle has a very simple parametric form. One of their greatest perks, the ability to describe a shape with just a handful of control points, is their greatest shortcoming at the same time. This is most likely the reason why Beziér curves are not the state of the art in technical engineering applications. However, Beziér curves certainly do provide an intuition for Non-Uniform Rational B-Splines or NURBS, which is their prevailing counterpart.

2.2 Non-Uniform Rational B-splines (NURBS)

2.2.1 Definition

Similarly to how Beziér curves are defined on the Bernstein polynomial basis, NURBS are defined on basis functions called basis splines (or more commonly B-splines), which are recursively defined on a *knot sequence* $(t_m)_{m=-\infty}^{\infty} \subset \mathbb{R}$ with $t_m \leq t_{m+1}$ for all $m \in \mathbb{Z}$.

Definition 2.2.1. The *B-splines of degree* 0 on a knot sequence (t_m) are defined as

$$N_{0,k}^{(t_m)}(t) := \begin{cases} 1 & \text{if } t \in [t_k, t_{k+1}), \\ 0 & \text{else.} \end{cases} \tag{2.5}$$

The *B-splines of degree* $p+1$ for $p \in \mathbb{N}$ are given by the *Cox-de-Boor recursion formula*

$$N_{p+1,k}^{(t_m)}(t) := \omega_{p,k}^{(t_m)}(t)\, N_{p,k}^{(t_m)}(t) + (1 - \omega_{p,k+1}^{(t_m)}(t))\, N_{p,k+1}^{(t_m)}(t), \tag{2.6}$$

where

$$\omega_{p,k}^{(t_m)}(t) := \begin{cases} \frac{t-t_k}{t_{k+p}-t_k} & \text{if } t_{k+p} \neq t, \\ 0 & \text{else.} \end{cases} \tag{2.7}$$

Remark. Instead of $N_{p,k}^{(t_m)}$ we write $N_{p,k}$ and explicitly refer to (t_m) when necessary. We restrict the domain of definition of $N_{p,k}$ to $[0,1)$ by setting $\lim_{m \to -\infty} t_m = 0$ and $\lim_{m \to \infty} t_m = 1$.

Definition 2.2.2. Let V a vector space. A *B-spline curve* over a set of control points $P = \{P_0, P_1, ...P_n\} \subset V$ is defined as

$$S_P(t) = \sum_{k=0}^{n} N_{p,k}(t)P_k.$$

Definition 2.2.3. A *NURBS curve* $C_P(t)$ of degree p with the control points $P = \{P_0, P_1, ..., P_n\} \subset V$, the control weights $w = (w_0, w_1, ...w_n) \subset \mathbb{R}$ and a knot sequence (t_m) is defined as

$$C_P(t) = \frac{\sum_{k=0}^{n} N_{p,k}(t)w_k P_k}{\sum_{k=0}^{n} N_{p,k}(t)w_k}. \tag{2.8}$$

Remark. Let $\dim V = d$. A NURBS curve can alternatively be understood as a projection of a B-spline curve on a transformed set of control points. For this purpose we define the embedding into the weighted vector space

$$\Phi_w : V \to V \times \mathbb{R}$$

that maps each control point $P_k = (p_1, ..., p_d) \in V$ onto $(w_k p_1, ..., w_k p_d, w_k) \in V \times \mathbb{R}$. We also have to define the projection map

$$\Phi^\dagger : V \times \mathbb{R} \to V$$

that maps each point on the B-spline curve $S_{\Phi(P)}(t) = (s_1, ..., s_d, s_{d+1})$ onto $(\frac{s_1}{s_{d+1}}, ..., \frac{s_d}{s_{d+1}})$. We can then define the NURBS curve as

$$C_P(t) = \Phi^\dagger(S_{\Phi_w(P)}(t))$$

which we will make use of later on.

The notion of the knot sequence (t_m) is commonly computationally simplified to that of a knot vector τ, since τ only contains a finite number of elements. For our purposes, we let

$$\tau = (\underbrace{t_0, ..., t_{p-1}}_{\text{p elements, all } =0}, t_p, ..., t_n, \underbrace{t_{n+1}, t_{n+p}}_{\text{p elements, all } =1}),$$

where $t_k = 0$ for $k \in \{0, ..., p-1\}$ and $t_k = 1$ for $k \in \{n+1, ..., n+p\}$. We still require the monotony property $t_k \leq t_{k+1}$ for all $k \in \{0, ..., n+p\}$.

2.2.2 De Boor's Algorithm

To efficiently calculate points on a B-spline curve, Carl-Wilhelm Reinhold de Boor devised an efficient algorithm, the construction of which rigorously demonstrates its correctness. Together with the embedding Φ_w and the projection $\Phi^\dagger$ from the Remark for Definition 2.2.3, this algorithm can also be used to calculate points on a NURBS curve.

Let $P = \{P_0, P_1, ..., P_n\}$ a set of control points, (t_m) a knot sequence and p the degree of the

B-spline curve $S(t)$. Then by the Cox-de-Boor recursion formula, we find

$$\begin{aligned} S(t) &= \sum_{k=0}^{n} N_{p,k}(t)\, P_k \\ &= \sum_{k=1}^{n} \omega_{p-1,k}(t)\, N_{p-1,k}(t)\, P_k \;+\; \sum_{k=0}^{n} \left(1-\omega_{p-1,k+1}(t)\right)\, N_{p-1,k+1}(t)\, P_k. \end{aligned}$$

Changing the limits of the second term, we can summarize the two terms as

$$\begin{aligned} S(t) &= \sum_{k=1}^{n} N_{p-1,k}(t)\, \underbrace{\Big[\omega_{p-1,k}(t)\, P_k \;+\; \left(1-\omega_{p-1,k}(t)\right)\, P_{k-1}\Big]}_{=:P_k^{(1)}(t)} \\ &= \sum_{k=1}^{n} N_{p-1,k}(t)\, P_k^{(1)}(t). \end{aligned}$$

Recursively defining

$$P_k^{(j)}(t) := \begin{cases} \omega_{p-j,k}\, P_k^{(j-1)}(t) \;+\; \left(1-\omega_{p-j,k}\right) P_{k-1}^{(j-1)}(t) & \text{if } j > 0, \\ P_k & \text{else,} \end{cases} \tag{2.9}$$

we can repeat this process up to $p-2$ more times, finding

$$S(t) = \sum_{k=p-1}^{n} P_k^{(p-1)}(t)\, N_{1,k}(t).$$

The iteration process in Equation 2.9 is the key step in de Boor's algorithm. It is already clear from this place that the points calculated on a B-spline curve are in fact also a cumultated convex combination of control points, just as it is the case with Beziér curves. As zero-values for $\omega_{i,j}$ (or $1-\omega_{i,j}$) can be completely omitted by checking multiplicity, we arrive at the following algorithm:

Algorithm 2 de Boor's algorithm

Input

$P = \{P_0, P_1, ..., P_n\}$ set of control points of the B-spline curve

$T = (t_0, t_1, ..., t_{n+p+1})$ knot vector of the B-spline curve

p degree of the B-spline curve

$t \in [t_0, t_{n+p+1})$ real number

Output

$C_P(t)$ the point on the NURBS curve w.r.t. to t

procedure DEBOOR(P, p, T, t)

Find k such that $t \in [t_k, t_{k+1})$

Let m be the multiplicity of t in the knot vector T

$P^{(0)} \leftarrow P$

for $i = 1, 2, ..., p - m$ **do**

for $j = k - p + i, ..., k - m$ **do**

$\omega_j^{(i)} \leftarrow \frac{t - t_j}{t_{j+p-i+1} - t_j}$

$P_j^{(i)} \leftarrow (1 - \omega_j^{(i)}) \cdot P_{j-1}^{(i-1)} + \omega_j^{(i)} \cdot P_j^{(i-1)}$

return $P_{k-m}^{(p-m)}) = C_P(t)$

Theorem 2.2.4. Algorithm 2 computes $S(t)$.

Proof. By construction. □

A visual representation of Algorithm 2 reveals that this algorithm is in fact a generalization of 1. Notice that in the algorithm $\omega_{p-i,j}$ was renamed to $\omega_j^{(i)}$.

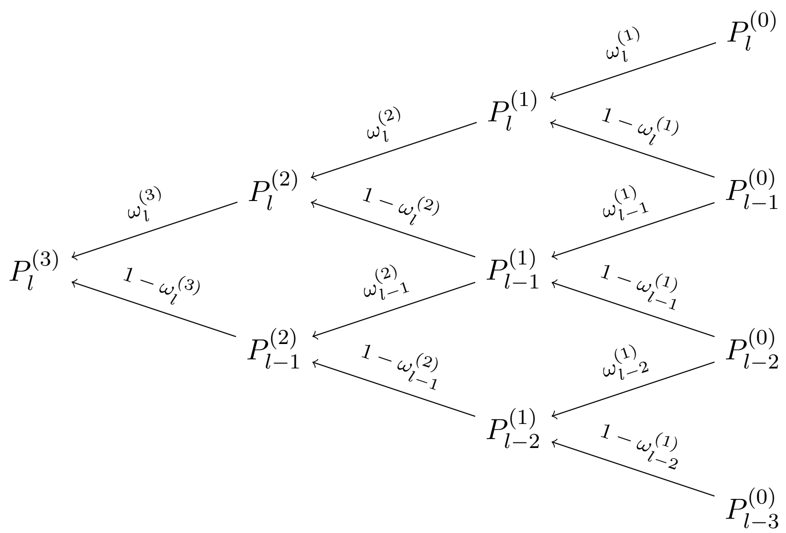


Figure 2.3: Algorithmic visualization.

Using the embedding map Φ_w and the projection map $\Phi^\dagger$, we can employ the strategy to calculate points on a NURBS curve $C_P(t)$:

1. Calculate $P_w = \Phi_w(P)$.

2. Use de Boor's algorithm to calculate points on S_{P_w}.

3. Project points onto V by applying $\Phi^\dagger$ to S_{P_w} to find $C_P(t)$.

2.2.3 Properties

2.3 Methods on NURBS Objects

2.3.1 Affine Transformations

2.3.2 The Frenet-Serret Apparatus

2.3.3 Finding Intersections

2.3.4 Interpolation

2.4 Jet Engine Design Specifics

2.4.1 Fundamental Terms

2.4.2 The S2M Net

2.4.3 Fillet Creation

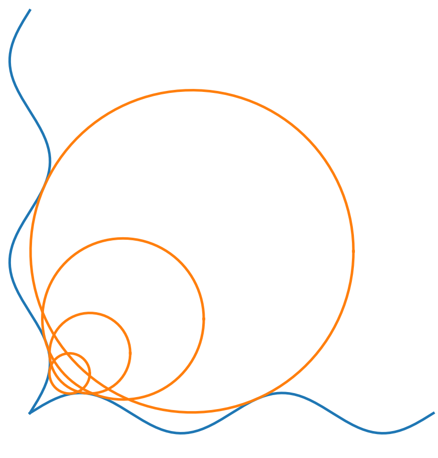


Figure 2.4: yeah

3 Results

3.1 Cooling Geometries And Their Parametrizations

3.1.1 Chambers

3.1.2 Turnarounds

3.1.3 Slots

3.1.4 Film Cooling Holes

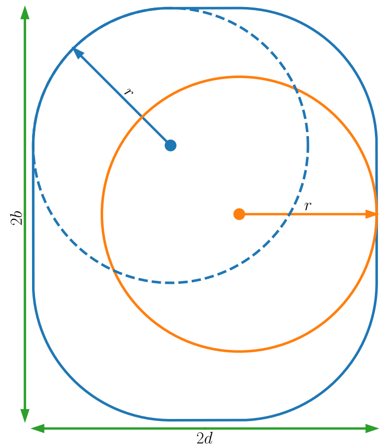


Figure 3.1: yeah

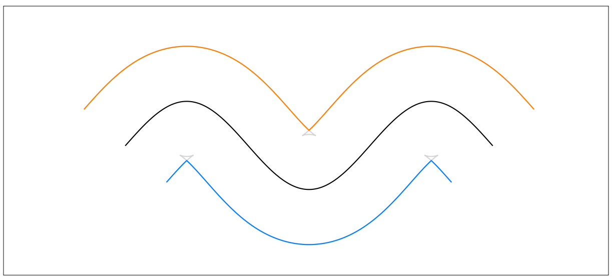


Figure 3.2: yeah

3.1.5 Impingement Inserts

3.2 Export for CENTAUR

3.3 Export for Open CASCADE

4 Discussion

4.1 Future Work

4.2 Conclusion

[Pie97]

5 References

[Béz68] Pierre E. Bézier. "How Renault Uses Numerical Control for Car Body Design and Tooling". In: *SAE Technical Paper Series*. SAE International, Feb. 1968. DOI: 10.4271/680010.

[Pie97] Les A. Piegl. *The NURBS book*. Springer, 1997, p. 646. ISBN: 3540615458.