Linear Matrix Formulation for Directional Tangency Constraints in Parametric Curve Design

Kumara U.G.N. [1]
[1] Department of Mechanical Engineering, University of Moratuwa, Sri Lanka.

ABSTRACT

In Computer-Aided Design (CAD), parametric curves enable the creation of arbitrary yet controllable shapes. During sketching, designers often apply geometric constraints such as fixed points or tangency, leading to non-linear systems that are costly to solve in real time. This work presents a linear matrix formulation for enforcing directional tangency constraints without optimization algorithms. The method was implemented in Python and C++, and integrated into a commercial turbomachinery design tool to maintain blade-edge tangency constraints interactively.

I. INTRODUCTION

Any parametric curve, such as a B-Spline or Bézier, can be expressed as

$$\mathbf{C}(u) = \sum_{i=0}^{n} R_{i,p}(u)\mathbf{P}_i$$

Where u is the parameter, $R_{i,p}$ is the coefficient for i^{th} point for degree p curve. P_i with $i = 0, 1, ..., n$ are the control points with $P_i = (x_i, y_i)$ for two-dimensional case. Matrix notation for this equation can be expressed as,

$$\mathbf{C}(u) = \mathbf{R}(u)\mathbf{P}$$

Where $\mathbf{R}(u) = [R_{0,p}(u), R_{1,p}(u), ..., R_{n,p}(u)]$ and $\mathbf{P} = [\mathbf{P}_0^T, \mathbf{P}_1^T, ..., \mathbf{P}_n^T]^T$.

II. LITERATURE REVIEW

Suppose that we are given a set of parameter values $\{u_r\}, r = 1, ..., R$ and required displacements (whether point or tangent) at those parameter values $\left\{\Delta D_r^{(k)}\right\}$ Assume that when we provide the displacements at the control points $\Delta\mathbf{P}_i$, we obtain the desired displacements at targeted points. Then,

$$\Delta\mathbf{P} = B^T(BB^T)^{-1}\Delta D \tag{1}$$

gives the minimum norm solution that minimizes the combined movement of the control points adhering to the given set of constraints. Where $B \in \mathbb{R}^{R\times N}$ is the coefficient matrix that connects the control point displacements to the displacements at the targeted points. Each row of B corresponds to a constraint at a parameter value u_r and each column corresponds to a control point displacement component (either x or y). The entries of B are derived from the basis functions evaluated at the parameter values. A full derivation of this formulation can be found in [1] which based their work on [2].

III. METHODOLOGY

A. Start and End tangency constraint

For a Bezier curve, the start tangent is always in the direction of the first 2 control points while the end tangent is in the direction of last two control points. This property also holds for B-Splines with uniform knots. We can exploit this property to manipulate start and end tangencies of these parametric curves.

Given that the start tangent should be $\mathbf{T}_s$, from the property above we can derive an expression connecting locations of first point $\mathbf{P}_1$ and second point $\mathbf{P}_2$.

$$\mathbf{P}_2 = \mathbf{P}_1 + t_s\mathbf{T}_s$$

Where t_s is the parameter that the start tangent vector get scaled by to reach the second point. Since we are dealing with the change in control point location,

$$\Delta\mathbf{P}_2 = \Delta\mathbf{P}_1 + \Delta t_s\mathbf{T}_s$$

Then we can define a new displacement vector $\Delta\tilde{\mathbf{P}} \in \mathbb{R}^{2(N-1)}$ that excludes $\Delta\mathbf{P}_2$ as an independent variable and enforce the constraint using,

$$\Delta\mathbf{P}' = A\Delta\tilde{\mathbf{P}} + \mathbf{T}_s\Delta t_s e_2$$

Where $A \in \mathrm{R}^{2N\times 2(N-1)}$ is the matrix that duplicated $\Delta\mathbf{P}_1$ into the second row while e_2 is the unit vector that takes $\Delta t_s\mathbf{T}_s$ into the second row to get added to $\Delta\mathbf{P}_1$ to finally create $\Delta\mathbf{P}_2$. An expanded form is given below where I_2 is the 2×2 identity matrix.

$$\begin{bmatrix}\Delta\mathbf{P}_1\\ \Delta\mathbf{P}_2\\ \vdots\\ \Delta\mathbf{P}_N\end{bmatrix} = \begin{bmatrix}I_2 & 0 & \dots & 0\\ I_2 & 0 & \dots & 0\\ \vdots & \vdots & \ddots & \vdots\\ 0 & 0 & \dots & I_2\end{bmatrix}\begin{bmatrix}\Delta\mathbf{P}_1\\ \vdots\\ \Delta\mathbf{P}_N\end{bmatrix} + \Delta t_s\mathbf{T}_s\begin{bmatrix}0\\ I_2\\ \vdots\\ 0\end{bmatrix}$$

After this modification, we can substitute new $\Delta\mathbf{P}'$ into the matrix equation.

$$\Delta\mathbf{P}A\Delta\tilde{\mathbf{P}} + \mathbf{T}_s\Delta t_s e_2 \Rightarrow \begin{cases}\Delta\mathbf{P}_1 = \Delta\mathbf{P}_1,\\ \Delta\mathbf{P}_2 = \Delta\mathbf{P}_1 + \mathbf{T}_s\Delta t_s,\\ \vdots\\ \Delta\mathbf{P}_N = \Delta\mathbf{P}_N.\end{cases}$$

Now, substitute into the constraint equation:

$$B\Delta\mathbf{P} = \Delta\mathbf{D} \quad\Longrightarrow\quad B(A\Delta\tilde{\mathbf{P}} + \mathbf{T}_s\Delta t_s e_2) = \Delta\mathbf{D}.$$

Rearranging,

$$(BA)\Delta\tilde{\mathbf{P}} + (Be_2)\mathbf{T}_s\Delta t_s = \Delta\mathbf{D}$$

Let us define:

$$\tilde{B} = BA, \qquad c = (Be_2)\mathbf{T}_s$$

Then the system becomes:

$$\tilde{B}\Delta\tilde{\mathbf{P}} + c\Delta t_s = \Delta\mathbf{D}$$

This is still a linear system,

$$\begin{bmatrix}\tilde{B} & c\end{bmatrix}\begin{bmatrix}\Delta\tilde{\mathbf{P}}\\ \Delta t_s\end{bmatrix} = \Delta D$$

Equivalently in compact block notation:

$$\begin{bmatrix}B_{\text{red}} & \mathbf{0} & c_x\\ \mathbf{0} & B_{\text{red}} & c_y\end{bmatrix}\begin{bmatrix}\Delta x_{\text{red}}\\ \Delta y_{\text{red}}\\ \Delta t_s\end{bmatrix} = \begin{bmatrix}\Delta D_x\\ \Delta D_y\end{bmatrix},$$

where $B_{\text{red}} \in \mathbb{R}^{R\times(N-1)}$ is B with column 2 removed, $c_x, c_y \in \mathbb{R}^R$. Now we can obtain the minimum-norm (row-space) solution:

$$x^\star = B_{\text{block}}^T\left(B_{\text{block}}B_{\text{block}}^T\right)^{-1}b,$$

B. Directional tangency constraint

According to the system of equations formulated earlier, to apply a tangency constraint, the required tangent vector displacement at the point is required. Which means both direction and the magnitude of the tangent vector at the location can be manipulated as needed.

$$D^{(1)}_{\text{new}} = D^{(1)}_{\text{old}} + \Delta D^{(1)}$$

However, this isn't required in traditional CAD applications. Often when we apply a tangency constraint, the aim is to align the the tangent vectors in the same direction. This only enforces a directional constraint and not a magnitude constraint. This is important because directional tangency constraint only reduces the DOF of the system by 1, giving more freedom to find a solution. Given that that target tangent vector at $u = u_r$ is $\mathbf{T_t}$, then $D^{(1)}_{\text{new}}(u_r)$ should be parallel to $\mathbf{T_t}$. Then their cross product must be equal to zero. i.e.

$$D^{(1)}_{\text{new}}(u_r) \times \mathbf{T_t} = 0$$

In matrix notation, we can use 2D 90° rotation matrix A to rotate $\mathbf{T_t}$ and perform dot product to get the same result.

$$(D^{(1)}_{\text{new}})^T A \mathbf{T_t} = 0 \quad \text{with} \quad A = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}$$

Substituting for $D^{(1)}_{\text{new}}$,

$$(D^{(1)}_{\text{old}} + \Delta D^{(1)})^T A \mathbf{T_t} = 0$$

We know that $\Delta D^{(1)}(u_r) = \sum_{i=0}^{n} R'_{i,p}(u_r)\Delta \mathbf{P}_i$ and we can express this in terms of its corresponding row in overall coefficient matrix $B \longrightarrow B_t$ and the vector $\Delta\mathbf{P}$ as $\Delta D^{(1)}(u_r) = B_t \Delta \mathbf{P}$. Use the row extraction matrix $E_t \in \mathrm{R}^{1\times R}$ to obtain $B_t \in \mathrm{R}^{1\times N}$.

$$B_t = E_t B \quad \text{with} \quad E_t = \begin{bmatrix} 0 & \dots 0 & 1 & 0 & \dots & 0 \end{bmatrix}$$

Then the constraint becomes,

$$(D^{(1)}_{\text{old}}(u_r) + B_t \Delta \mathbf{P})^T A \mathbf{T}_t = 0$$

Move known term to RHS and apply the transpose to the bracketed term.

$$(B_t \Delta \mathbf{P})^T A \mathbf{T_t} = -(D^{(1)}_{\text{old}}(u_r))^T A \mathbf{T_t}$$

After taking the transpose of the full equation and simplified, we get the following relation.

$$\mathbf{T_t}^T A^T B_t \Delta \mathbf{P} = -\mathbf{T_t}^T A^T D^{(1)}_{\text{old}}(u_r)$$

This forms a new equation in the form of $B_{tan}\Delta\mathbf{P} = b_{tan}$ with,

$$B_{tan} = \mathbf{T_t}^T A^T B_t \quad \text{and} \quad b_{tan} = -\mathbf{T_t}^T A^T D^{(1)}_{\text{old}}(u_r)$$

Now we can generalize this, for any R_t number of tangency constraints with,

$$B_{\text{tan}} = \begin{bmatrix} \mathbf{T}_1^T A^T B_{t,1} \\ \vdots \\ \mathbf{T}_{R_t}^T A^T B_{t,R_t} \end{bmatrix} \quad b_{\text{tan}} = - \begin{bmatrix} \mathbf{T}_1^T A^T D^{(1)}_{\text{old}}(u_1) \\ \vdots \\ \mathbf{T}_{R_t}^T A^T D^{(1)}_{\text{old}}(u_{R_t}) \end{bmatrix}$$

Then the combined system becomes (with other M number of constraints),

$$\underbrace{\begin{bmatrix} B \\ B_{\text{tan}} \end{bmatrix}}_{B_m \in \mathbb{R}^{(M+R_t)\times 2N}} \quad \Delta\mathbf{P} = \quad \underbrace{\begin{bmatrix} \Delta D \\ b_{\text{tan}} \end{bmatrix}}_{\Delta D_{\text{modified}} \in \mathbb{R}^{M+R_t}}$$

Then the minimum-norm solution:

$$\Delta\mathbf{P}^\star = B_m^T (B_m B_m^T)^{-1} \Delta D_{\text{modified}}$$

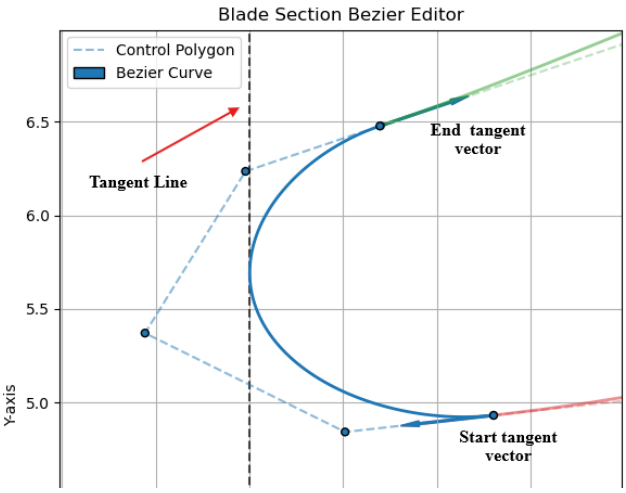

Figure 1: Blade editor interface with LE curve tangent to both PS,SS curves and stream surface boundary.

C. Unified formulation of both constraints

These two results can be combined to form a single system that can handle both start/end tangency constraints and directional tangency constraints at arbitrary parameter values. If any point has to be fixed, that can be handled by adding coefficient zero to respective columns.

IV. RESULTS

The blade editor interface using Python is shown in Fig. 1. The leading edge and trailing edge curves are constrained to be tangent to the specified directions (taken from pressure side and suction side curve end tangent vectors) at the start and end points respectively. The solver successfully maintains tangency constraints in real time, demonstrating its effectiveness.

V. CONCLUSION

A compact linear formulation for enforcing start/end and directional tangency constraints in parametric curves was presented. While this work focuses on 2D constraints, these planar curves act as cross-sections that are lofted span-wise to generate the full 3D blade surface, ensuring smooth transitions along the blade height. Its successful integration into a turbomachinery blade design tool demonstrates both accuracy and computational efficiency.

VI. ACKNOWLEDGMENT

This formulation was derived during the internship of the author at Turbogen Consultancy Pvt. Ltd. for the use of Bezier curve editor for the axial blade design module of their commercial software Turbotides. The author would like to thank Turbogen for the opportunity and support provided during the internship.

REFERENCES

[1] Les Piegl and Wayne Tiller. "Curve and Surface Fitting". In: *The NURBS Book*. Ed. by Les Piegl and Wayne Tiller. Berlin, Heidelberg: Springer, 1997, pp. 361–453. ISBN: 978-3-642-59223-2. DOI: 10.1007/978-3-642-59223-2_9. (Visited on 06/10/2025).

[2] Barry Fowler and Richard Bartels. "Constraint-Based Curve Manipulation". In: *IEEE Comput. Graph. Appl.* 13.5 (Sept. 1993), pp. 43–49. ISSN: 0272-1716. DOI: 10.1109/38.232098. (Visited on 10/18/2025).