www.cad-journal.net
Computer-Aided Design AND APPLICATIONS

ITA Spline: A Circular-Arc-Based G¹-Continuous Curve for 3D Point Sequences

Kazuya Mori[1]

[1]Former 3D CAD Software Engineer, mt2967+ADS@gmail.com

Abstract. This document introduces the ITA Spline (Isosceles Triangle Arc Spline), a method for constructing a G¹-continuous curve through a sequence of points in three-dimensional space using only circular arcs. The method is derived from elementary geometry and yields analytical expressions for arc length and curvature. Its unique geometric formulation eliminates the degree of freedom ambiguity inherent in biarc interpolation, resulting in a uniquely determined circular arc spline. Applications include CAD modeling, walkthrough path generation, and smooth attitude control on the unit sphere.

Keywords: circular arc, G¹-continuous curve, 3D interpolation
DOI: https://doi.org/10.14733/cadaps.2027.101-118

1 INTRODUCTION

In the field of computer graphics, several classical methods exist for interpolating a sequence of points that define an arbitrary curve, such as Lagrange interpolation and B-spline interpolation. These techniques have long been used in CAD systems and related applications.
This paper introduces a method, referred to as ITA Spline, which smoothly connects N points in three-dimensional space using 2(N−1) circular arcs with different radii of curvature, achieving G¹-continuity. Despite its simple construction, the resulting curve exhibits no visual unnaturalness, allows easy computation of curve length, and has the notable property of reproducing a perfect circle without error when the input points correspond to the vertices of a regular polygon.
The method is simple in its purpose, computational procedure, and resulting behavior, making it easy to understand. Its relationship to existing biarc-based approaches is also discussed.

2 BACKGROUND AND MOTIVATION

In the 1990s, the author was involved in the development of a three-dimensional CAD system for plant design at a Japanese chemical manufacturer. At that time, real-time rendering on engineering workstations (EWS) was difficult, so walkthrough videos were generated by running overnight batch processing after defining a walking path in advance.

To determine this path, smooth movement through space and accurate measurement of walking time were required. For this purpose, the author devised a curve interpolation method using only

simple geometric primitives—circular arcs. This interpolation technique was also applicable to representing the three-dimensional shapes of hoses and tubes placed within the plant. These shapes could be constructed by combining partial tori, which serve as geometric primitives in solid modeling.

A literature survey conducted in recent years, supplemented by AI-assisted searching as of 2026, confirmed that this method has not been previously published. Therefore, the author names the technique Isosceles Triangle Arc Spline (ITA Spline) and summarizes it in this paper.

A concrete example—generating a hose from a sequence of points—is shown in Figure 1.

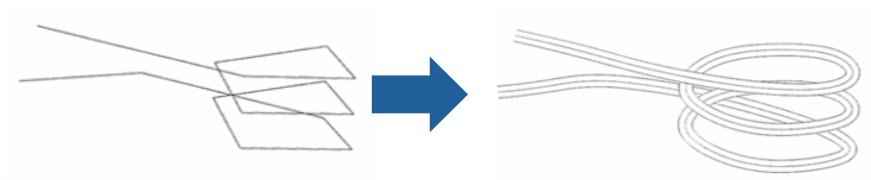


Figure 1: ITA spline example.

3 BASIC DEFINITION

3.1 Isosceles Triangle and Base Tangent Arc

We begin by explaining, using elementary geometry, the relationship between an isosceles triangle and the circular arc associated with it, which forms the basis of the ITA Spline method. (See Figure 2 and Figure 3)

Consider an isosceles triangle △ABC defined as follows:

$$\mathrm{AB} = \mathrm{AC} = \mathrm{L}$$

$$\angle \mathrm{BAC} = \alpha$$

A circle can be constructed that passes through points B and C and is tangent to sides AB and AC. The circular arc of this circle lying inside △ABC is defined as the **base tangent arc** of the isosceles triangle. For convenience in this paper, vertex A is called the **apex**, vertices B and C are called **child vertices**, and the edges connecting the apex to the child vertices are called **side edges**.

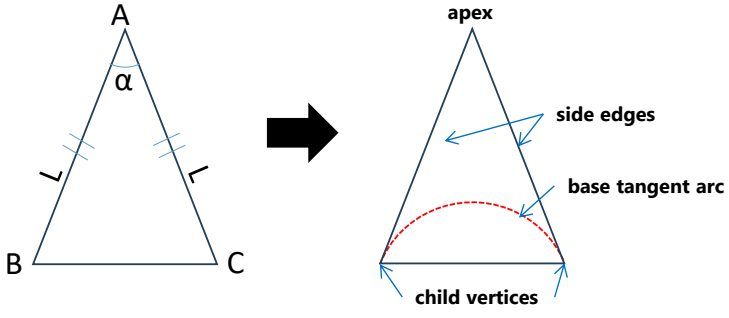


Figure 2: Base tangent arc in isosceles triangle.

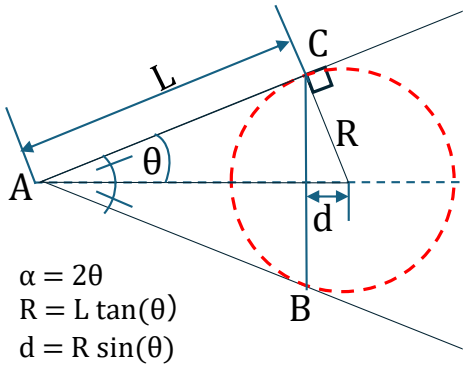


Figure 3: Explanation using elementary geometry.

3.2 Properties of Isosceles Triangles and Base Tangent Arcs

Consider two isosceles triangles arranged such that:

- They touch at one child vertex, and
- One side edge from each triangle lies on the same straight line (a **common side line**). (See Figure 4)

In this configuration, the two base tangent arcs connect smoothly at the shared child vertex, achieving **G¹-continuous connection**.

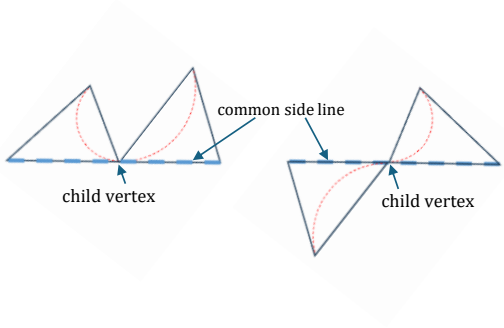


Figure 4: Chain of Isosceles Triangles.

Furthermore, consider rotating one of the isosceles triangles around the common side line by an arbitrary angle. (See Figure 5) The rotated triangle no longer lies in the same plane and appears to "pop out" of the page. Even in this case, the two base tangent arcs share a common tangent direction at the child vertex and therefore connect smoothly. The two side edges not lying on the common side line become **skew lines** in three-dimensional space.

3.3 Definition of the ITA Spline

Given a sequence of points P_i (i=1,2,…,n) in three-dimensional space R^3, we define the **ITA Spline** as a curve interpolation method satisfying the following three conditions:
A) The curve is constructed using circular arcs as its basic elements.
B) The curve passes through every point P_i .

C) Each segment between P_i and P_{i+1} consists of **two circular arcs**, giving a total of 2(n-1) arcs, all connected with **G^1 continuity**.

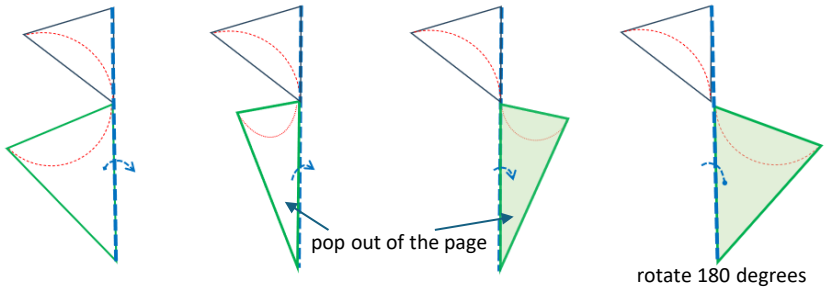


Figure 5: Rotation around the common side line.

3.4 Construction Procedure

The construction procedure is described below. (See Figure 6)

(1) Constraints on the Point Sequence in R^3

Let the angle between segments P_i P_{i+1} and P_{i+1} P_{i+2} be α, where

$$0<\alpha<\pi$$

(i.e., the points do not lie on a straight line).

For closed curves, the starting point P_1 and the ending point P_{n+1} (virtual) are set equal. Although the figure appears planar, the sequence of points is not limited to a plane.

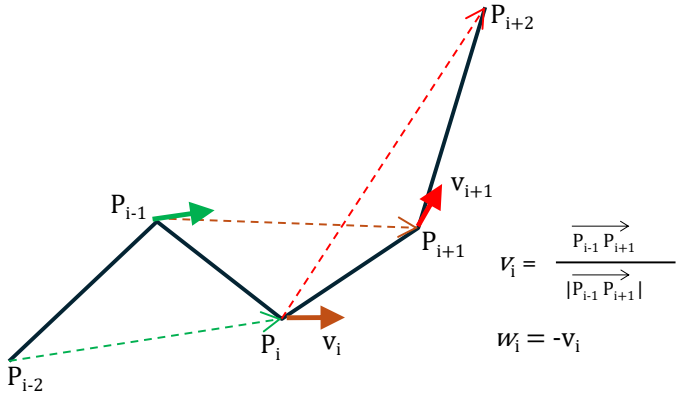


Figure 6: Points and directions in R^3.

(2) Direction Vectors

For each vertex P_i, define a direction vector $\mathbf{v_i}$ as the normalized vector of the difference between its neighboring points. Also define $\mathbf{w_i} = -\mathbf{v_i}$. For endpoints P_1 and P_n, use P_1 in place of P_0, and P_n in place of P_{n+1}. For closed curves, use P_n for P_0 and P_1 for P_{n+1}.

Focus on the pairs (P_i, $\mathbf{v_i}$) and (P_{i+1}, $\mathbf{v_{i+1}}$). (See Figure 7)

Note that other definitions of direction vectors are possible, and as described later, they affect the shape of the interpolated curve.

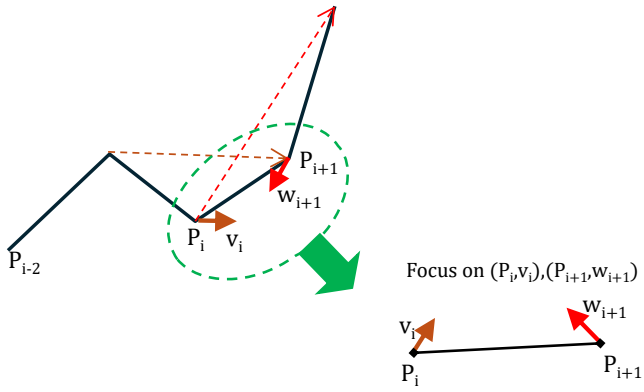


Figure 7: Focus on a single line segment.

(3) Auxiliary Lines

Draw a line through P_i in the direction $\mathbf{v}_i$, and another through P_{i+1} in the direction $\mathbf{w}_{i+1}$. If $\mathbf{v}_i$ and $\mathbf{w}_{i+1}$ are not coplanar, these two lines become **skew lines**. The procedure works regardless of whether the vectors lie in the same plane. (See Figure 8)

Let the position vectors of P_i and P_{i+1} be $\mathbf{p}_i$ and $\mathbf{p}_{i+1}$. Consider moving points S and T on the two auxiliary lines. For real numbers s and t, the position vectors of S and T are:

$$\mathbf{x} = \mathbf{p}_i + s\,\mathbf{v}_i$$
$$\mathbf{y} = \mathbf{p}_{i+1} + t\,\mathbf{w}_{i+1}$$

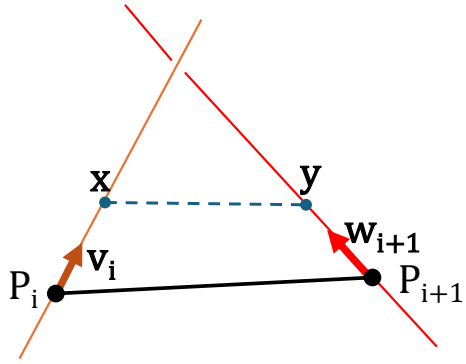


Figure 8: Auxiliary (skew) lines.

(4) Moving Points and Construction of Two Isosceles Triangles

Let Q be the midpoint of segment ST. Points S and T start from P_i and P_{i+1}, respectively, and move at unit speed along directions $\mathbf{v}_i$ and $\mathbf{w}_{i+1}$.

We show that two isosceles triangles $\triangle P_i$ SQ and $\triangle P_{i+1}$TQ can be constructed such that:

$$P_i S = QS,\ P_{i+1} T = QT \qquad (3.1)$$

(See Figure 9)

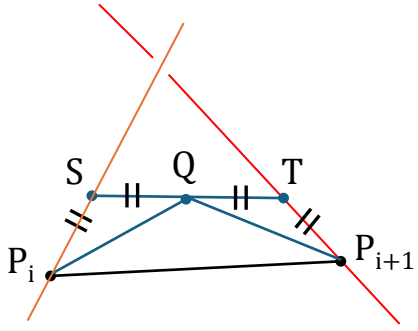


Figure 9: Construction of two isosceles triangles.

Since

$$P_i S = s,\ P_{i+1} T = t,\ \ QS + QT = |\mathbf{x}-\mathbf{y}| \qquad (3.2)$$

we obtain:

$$|\mathbf{x}-\mathbf{y}| = s+t$$

and from symmetry,

$$s = t$$

Substituting into the above yields the following equation (where <,> denotes the inner product):

$$2(1+\langle \mathbf{v}_i, \mathbf{w}_{i+1}\rangle)s^2 - 2\langle \mathbf{p}_i - \mathbf{p}_{i+1}, \mathbf{v}_i - \mathbf{w}_{i+1}\rangle s - |\mathbf{p}_i - \mathbf{p}_{i+1}|^2 = 0 \qquad (3.3)$$

If $\langle \mathbf{v}_i, \mathbf{w}_{i+1}\rangle \neq -1$,
the equation becomes a quadratic equation in s, which has a positive solution.

$$s = \frac{\langle p_i - p_{i+1}, v_i - w_{i+1}\rangle + \sqrt{\langle p_i - p_{i+1}, v_i - w_{i+1}\rangle^2 + 2|p_i - p_{i+1}|^2(1+\langle v_i, w_{i+1}\rangle)}}{2(1+\langle v_i, w_{i+1}\rangle)} \qquad (3.4)$$

If $\langle \mathbf{v}_i, \mathbf{w}_{i+1}\rangle = -1$,
the equation becomes linear in s, corresponding geometrically to the case where $\mathbf{v}_i$ and $\mathbf{w}_{i+1}$ are parallel in the opposite direction (See Figure 10).

$$s = \frac{|p_i - p_{i+1}|^2}{4\langle \mathbf{p}_i - \mathbf{p}_{i+1}, \mathbf{w}_{i+1}\rangle} \qquad (3.5)$$

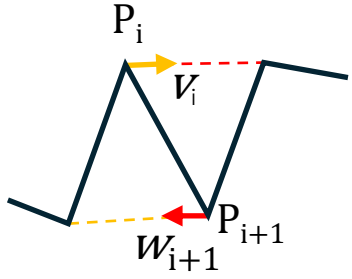


Figure 10: Opposite direction.

Thus, two isosceles triangles are determined, and their base tangent arcs connect smoothly.(Figure 11)

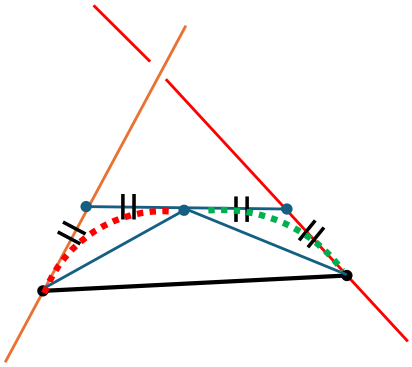


Figure 11: Arcs connect smoothly.

Repeating the same procedure for the next point P_{i+2} yields another smoothly connected arc. The two isosceles triangles that meet at P_{i+1} share a common side, so their base tangent arcs join smoothly. (Figure 12)
By repeating this process, all points in the sequence are connected smoothly by circular arcs.

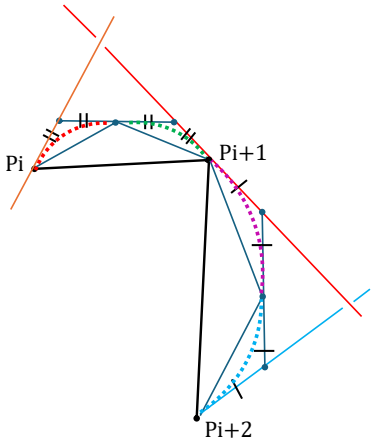


Figure 12: Chain of arcs.

4 DEGREES OF FREEDOM IN VERTEX DIRECTION VECTORS

We begin with the simplest case: a set of three points forming a triangle (a right triangle with side lengths 3, 4, and 5). By identifying the starting and ending points, we treat the configuration as a closed curve. For each vertex, the direction vector is chosen to be parallel to the opposite side (See Definition 3.4(2) in Section 3). The resulting ITA Spline is shown in Figure 13. The right-hand figure illustrates the arrangement of the circular arcs using color coding.

On the other hand, elementary geometry tells us that a triangle has a unique circumcircle. If the direction vectors at each vertex are assigned so that they are perpendicular to the line segments connecting the circumcenter to each vertex (Left Figure 14), the resulting ITA Spline becomes exactly the circumcircle.

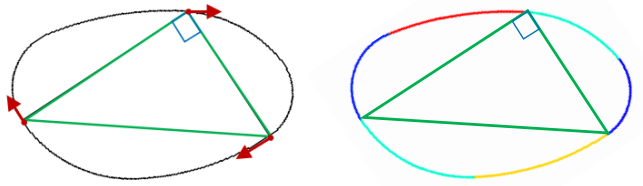


Figure 13: Opposite-side-parallel type.

The circumcenter is the intersection point of the perpendicular bisectors of the three sides (Right Figure 14).

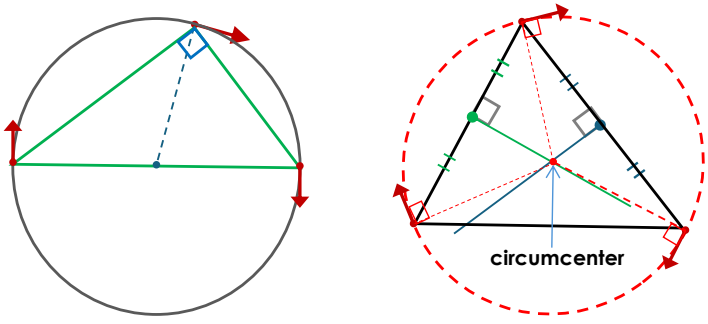


Figure 14: Circumcenter-based type.

Thus, the shape of the interpolated curve varies depending on how the direction vectors at the vertices are chosen—there are infinitely many possibilities. In this report, we focus on the two representative choices described above, which we refer to as the opposite-side-parallel type and the circumcenter-based type.

5 EXAMPLE FIGURES

This section presents several examples of ITA Spline curves computed from given point sequences (shown in red). Three examples are planar curves, and two are spatial curves. For 3D visualization, the software *Linear Graph 3D* was used.

(1) Three Points (Open Curve)

For the opposite-side-parallel type, the direction vector $\mathbf{v_2} = -\mathbf{w_2}$ is computed by normalizing the difference vectors of the neighboring points. The direction vector at P_1 is:

$$v_1 = \frac{\overrightarrow{P_1P_2}}{|\overrightarrow{P_1P_2}|}$$

The vector $\mathbf{v_3} = -\mathbf{w_3}$ is computed in the same manner. For the circumcenter-based type, refer to the source code in the Appendix. Since the endpoint direction vectors are shared, the difference between the two types is relatively small in this case.(Figure 15).

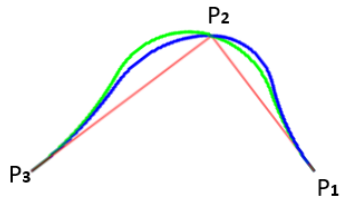


Figure 15: Open curve(blue: opposite-side-parallel, green: Circumcenter-based).

(2) Three Points (Closed Curve)

Here we set $P_0 = P_3$. Similarly, $\mathbf{v_3} = -\mathbf{w_3}$ is computed by treating $P_4 = P_1$. Both curves pass through the same three points, but their shapes differ significantly.(Figure 16).

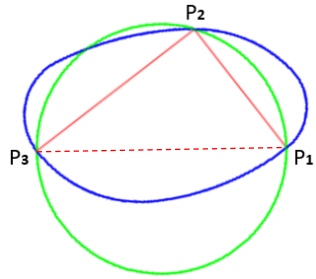


Figure 16: Closed curve (blue: opposite-side-parallel, green: Circumcenter-based).

(3) Cyclic Pentagon

Points P_1, P_2, P_3, P_4, P_5 lie on the same circle, and the curve is treated as closed. (See Figure 17) The circumcenter-based ITA Spline becomes the unique circle (green curve). In contrast, the opposite-side-parallel ITA Spline exhibits noticeable distortion (blue curve).

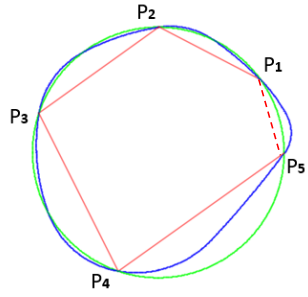


Figure 17: Cyclic Pentagon (blue: opposite-side-parallel, green: Circumcenter-based).

(4) Helix Curve

A helix curve is defined by the following parametric equations:

$$\begin{cases} x(t)=\cos(t) \\ y(t)=\sin(t) \\ z(t)=ht \end{cases}$$
$$h \geqq 0,\ 0 \leqq t \leqq 6\pi \tag{5.1}$$

We sample five equally spaced points over one period and identify the start and end points. Set h=0.25(Figure 18). The difference in direction-vector types appears only in the segment between these points, while no difference is seen along the uniformly sampled circular part.

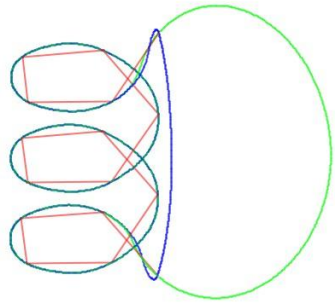


Figure 18: Helix (Closed curve) (red: sampling, blue: opposite-side-parallel, green: Circumcenter-based).

Figure 19 shows the original curve overlaid with the ITA Spline(five equally spaced points over one period). Except near the start and end points, the ITA Spline provides good interpolation.

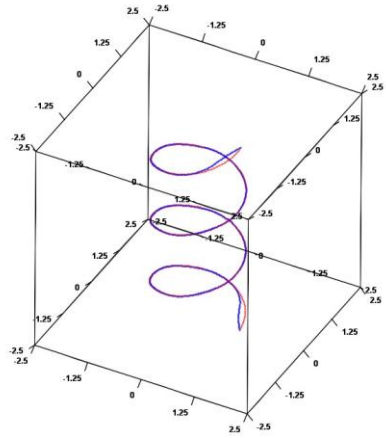


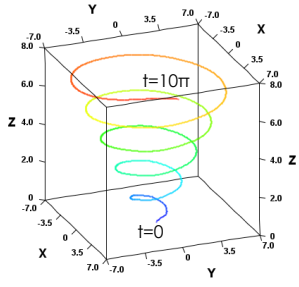
Figure 19: Helix (Open curve) (red: formula, blue: opposite-side-parallel).

(5) Tornado-Shaped Spatial Curve

A three-dimensional tornado-shaped curve (See Figure 20) is defined by the following parametric equations:

$$\begin{cases} x(t)=(a+bt)\cos(t) \\ y(t)=(a+bt)\sin(t) \\ z(t)=ht \end{cases}$$

$$a>0, b\geqq 0, h\geqq 0,\ 0\leqq t\leqq 10\pi \tag{5.2}$$



$a=1, b=0.20, h=0.25, 0\leqq t\leqq 10\pi$

Figure 20: Tornado-shaped curve.

The point sequence is sampled at five equal intervals per period (red curve in Figure 21). Applying the ITA Spline (opposite-side-parallel type) yields the interpolated curve (blue curve in Figure 21). As in the helix case, even with the circumcenter-based type, no difference is observed along the revolving part.

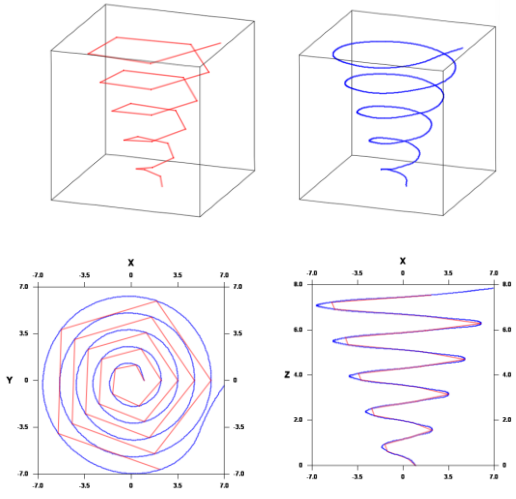


Figure 21: Comparison (sampling, ITA).

The difference between the original curve and the ITA Spline is shown in Figure 22: the original curve is drawn in red, and the interpolation in blue. The deviation is small except near the endpoints.

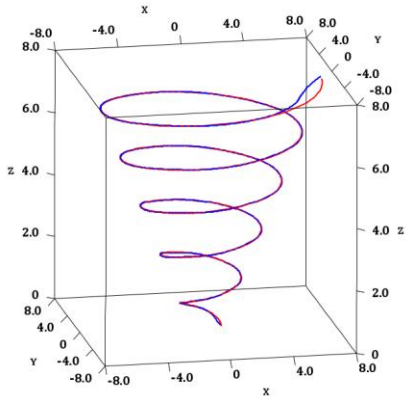


Figure 22: Comparison (red: formula, blue: ITA).

The arc length of the original curve can be computed analytically.

$$\text{arc length} = \int \sqrt{(bt + a)^2 + b^2 + h^2}\, \mathrm{d}t$$

Specifically, the arc length over the interval

$$t \in \left[\frac{4\pi}{5}, \frac{44\pi}{5}\right]$$

is

$$101.30638\ldots$$

which can be expressed using elementary functions (excluding endpoint regions). The ITA Spline arc length, obtained by summing the lengths of 40 arcs, is

$$101.285079$$

giving a relative error of approximately **0.02%**.

(6) Torus Knot

As an example of a (2,3)-type torus knot, the entire curve is sampled at 7 equally spaced points. (opposite-side-parallel type) See Figure 23.

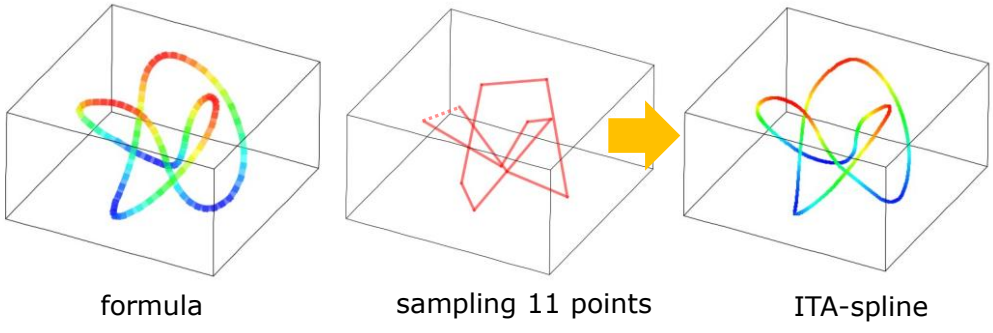


Figure 23: Torus knot (2,3)-type.

Table 1 shows the ratio of the ITA Spline length to the theoretical curve length when the number of divisions is varied. As the number of interpolation points increases, the ITA Spline length approaches the theoretical value. See Figure 24.

number of divisions	7	11	21	100
Opposite-side-parallel	90.7%	96.9%	99.7%	100.1%
Circumcenter-based	91.8%	98.3%	99.9%	99.9%

Table 1: Division count and length ratio.

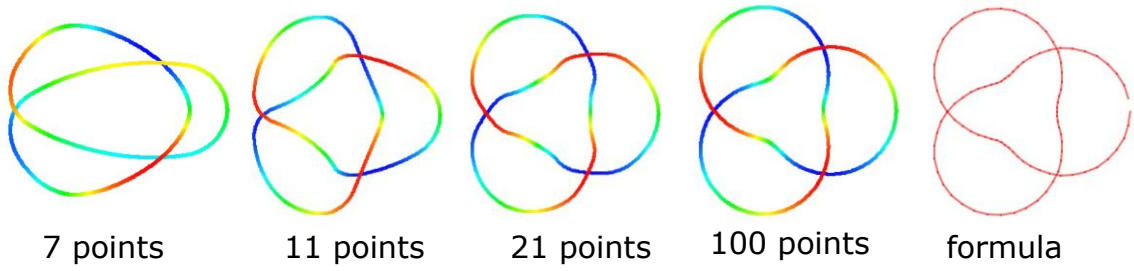


Figure 24: Planar diagram of the torus knot with variable division counts.

6 CHARACTERISTICS

Based on the preceding explanations, the main characteristics of the ITA Spline are summarized below.

1. **Curve construction using circular arcs**
   For a sequence of N points in three-dimensional space, the method constructs an interpolated curve consisting of **two circular arcs between each pair of points**, for a total of 2(N-1) arcs. The resulting curve passes through all N points and achieves **G^1-continuous connection**.
2. **Analytical derivation using elementary geometry**
   Since the derivation relies only on elementary geometry, vector calculations, and quadratic equations, analytical (exact) expressions can be obtained for geometric properties such as position, arc length, and curvature.
3. **Direction vectors at each vertex**
   A direction vector must be assigned to each point in the sequence. Representative definitions include: (a) **Opposite-side-parallel type**, and (b) **Circumcenter-based type**. When the circumcenter-based type is used, a set of points lying on a circle can reproduce the circle **exactly**, without approximation error.
4. **Visual smoothness and curvature discontinuity**
   Although each circular arc is connected with G^1 continuity, the radius and direction of curvature are discontinuous at the connection points. Therefore, caution is required when applying the method to motion-planning or conveyor-path applications.
5. **Practical applications since the 1990s**
   The method was originally devised in the 1990s and has been used for defining walkthrough paths and for representing free-form hoses (a type of piping) using partial-torus primitives in 3D CAD systems. Complex spatial curves can be modeled by combining only circular arcs.
6. **Simple software implementation**

Implementation requires only three-dimensional vector operations and does not depend on any special external libraries.

7. **Computational efficiency**
   Each segment can be computed in constant time, and the overall computational complexity is **O(N)**.
8. **Numerical stability**
   In the computation of $s$for the isosceles triangles (Definition 3.4.(4)), when the inner product approaches -1, the equation reduces to a linear form, ensuring numerical stability.

7 IMPLEMENTATION METHOD

This section describes the basic data structures and function set required to implement the ITA Spline in C++. The complete source code is provided in the Appendix.

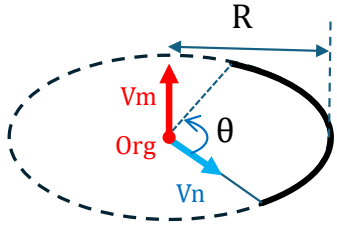
(1) Data Structure for Circular Arcs

A structure strct_arc is defined to represent a circular arc. (Figures 24 and 25) In this structure:

- **Vm** denotes the main axis, specifying the direction vector of the rotation axis of the circle (arc).
- **Vn** specifies the direction vector pointing toward one endpoint of the arc.
-

```
// circular arc structure
struct strct_arc {
      Vct3 Org;          //arc origin
      Vct3 Vm;           //main vector
      Vct3 Vn;           //auxiliary vector
      double Ang_rad; //center angle(rad)
      double Radius;     //radius
};
```

Figure 24: Arc structure（C++).



θ= Ang_rad
R= Radius

Figure 25: Arc representation with vectors.

(2) Definition of Point Sequences in 3D Space

The following components are used:

- **Vct3**: a class representing a 3-dimensional vector. The inner product operator is defined as "|", and the cross product operator as "^".
- **std::vector<Vct3>**: an array of 3D vectors representing the point sequence.

(3) Function to Generate Circular Arcs from a Point Sequence

cpp

int ITA_Points_to_Arcs(std::vector<Vct3> vecPt, int Type, std::vector<strct_arc> &vecArc)
This function converts a sequence of points in 3D space into a sequence of ITA Spline circular arcs. For each segment, two isosceles triangles are constructed, and two base tangent arcs are generated.

(4) Function to Compute Total Arc Length

cpp
double ArcLength(std::vector<strct_arc> vecArc)
Returns the total length of the circular-arc spline.

(5) Function to Obtain Coordinates at a Given Distance from the Start

cpp
int Point_in_Arcs(std::vector<strct_arc> vecArc, double curve_length, Vct3& Pt, double &r1)
Given a distance measured from the starting point of the arc sequence, this function outputs the corresponding coordinate and the radius of curvature at that location.
This is the fundamental function for treating the ITA Spline as a **parametric curve**, and can be used for animation or generating walking paths.

(6) Program to Generate ITA Spline Data from Helix or Tornado-Shaped Curves

cpp
int main()
Using the functions described above, the program performs the following steps:

1. Generate a sequence of spatial points
2. Convert the point sequence into ITA Spline circular arcs
3. Generate a sequence of points along the interpolated curve

8 APPLICATION TO ATTITUDE CONTROL

This section demonstrates the potential application of the ITA Spline to **attitude control**. Consider a video camera mounted on a spacecraft, whose viewing direction can be freely adjusted. As part of the imaging scenario, a sequence of viewing directions is predetermined, represented by N unit direction vectors. The camera begins by pointing along the first direction vector, and then continuously rotates toward the second, third, and subsequent directions. (For simplicity, rotation around the optical axis is ignored.)

If each direction vector is regarded as a point on the unit sphere, the change in viewing direction during motion can be visualized as a **curve on the sphere**.
There exist infinitely many possible motion paths on the sphere. A well-known method is to connect two points on the sphere using a **great-circle arc**. Although simple, this approach does **not** provide G^1-continuity at the junctions between segments. Because the angular velocity is discontinuous, the resulting video may exhibit noticeable "jerkiness." Spline-based smoothing techniques are often used to mitigate this issue. The ITA Spline can be applied precisely to this part of the problem.

(1) Preparation

As shown in Figure 26, consider three points P_1, P_2, P_3 on the sphere arranged so that the triangle has a right angle at P_2=90° .

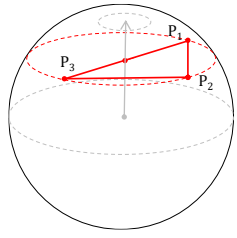


Figure 26: Right triangle on a sphere.

(2) Existence of an ITA Spline Passing Through the Three Vertices
As demonstrated in the examples section, an ITA Spline **closed curve** exists that passes through these three vertices.(Figure 27)

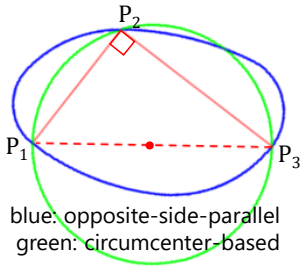


Figure 27: Closed curves on 3 points.

(3) Comparison with SLERP
A basic method for interpolating between two directions is **spherical linear interpolation (SLERP)**, shown as the green curve in Figure 28. Each interpolated segment lies on a great circle. While SLERP is simple, the connections at the vertices are **not G¹-continuous**.

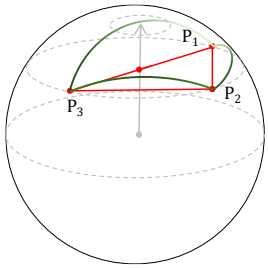


Figure 28: (green) SLERP curves.

(4) Applying the ITA Spline (Opposite-Side-Parallel Type)
To achieve G¹-continuity, the ITA Spline (opposite-side-parallel type) can be applied. The resulting ITA closed curve (green curve in Figure 29) passes through the three vertices and consists of portions that extend outside the sphere and portions that lie inside it.

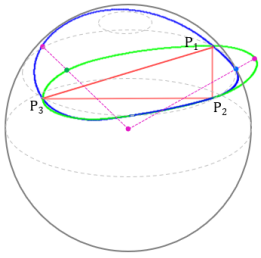


Figure 29: (green)ITA-spline, (blue)projection onto the sphere.

(5) Projection onto the Sphere

Each point on the ITA Spline is connected to the origin of the sphere, and the resulting vector is normalized (i.e., projected onto the sphere). The trajectory of the endpoints of these normalized vectors (blue curve in Figure 29) lies entirely on the sphere. Since this blue curve is the **normalized image of a G^1-continuous curve**, it is also **G^1-continuous**. This provides a new control curve suitable for smooth attitude transitions.

9 HISTORICAL NOTE

The ITA Spline method was originally conceived by the author in 1996 during the development of a 3D CAD system for plant design. The first implementation was written in Fortran. No external literature was referenced at that time.

During the preparation of this document, AI-based search tools (such as Copilot) were used to confirm the originality of the ITA Spline method and to check for related prior work.

For the example figures, the visualization software "RINEARN Graph 3D" was used to display and export three-dimensional plots. (https://www.rinearn.com/ja-jp/graph3d/)

10 CONCLUSIONS

This paper has described the construction and implementation of an interpolating curve based on the ITA Spline method for a sequence of points in three-dimensional space.

The ITA Spline was developed to meet the practical needs of plant-design environments—specifically, achieving G^1-smoothness and manufacturability using partial-torus primitives.

When the method was originally conceived thirty years ago, neither modern search systems nor AI tools were available, and it was not possible to investigate the originality of the approach. However, the rapid development of AI in recent years has clarified the following points:

1. Approximating—or, more precisely, interpolating—a short curve segment using two circular arcs is a well-known topic in the CAD/CAM field, formulated as the biarc problem. The input to the biarc problem consists of two points and their tangents (direction vectors). Although this is the same type of input, the ITA Spline does not require prescribed tangents; they are generated automatically.
2. In general, infinitely many circular arcs satisfy these conditions because one degree of freedom remains; therefore, the solution is not unique.
3. Biarc methods employ various analytical techniques to determine this remaining degree of freedom and obtain a specific solution. Since these methods are used in applications such as machine-tool path control and vehicle-body design, accuracy analysis is required. For example, the spatial biarc method [5] requires two points and two prescribed direction vectors, and by adjusting the lengths of the direction vectors, it generates a suitable spatial biarc for controlling a concentric cable-driven manipulator in 3-D space.
4. In contrast, the ITA Spline is a fully deterministic G^1-interpolating curve that requires only a sequence of points. The underlying principle is the connection of base tangent arcs derived from chains of isosceles triangles. By formulating the problem geometrically and imposing a natural condition (Definition 3.4.(4): choosing Q as the midpoint of S and T, and solving with s=t), the degree of freedom disappears, yielding a **unique** solution. Moreover, this approach works uniformly in both two and three dimensions.

The ITA Spline, while using circular arcs as its fundamental elements, operates naturally in 3D space and resolves the degree-of-freedom issue under natural geometric conditions. As a uniquely determined circular-arc spline, it occupies a position distinct from existing biarc methods.
It is the author's hope that this method will contribute, even in a small way, to the future development of the CAD/CAM field.

11 ACKNOWLEDGEMENTS

The author is grateful to his former colleagues from the 1990s CAD development team in Shinagawa, whose collaboration during the migration from EWS to Windows systems and in CAD operational support contributed to the early development of this work.

REFERENCES

[1] Bertolazzi, E.; Frego, M.: A Note on Robust Biarc Computation, Computer-Aided Design and Applications, 16(5), 2019. https://doi.org/10.14733/cadaps.2019.822-835.
[2] Farin, G. E.: Curves and Surfaces for Computer-Aided Geometric Design: A Practical Guide. Academic Press, 1988.
[3] Shoemake, K.: Animating Rotation with Quaternion Curves. ACM SIGGRAPH, 1985.
[4] Meek, D.; Walton, D.: Approximating Smooth Planar Curves by Arc Splines, Journal of Computational and Applied Mathematics, 59(2), 221–231, 1995. https://doi.org/10.1016/0377-0427(94)00029-Z.
[5] Li, Z.; et al.: A Spatial Biarc Method for Inverse Kinematics and Configuration Planning of Concentric Cable-Driven Manipulators, IEEE Transactions on Systems, Man, and Cybernetics: Systems, 2021. https://doi.org/10.1109/TSMC.2021.3092012.