Proceedings of CONF-CIAP 2026 Symposium: International Conference on Atomic Magnetometer and Applications
DOI: 10.54254/2753-8818/2026.HZ29024

N-order Generalized Trigonometric Bézier Curves for Geometric Design and Engineering Applications

Payton Hu

Tsinghua International School, Beijing, China
paytonhu2007@icloud.com

Abstract. Bézier curves have proven to be one of the most powerful tools in geometric design and engineering due to their simplicity, efficiency, and flexibility. This paper discusses the mathematical theory and algorithms behind the construction of Bézier curves, including the classical de Casteljau method and cubic spline extensions, and introduces an online Streamlit-based tool for visualizing their applications. By generalizing these algorithms, our framework provides enhanced shape adjustability in curve design and facilitates interactive exploration of their properties. The tool demonstrates Bézier curves' wide-ranging applications in computer-aided design, generative digital art, airfoil design optimization, and motion planning for autonomous systems. These examples highlight how a single mathematical framework can unify diverse domains, reducing fragmentation between theory and practice. In addition to illustrating established algorithms, we show how Bézier-based parameterizations enable intuitive geometric manipulation and maintain stability under affine transformations. The educational utility of our platform also helps students and engineers bridge the gap between abstract mathematics and real-world engineering challenges. Future extensions of this work include expanding to three-dimensional curve and surface generation, incorporating optimization methods for design automation, and applying the framework to real-time simulations. Overall, our study highlights the enduring importance of Bézier curves as a versatile and practical tool that connects mathematics, engineering, and digital creativity.

Keywords: Bezier Curve, Applications, Engineering, Computer aided design, Splines

1. Introduction

Imagine the elegant curves on the outside of a sports car, or the seamless movement of a video game character. The ability to craft these smooth animations and curves is made possible by mathematical tools that model curvature with control and efficiency. One of the most popular tools is the Bezier curve. Named after Pierre Bézier [1], a French Engineer who designed the exterior of Renault cars and publicized the Bezier curve in the 1960's, the precision and control these simple curves revolutionized digital art and design.

The Bezier curve allows the generation of smooth, scalable and intricate curves easily with just a few control points [2]. Its elegance lies in its simplicity: a curve can be constructed by just using recursive linear interpolation across a few points on the coordinate plane (which is also known as de

© 2025 The Authors. This is an open access article distributed under the terms of the Creative Commons Attribution License 4.0 (https://creativecommons.org/licenses/by/4.0/).

DOI: 10.54254/2753-8818/2026.HZ29024

Casteljau algorithm). Computers only need to store a small amount of data to create these manipulatable curves. The efficiency of Bezier curves makes them extremely useful in digital design, such as font design, graphics editors, and animation.

In addition, properties such as continuity and affine invariance of Bézier curves ensure that no unexpected distortions occur, which is why they play a significant role in trajectory planning and airfoil design. Their predictability and stability are crucial for tasks that require optimization and control.

This paper investigates how the mathematical functions and core algorithms behind the Bezier curve constructs highly flexible and intuitive curves. In parallel to theory, we also developed a Streamlit-Based tool to visualize and show Bezier curves' practical applications. The tool is created for practical and educational purposes, teaching users about how these curve function and showing how they bridge the gap between mathematical theory and real-life utility in engineering, digital art, and physics. It offers a hands-on method to explore the curves.

Through the demonstration of Bezier curves in different fields of study, we make the theory of Bezier curves more accessible. The tool allows users to see the connection between abstract math and engineering and design.

2. Definition and theory of Bezier curves

Algorithms used to generate Bezier curves are defined and discussed in this section.

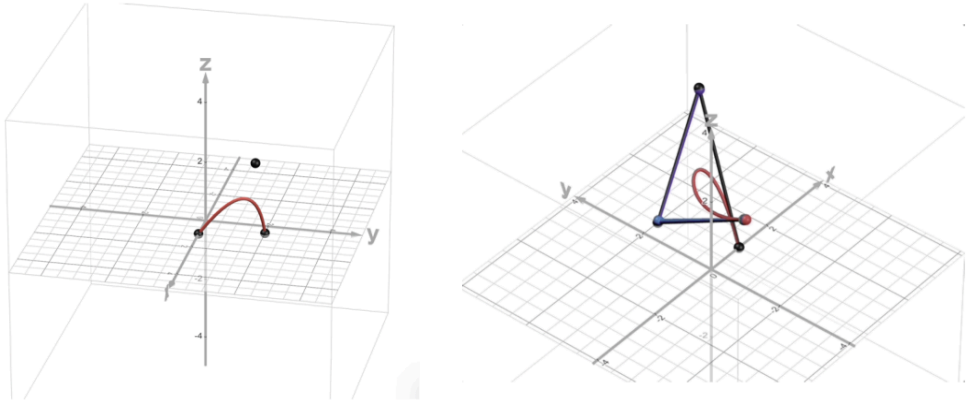


Figure 1. Three-dimensional visualizations of Bézier curves. (Left) Visualization of a Cubic Bezier Curve in R3. A cubic Bézier curve (n=3) constructed from four control points (black dots). The red curve illustrates how the polynomial interpolation produces a smooth trajectory lying within the convex hull of its control polygon. (Right) Visualization of a Quartic Bezier Curve in R3. A quartic Bézier curve (n=4) defined by five control points, with the control polygon shown in blue. Compared to the cubic case, the quartic formulation provides additional flexibility in shaping the curve, while still preserving properties such as affine invariance and containment within the convex hull

2.1. Bernstein polynomial form

Definition:

Given control points $P_0, \ldots, P_n \in R^2$, the Bezier curve is:

DOI: 10.54254/2753-8818/2026.HZ29024

$$B(t)=\sum_{i=0}^{n}\binom{n}{i}(1-t)^{n-i}t^{i}P_i=\sum_{i=0}^{n}b_i(t)P_i\ t\in[0,1]. \tag{1}$$

$\binom{n}{i}(1-t)^{n-i}t^{i}=b_i(t)$ are the Bernstein basis polynomials. The Bezier curve lies in the convex hull (a "rubber band" region formed around the control points) formed by $n+1$ control points, and its endpoints are $B(0)=P_0, B(1)=P_n$. For $t\notin[0,1]$ the polynomial extrapolates beyond the endpoints, and the shape does not change (see Fig. 1).

Properties:

- Affine invariance

• Any affine transformation to the control points still results in the same Bezier Curve.

Let the scalar of the Berstein polynomial be: $b_i(t)=\binom{n}{i}(1-t)^{n-i}t^{i}$, then, for any affine map $A(x)=Mx+b$, $A(B(t))=\sum_i b_i(t)A(P_i)$

Proof:

Using linearity and the definition: $\sum_i b_i(t)=1$

$$A(B(t))=M\left(\sum_i b_i(t)P_i\right)+b=\sum_i b_i(t)MP_i+\left(\sum_i b_i(t)\right)b=\sum_i b_i(t)(MP_i+b)=\sum_i b_i(t)A(P_i) \tag{2}$$

- Variation Diminishing

• The curve cannot have oscillations that are not already implied by the control polygon, the control polygon being the polygon formed by connecting all the control points in sequence.

• If the control polygon is monotone in a direction, the curve is also monotone in that direction.

• Edits to the curve/control points are stable.

• The Bernstein polynomials are totally positive, which implies the variation diminishing property.

- Convex Hull

• When $t\in[0,1]$, the curve lies within the control polygon.

2.2. de Casteljau's Algorithm

The method of using repeated affine linear interpolation to construct $B(t)$ is called the de Casteljau's Algorithm. It starts from the first control point of the control polygon, and through a series of linear interpolations of ratio t with neighboring points, $B(t)$ forms [2].

Definition:

Starting with level 0 $P_i^0=P_i$:

$$P_i^k(t)=(1-t)P_i^{k-1}(t)+tP_{i+1}^{k+1}(t) \tag{3}$$

And $B(t)=P_0^n(t)$. The Bezier curve is created through multiple control polygons inside the previous, imagine a dot on the side of a line with endpoints at the sides of a triangle with vertices on

DOI: 10.54254/2753-8818/2026.HZ29024

the edges of a quadrilateral etc.

Each “level” is a control polygon inside the previous one; the levels nest and visually converge to $B(t)$. The algorithm is equivalent to the Bernstein polynomial form, while this method is geometric, the Berstein polynomial form is algebraic (see Fig. 2).

Properties:

- Can be split into multiple Bezier curve segments at any t
- Numerically stable
- Convex Hull

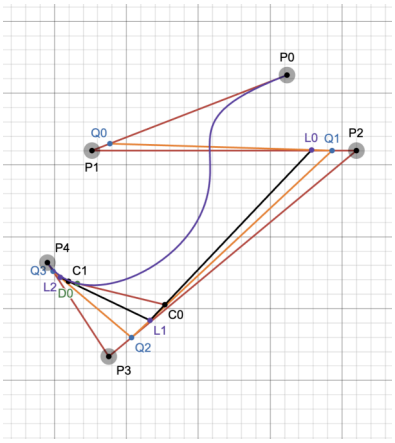


Figure 2. Visualization of the construction of a quartic Bézier curve using de Casteljau’s algorithm. The five control points P0, P1, P2, P3, P4 are shown in black. Linear interpolations between adjacent control points (L0, L1, L2, L3) are drawn in orange. Subsequent interpolations between these points (Q0, Q1, Q2) are shown in blue, followed by the next level (C0, C1) in green. The final interpolation between C0 and C1 yields the point on the Bézier curve corresponding to the chosen parameter value t. The recursive nature of the algorithm constructs the curve point by point, illustrating the geometric equivalence to the Bernstein polynomial formulation

2.3. Arc length, parameter speed, and curvature

2.3.1. Arc length

Bezier curves are parametric, thus the length of the curve from $t=0$ to $t=1$ can be obtained via the integral:

$$L=\int_0^1 \sqrt{(x'(t))^2+(y'(t))^2}dt=\int_0^1 |B'(t)|dt \tag{4}$$

This can also be generalized to R_3 where the arc length can be obtained via the integral:

$$L=\int_0^1 \sqrt{(x'(t))^2+(y'(t))^2+(z'(t))^2}dt=\int_0^1 |B'(t)|dt \tag{5}$$

Besides just being smooth curves, the Bezier curve also can model a natural speeding-up and slowing-down of time. The movement speed of the point as t increases constantly is not consistent and varies as it moves along the curve. This is because the rate of change of $x(t)$ and $y(t)$ is not

DOI: 10.54254/2753-8818/2026.HZ29024

constant. This is used in animation to create natural easing, for a constant speed, a reparameterization by arc length is necessary.

2.3.2. Parameter speed

Velocity:

$$B'(t)=n\sum_{i=0}^{n-1} b_i^{n-1}(t)\Delta P_i,\ \Delta P_i=P_{i+1}-P_i \tag{6}$$

Speed:

$$v(t)=|B'(t)|$$

Acceleration:

$$B''(t)=n(n-1)\sum_{i=0}^{n-2} b_i^{n-2}(t)\Delta^2 P_i,\ \Delta^2 P_i=\Delta P_{i+1}-\Delta P_i \tag{7}$$

The maximum speed occurs when $B'(t)$ is orthogonal to $B''(t)$:

$$v_{max}(t)=\left|B'(t)\right|,\frac{d}{dt}\left(v(t)^2\right)=2B'(t)\bullet B''(t)=0 \tag{8}$$

2.3.3. Curvature

Curvature is a value/function that determines that reciprocal of the radius of the osculating circle at a point on a function, see Figure 3. A larger value of κ means the curve is sharper (see Fig. 3).

$$\kappa(t)=\frac{|T'(t)|}{|r'(t)|} \tag{9}$$

For a Bezier curve $B(t)=(x(t),y(t))$ of degree n,

$$B'(t)=n\sum_{i=0}^{n-1} b_i^{n-1}(t)\Delta P_i,\ B''(t)=n(n-1)\sum_{i=0}^{n-2} b_i^{n-2}(t)\Delta^2 P_i \tag{10}$$

$$\kappa(t)=\frac{|x'(t)y''(t)-y'(t)x''(t)|}{\left(x'(t)^2+y'(t)^2\right)^{\frac{3}{2}}} \tag{11}$$

Curvature is useful for determining centripetal acceleration which can help check if roads are safe for driving at certain speeds or the g-force pilots on fighter jets and passengers on amusement park rides experience.

DOI: 10.54254/2753-8818/2026.HZ29024

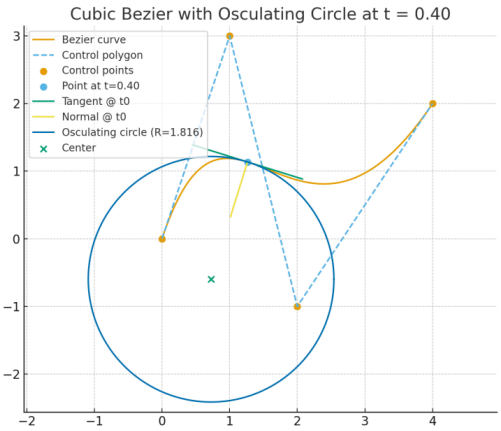


Figure 3. Cubic Bezier with Osculating Circle at t=0.40. The Bézier curve (orange) is defined by four control points (yellow markers) connected by the control polygon (blue dashed lines). The point on the curve corresponding to t=0.40is highlighted in cyan, with its tangent and normal vectors shown in yellow. The osculating circle (blue) centered at the green cross represents the local curvature, with radius R=1.816, corresponding to curvature κ=1/R. This construction illustrates how curvature can be evaluated along a Bézier curve to characterize local geometric properties relevant to applications such as motion planning and aerodynamics

Since Bezier curves can model trajectories and paths, curvature is extremely useful, Using the curvature formula to find the curvature of Bezier curves created in Bernstein polynomial form is extremely simple.

2.3.4. Joining segments of Bezier curves

Derivatives of endpoints:

For a quartic Bezier curve $B(t)$ with control points P_0, P_1, P_2, P_3

$$B(0)=P_0, B(1)=P_3, B'(0)=3(P_1-P_0), B'(1)=3(P_3-P_2) \tag{12}$$

Continuous connections of Bezier segment k and $k+1$ with control points P_i^k, P_i^{k+1} :

C^0 the connection is continuous: $P_3^k=P_0^{k+1}$

C^1 the first derivatives are continuous: $P_2^k, P_3^k, P_0^{k+1}, P_1^{k+1}$ must be collinear with $B'(1^-)=3(P_3^k-P_2^k)=3(P_1^{k+1}-P_0^{k+1})=B'(0^+)$

C^2 the second derivatives are continuous: $B''(1^-)=6(P_3^k-2P_2^k+P_1^k)=6(P_2^{k+1}-2P_1^{k+1}+P_0^{k+1})=B''(0^+)$

2.3.5. Degree elevation

Raising the degree of a Bezier curve from n to $n+1$ is useful for segment unification.

Theorem:

DOI: 10.54254/2753-8818/2026.HZ29024

$$Q_0=P_0,\ Q_{n+1}=P_n,\ Q_i=\frac{i}{n+1}P_{i-1}+\left(1-\frac{i}{n+1}\right)P_i,\ 1\le i\le n.$$

$$\sum_{i=0}^{n} b_i^n(t)P_i=\sum_{i=0}^{n+1} b_i^{n+1}(t)Q_i \tag{13}$$

2.4. Catmull-rom splines

A spline is a function that is constructed from multiple polynomials and are joined at knots. Their connections can have smoothness levels, such as continuous first derivatives. The Catmull-Rom spline is a curve that passes through a sequence of waypoints. It is a special case of the Cubic Hermite Spline [3,4].

Definition:

Given a sequence of nodes, $P_0,P_1,\ldots,P_N\in R^d$

Let the spline be $C(t_i)=P_i,\ (i=0,\ldots,N)$, the spline exhibits C^1- continuity, a continuous first derivative, so at each interior node t_i.

Choose a parameter sequence that controls the timing of each node: $\{t_i\}$

$$t_0=0,t_{i+1}=t_i+|P_{i+1}-|P_i|^{\alpha} \tag{14}$$

When $\alpha=0$, the t values are uniform, when $\alpha=1$, the t values are directly proportional to distance.

Defined the tangents at each node $P_i,\ (1<i<N-1)$:

$$m_i=\frac{1-\tau}{2}\ \frac{P_{i+1}-P_{i-1}}{t_{i+1}-t_{i-1}} \tag{15}$$

$\tau\in[0,1)$ and is the tension parameter (controls stretch and shrink, higher τ creates sharper curves). The default value is 0. At uniform t interval value, where $\alpha=0$ and $t_{i+1}-t_{i-1}$ is constant, the tangents can be simplified to:

$$m_i=\frac{1-\tau}{2}\ (P_{i+1}-P_{i-1}) \tag{16}$$

Define a normalized parameter s on the interval $[t_i, t_{i+1}]$:

$$s=\frac{t-t_i}{t_{i+1}-t_i} \tag{17}$$

To guarantee the curve segments pass through the points P_i, we use Cubic Hermite Spline, used to interpolate between points based on two tangent vectors. Let $C_i(s)$ be the Hermite spline segment:

$$C_i(s)=h_{00}(s)P_i+h_{10}(s)(t_{i+1}-t_i)m_i+h_{01}(s)P_{i+1}+h_{11}(s)(t_{i+1}-t_i)m_{i+1} \tag{18}$$

The basis functions:

$$h_{00}(s)=2s^3-3s^2+1,\ h_{10}(s)=s^3-2s^2+s,\ h_{10}(s)=-2s^3+3s^2,\ h_{11}(s)=s^3-s^2 \tag{19}$$

DOI: 10.54254/2753-8818/2026.HZ29024

The segment ensures the curve goes through each node $C_i(0)=P_i$ and $C_i(1)=P_{i+1}$, and gives desired in and out directions, $C_i'(0)=(t_{i+1}-t_i)m_i$ and $C_i'(1)=(t_{i+1}-t_i)m_{i+1}$.

When tension Is 0: $\tau=0$, and time is uniform: constant $t_{i+1}-t_{i-1}$, the segments can be written directly from four consecutive points, the segment $[P_i, P_{i+1}]$ can be expressed as:

$$C_i(s)=\frac{1}{2}\left[(2P_i)+(-P_{i-1}+P_{i+1})s+(2P_{i-1}-5P_i+4P_{i+1}-P_{i+2})s^2+(-P_{i-1}+3P_i-3P_{i+1}+P_{i+2})s^3\right] \quad (20)$$

, $s\in[0,1]$.

This can be converted to a Bezier Curve, for each segment $[P_i,P_{i+1}]$ with two neighboring points P_{i-1} and P_{i+2}, the control points of the Bezier Curve in uniform t intervals and zero tension are:

$$B_0=P_i,\ B_1=P_i+\frac{1}{6}(P_{i+1}-P_{i-1}),B_2=P_{i+1}+\frac{1}{6}(P_{i+2}-P_i),\ B_3=P_{i+1} \quad (21)$$

When the t intervals are not constant and when tension is not zero the control points become:

$$B_0=P_i,\ B_1=P_i+\frac{1-\tau}{6}\left(\frac{P_{i+1}-P_{i-1}}{t_{i+1}-t_{i-1}}\right),B_2=P_{i+1}+\frac{1-\tau}{6}\left(\frac{P_{i+2}-P_i}{t_{i+2}-t_i}\right),\ B_3=P_{i+1} \quad (22)$$

In sum, the Catmull-Rom Spline is a sequence of cubic Hermite segments chained together that create a smooth curve through a sequence of points that can be easily converted into Bezier Curves [5,6].

3. Application of Bezier curves

Applied mathematics have useful impact in many industries. For example, stochastic modeling, monte carlo methods, statistics are widely used across many industries for modeling real-world problems. Bezier curves are also important assets in terms of application in the computer-aided graphics, 3D graphical design, aerospace, automotive industries. We designed an online tool for several application instances of Bezier curves for the purpose of demonstration its practical use in real world problems.

We designed and coded a stream-lit based tool to show these applications of Bezier curves.

3.1. Computer aided design

3.1.1. Bezier curves generation

Bezier curves are generated using the same definition in 2.1:

Given control points $P_0,\ldots,P_n\in R^2$, the Bezier curve is:

$$B(t)=\sum_{i=0}^{n}\binom{n}{i}(1-t)^{n-i}t^iP_i=\sum_{i=0}^{n}b_i(t)P_i\ t\in[0,1]. \quad (23)$$

For a cubic Bezier curve with control points P_0, P_1, P_2, P_3

DOI: 10.54254/2753-8818/2026.HZ29024

$$B'(0)=3(P_1-P_0), B'(1)=3(P_3-P_2) \tag{24}$$

Important Theories:

- De Casteljau's algorithm:

$$P_i^k(t)=(1-t)P_i^{k-1}(t)+tP_{i+1}^{k+1}(t) \tag{25}$$

- Catmull – Rom spline to cubic Bezier. For nodes P_0, P_1, P_2, P_3 the Catmull-Rom segment is the cubic Bezier:

$$B_0=P_i, B_1=P_i+\frac{1}{6}(P_{i+1}-P_{i-1}), B_2=P_{i+1}+\frac{1}{6}(P_{i+2}-P_i), B_3=P_{i+1} \tag{26}$$

Code tools:

- Bezier_curve(control_points, num_points)

• draws the Bezier curve through sampling at many t values, the number of samples is controllable, with more samples creating a smoother curve.

- de_casteljau_levels(points, t)

• Shows how the curve is constructed at different values of t.

• Finds the exact point of the curve at a certain value of t.

• Animates the construction.

- catmull_rom_to_Bezier(points, closed)

• Uses Catmull-Rom splines to create a curve through predetermined way points.

• Converts the Catmull-Rom segments into Bezier segments.

- polygon_to_cubic_segments(points)

• Turns straight polygon segments to Bezier curves so polygons could be constructed using the same system.

Results:

- Generates and visualizes Bezier curves up to the 5th degree, with the option to choose control points coordinates through input or dragging.

- Animates the construction of Bezier curves using De Casteljau's algorithm.

- Converts segments of line through waypoints into Bezier chains.

3.1.2. Generative art using Bezier curves

Bezier curves are smooth and scalable, entire images and artworks can be exclusively created from segments of Bezier curves. The theories used are the same as the previous subsection.

Flower Generator (see Figure A1):

Goal: Assemble a flower with petals around a center circle. Each petal created by two mirrored Bezier curves.

For each petal shape, in local coordinates, define 3 points for a half petal, and evaluate the curve they form with Bezier curves. Rotate each petal by $\frac{2k\pi}{N}$ and attach to the center of the flower. Fill the petal with either predetermined or user inputted colors.

- Mountains and trees (see Figure A2):

Goal: Generate a scenery with layered mountains and trees on them.

• Mountain Layers:

For each mountain layer:

DOI: 10.54254/2753-8818/2026.HZ29024

1. Sample peak abscissae and heights within prescribed range

2. Fill to a predetermined opacity and color to the bottom of the page through a closed Bezier loop

- Tree formation:

• Sample tree positions from back to front

• Use a closed Bezier to form a small quadrilateral with preset opacity and color

• Use 2-3 closed Beziers with a rounded triangle shape for the foliage

- Sky:

• Place nodes long a horizontal axis with varied vertical distance from it, convert it to a Bezier chain, and fill with a preset opacity and color

• Produces soft layered bands

Abstract Curves (see Figure A3):

Goal: Generate layered, colorful, and smooth curves that resemble an abstract artwork.

For each curve, draw 3 to 6 points uniformly in $[-r,r]^2$, convert the waypoints into a Bezier chain, assign a color and repeat for n curves.

Galaxy Streamers (see Figure A4):

Goal: Generate logarithmic spirals that resemble the shape of the milky way galaxy.

For each arm k , offset by phase $\frac{2\pi k}{A}$, perturb points perpendicular to the local tangent with increasing amplitude. Covert the waypoints into a Bezier chain and assign color to each chain.

Topo City Skyline, see Figure A5:

Goal: Create a smooth silhouette that resembles a city skyline.

Sample building heights along the x -axis and convert the skyline into a single Bezier chain. Shift the Bezier chain down by i -step and fill the area below with colors of varying hue and opacity. Sample random points under the silhouette and scatter yellow dots to create a lighting effect.

3.1.3. Shape morphing

Concept: Given two closed shapes A and B , find a continuous family of shapes that change gradually from one shape to another, creating the shape morphing effect. If the set of shapes is $\{S(t)\}$, $t\in[0,1]$, then $S(0)=A$, $S(1)=B$.

Process:

1. Normalize the shapes:

Center each shape so their “center of mass” is at the same point. Scale each shape to a common size.

2. Prepare for blending:

Place N points around each shape’s outline with each spacing in terms of arc length. Rotate one shape’s points to minimize the average squared distance to the other. Convert the edge between each consecutive point on each shape to a cubic Bezier segment.

3. Blend the control points

For each segment k and its 4 control points, blend where:

$$C_{k,j}(t)=(1-t)C_{k,j}^{A}+tC_{k,j}^{B},\ j=0,1,2,3 \tag{27}$$

and do this for all segments of k . The collection of $\{C_{k,j}(t)\}$ becomes $S(t)$.

4. Draw the in between shapes

DOI: 10.54254/2753-8818/2026.HZ29024

Evaluate each Bezier with a set of samples and connect the point.

3.2. Airfoil

An airfoil is the 2D cross-section of the wing, propeller, or turbine of a plane. The NACA family is a widely used naming convention of wings, where the digits after encode camber and thickness distributions. Bezier curves are useful for airfoil because of their smoothness, easy geometric constraints, and stable editing [7].

Theory:

The upper and lower surfaces of the airfoil are expressed as functions of the normalized curve $x\in[0,1]$. $y_u(x)$ being the upper surface function and $y_l(x)$ being the lower surface. Let the Bernstein polynomial $b_i(x)=\binom{n}{i}(1-x)^{n-i}x^i$, then:

$$y_u(x)=\sum_{i=0}^{n_u} u_i b_i(x),\ y_l(x)=\sum_{i=0}^{n_l} l_i b_i(x) \tag{28}$$

The set of coefficients $\{u_i\}$ and $\{l_i\}$ are the y -coordinates of the control points of the curve.

An improvement to this is to use CST, or Class-Shape Transformation:

The surface has a class term $C(x)$ and a shape term that are the Berstein polynomials. A common class term for a thick leading edge and a thin trailing edge is:

$$C(x)=x^{\frac{1}{2}}(1-x)$$

$$y_u(x)=C(x)\sum_{i=0}^{n_u} A_{ui}b_i(x)+\frac{t_{TE}}{2}x,\ y_l(x)=-C(x)\sum_{i=0}^{n_l} A_{li}b_i(x)+\frac{t_{TE}}{2}x \tag{29}$$

$\frac{t_{TE}}{2}x$ sets the gap for the trailing edge and $\{A_{ui}\}$ and $\{A_{li}\}$ control the shape.

To ensure a smooth generation of the airfoil surface, there should be more samples at places where curvature is high, so we sample x with cosine clustering:

$$x(\theta)=\frac{1}{2}(1-\cos\theta),\ \theta\in[0,\pi] \tag{30}$$

- Fitting known airfoil (See Fig. 4 for demonstration of the NACA 2408 airfoil): Solve a least-squares problem for the sets of coefficients and the trailing edge.
- Generating new airfoils: Set custom n value, coefficients and trailing edge.

DOI: 10.54254/2753-8818/2026.HZ29024

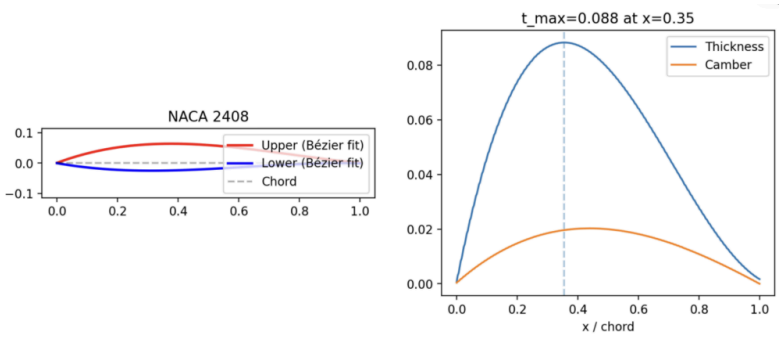


Figure 4. Bezier visualization of the NACA 2408 Airfoil with thickness and camber graph. (Left) Upper (red) and lower (blue) surfaces of the airfoil fitted using Bézier representations, with the chord line shown in gray. (Right) Corresponding thickness (blue) and camber (orange) distributions as functions of the normalized chord position x/c. The maximum thickness is tmax=0.088occurring at x/c=0.35. The figure demonstrates how Bézier parameterization captures aerodynamic surface geometry and enables quantitative analysis of shape descriptors such as camber and thickness

3.3. Trajectory and motion planning

Bezier curves' smoothness make them great for trajectory and motion planning. The control points of the Bezier curve can easily manipulate and shape paths.

Planning the trajectory

To plan a planar trajectory, the boundary conditions for the curve $B(t)$ are:

$$B(0)=p_{start}, B(1)=p_{end}, B'(0)=heading_0, B'(1)=heading_1$$

To ensure the trajectory's steering or jerk is kept at a minimum, $\int_0^1 |B'''(t)|^2 dt$ should be minimized.

To make sure that the trajectory is safe for vehicles to drive, the curvature:

$$\kappa(t)=\frac{|x'(t)y''(t)-y'(t)x''(t)|}{\left(x'(t)^2+y'(t)^2\right)^{\frac{3}{2}}} \tag{31}$$

Of the curve must be kept below the tire static friction limit by having an upper bound for $B''(t)$ and setting a maximum speed.

For a static obstacle O and a margin of r, sample a set of points from the edge of the obstacle and keep the control polygon outside of the Minkowski-expanded obstacle $O^{\oplus r}$ (See Fig. 5 for demonstration)

$$\{P_0, P_1 \ldots, P_n\} \cap O^{\oplus r} \tag{32}$$

DOI: 10.54254/2753-8818/2026.HZ29024

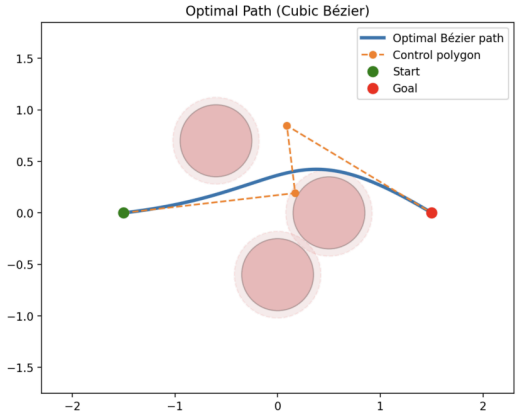


Figure 5. Bezier path planning. The trajectory (red curve) is generated to connect the specified start and end points while satisfying boundary conditions on position and heading. Control points (black markers) define the curve geometry, ensuring smooth curvature within feasible limits for vehicle dynamics. Obstacles (gray shaded regions) are avoided by constraining the control polygon outside the Minkowski-expanded obstacle boundaries. This example highlights how Bézier curves enable smooth, collision-free trajectory generation for motion planning applications in autonomous driving and robotics

4. Conclusion

In this study, we created a stream-lit based tool to showcase the application of Bezier curves. This tool includes interactive Bezier graphs, Bezier generative art, Bezier airfoil design, and Bezier path planning. It is easily accessible and clearly showcases the variety of use cases of these curves. We believe our work is meaningful because it shows the power of Bezier curves to create a unified pipeline for very different tasks. It reduces the fragmentation between subjects such as graphics, design and robotics.

We believe our work is meaningful because it highlights the power of Bézier curves to create a unified pipeline for tasks that span very different domains. By providing a single mathematical framework, Bézier curves reduce fragmentation between fields such as computer graphics, industrial design, and robotics, and illustrate how theoretical tools can translate into practical implementations.

Limitations of the current tool include limited customizability for generative art and the restriction to two-dimensional airfoil CSTs and curve visualizations. Future work could extend the framework to three-dimensional surfaces, incorporate more advanced optimization routines, and broaden the scope of applications in real-time simulation, engineering education, and automated design workflows.

Acknowledgments

During the preparation of this manuscript/study, the author(s) used generative AI for the purposes of polishing code. The authors have reviewed and edited the output and take full responsibility for the content of this publication.

References

[1] Laurent, P.-J., & Sablonnière, P. (2001, September). Pierre Bézier: An engineer and a mathematician - sciencedirect. ScienceDirect. https: //www.sciencedirect.com/science/article/abs/pii/S0167839601000565

DOI: 10.54254/2753-8818/2026.HZ29024

[2] Fuda, C., Ramanantoanina, A., & Hormann, K. (2024, March 9). A comprehensive comparison of algorithms for evaluating rational Bézier curves. Padova University Press. https: //drna.padovauniversitypress.it/system/files/papers/DRNA-2024-3-09.pdf
[3] Dunlop, R. (2025, July 26). Introduction to Catmull-Rom Splines. Introduction to catmull-rom splines. https: //www.mvps.org/directx/articles/catmull/Author 1, A.B.; Author 2, C. Title of Unpublished Work. Abbreviated Journal Name year, phrase indicating stage of publication (submitted; accepted; in press).
[4] Catmull, E., & Rom, R. (n.d.). A CLASS OF LOCAL INTERPOLATING SPLINES . https: //muugumuugu.github.io/bOOkshelF/standalone/Class%20of%20Local%20Interpolating%20Splines.pdf
[5] Finn, D. (2024, December 10). Cubic Curves and Cubic Hermite Interpolation. Rose Hulman Institute of Technology. https: //www.rose-hulman.edu/~finn/CCLI/Notes/day08.pdf
[6] Ron, A. (2010, October 26). Cubic Hermite Spline Interpolation II. Computer Sciences, University of Wisconsin, Madison. https: //pages.cs.wisc.edu/~amos/412/lecture-notes/lecture13.pdf
[7] Agarwal, D., & Sahu, P. (2022). A Unified Approach for Airfoil Parameterization Using Bezier Curves. Computer Aided Design and Applications. https: //www.cad-journal.net/files/vol_19/CAD_19(6)_2022_1130-1142.pdf

Appendix A

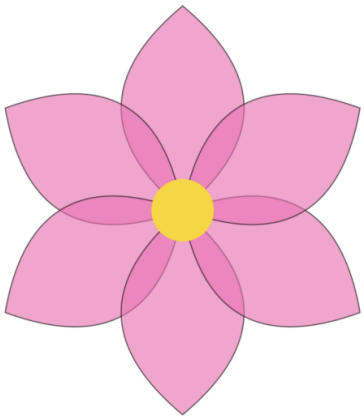


Figure A1. Pink Bezier Flower

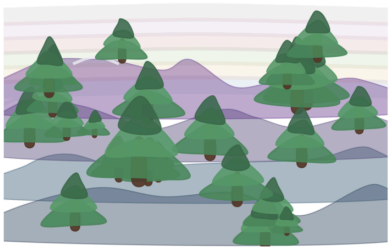


Figure A2. Bezier mountains and trees

DOI: 10.54254/2753-8818/2026.HZ29024



Figure A3. Bezier abstract curve art



Figure A4. Bezier galaxy streamer

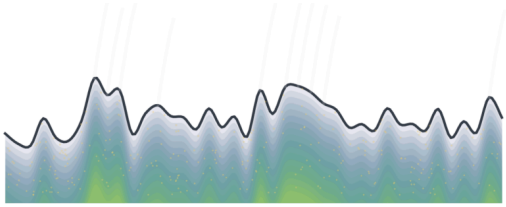


Figure A5. Bezier city skyline