# Rooting Out Self-Intersection: An Algebraic Shape-Optimization Barrier

Nima Kakhsaz Malayeri
Department of Electrical Engineering, Amirkabir University of Technology, Tehran, Iran
✉E-Mail: Nima.kakhsaz.m@gmail.com

## ABSTRACT

Self-intersection is a fundamental feasibility constraint in shape optimization: a self-crossing boundary leaves its interior, normal field, and finite-element mesh ill-defined, yet most existing barriers rely on heuristic geometric-proximity measures rather than certifying self-intersection directly. We propose a self-intersection barrier grounded in algebraic detection. We derived two bivariate polynomials from a curve's Fourier coefficients whose common roots correspond exactly to its self-intersections, and solved the resulting system using the existing colleague-pencil linearization of the Bézout resultant in a Chebyshev basis, evaluating resultant coefficients via the Fast Fourier Transform. We proved that the resulting barrier diverges if and only if the curve self-intersects, remains continuous everywhere on its domain, and reduces constrained search to unconstrained optimization over the Fourier coefficients. We identified and corrected three failure modes in the underlying polynomial system, with numerical reliability ensured by forward-error certification, Newton polishing, and a numerical safety net. We validated the detection method on six analytically solvable benchmark curves, matching MATLAB's built-in solver while running ~35× faster. In shape-optimization tests against a heuristic proximity barrier, our approach converged using ~17× fewer function evaluations, ~37× less runtime, and reached a 24× lower final cost, while tracking target geometries more closely — enforcing simplicity by construction rather than rejecting it after the fact.

**Keywords:** Shape Optimization, Closed Curves, Fourier Representation, Barrier Function
**MSC Classification:** 49Q10, 65H04, 13P15, 65F15

## 1. Introduction

At the core of many physics-based engineering simulations lies a formidable challenge: discovering optimal geometries that push the boundaries of performance. This often requires finding a boundary curve $\boldsymbol{\tau} : [0,1) \to \mathbb{R}^2$ that minimizes an objective $j(\boldsymbol{\tau})$, subject to $\boldsymbol{\tau}$ belonging to $\mathcal{J}$, the set of connected, bounded, closed, piecewise-smooth, non-self-intersecting curves. Formally:

$$\boldsymbol{\tau}^* = \underset{\boldsymbol{\tau}}{\operatorname{argmin}}\, j(\boldsymbol{\tau}) \quad \text{subject to} \quad \boldsymbol{\tau} \in \mathcal{J} \tag{1}$$

Such problems arise across many engineering applications: floating bridge pontoons (Lu et al. 2025), aircraft wings and airfoils under aerodynamic loading (Chen et al. 2026), bodies immersed in Navier–Stokes flow (Garcke et al. 2018), peristaltic pumps (Bonnet et al. 2020), acoustic scatterers (Hiptmair et al. 2018), aeroelastic wing–engine systems (Namani Koureh et al. 2026), and stellarator plasma boundaries (Paul et al. 2021). In each case, the boundary must stay physically admissible throughout optimization. Self-intersection is the most consequential failure mode. Whether the optimizer is gradient-based (Nocedal and Wright 2006) or a machine-learning model (Chen et al. 2026), the boundary can drift into a self-intersecting configuration even from a feasible starting shape. Once self-intersection happens, in many cases, the objective function can no longer be evaluated: the interior and exterior of the domain are no longer well defined, the boundary normal field breaks down, and any finite-element discretization becomes invalid. Preventing self-intersection is therefore not a refinement but a precondition for the optimization problem to be well posed.

In practice, searching $\mathcal{J}$ for an optimal design $\boldsymbol{\tau}^*$ requires parametrizing the boundary as $\boldsymbol{\tau}(\boldsymbol{\theta})$ using a finite design vector $\boldsymbol{\theta}$. Common basis choices include closed splines and Bézier curves (Piegl and Tiller 1997; Pekerman et al. 2008), Fourier series (Zahn and Roskies 1972; Deng et al. 2022), and level sets (van Dijk et al. 2013). Most requirements defining $\mathcal{J}$ are readily satisfied by such a parametrization: connectedness, boundedness, and piecewise smoothness

follow directly from the choice of basis. Closedness and non-self-intersection do not. A curve built from open polynomial or spline segments is closed only if its endpoints and derivatives are explicitly matched — a condition that must be imposed, not inherited from the basis. Non-self-intersection is harder still: it requires checking whether two distinct parameter values map to the same point, and standard parametrizations offer no built-in way to rule this out.

Even with a fast test for self-intersection, directly enforcing the hard feasibility condition in problem (1) gives the optimizer no way back once it strays into infeasible territory. When a self-intersecting shape occurs, repeated resampling of the design parameters $\boldsymbol{\theta}$ can fail to find a valid design within a set retry limit, stalling the optimization. The standard fix to this problem is to replace the hard constraint with a barrier function:

$$\boldsymbol{\tau}^* = \underset{\boldsymbol{\tau}}{\operatorname{argmin}} \left( j(\boldsymbol{\theta}) + \frac{1}{\varphi(\boldsymbol{\theta})} \right) \quad \text{subject to} \quad \boldsymbol{\theta} \subset \mathbb{R}^n$$

Here, $\varphi(\boldsymbol{\theta})$ must be constructed in a way that continuously approaches zero as $\boldsymbol{\theta}$ approaches a self-intersecting design. This gives the optimizer a guiding signal that steers the search away from invalid regions, avoiding the abrupt loss of information that hard rejection in problem (1) causes.

Existing strategies for building such a barrier fall into three categories. The first restricts the design space so self-intersecting configurations are excluded by construction, without an explicit barrier — as in polygonal primitives (Norato 2018, Zelickman and Guest 2025). The second builds a barrier from an indirect proxy for self-intersection: implicit level-set representations prevent self-intersection while naturally supporting topological changes (van Dijk et al. 2013), while non-local repulsion energies instead penalize curve self-proximity (Walker 2016; Wu et al. 2019; Yu et al. 2021; Sassen et al. 2024) — for example, through global radius-of-curvature penalties (Paul et al. 2021). The third detects self-intersections dynamically and resolves them numerically as they occur (Pekerman et al. 2008).

Each mentioned strategy presents a fundamental trade-off. Constructive constraints limit geometries to a small subset of space $\mathcal{J}$, while level-set methods carry high overhead from Hamilton–Jacobi PDEs and offer no direct intersection criterion. Repulsion energies scale quadratically with number of sampling points, and their sampling-based nature can miss self-intersections along complex boundaries. Direct detection methods also lack a continuous signal to prevent collisions proactively and is too slow for iterative use. Ultimately, current methods force a choice between coarse proxies and high computational cost.

We close this gap by combining two previously separate ideas: algebraic elimination theory, which characterizes self-intersection algebraically rather than approximately, and the continuous-barrier approach, which can be incorporated directly into a gradient-free optimization loop. The result replaces the heuristic proxies of existing methods with an algebraically grounded, continuous self-intersection barrier.

For curves with polynomial parametrizations, elimination theory gives a direct path to detecting self-intersections: the condition is mathematically equivalent to finding common roots of two bivariate polynomials corresponding to curve coordinates. Resultant methods are the standard computational tool for this, transforming the bivariate polynomial system into a univariate polynomial whose roots — typically computed via generalized eigenvalue decomposition (Sorber et al. 2014) — identify the common solutions. These formulations range from the classical Sylvester matrix (Noferini and Townsend 2016) to better-conditioned, Chebyshev-based Bézout matrices (Nakatsukasa et al. 2015; Gnazzo 2020), alongside Cayley (Chionh et al. 2002) and Macaulay (Jónsson and Vavasis 2005) resultants for higher-dimensional systems. These resultant matrices are linearized using companion matrices in the monomial basis (Amiraslani et al. 2009; Eidelman et al. 2008; Fasino and Gemignani 2003; Frederix et al. 2013) or colleague-pencil matrices in the Chebyshev basis (Good 1961; Boyd 2002). Alternative approaches to finding common roots include homotopy continuation (Sommese

and Wampler 2005; Telen et al. 2020), Gröbner-basis elimination (Buchberger 1998; Cox et al. 2015), and subdivision-based intersection techniques (Manocha and Demmel 1994; Bini and Di Marco 2006; Bini 1996). Resultant formulations, however, can become ill-conditioned for high-degree or nearly degenerate systems (Noferini and Townsend 2016; Lawrence and Corless 2014; Bini and Fiorentino 2000).

Polynomial and piecewise-polynomial representations dominate commercial computer-aided design (CAD) systems, thanks to their well-developed theory, but closedness is not automatic — it must be enforced through explicit continuity constraints, adding geometric restrictions that further complicate optimization. Existing elimination-based methods have yet to be paired with a representation that is closed by construction. We therefore propose a truncated Fourier representation, where periodicity and smoothness are intrinsic to the basis. A truncated Fourier curve is not inherently free of self-intersection (Kalmykov and Kovalev 2025), but its trigonometric coordinates can be mapped to an polynomial representation through substitution, allowing elimination methods to detect self-intersection. We derive the formulations needed to adapt polynomial elimination machinery to this setting, constructing a self-intersection barrier for Fourier-parametrized boundaries.

In doing so, we extend the resultant-based common-root formulation of Gnazzo (2020) — originally developed for polynomial systems — to self-intersection detection for arbitrary truncated Fourier-series curves. This extension addresses three failure modes unique to Fourier representations: a removable singularity, resultant degeneracy from sparse Fourier coefficients, and an incorrect parameter-recovery formula. We resolve each issue and combine the resultant's nullity with Newton–Raphson polishing to distinguish genuine intersections, tangential contact, and numerical near-misses.

To keep this efficient, we replace the Vandermonde-based coefficient inversion of Gnazzo (2020) with polynomial coefficients evaluated via Fast Fourier Transforms at Chebyshev–Lobatto nodes (Trefethen 2000; Trefethen 2013). We solve the resulting matrix polynomial using colleague-pencil linearization in the Chebyshev basis, and certified per-root forward-error bounds verify numerical reliability (Tisseur 2000). To our knowledge, this is the first self-intersection barrier for Fourier-parametrized curves grounded in algebraic detection. It remains continuous across its domain, reducing the constrained search over $\mathcal{J}$ to an unconstrained optimization over the Fourier coefficients themselves.

## 2. Algebraic Path to a Self-Intersection Barrier

Fig. 1 illustrates an arbitrary cross-section D with boundary $\partial$D, represented by the curve $\boldsymbol{\tau}$. To solve the optimization problem (2), we parametrize $\boldsymbol{\tau}$ using the truncated Fourier series representation:

$$\boldsymbol{\tau}(s) = \begin{bmatrix} a_{x_0} \\ a_{y_0} \end{bmatrix} + \sum_{n=1}^{N} \left( \begin{bmatrix} a_{x_n} \\ a_{y_n} \end{bmatrix} \cos 2\pi n s + \begin{bmatrix} b_{x_n} \\ b_{y_n} \end{bmatrix} \sin 2\pi n s \right) \quad , \quad s \in [0,1] \tag{2}$$

Where $a_{x_n}, b_{x_n}, a_{y_n}, b_{y_n}$ are the Fourier coefficients that form the parametrization (design) vector $\boldsymbol{\theta} = \left[ [a_{x_n}], [b_{x_n}], [a_{y_n}], [b_{y_n}] \right]$.

The selected basis functions in (2) ensure that $\boldsymbol{\tau}$ is continuous, smooth, and periodic (and thus closed), while constraining $\boldsymbol{\theta}$ directly bounds the curve itself. Moreover, the truncated Fourier representation is sufficiently flexible to represent any boundary in $\mathcal{J}$, as the harmonic cutoff $N$ increases (Boyd 2002).

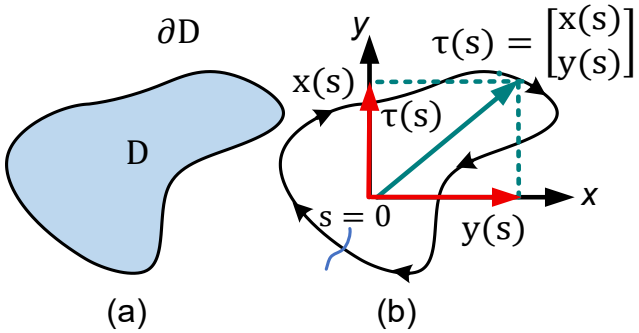


**Fig. 1** (a) The physical domain D and boundary ∂D (b) Parametrization of the boundary curve

A self-intersection occurs if and only if there exist two distinct parameters $(s_1, s_2)$ that map to the same point on the boundary curve:

$$\exists\, s_1, s_2 \in [0,1) \quad , \quad s_1 \neq s_2 \quad \text{such that} \quad \boldsymbol{\tau}(s_1) = \boldsymbol{\tau}(s_2) \tag{3}$$

Substituting the Fourier representation (2) into Equation (3) yields:

$$\sum_{n=1}^{N} \left( \begin{bmatrix} a_{x_n} \\ a_{y_n} \end{bmatrix} (\cos 2\pi n s_1 - \cos 2\pi n s_2) + \begin{bmatrix} b_{x_n} \\ b_{y_n} \end{bmatrix} (\sin 2\pi n s_1 - \sin 2\pi n s_2) \right) = 0 \tag{4}$$

While this equation describes self-intersections, it includes the trivial solution $s_1 = s_2$. Moreover, the system's coupled trigonometric structure prevents the direct use of algebraic elimination methods such as resultants — a limitation we will address.

## 2.1 Fourier Representation

Because equation (4) admits the trivial root $(s_1 = s_2)$, we introduce two new variables, $\sigma$ and $\delta$:

$$\sigma = \pi(s_1 + s_2) \qquad \delta = \pi(s_1 - s_2)$$

From the product-to-sum identities:

$$\cos 2\pi n s_2 - \cos 2\pi n s_1 = 2 \sin n\sigma \sin n\delta$$
$$\sin 2\pi n s_1 - \sin 2\pi n s_2 = 2 \sin n\delta \cos n\sigma$$

Substituting these into (4) gives:

$$\sum_{n=1}^{N} \left( \begin{bmatrix} b_{x_n} \\ b_{y_n} \end{bmatrix} \cos n\sigma - \begin{bmatrix} a_{x_n} \\ a_{y_n} \end{bmatrix} \sin n\sigma \right) \sin n\delta = 0$$

Setting

$$u = \cos\sigma \quad v = \cos\delta$$

and using the following Chebyshev identities ($T_n$,$U_n$: Chebyshev polynomials of the first and second kind)

$$\cos n\theta = T_n(\cos\theta) \quad \sin n\theta = \sin\theta\, U_{n-1}(\cos\theta)$$

gives:

$$\sin\delta \sum_{n=1}^{N} \left( \begin{bmatrix} b_{x_n} \\ b_{y_n} \end{bmatrix} T_n(u) - \begin{bmatrix} a_{x_n} \\ a_{y_n} \end{bmatrix} \sin\sigma\, U_{n-1}(u) \right) U_{n-1}(v) = 0$$

Dividing both sides by $\sin\delta$ removes the trivial solution $s_1 = s_2$ and yields:

$$\sum_{n=1}^{N} \left( \begin{bmatrix} b_{x_n} \\ b_{y_n} \end{bmatrix} T_n(u) - \begin{bmatrix} a_{x_n} \\ a_{y_n} \end{bmatrix} \sin\sigma\, U_{n-1}(u) \right) U_{n-1}(v) = 0$$

Letting $w := \sin\sigma$, this equation reduces to two linear equations combined with the Pythagorean identity:

$$\begin{aligned} wA_x(u,v) + B_x(u,v) &= 0 \\ wA_y(u,v) + B_y(u,v) &= 0 \end{aligned} \qquad u^2 + w^2 - 1 = 0 \tag{5}$$

where

$$A_x(u,v) = -\sum_{n=1}^{N} a_{x_n} U_{n-1}(u) U_{n-1}(v) \quad B_x(u,v) = \sum_{n=1}^{N} b_{x_n} T_n(u) U_{n-1}(v)$$
$$A_y(u,v) = -\sum_{n=1}^{N} a_{y_n} U_{n-1}(u) U_{n-1}(v) \quad B_y(u,v) = \sum_{n=1}^{N} b_{y_n} T_n(u) U_{n-1}(v)$$

with $u, v, w \in [-1,1]$. To reduce (5) to a system of two bivariate polynomials, we eliminate $w$ via Lemmas 1–3.

**Lemma 1 (Linear compatibility)**. If $(u, v, w)$ satisfies (5) with $\left(A_x(u,v), A_y(u,v)\right) \neq (0,0)$, then

$$p_1(u,v) \coloneqq A_x(u,v)B_y(u,v) - A_y(u,v)B_x(u,v) = 0 \tag{6}$$

*Proof.* The linear sub-system of Equation (5) reads:

$$\begin{bmatrix} A_x(u,v) & B_x(u,v) \\ A_y(u,v) & B_y(u,v) \end{bmatrix} \begin{bmatrix} w \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

A solution for $w$ exists only if this matrix is singular, which gives (6).

**Lemma 2 (Quadratic compatibility)**. Under the hypotheses of Lemma 1, if $(u, v, w)$ additionally satisfies (6), then

$$p_2(u,v) \coloneqq B_x^2(u,v) + A_x^2(u,v)(u^2 - 1) = 0$$

*Proof.* Define

$$K_x \coloneqq B_x^2(u,v) - A_x^2(u,v)(1-u^2) = p_2(u,v)$$
$$K_y \coloneqq B_y^2(u,v) - A_y^2(u,v)(1-u^2)$$

Squaring and adding the linear equations in (5) gives:

$$w^2\left(A_x^2(u,v) + A_y^2(u,v)\right) = B_x^2(u,v) + B_y^2(u,v)$$

Substituting this into the Pythagorean equation of system (5) gives:

$$\left(A_x^2(u,v) + A_y^2(u,v)\right)(1-u^2) - \left(B_x^2(u,v) + B_y^2(u,v)\right) = 0$$

this is equivalent to

$$K_x + K_y = 0$$

A direct computation gives:

$$A_y^2(u,v)K_x - A_x^2(u,v)K_y = -p_1(u,v)\left(A_y(u,v)B_x(u,v) + A_x(u,v)B_y(u,v)\right) = 0$$

by Lemma 1. Combining this with $K_x + K_y = 0$ gives:

$$\left(A_x^2(u,v) + A_y^2(u,v)\right)K_x = 0$$

Since $A_x^2(u,v) + A_y^2(u,v) > 0$, $K_x = 0$.

**Lemma 3 (Degenerate case)**. If $A_x(u,v) = A_y(u,v) = 0$, (5) reduces to $B_x(u,v) = B_y(u,v) = 0$. Every such solution satisfies $p_1(u,v) = p_2(u,v) = 0$:

$$p_1(u,v) = A_x(u,v)\cdot 0 - A_y(u,v)\cdot 0 = 0$$
$$p_2(u,v) = 0 + 0(u^2-1) = 0$$

*Remark.* The converse fails here: $p_1(u,v)$ vanishes identically regardless of $B_y(u,v)$, and $p_2(u,v) = B_x^2(u,v)$ does not constrain $B_y(u,v)$. Therefore, Equation (7) below may contain extraneous roots with $B_x(u,v) = 0$, $B_y(u,v) \neq 0$.

**Theorem 1 (Elimination of w)**. Every solution of (5) with $(u,v) \in [-1,1]^2$ satisfies

$$\begin{matrix} p_1(u,v) = 0 \\ p_2(u,v) = 0 \end{matrix}, \quad (u,v) \in [-1,1]^2 \tag{7}$$

*Proof.* Lemmas 1–3.

## 2.2 Bivariate Root Resolution

Candidate self-intersections of the boundary curve $\boldsymbol{\tau}$ are represented by system (7). This bivariate polynomial system is solved via an algebraic elimination approach that combines the Bézout resultant with colleague-pencil linearization, following Gnazzo (2020). Both are well-established techniques, so for brevity we relegate the explicit matrix constructions, basis transformations, and coefficient mappings to Appendix A and outline only the core methodology below.

The bivariate polynomials $p_1(u,v)$ and $p_2(u,v)$ admit the standard monomial expansions

$$p_1(u,v) = \sum_{i=0}^{2N-1}\sum_{j=0}^{2N-2} p_{1_{ij}} u^i v^j \quad , \quad p_2(u,v) = \sum_{i=0}^{2N}\sum_{j=0}^{2N-2} p_{2_{ij}} u^i v^j$$

where the coefficient arrays $p_{1_{ij}}$ and $p_{2_{ij}}$ are computed directly from the parametrization vector $\boldsymbol{\theta}$.

For better numerical conditioning, $p_1(u,v)$ and $p_2(u,v)$ are transformed into the Chebyshev basis:

$$p_1(u,v) = \sum_{i=0}^{2N-1}\sum_{j=0}^{2N-2} p_{1_{ij}}^{(\mathrm{ch})} T_i(u)T_j(v) \quad , \quad p_2(u,v) = \sum_{i=0}^{2N}\sum_{j=0}^{2N-2} p_{2_{ij}}^{(\mathrm{ch})} T_i(u)T_j(v) \tag{8}$$

Eliminating $u$ via the Bézout resultant projects the common roots onto a univariate matrix singularity condition in $v$:

$$p_1(u,v) = p_2(u,v) = 0 \rightarrow \det\left(\mathrm{B}_{p1,p2}(v)\right) = 0$$

Here, $\mathrm{B}_{p1,p2}(v)$ is the Bézout resultant matrix, whose entries are univariate polynomials in $v$. To avoid solving this nonlinear determinant condition directly, colleague-pencil linearization is applied instead, converting the root-finding problem into a standard generalized eigenvalue problem:

$$\det(\mathcal{L}_0 - v\mathcal{L}_1) = 0$$

where $\mathcal{L}_0$ and $\mathcal{L}_1$ are constant matrices. This equation removes the explicit dependence on $v$ from the determinant condition on $(\det \mathrm{B}_{p1,p2}(v) = 0)$, so that candidate roots $v^*$ can be computed directly.

After computing candidate points $(v^*)$, each corresponding $u^*$ is recovered from the eigenvector of the pencil $(\mathcal{L}_0 - v^*\mathcal{L}_1)$, provided $v^*$ is a simple root. If $v^*$ is a repeated root, the nullspace of $(\mathcal{L}_0 - v\mathcal{L}_1)$ no longer isolates $u^*$ uniquely; in this case, $u^*$ is instead recovered by substituting $v^*$ into (7) and solving for the common roots of the resulting univariate polynomials $p_1(u,v^*)$ and $p_2(u,v^*)$ using the same Bézout resultant machinery.

For each pair $(u^*,v^*)$ satisfying (7), $A_x^2(u^*,v^*) + A_y^2(u^*,v^*)$ is evaluated first. By Theorem 1, if this quantity is nonzero, $w^*$ is recovered as:

$$w^* = -\frac{\left(A_x(u^*,v^*)B_x(u^*,v^*) + A_y(u^*,v^*)B_y(u^*,v^*)\right)}{A_x^2(u^*,v^*) + A_y^2(u^*,v^*)}$$

If instead $A_x(u^*,v^*) = A_y(u^*,v^*) = 0$, $B_x(u^*,v^*)$ and $B_y(u^*,v^*)$ are evaluated; if both vanish, $(u^*,v^*)$ is a genuine root by Lemma 3, and $w^*$ is recovered from the Pythagorean equation (5) as:

$$w^* = \pm\sqrt{1-u^{*2}}$$

Given a triplet $(u^*,v^*,w^*)$, the self-intersection location is recovered by first using

$$\sigma = \mathrm{atan2}(w^*,u^*) \quad \delta = \arccos(v^*) \tag{9}$$

and finally

$$s_1 = \mathrm{mod}\left(\frac{\sigma+\delta}{2\pi},1\right) \quad s_2 = \mathrm{mod}\left(\frac{\sigma-\delta}{2\pi},1\right) \tag{10}$$

To remove spurious solutions, $s_1$ and $s_2$ from Equation (10) must be checked against the original Equation (4). Note that, owing to the symmetry of mapping (9), the negative branch of arccos $v^*$ is dropped: in the degenerate case ($A_x(u,v) = A_y(u,v) = 0$), the formula for $w^*$ has ambiguous sign, and both signs in the Pythagorean formula above can be valid.

*Remark*. Geometrically, on the periodic parameter circle $s \in [0,1)$, $\delta$ fixes the parameter separation between candidate points, while $\sigma$ determines their midpoint. Negating $w^*$ reflects this midpoint via $\sigma \to -\sigma$, mapping the primary candidate pair $(s_1, s_2)$ to its mirrored counterpart $(1-s_2, 1-s_1)$. Because the reflection $s \to 1-s$ traverses the parameter circle in the opposite direction, these two branches correspond to physically distinct candidate locations. Consequently, in the degenerate case, both $(s_1, s_2)$ and $(1-s_2, 1-s_1)$ must be checked against Equation (4) to filter spurious roots.

### *2.2.1 Root Certification*

Suppose solving $\det \mathrm{B}_{p1,p2}(v) = 0$ yields the numerical root $\widetilde{v^*}$. Owing to finite-precision arithmetic, $\widetilde{v^*}$ generally differs from the exact root $v^*$. An inclusion radius $r_{\widetilde{v^*}}$ is therefore sought such that the disk centered at $\widetilde{v^*}$ is guaranteed to contain the exact root (Tisseur 2000):

$$D(\widetilde{v^*}, r_{\widetilde{v^*}}) = \{z \in \mathbb{C} : |z - \widetilde{v^*}| \le r_{\widetilde{v^*}}\} \tag{11}$$

If this inclusion disk intersects the target interval $[-1,1]$, the computed root is retained as a candidate, with the inclusion radius providing a measure of its numerical reliability. Gnazzo (2020) estimates $r_{\widetilde{v^*}}$ by analyzing the forward error of this determinant condition, yielding

$$|v^* - \widetilde{v^*}| \le r_{\widetilde{v^*}} = \sigma_{\min}\left(\mathrm{B}(\widetilde{v^*})\right) \frac{\|\widetilde{\boldsymbol{v}^*}\|_2^2}{|\widetilde{v^*}| \left|\widetilde{\boldsymbol{v}^*}^{\mathrm{H}} \mathrm{B}'(\widetilde{v^*}) \widetilde{\boldsymbol{v}^*}\right|}$$

where $\widetilde{\boldsymbol{v}^*}$ denotes the eigenvector corresponding to the computed eigenvalue $\widetilde{v^*}$, and $\mathrm{B}'^{(\widetilde{v^*})}$, $\|\cdot\|_2$, and $\sigma_{min}$ denote the derivative of $\mathrm{B}(v)$ with respect to $v$, the 2-norm, and the minimum singular value, respectively. Appendix A provides a closed-form expression for $\mathrm{B}'^{(\widetilde{v^*})}$.

## 2.3 Optimization Barrier

Hard constraints that simply reject intersecting designs waste iterations without guiding the search, and existing continuous-barrier alternatives avoid this but remain heuristic rather than algebraically grounded (Walker 2016; Wu et al. 2019; Yu et al. 2021; Sassen et al. 2024). We build an algebra-based continuous barrier on the root-finding method from Section 2.2: it steers the Fourier coefficients away from intersecting configurations, with a penalty that grows smoothly as a design nears the boundary. This gives the optimizer a consistent signal, rather than the abrupt swings a hard constraint produces.

### *2.3.1 Barrier Introduction*

To construct such a self-intersection barrier, we first introduce quantities:

$$\begin{aligned} \lambda_v &= \{v \in \mathbb{C} : \det(\mathcal{L}_{0_v} - v\mathcal{L}_{1_v}) = 0\} \\ \lambda_u &= \{u \in \mathbb{C} : \det(\mathcal{L}_{0_u} - u\mathcal{L}_{1_u}) = 0\} \\ \lambda_{\mathrm{w}} &= \{(u,v) : u \in \lambda_u, v \in \lambda_v\} \end{aligned}$$

where we obtain the pair $(\mathcal{L}_{0_v}, \mathcal{L}_{1_v})$ via colleague-pencil linearization by eliminating $v$, and construct the pair $(\mathcal{L}_{0_u}, \mathcal{L}_{1_u})$ by eliminating $u$ from system (7). We define barrier $\varphi(\boldsymbol{\theta})$ as:

$$\varphi(\boldsymbol{\theta}) = \varsigma_v(\boldsymbol{\theta}) + \varsigma_u(\boldsymbol{\theta}) + \varsigma_w(\boldsymbol{\theta}) \tag{12}$$

given that:

$$\begin{aligned} \varsigma_v(\boldsymbol{\theta}) &= \mathrm{softMin}\left(\left\{\bar{\sigma}\left(\mathrm{B}_v\left(\Re(v)\right)\right) + \psi(|v|)\right\}\right), \quad v \in \lambda_v \\ \varsigma_u(\boldsymbol{\theta}) &= \mathrm{softMin}\left(\left\{\bar{\sigma}\left(\mathrm{B}_u\left(\Re(u)\right)\right) + \psi(|u|)\right\}\right), \quad u \in \lambda_u \\ \varsigma_w(\boldsymbol{\theta}) &= \mathrm{softMin}\left(\left\{f\left(\Re(u), \Re(v)\right) + \psi(|v|) + \psi(|u|)\right\}\right), \quad (u,v) \in \lambda_{\mathrm{w}} \end{aligned} \tag{13}$$

where $f(u,v)$ is given by:

$$f(u,v) = \left( \left( \left(A_x B_x + A_y B_y\right)^2 + \left(A_x^2 + A_y^2\right)^2 (u^2 - 1) \right)^2 + (p_1^2 + p_2^2) \right) + \left( e^{-\beta\left(A_x^2 + A_y^2\right)} \left(B_x^2 + B_y^2\right) \right) \quad (14)$$

Here, $\beta$ is a large number (typically $10^6$) and $\psi$ is given by:

$$\psi(|z|) = \begin{cases} 0 & |z| \le 1 \\ (|z| - 1)^2 & |z| > 1 \end{cases}$$

and $\bar{\sigma}(\mathrm{B})$is calculated by:

$$\bar{\sigma}(\mathrm{B}) = \mathrm{softMin}(\{\sigma_i\})$$

given that:

$$\mathrm{softMin}(\{x_i\}) = \sum_i x_i \frac{e^{-\alpha x_i}}{\sum_j e^{-\alpha x_j}}$$

Here, $\alpha$ is a large number (typically $10^6$). We provide a MATLAB implementation of the barrier function in Online Resource 1.

### *2.3.2 Barrier Justification*

Replacing the softMin operations in Equations (13) with a hard minimum yields the algebraic necessary condition, $\varsigma_v(\boldsymbol{\theta}) = \varsigma_u(\boldsymbol{\theta}) = 0$ — real, in-range colleague-pencil eigenvalues belonging to interval [−1, 1]. This condition is necessary but not sufficient: a real root makes $\mathrm{B}_u$ or $\mathrm{B}_v$ singular regardless of range, and even when real and in-range $u^*$ and $v^*$ both exist, no particular pairing is guaranteed to jointly satisfy the system (5) and survive a successful $w$-recovery. We could discard invalid roots outright and enforce both constraints directly for $\mathrm{B}_u$ and $\mathrm{B}_v$, but as $\boldsymbol{\theta}$ varies continuously, a root can cross the real segment $[-1,1]$ in the complex plane. That crossing would create a discontinuous jump in the barrier at bifurcation points. Therefore, we instead retain every root and suppress invalid ones continuously rather than discarding them. The next two theorems make this precise.

**Lemma 4 (Necessary-condition vanishing).** As $\alpha \to \infty$, $\varsigma_v(\boldsymbol{\theta})$ vanishes if and only if $\lambda_v$ contains a real root $v^* \in [-1,1]$; otherwise, it remains bounded away from zero. The same holds for $\varsigma_u(\boldsymbol{\theta})$ with respect to $\lambda_u$.

*Proof.* The softMin operation suppresses each invalid root in $\lambda_v$ in two ways. First, (13) uses only the real part of each extracted root, so $\mathrm{B}_v$ reaches exact singularity only when the root is real; a complex root instead yields a generally nonzero value, which softMin suppresses. Second, $\psi$ penalizes any retained real or complex outlier root by its modulus, while leaving in-range roots (inside the unit circle) unaffected. As a result, a genuine, real, in-range root contributes nothing to the softMin sum (to machine precision) while every other root contributes a small positive residue that decays exponentially in $\alpha$. So $\varsigma_v(\boldsymbol{\theta})$ — and, similarly, $\varsigma_u(\boldsymbol{\theta})$ — vanishes as $\alpha \to \infty$ whenever a genuine root exists.

**Lemma 5 (Pairing sufficiency).** For $\alpha, \beta \to \infty$ and $(u,v) \in \lambda_u \times \lambda_v$, $f(u,v) = 0$ together with $\psi(|u|) = \psi(|v|) = 0$ holds if and only if $u$ and $v$ are both real, both lie in $[-1,1]$, and jointly satisfy the $w$-recovery relation (5) — that is, $(u,v)$ is a genuine candidate self-intersection pairing.

*Proof.* We evaluate $\varsigma_v(\boldsymbol{\theta})$ and $\varsigma_u(\boldsymbol{\theta})$ independently, and each encodes only the necessary condition, so no single $(u,v)$ pairing alone constitutes evidence of a genuine self-intersection. We therefore introduce $f(u,v)$ in (14) to verify correct $w$-recovery across the two branches. When $A_x$ and $A_y$ do not both vanish, the first term of $f(u,v)$ checks the correctness of the recovered $w$ (through Pythagorean identity) and we choose $\beta$ large enough that the second term's contribution stays small. When $A_x = A_y = 0$, the first term in $f$ vanishes identically and $f$ reduces to $B_x^2 + B_y^2$, which provides the sufficient condition for the degenerate branch.

However, $f(u,v) = 0$ only verifies algebraic $w$-recovery; valid algebraic recovery can still occur for a pair $(u,v)$ where $u$ or $v$ falls outside the domain $[-1,1]$. We therefore apply the same safeguard function $\psi$ to each $(u,v)$ pair via $\psi(|u|) + \psi(|v|)$.

**Theorem 2 (Barrier characterization).** For $\alpha, \beta \to \infty$, $\varphi(\boldsymbol{\theta}) \to 0$ if and only if the Fourier curve $\boldsymbol{\tau}(\boldsymbol{\theta})$ has at least one genuine self-intersection.

*Proof.* We evaluate the term $\varsigma_w$ over every pairing drawn from the full, unfiltered candidate sets $\lambda_v$ and $\lambda_u$. A pairing that satisfies both necessary conditions and drives $\varsigma_w$ to zero indicates a genuine self-intersection, so all three $\varsigma$-terms become small to machine precision (Lemmas 4 and 5). A pairing that satisfies only the necessary conditions, by contrast, leaves $\varsigma_w$ bounded away from zero. Since all three terms are non-negative ($f, \bar{\sigma}, \psi \geq 0$), their sum stays small only for a self-intersecting curve. And because we retain the genuine crossing throughout the self-intersection domain — not just at its boundary — the barrier gives a useful signal everywhere, not only near the threshold.

*Remark (Boundary cases).* Two limiting cases confirm this. When no genuine self-intersection exists, every retained root is either complex — so no real root drives $\mathrm{B}_u$ or $\mathrm{B}_v$ to a singularity — or real but out of bounds, where $\psi$ keeps the penalized term positive despite $\bar{\sigma}$ vanishing; either way, $\varsigma_v(\boldsymbol{\theta})$ and $\varsigma_u(\boldsymbol{\theta})$ stay bounded away from zero. When multiple genuine crossings coexist, each instead contributes a near-zero term; since softMin cannot fall below the smallest input, $\varsigma_v(\boldsymbol{\theta})$, $\varsigma_u(\boldsymbol{\theta})$, and $\varsigma_w(\boldsymbol{\theta})$ all remain near zero regardless of how many valid candidates tie.

**Theorem 3 (Continuity).** $\varphi(\boldsymbol{\theta})$ is continuous.

*Proof.* $\varphi(\boldsymbol{\theta})$ is continuous because four properties hold together: singular values vary continuously with the matrix entries (Horn and Johnson 2012); polynomial roots, as a multiset, vary continuously with the coefficients, so their real and imaginary parts vary continuously with $\boldsymbol{\theta}$ (Harris and Martin 1987); $\psi$ is continuous across $|z| = 1$; and every root in $\lambda_u$ and $\lambda_v$ is retained (softMin) rather than filtered by reality or magnitude.

## 3. Numerical Results

We validate the proposed methodology in two experiments. Section 3.1 tests the self-intersection detection algorithm from Sections 2.1 and 2.2. Section 3.2 evaluates the barrier from Section 2.3 within an optimization setting.

### 3.1 Self-Intersection Detection Validation

Appendix B provides the complete implementation of the self-intersection detection algorithm; we validate it here instead, since the barrier is built directly on this detection and inherits its reliability. The implementation routes between three paths: (1) PAP, the direct algebraic path, which solves Equation (7) and checks candidates against Equation (4); (2) NPAP, the same algebraic path refined by Newton–Raphson polishing; and (3) NFBP, a numerical fallback triggered when the resultant matrix B becomes degenerate, as happens for single-harmonic or sparse-coefficient curves.

Newton polishing augmentation to algebraic path does more than its name suggests. It resolves three distinct problems. First, it refines every candidate to machine precision. Second, it separates genuine roots from noise artifacts, in two ways. A near-miss — where the curve passes close to itself but never touches — has no true root nearby, so polishing diverges or oscillates instead of converging, and the candidate is rejected. In the second, when coefficient noise splits one genuine root, such as the coincident crossing in a quatrefoil, into several nearby candidates, polishing draws them back to a single point. By contrast, genuinely distinct self-intersections that lie close together do not share a single convergence point, so polishing refines them independently. Third, polishing confirms tangential (higher-order ($d > 1$) self-intersections directly, since the Jacobian of the underlying system vanishes exactly at these points.

We validate the proposed self-intersection detection method four ways: against curves with analytically known self-intersections, against MATLAB's geometric solver, against the polynomial-curve method of Pekerman et al. (2008), and at scale for computational performance.

We first test six benchmark curves embedded inside the unit square, each designed to stress a specific algorithmic safeguard: a simple rank-one intersection, a quatrefoil with multiple passes through a single root, a tangential intersection, an intersection with clustered roots nearby, a degenerate case ($A_x = A_y = 0$), and a near-degenerate case created by perturbing one Fourier coefficient by $10^{-3}$. As Fig. 2 shows, the Newton Polished Algebraic Path (NPAP) resolves the self-intersections correctly in every case.

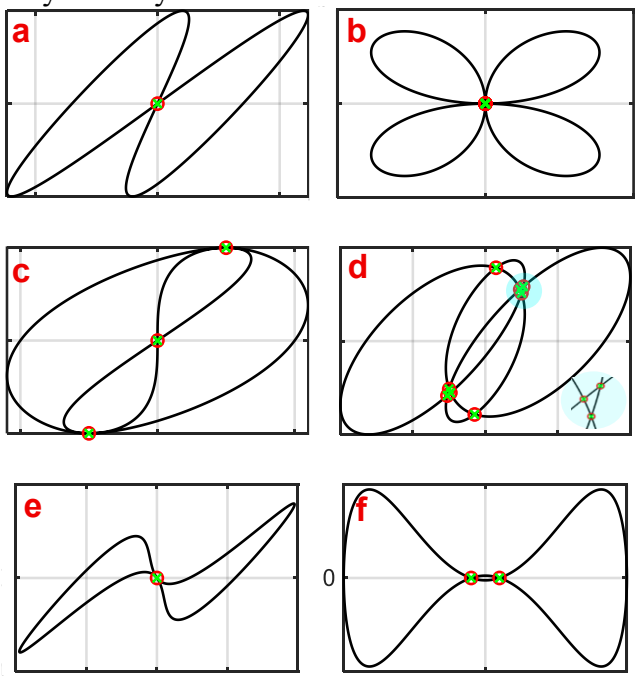


**Fig. 2** Validation on six hand-picked cases, each targeting a specific algorithmic safeguard NPAP resolves all six cases correctly. Fig. 2a shows a simple rank-one self-intersection in an eight-shaped curve as a baseline. Fig. 2b shows a quatrefoil where multiple parameter pairs converge to a single intersection; Newton polishing reduces the accumulated numerical error to near machine precision, collapsing the apparent cluster into one point. Fig. 2c shows higher-order, tangential self-intersections, which the algorithm identifies correctly. Fig. 2d shows a clustered self-intersection: the algorithm keeps the distinct nearby roots separate instead of merging them. Fig. 2e shows a degenerate 0/0 form in $w$-recovery at the origin, which the algorithm also resolves correctly. Fig. 2f shows a clustered-root case created by perturbing one Fourier coefficient, which splits the degree-two intersection at the origin into two closely spaced intersections; the algorithm detects both.

To further assess the detection method, we compare NPAP against MATLAB's built-in polyxpoly function. Unlike NPAP, which operates directly on the algebraic representation of the Fourier curve, polyxpoly detects intersections geometrically: it samples points along the curve, connects consecutive samples with line segments, and tests these segments for intersection. Its result therefore depends on the chosen discretization rather than the curve's algebraic structure. We randomly generate a self-intersecting degree-10 Fourier curve, discretize it with 4000 points, and apply both polyxpoly and NPAP. Fig. 3 and Fig. 4 show the results.

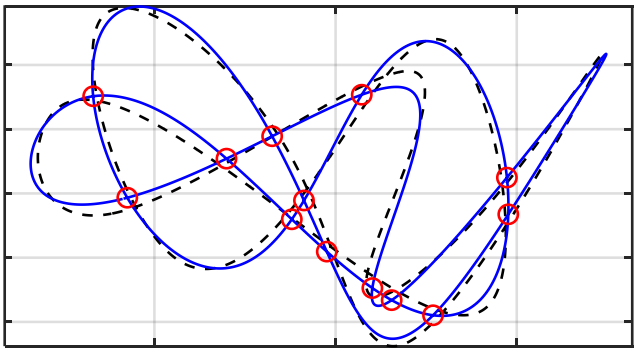

**Fig. 3** NPAP (red circles) vs. polyxpoly's polygonal approximation (dashed), $N = 6$ approximation of original curve (solid)

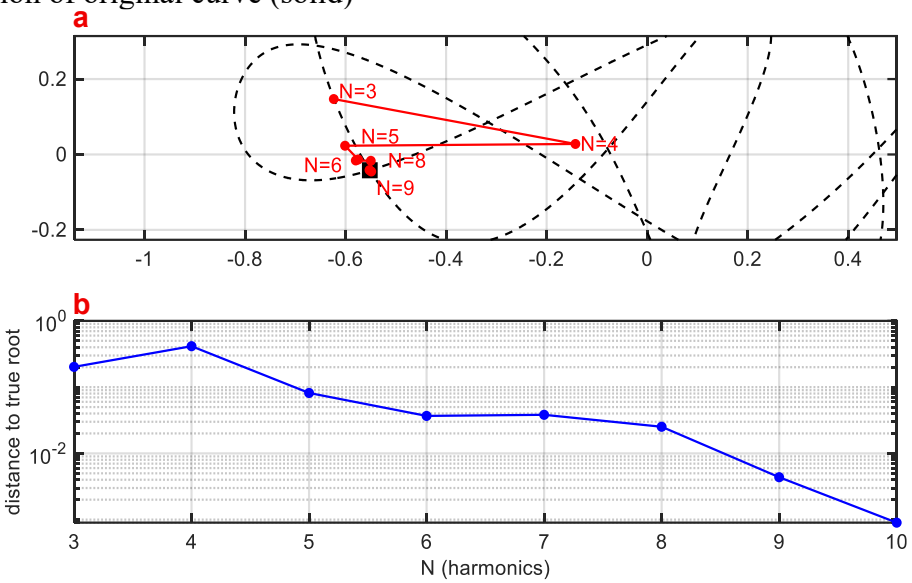


**Fig. 4** (a) Convergence of the detected intersection (solid circles) toward polyxpoly's approximation (dashed) as harmonics increase (b) Distance between the two calculations

Fig. 3 shows that NPAP correctly identifies all self-intersections of the ten-harmonic Fourier curve (red hollow circles), even when the curve is truncated to six harmonics (solid line). The dashed line shows the full ten-harmonic curve and its intersections as found by MATLAB's polyxpoly. Fig. 4 confirms the same trend: as harmonic order increases, the truncated series' self-intersections move closer to the intersections MATLAB's polyxpoly finds.

Pekerman et al. (2008) address self-intersection detection for polynomial Bézier and B-spline curves. To compare against their work, we approximate their degree-six Bézier test curve with truncated Fourier series at increasing harmonic orders and run each approximation through NPAP (Fig. 5). NPAP correctly identifies one self-intersection at every truncation from $N = 2$ to $N = 7$. Over this range, the Fourier-approximation residual falls from $8.66 \times 10^{-4}$. to $1.32 \times 10^{-14}$, while runtime rises modestly from 15 ms to 44 ms — comparable to Pekerman's 38 ms. This confirms the method works beyond curves natively represented in the Fourier basis.

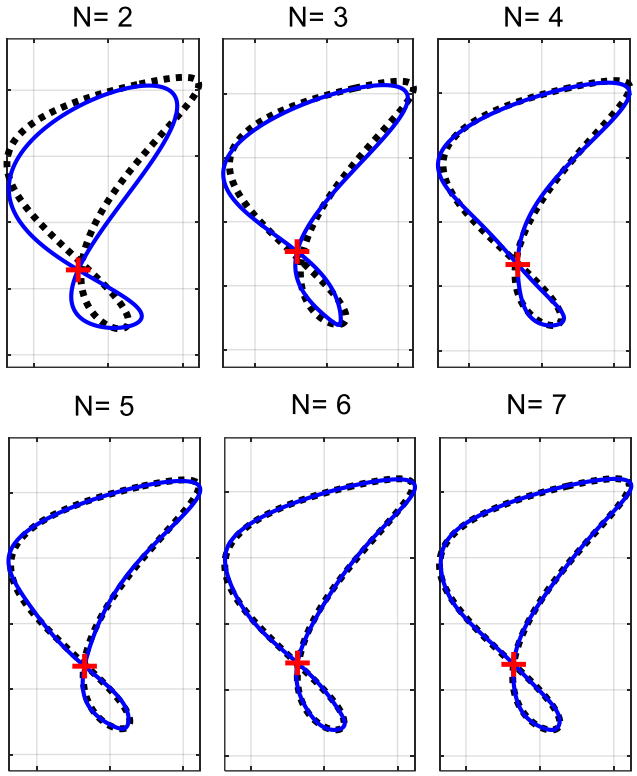


**Fig. 5** Self-intersection results for Pekerman's Bézier curve (dashed) against its Fourier approximations (solid) resolved using NPAP

As a final validation step, we test the method's performance at scale. We generate 50 random Fourier curves per harmonic order ($3 \leq N \leq 10$) and evaluate them with three approaches: PAP, NFBP (400/4000 seed points), and the proposed full pipeline. For each, we record the detected-intersection count, runtime, and worst-case inclusion-disk radius as a reliability measure, running all tests on an 8-core AMD processor at 4.2 GHz. Table 1 presents the results.

**Table 1** Comparison between PAP, NFBP, and proposed (full pipeline) methods, comparing number of roots and average runtime and inclusion disk radius

| N | Total | | | Average Runtime (ms) | | | Condition |
|---|---|---|---|---|---|---|---|
| | PAP | NFBP | Proposed | PAP | NFBP | Proposed | Max ($r_{\widetilde{v^*}}$) |
| 3 | 188 | 188 | 188 | 7.1 | 45.3 | 17.3 | 1.7E-13 |
| 4 | 371 | 371 | 371 | 8.1 | 47.5 | 18.3 | 1.4E-11 |
| 5 | 599 | 596 | 599 | 17.3 | 45.6 | 27.5 | 2.8E-11 |
| 6 | 908 | 901 | 908 | 39.5 | 471.9 | 49.7 | 5.4E-11 |
| 7 | 1264 | 1256 | 1264 | 89.4 | 471.8 | 99.6 | 2.6E-11 |
| 8 | 1662 | 1639 | 1662 | 192.2 | 471.7 | 202.4 | 6.7E-10 |
| 9 | 1985 | 1968 | 1985 | 373.5 | 481.3 | 383.7 | 3.4E-11 |
| 10 | 2596 | 2573 | 2593 | 712.0 | 483.8 | 722.2 | 2.2E-6 |

The proposed method, PAP, and NFBP agree closely for $N < 5$, with an exception worth tracing. At $N = 5$, NFBP reports fewer roots than the proposed method and PAP; tracing the discrepancy confirms the proposed method's count is correct: this is because the NFBP simply cannot resolve tangential self-intersections correctly, although rare they can happen even in randomly generated curves. At higher orders ($N \geq 6$), NFBP consistently undercounts relative to the algebraic method by 7–23 roots, since growing intersection density outpaces its fixed grid resolution. The proposed method matches the PAP count across $6 \leq N \leq 9$. At $N = 10$, PAP reports extra roots that the proposed method correctly rejects. These extra roots trace back to a single high-degree intersection: without Newton polishing, PAP resolves it as a cluster of

separate roots instead of one. This also matches the sharp rise in maximum inclusion-disk radius.

Runtime scaling reflects the same structural differences. PAP's runtime grows smoothly, driven by the cubic complexity of the colleague-pencil generalized eigenvalue solve. NFBP spikes sharply between $N = 5$ and $N = 6$, as its required seed count jumps from 400 to 4000. The proposed hybrid method carries a fixed ~10.2 ms safety-net overhead, which falls from over 100% of total runtime at $N = 3$ to under 1.5% at $N = 10$. Aside from the $N = 10$ exception above, the maximum recorded inclusion-disk radius stays small throughout, showing that forward-error bounds remain controlled and reliability stays high as harmonic order increases. Fig. 6 gives further examples of randomly generated closed curves, at various harmonic orders, with self-intersections detected by the proposed method.

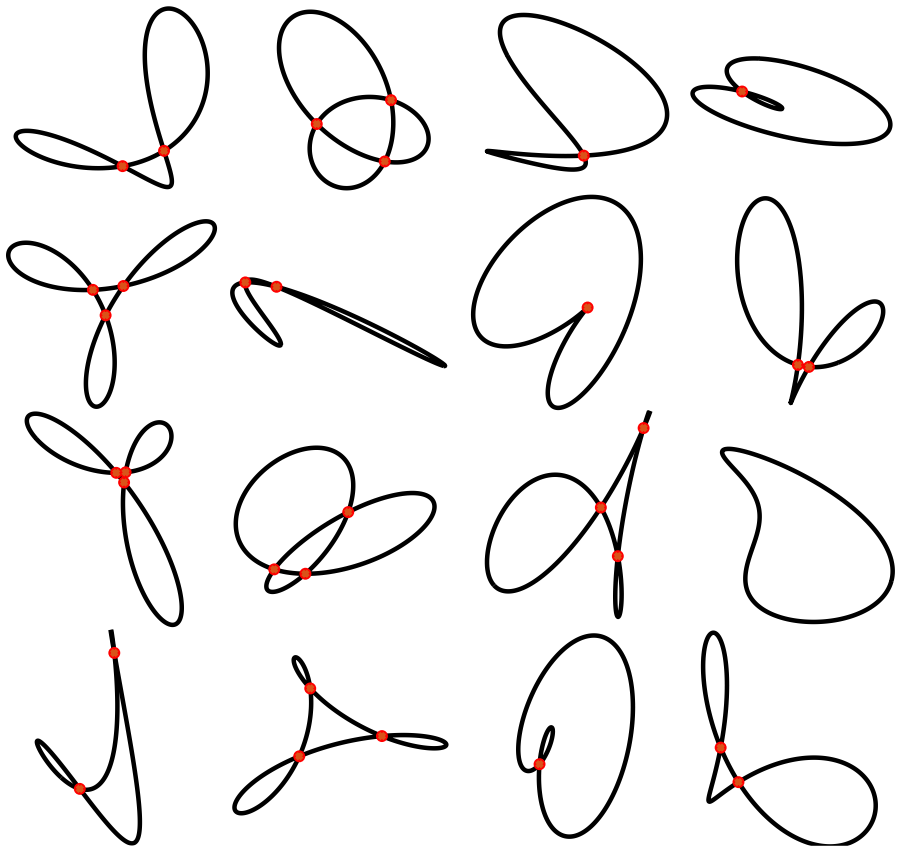

**Fig. 6** Gallery of randomly generated self-intersecting curves with detected self-intersections

## 3.2 Barrier Validation

We validate the proposed barrier within an optimization scheme by deforming a non-self-intersecting, oval-shaped seed curve toward an eight-shaped self-intersecting target, while minimizing the cost function

$$j(\boldsymbol{\theta}) = \lambda\left(\frac{\varphi(\boldsymbol{\theta})}{\varphi(\boldsymbol{\theta}_0)}\right) + \int_0^1 \left(x_{\boldsymbol{\theta}}(s) - x_f(s)\right)^2 + \left(y_{\boldsymbol{\theta}}(s) - y_f(s)\right)^2 ds \qquad (15)$$

using the Parallelized Complex optimization method (Namani Koureh et al. 2026), where $\lambda$ is a scaling factor, $\boldsymbol{\theta}_0$ is the Fourier coefficients of the initial curve, and $\boldsymbol{\tau}_f$ is the target curve. Fig. 7 shows the deformation toward, $\boldsymbol{\tau}_f$ starting from the same $\boldsymbol{\theta}_0$ for $\lambda = 0$ and $\lambda = 0.1$.

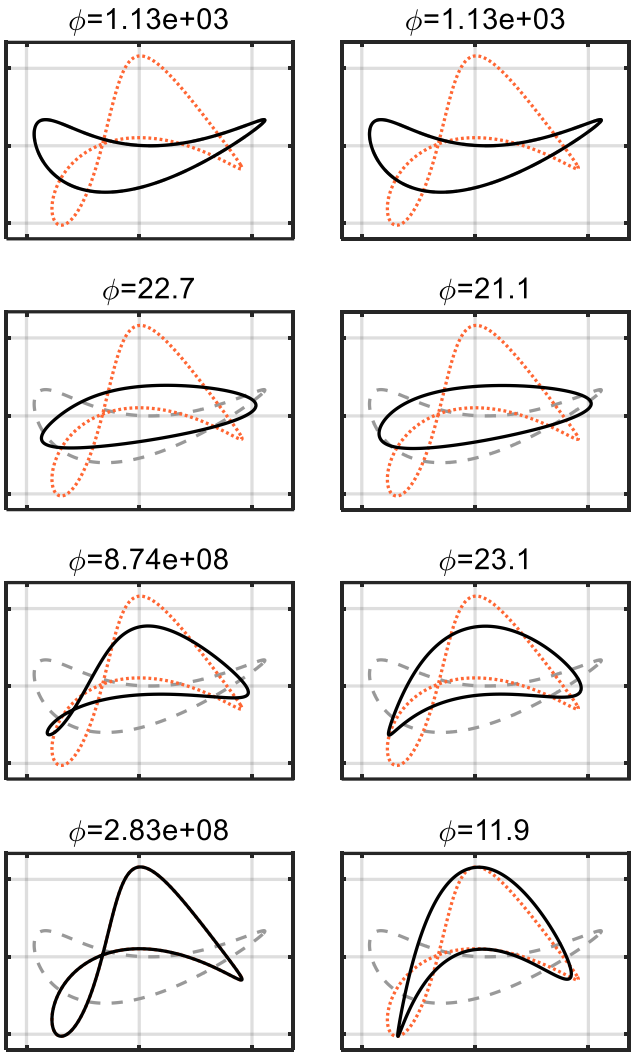


**Fig. 7** Linearly spaced snapshots of the optimization for $\lambda = 0$ (left column) and $\lambda = 0.1$ (right column): initial (dash), target (dot), current state (solid), and recorded value of the barrier ($\phi$) at each stage

With $\lambda = 0$, the curve converges exactly to the self-intersecting target, and $\varphi(\boldsymbol{\theta})$ climbs. For $\lambda = 0.1$, the optimizer instead converges to a simple, non-self-intersecting curve that closely matches the target's overall extent.

We study the effect of $\lambda$ by doing a $\lambda$ sweep and plotting the resulting optimized curves in Fig. 8. This reveals a failure mode, generally unimportant in practice: as $\lambda \to \infty$, the optimal curve collapses to a degenerate straight line, since such a configuration cannot self-intersect. To guard against this, we introduce:

$$j_{\mathrm{d}} = \left( \sum_{i=1}^{N} i \left( a_{x_i} b_{y_i} - a_{y_i} b_{x_i} \right)^2 \right)^{-1} \tag{16}$$

Equation (16) complements the self-intersection barrier by guarding against this distinct failure mode: collapse to a zero-area, dimensionally degenerate configuration. By Green's theorem, the signed area enclosed by the Fourier-parametrized curve equals $\Sigma_{\mathrm{i}} \pi i \left( a_{x_i} b_{y_i} - a_{y_i} b_{x_i} \right)$. Equation (16) vanishes only when every term in this sum vanishes individually, forcing the enclosed area to zero. Two configurations reach zero net area this way: two or more lobes of canceling signed orientation, or genuine collapse onto a lower-dimensional set with no enclosed area at all. The first case requires the curve to self-intersect where the lobes meet, so $\varphi(\boldsymbol{\theta})$ already excludes it before Equation (16) plays any role. Equation (16) is needed only for

the second case, which $\varphi(\boldsymbol{\theta})$ does not cover. Since any genuine simple closed curve encloses strictly positive area by the Jordan Curve Theorem, no valid, non-self-intersecting configuration can ever drive this penalty to infinity — only large.

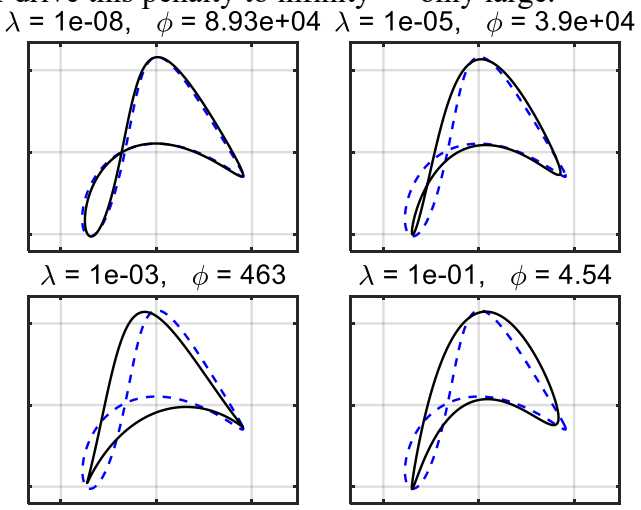


**Fig. 8** Barrier value $\varphi$ and optimization results for a given $\lambda$

As a final test, we compare our barrier against the tangent-point energy $E(\boldsymbol{\theta})$ of Yu et al. (2021), which replaces $\frac{1}{\varphi(\boldsymbol{\theta})}$ in problem (15). For planar vectors $\boldsymbol{p}_1$ and $\boldsymbol{p}_2$ on the curve with unit tangent $T(\boldsymbol{p}_1)$, the barrier $E(\boldsymbol{\theta})$ is calculated via summing over sampling quadrature points spaced $\Delta s$ on the curve as:

$$r(\boldsymbol{p}_1, \boldsymbol{p}_2) = \frac{(\boldsymbol{p}_1 - \boldsymbol{p}_2)^2}{2|T(\boldsymbol{p}_1) \times (\boldsymbol{p}_1 - \boldsymbol{p}_2)|} \quad , \quad E(\boldsymbol{\theta}) = \sum_i \sum_{j \neq i} r\left(\boldsymbol{p}_{1_i}, \boldsymbol{p}_{2_j}\right)^{-p} \Delta s^2 \tag{17}$$

Here, $\boldsymbol{a} \times \boldsymbol{b} := a_1 b_2 - a_2 b_1$ denotes the scalar (perp-dot) cross product for planar vectors $\boldsymbol{a} = (a_1, a_2)$ and $\boldsymbol{b} = (b_1, b_2)$. Unlike $\varphi(\boldsymbol{\theta})$, which certifies self-intersection through an algebraically grounded condition, $E(\boldsymbol{\theta})$ measures purely geometric proximity: it grows large whenever any two sampled points lie close together relative to the local curvature, whether or not they mark a genuine self-intersection. We select a two-harmonic Fourier curve with coefficients:

$$a_x = [0.75 \;\; 0.15] \quad , \quad a_y = [0.12 \;\; -0.15]$$
$$b_x = [0.5 \;\; t] \quad , \quad b_y = [0.55 \;\; 0]$$

Here, we sweep $0 \leq t \leq 1.5$, changing target shape $\boldsymbol{\tau_f}$ (Fig. 9a), recording the values of the barrier functions $\varphi(\boldsymbol{\theta})$ and $E(\boldsymbol{\theta})$ throughout the sweep (Fig. 9b), and solving the optimization problem using at $t = 0.95$ as illustrated in Fig. 9c. Additionally, Table 2 compares results of the optimization problem using 16 sampling points for the $E$ barrier.

**Table 2** Optimization comparison: $\varphi$-barrier vs. tangent-point-energy barrier, same target, initial guess, and optimizer

| Metric | $\varphi$-barrier | $E$-barrier | Ratio $(\frac{E}{\varphi})$ |
|---|---|---|---|
| Function evaluations | 1823 | 31734 | 17.4 |
| Runtime (s) | 2.5 | 93 | 37.2 |
| Final cost | 0.00357427 | 0.0867745 | 24.27 |

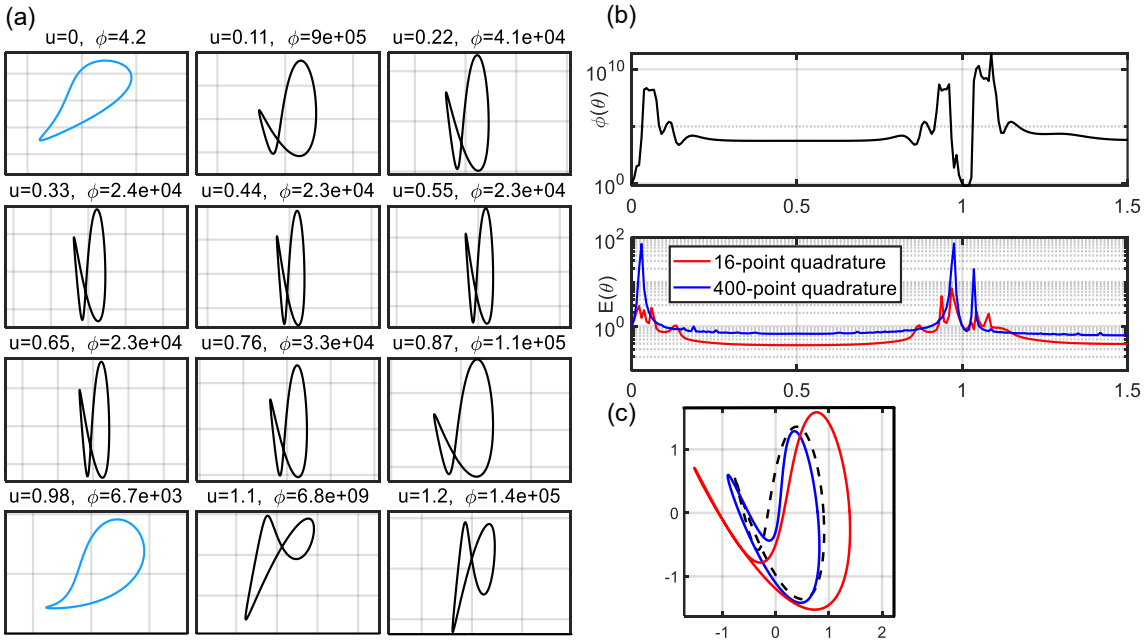


**Fig. 9** (a) Sweep of target curve, (b) Comparison of barriers $\varphi(\boldsymbol{\theta})$ and $E(\boldsymbol{\theta})$ throughout the sweep, and (c) optimization results of proposed (blue) and $E(\boldsymbol{\theta})$ (red) for $t = 0.95$ (dashed)

As Fig. 9a shows, sweeping $0 \leq t \leq 1.5$, the target curve starts in a non-self-intersecting configuration, enters a self-intersecting regime, briefly reverts to non-self-intersecting near $t = 1$, then resumes a self-intersecting geometry. Both the proposed barrier $\varphi(\boldsymbol{\theta})$ and the heuristic barrier from Yu et al. (2021) spike sharply at the transitions between the two regimes (Fig. 9b). However, $E(\boldsymbol{\theta})$ with only 16 sampling points shows a lower peak than the 400-point grid, making self-intersection detection less distinct, as expected.

$\varphi(\boldsymbol{\theta})$ maintains a consistently strong signal throughout $0.1 \leq t \leq 1$, the self-intersecting regime, because of its algebraic formulation, while $E(\boldsymbol{\theta})$ gradually decays because of its grid-based and heuristic nature. $E(\boldsymbol{\theta})$ also fails to capture the brief interval near $t = 1$ where the target curve has no self-intersections; $\varphi(\boldsymbol{\theta})$, in contrast, drops sharply to near zero there and recovers for $t \geq 1$ as self-intersection returns. At $t = 0.95$ (Table 2), $\varphi(\boldsymbol{\theta})$ resolves the optimization significantly faster, needs fewer function evaluations, and tracks the target curve's silhouette more closely, and detects self-intersection more sharply in comparison.

Heuristic barriers such as $E(\boldsymbol{\theta})$ share three fundamental shortcomings. First, because they rely on spatial grids, they are highly sensitive to grid density: raising the resolution from 16 to 400 points heavily burdens computation, requiring over 10 minutes of runtime. Second, without algebraic grounding, they miss self-intersecting regions entirely through sampling aliasing — they detect initial boundary crossings quickly but give little signal within the self-intersecting domain itself. Third, $E(\boldsymbol{\theta})$ is extremely sensitive to the scaling parameter $\lambda$: raising $\lambda$ from 1.5 (Fig. 9c) to 3 pushes the final geometry toward an oval shape, a distortion entirely absent from $\varphi(\boldsymbol{\theta})$. These unpredictable dynamics make heuristic barriers hard to analyze theoretically in large-scale, complex engineering problems.

# 4. Conclusion

We proposed a continuous shape-optimization barrier built on an algebraic self-intersection condition for closed, truncated Fourier-series curves. We substituted the Fourier representation into the self-intersection condition, dropped the trivial solution, and reduced the trigonometric condition to a common-root problem for two bivariate polynomials. We expanded these polynomials in the Chebyshev basis, reduced them via Bézout elimination, and reduced the problem further to finding the singularities of a resultant matrix. We linearized this matrix as a

colleague-pencil generalized eigenvalue problem (Gnazzo 2020) and assessed its reliability using the forward-error bound that Tisseur (2000) established.

Three failure modes surfaced in the algebraic pipeline: a removable singularity in the auxiliary variable, resultant degeneracy from sparse Fourier coefficients, and an incorrect parameter-recovery formula. We addressed all three. Our final implementation combined eigenvector-based root extraction with a nullspace-dimension check for clustered roots, Newton–Raphson polishing, and a numerical safety net. We verified every candidate intersection against the original curve and rejected the extraneous roots that the elimination step introduced.

We ran four experiments to validate the self-intersection detection approach. First, we tested six benchmark curves with known intersection behavior — tangential intersections, clustered roots, degeneracies, and near-misses — and the method handled every one correctly. Second, we compared the method against MATLAB's polyxpoly and found a close match, achieving roughly 35× faster performance, with the Fourier-based locations converging to the polygonal reference as we increased the harmonic order. Third, we compared the method against the polynomial-curve approach of Pekerman et al. (2008) and observed consistent behavior across truncation orders, which extended the approach to polynomial Bézier curves via Fourier approximation. Fourth, we ran a performance study on 50 random curves at each of eight harmonic orders, which confirmed that the algebraic approach scaled well.

We used the same algebraic construction to build a continuous shape-optimization barrier. It replaced a hard constraints accept/reject behavior with a continuously varying penalty, which made the set $\mathcal{J}$ measurable. With the barrier off, the optimizer converged to the self-intersecting target. With the barrier on, it converged instead to a simple curve that tracked the target while avoiding self-intersection. Finally, we compared our barrier against the heuristic tangent-point-energy barrier of Yu et al. (2021) and achieved faster convergence, fewer function calls, and tighter tracking of the target geometry. Across the sweep into and out of self-intersection, $\varphi$ stayed persistently elevated throughout the self-intersecting region, while the tangent-point energy spiked only briefly and sporadically, leaving most of that region without a strong signal.

In sum, we built a continuous, algebraically grounded self-intersection barrier, ready for direct use in shape-optimization pipelines — one that detected and actively avoided self-intersections rather than flagging them after the fact. It relied on a practical, numerically assessable detection primitive for Fourier-represented closed curves, which supplied the algebraic condition on which we built the barrier.

## Statements and Declarations

**Competing Interests:** The author has no competing interests to declare that are relevant to the content of this article.

**Funding:** The author received no financial support or funding for the research, authorship, and/or publication of this article.

**Data and Code/Software Availability:** No external datasets were analyzed or generated during the current study. The source code for the barrier cost function and root detection algorithm developed in this study has been submitted as supplementary source files with the manuscript. Interested parties may also contact the corresponding author via email for inquiries regarding the code or computational implementations.

**Authors' Contributions:** The author confirms sole responsibility for the conceptualization, mathematical formulation, algorithm design, software implementation of the barrier cost function and root detection algorithm, and writing of this manuscript.

**Ethics Approval:** Not applicable — this study did not involve human participants, human data, or animal subjects.

**Use of Large Language Models:** editing language, ordering references, formatting, extracting DOIs, catching notation errors, and finding MSC codes

## References

Amiraslani, A., Corless, R.M., Lancaster, P.: Linearization of matrix polynomials expressed in polynomial bases. IMA J. Numer. Anal. 29(1), 141–157 (2009). doi.org/10.1093/imanum/drm051

Bini, D.A.: Numerical computation of polynomial zeros by means of Aberth's method. Numer. Algorithms 13, 179–200 (1996). doi.org/10.1007/BF02207694

Bini, D.A., Fiorentino, G.: Design, analysis, and implementation of a multiprecision polynomial rootfinder. Numer. Algorithms 23, 127–173 (2000). doi.org/10.1023/A:1019199917103

Bini, D.A., Di Marco, G.: Computing curve intersection by means of simultaneous iterations. Numer. Algorithms 43(2), 151–175 (2006). doi.org/10.1007/s11075-006-9048-0

Bonnet, M., Liu, R., Veerapaneni, S.: Shape optimization of Stokesian peristaltic pumps using boundary integral methods. Adv. Comput. Math. 46, 18 (2020). doi.org/10.1007/s10444-020-09761-7

Boyd, J.P.: Chebyshev and Fourier Spectral Methods, 2nd edn. Dover Publications, Mineola (2002)

Buchberger, B.: Introduction to Gröbner bases. In: Buchberger, B., Winkler, F. (eds.) Gröbner Bases and Applications, pp. 3–31. Cambridge University Press, Cambridge (1998)

Chen, Y., Li, J., Song, C., et al.: Aerodynamic shape optimization with effective deep-learning-based geometric filtering: from a cylinder to a wing. Struct. Multidiscip. Optim. 69, 25 (2026). doi.org/10.1007/s00158-025-04235-0

Chionh, E.H., Goldman, R.N., Zhang, X.: Fast computation of the Bezout and Dixon resultant matrices. J. Symb. Comput. 33(1), 13–29 (2002). doi.org/10.1006/jsco.2001.0462

Cox, D.A., Little, J., O'Shea, D.: Ideals, Varieties, and Algorithms, 4th edn. Springer, Cham (2015)

Deng, F., Xue, C., Qin, N.: Parameterizing airfoil shape using aerodynamic performance parameters. AIAA J. 60(7), 4399–4412 (2022). doi.org/10.2514/1.J061464

Eidelman, Y., Gemignani, L., Gohberg, I.: Efficient eigenvalue computation for quasiseparable Hermitian matrices under low rank perturbations. Numer. Algorithms 47, 253–273 (2008). doi.org/10.1007/s11075-008-9172-0

Fasino, D., Gemignani, L.: Direct and inverse eigenvalue problems for diagonal-plus-semiseparable matrices. Numer. Algorithms 34, 313–324 (2003). doi.org/10.1023/B:NUMA.0000005402.66868.af

Frederix, K., Delvaux, S., Van Barel, M.: An algorithm for computing the eigenvalues of block companion matrices. Numer. Algorithms 62, 261–287 (2013). doi.org/10.1007/s11075-012-9579-5

Garcke, H., Hinze, M., Kahle, C., Lam, K.F.: A phase field approach to shape optimization in Navier–Stokes flow with integral state constraints. Adv. Comput. Math. 44, 1345–1383 (2018). doi.org/10.1007/s10444-018-9586-8

Gnazzo, M.: Structured Matrix Computations and Applications in Polynomial Systems. PhD thesis, University of Pisa, Pisa (2020)

Good, I.J.: The colleague matrix, a Chebyshev analogue of the companion matrix. Q. J. Math. 12(1), 61–68 (1961). doi.org/10.1093/qmath/12.1.61

Harris, G., Martin, C.: The roots of a polynomial vary continuously as a function of the coefficients. Proc. Am. Math. Soc. 100(2), 390–392 (1987). doi.org/10.1090/S0002-9939-1987-0884486-8

Hiptmair, R., Scarabosio, L., Schillings, C., Schwab, C.: Large deformation shape uncertainty quantification in acoustic scattering. Adv. Comput. Math. 44, 1475–1518 (2018). doi.org/10.1007/s10444-018-9594-8

Horn, R.A., Johnson, C.R.: Matrix Analysis, 2nd edn. Cambridge University Press, Cambridge (2012). doi.org/10.1017/9781139020411

Jónsson, G., Vavasis, S.: Accurate solution of polynomial equations using Macaulay resultant matrices. Math. Comput. 74(249), 221–262 (2005). doi.org/10.1090/S0025-5718-04-01722-3

Kalmykov, S., Kovalev, L.V.: A sharp bound on the number of self-intersections of a trigonometric curve. J. Math. Anal. Appl. 543(2), 128995 (2025). doi.org/10.1016/j.jmaa.2024.128995

Lawrence, P.W., Corless, R.M.: Stability of rootfinding for barycentric Lagrange interpolants. Numer. Algorithms 65, 447–464 (2014). doi.org/10.1007/s11075-013-9770-3

Lu, C., Xu, L., Guo, A., et al.: Shape optimization of floating bridge pontoons with mooring constraints under wave actions. Struct. Multidiscip. Optim. 68, 53 (2025). doi.org/10.1007/s00158-025-03985-1

Manocha, D., Demmel, J.: Algorithms for intersecting parametric and algebraic curves I: simple intersections. ACM Trans. Graph. 13(1), 73–100 (1994). doi.org/10.1145/174462.174617

Nakatsukasa, Y., Noferini, V., Townsend, A.: Computing the common zeros of two bivariate functions via Bézout resultants. Numer. Math. 129(1), 181–209 (2015). doi.org/10.1007/s00211-014-0649-4

Namani Koureh, S., Shahverdi, H., Kakhsaz Malayeri, N., Amoozgar, M.: Parallelized complex method for multidisciplinary optimization of aeroelastic performance in thin-walled wing–engine system using bidirectional curvilinear fiber pathing. Struct. Multidiscip. Optim. 69, 157 (2026). doi.org/10.1007/s00158-026-04311-z

Nocedal, J., Wright, S.J.: Numerical Optimization. Springer, New York (2006). doi.org/10.1007/978-0-387-40065-5

Noferini, V., Townsend, A.: Numerical instability of resultant methods for multidimensional rootfinding. SIAM J. Numer. Anal. 54(2), 719–743 (2016). doi.org/10.1137/15M1022513

Norato, J.A.: Topology optimization with supershapes. Struct. Multidiscip. Optim. 58, 415–434 (2018). doi.org/10.1007/s00158-018-2034-z

Paul, E.J., Landreman, M., Antonsen, T.: Gradient-based optimization of 3D MHD equilibria. J. Plasma Phys. 87(2), 905870215 (2021). doi.org/10.1017/S0022377821000283

Pekerman, D., Elber, G., Kim, M.S.: Self-intersection detection and elimination in freeform curves and surfaces. Comput. Aided Des. 40(2), 150–159 (2008). doi.org/10.1016/j.cad.2007.10.004

Piegl, L., Tiller, W.: The NURBS Book, 2nd edn. Springer, Berlin Heidelberg (1997)

Sassen, J., Schumacher, H., Rumpf, M., Crane, K.: Repulsive shells. ACM Trans. Graph. 43(4), Article 140 (2024). doi.org/10.1145/3658174

Sommese, A.J., Wampler, C.W.: The Numerical Solution of Systems of Polynomials Arising in Engineering and Science. World Scientific, Singapore (2005). doi.org/10.1142/5763

Sorber, L., Van Barel, M., De Lathauwer, L.: Numerical solution of bivariate and polyanalytic polynomial systems. SIAM J. Numer. Anal. 52(4), 1551–1572 (2014). doi.org/10.1137/130932387

Telen, S., Van Barel, M., Verschelde, J.: A robust numerical path tracking algorithm for polynomial homotopy continuation. SIAM J. Sci. Comput. 42(6), A3610–A3637 (2020). doi.org/10.1137/19M1288036

Tisseur, F.: Backward error and condition of polynomial eigenvalue problems. Linear Algebra Appl. 309(1), 339–361 (2000). doi.org/10.1016/S0024-3795(99)00063-4

Trefethen, L.N.: Spectral Methods in MATLAB. SIAM, Philadelphia (2000). doi.org/10.1137/1.9780898719598

Trefethen, L.N.: Approximation Theory and Approximation Practice. SIAM, Philadelphia (2013). doi.org/10.1137/1.9781611972413

van Dijk, N.P., Maute, K., Langelaar, M., et al.: Level-set methods for structural topology optimization: a review. Struct. Multidiscip. Optim. 48, 437–472 (2013). doi.org/10.1007/s00158-013-0912-y
Walker, S.W.: Shape optimization of self-avoiding curves. J. Comput. Phys. 311, 275–298 (2016). doi.org/10.1016/j.jcp.2016.02.011
Wu, Z., Jiang, W., Luo, H., Cheng, L.: A novel self-intersection penalty term for statistical body shape models and its applications in 3D pose estimation. Appl. Sci. 9(3), 400 (2019). doi.org/10.3390/app9030400
Yu, C., Schumacher, H., Crane, K.: Repulsive curves. ACM Trans. Graph. 40(2), Article 10 (2021). doi.org/10.1145/3439429
Zahn, C.T., Roskies, R.Z.: Fourier descriptors for plane closed curves. IEEE Trans. Comput. C-21(3), 269–281 (1972). doi.org/10.1109/T-C.1972.223549
Zelickman, Y., Guest, J.K.: Introducing a general polygonal primitive for feature mapping-based topology optimization. Struct. Multidiscip. Optim. 68, 99 (2025). doi.org/10.1007/s00158-025-04034-7

## **Appendix A** – Bézout Resultant and Colleague-Pencil Linearization

### A.1 Matrix Representation of Polynomials

To compute the coefficients of polynomials $p_1(u,v)$ and $p_2(u,v)$ in terms of the Fourier coefficients of $a_{x_n}, b_{x_n}, a_{y_n}, b_{y_n}$ the following three definitions are introduced.

*Definition* A1*:* The matrix representation F of a polynomial $f(u,v)$ is defined as:

$$f(u,v) = \sum_{i=0}^{n_u}\sum_{j=0}^{n_v} f_{ij}u^i v^j \quad \leftrightarrow \quad \mathrm{F} = \left[f_{(i+1)(j+1)}\right] \tag{A1}$$

*Definition* A2: Multiplying two polynomials $f(u,v)$ and $g(u,v)$ with matrix representations F and G, respectively, results in another polynomial $h(u,v)$, which has the matrix representation H given by:

$$h(u,v) = f(u,v)g(u,v) = \sum_{i=0}^{n_{u_h}}\sum_{j=0}^{n_{v_h}} h_{ij}u^i v^j \quad , \quad \mathrm{H} = \mathrm{F} \star \mathrm{G}$$

where $n_{u_h} = n_{u_f} + n_{u_g}$, $n_{v_h} = n_{v_f} + n_{v_g}$, and $\star$ denotes matrix convolution operation. Similarly, for the addition of $f(u,v)$ and $g(u,v)$, the matrix representation H is given by:

$$h = f + g = \sum_{i=0}^{n_{u_h}}\sum_{j=0}^{n_{v_h}} h_{ij}u^i v^j \quad , \quad \mathrm{H} = \mathrm{F} \oplus \mathrm{G} \tag{A2}$$

where operation $\oplus$ is defined as:

$$\mathrm{F} \oplus \mathrm{G} = \mathrm{F_p} + \mathrm{G_p}$$

with $\mathrm{F}_p$ and $\mathrm{G}_p$ representing F and G matrices padded with zeros on the right column and bottom row to match the target dimensions defined by $n_{u_h} = \max\left(n_{u_f}, n_{u_g}\right)$ and $n_{v_h} = \max\left(n_{v_f}, n_{v_g}\right)$.

*Definition* A3: The iterative relations for the Chebyshev polynomials of the first and second kind are given by:

$$\begin{aligned} T_n(x) &= 2xT_{n-1}(x) - T_{n-2}(x) \quad , \quad \begin{cases} T_0(x) = 1 \\ T_1(x) = x \end{cases} \\ U_n(x) &= 2xU_{n-1}(x) - U_{n-2}(x) \quad , \quad \begin{cases} U_0(x) = 1 \\ U_1(x) = 2x \end{cases} \end{aligned} \tag{A3}$$

Using (A3), the matrix representations of the first and second kind Chebyshev polynomials are:

$$\mathrm{T}_n = 2[0 \quad 1] * \mathrm{T}_{n-1} \oplus (-\mathrm{T}_{n-2}) \quad , \quad \begin{cases} T_0 = 1 \\ T_1 = [0,1] \end{cases}$$

$$\mathrm{U}_n = 2[0 \quad 1] * \mathrm{U}_{n-1} \oplus (-\mathrm{U}_{n-2}) \quad , \quad \begin{cases} U_0 = 1 \\ U_1 = [0,2] \end{cases}$$

Combining Definitions A1 through A3 yields the matrix representations of the polynomials $A_x$, $B_x$, $A_y$, and $B_y$ given in (5) as:

$$\mathrm{A}_x = -\bigoplus_{n=1}^{N} a_{x_n} \mathrm{U}_{n-1} \star (\mathrm{U}_{n-1})^{\mathrm{T}} \qquad \mathrm{B}_x = \bigoplus_{n=1}^{N} b_{x_n} \mathrm{T}_n \star (\mathrm{U}_{n-1})^{\mathrm{T}}$$

$$\mathrm{A}_y = -\bigoplus_{n=1}^{N} a_{y_n} \mathrm{U}_{n-1} \star (\mathrm{U}_{n-1})^{\mathrm{T}} \qquad \mathrm{B}_y = \bigoplus_{n=1}^{N} b_{y_n} \mathrm{T}_n \star (\mathrm{U}_{n-1})^{\mathrm{T}}$$

Consequently, the matrix representations of polynomials $p_1(u,v)$ and $p_2(u,v)$ are given by:

$$\begin{aligned} \mathrm{P}_1 &= -\mathrm{A}_y \star \mathrm{B}_x \oplus \mathrm{A}_x \star \mathrm{B}_y \\ \mathrm{P}_2 &= \mathrm{B}_x \star \mathrm{B}_x \oplus [-1,0,1] \star \mathrm{A}_x \star \mathrm{A}_x \end{aligned} \tag{A4}$$

thereby completing the derivation of matrix representations for polynomials $p_1(u,v)$ and $p_2(u,v)$.

## A.2 Monomial to Chebyshev Transfer

To transfer the monomial matrix representation of a bivariate polynomial introduced by Equation (A1) to a Chebyshev representation of Equation (7), the monomial-based representation of $T_i(x)$ is padded with trailing zeros so it can be concatenated for various $i$ as:

$$\mathrm{C}_\mathrm{n} = [\mathrm{T}_0^{\mathrm{T}} \; \mathrm{T}_1^{\mathrm{T}} \cdots \mathrm{T}_\mathrm{n}^{\mathrm{T}}] = [\mathrm{T}_0 \; \mathrm{T}_1 \cdots \mathrm{T}_\mathrm{n}]^{\mathrm{T}}$$

Consequently, $u^i$ and $v^j$ have the matrix representations:

$$[u^i] = \mathrm{C}_i^{-1} \begin{bmatrix} 0 \\ \vdots \\ 1 \end{bmatrix}_{i \times 1} \quad , \quad [v^j] = \left( \mathrm{C}_j^{-1} \begin{bmatrix} 0 \\ \vdots \\ 1 \end{bmatrix}_{j \times 1} \right)^{\mathrm{T}}$$

Since $\mathrm{C}^{-1} = \mathrm{C}^{\mathrm{T}}$, the Chebyshev matrix representation of a monomial matrix $\mathrm{P}_{ij}$ is given by:

$$\mathrm{P}_{ij}^{(\mathrm{Ch})} = \mathrm{C}^{\mathrm{T}} \mathrm{P}_{ij} \mathrm{C}$$

yielding the desired transformation law between monomial and Chebyshev representations of bivariate polynomials.

## A.3 Bézout Resultant

With the polynomials expressed in the Chebyshev basis, the Bézout resultant is constructed directly from these. The polynomials $p_1(u,v)$ and $p_2(u,v)$ in the Chebyshev basis are given by:

$$p_1(u,v) = \sum_{i=0}^{2N-1} \sum_{j=0}^{2N-2} {p_1}_{ij}^{(\mathrm{ch})} T_i(u) T_j(v) \quad , \quad p_2(u,v) = \sum_{i=0}^{2N} \sum_{j=0}^{2N-2} {p_2}_{ij}^{(\mathrm{ch})} T_i(u) T_j(v)$$

Assuming $v$ as hidden variable gives:

$$p_{1_v}(u) = \sum_{i=0}^{2N-1} \alpha_{1_i}(v)\, T_i(u) \quad , \quad p_{2_v}(u) = \sum_{i=0}^{2N} \alpha_{2_i}(v) T_i(u)$$

where:

$$\alpha_{1_i}(v) = \sum_{j=0}^{2N-2} \mathrm{C}_{ij}^{(1)} T_j(v) \quad , \quad \alpha_{2_i}(v) = \sum_{j=0}^{2N} \mathrm{C}_{ij}^{(2)} T_j(v)$$

The Bézout resultant polynomial $b(t,u)$ is defined as:

$$b(t,u) := \frac{p_{1_v}(u)p_{2_v}(t) - p_{1_v}(t)p_{2_v}(u)}{u-t} \quad , \quad t \neq u \tag{A5}$$

Since the numerator of $b(t,u)$ is a bivariate polynomial and vanishes at $u = t$, it must have a factor of $(u-t)$. Hence, the function $b(t,u)$ can be extended continuously on the domain $u = t$ as:

$$b(u=t) := \lim_{t\to u} b(t,u) = p_{2_v}(t)\frac{\mathrm{d}p_{1_v}(t)}{\mathrm{d}t} - p_{1_v}(t)\frac{\mathrm{d}p_{2_v}(t)}{\mathrm{d}t}$$

Consequently, $b(t,u)$ is a polynomial function in $u$, $t$, and the hidden variable $v$, and can be represented by a trivariate Chebyshev polynomial:

$$b(t,u) = \sum_{k=0}^{4N-4}\sum_{j=0}^{2N-1}\sum_{i=0}^{2N-1} b_{ijk}T_i(u)T_j(t)T_k(v)$$

where entries of tensor $b_{ijk}$ can be computed using a three-fold Chebyshev interpolation on Chebyshev–Lobatto points via the Fast Fourier Transform (FFT). If $(u^*, v^*)$ is a common root of the system, (A5) shows that the numerator of $b$ vanishes for every $t$. Hence, the Bézout resultant matrix $\mathrm{B}\big(p_{1_v}, p_{2_v}\big)$, which is defined as:

$$\mathrm{B}\big(p_{1_v}, p_{2_v}\big) := \sum_{k=0}^{4N-4} \mathrm{A}_k T_k(v) \quad , \quad \mathrm{A}_k = b_{ijk} \tag{A6}$$

must be singular for $v = v^*$. Additionally, using definition (A6), the partial derivative of B with respect to $v$ is given by:

$$\frac{\partial \mathrm{B}}{\partial v} = \sum_{k=1}^{4N-4} k\mathrm{A}_k U_{k-1}(v) \quad , \quad \mathrm{A}_k = b_{ijk} \tag{A7}$$

The partial derivative (A7) is utilized in the computation of the forward-error bound.

### A.4 Colleague-Pencil Matrix Linearization

Solving $\det\big(\mathrm{B}(v)\big) = 0$ directly would require finding the roots of a high-degree polynomial in $v$. The colleague-pencil construction below avoids this by converting the determinant condition into a standard generalized eigenvalue problem, solvable with numerically robust linear algebra routines.

Matrices $\mathcal{L}_0$ and $\mathcal{L}_1$ are given by:

$$\mathcal{L}_0 = \begin{bmatrix} 0 & \mathrm{I} & 0 & & \\ \mathrm{I} & 0 & \mathrm{I} & & \\ & \ddots & \ddots & \ddots & \\ & & \mathrm{I} & 0 & \mathrm{I} \\ -\mathrm{A}_0 & \cdots & -\mathrm{A}_{d-3} & \mathrm{A}_d - \mathrm{A}_{d-2} & -\mathrm{A}_{d-1} \end{bmatrix} \quad , \quad \mathcal{L}_1 = \begin{bmatrix} \mathrm{I} & & & & \\ & 2\mathrm{I} & & & \\ & & \ddots & & \\ & & & 2\mathrm{I} & \\ & & & & \mathrm{A}_d \end{bmatrix}$$

where $\mathcal{L}_0$ and $\mathcal{L}_1$ components are constant blocks, in contrast to the entries of the Bézout resultant matrix $\mathrm{B}(v)$. Consider the generalized eigenvalue problem for the pencil matrix $\mathcal{L}(\lambda) = \mathcal{L}_0 - \lambda\mathcal{L}_1$:

$$(\mathcal{L}_0 - \lambda\mathcal{L}_1)\boldsymbol{v} = 0 \tag{A8}$$

where $\boldsymbol{v} = [\boldsymbol{v}_0^{\mathrm{T}}\ \boldsymbol{v}_1^{\mathrm{T}} \cdots \boldsymbol{v}_d^{\mathrm{T}}]$, $\boldsymbol{v}_i$s are column vectors, and $\boldsymbol{v}_0$ is the eigenvector of matrix $\mathrm{B}(v)$ for a common root $v = v^*$. Using the explicit block structures of $\mathcal{L}_0$ and $\mathcal{L}_1$ and writing out the first block row of vector Equation (A8) gives:

$$\boldsymbol{v}_1 - \lambda\boldsymbol{v}_0 = 0$$

Multiplying the intermediate rows similarly results in:

$$\boldsymbol{v}_{i+1} = 2\lambda\boldsymbol{v}_i - \boldsymbol{v}_{i-1}$$

And finally, the bottom row gives:

$$-\mathrm{A}_0\boldsymbol{v}_0 - \mathrm{A}_1\boldsymbol{v}_1 - \cdots - \mathrm{A}_{d-3}\boldsymbol{v}_{d-2} + (\mathrm{A}_d - \mathrm{A}_{d-2})\boldsymbol{v}_{d-1} - \mathrm{A}_{d-1}\boldsymbol{v}_d - \lambda(2\mathrm{A}_d\boldsymbol{v}_d) = 0$$

Defining $\boldsymbol{v}_i := T_i(\lambda)\boldsymbol{v}_0$ and substituting it in the last row for various $i$ gives:

$$-\left(\sum_{i=0}^{d-1} \mathrm{A}_i\, T_i(\lambda)\right)\boldsymbol{v}_0 + \mathrm{A}_d T_{d-2}(\lambda)\boldsymbol{v}_0 - \mathrm{A}_{d-2} T_{d-1}(\lambda)\boldsymbol{v}_0 - 2\lambda \mathrm{A}_d T_d(\lambda)\boldsymbol{v}_0 = 0 \qquad \text{(A9)}$$

Using the Chebyshev property $T_{d-2}(\lambda) - 2\lambda T_d(\lambda) = -T_d(\lambda)$ simplifies (A9) as:

$$\mathrm{A}_d\big(T_{d-2}(\lambda) - 2\lambda T_d(\lambda)\big)\boldsymbol{v}_0 = -\mathrm{A}_d T_d(\lambda)\boldsymbol{v}_0$$

Combining all terms yields:

$$\left(\sum_{i=0}^{d} \mathrm{A}_i\, T_i(\lambda)\right)\boldsymbol{v}_0 = 0$$

Showing that $\lambda = v^*$.

# **Appendix B** – Self-Intersection Detection Algorithm

This appendix details the full self-intersection detection pipeline validated in Section 3.1. Fig. 10 summarizes the algorithm's routing between three paths — the Pure Algebraic Path (PAP), the Newton Polished Algebraic Path (NPAP), and the Numerical Fallback Path (NFBP) — which together address several numerical pitfalls and degenerate cases: (1) avoiding division by zero during degenerate step evaluations; (2) distinguishing near-misses from genuine self-intersections; (3) handling roots with high multiplicities; (4) mitigating errors from finite machine precision; (5) applying distinct tolerances for accepting or rejecting candidate roots; and (6) rejecting spurious roots.

As illustrated, the overall algorithm routes between these three paths rather than committing to one in advance. Detection begins with the algebraic construction of Section 2.1; if the resulting resultant matrices are non-degenerate, root extraction proceeds along the Pure Algebraic Path, with Newton polishing applied wherever numerical tolerances alone cannot resolve a genuine intersection from a near-miss, yielding the Newton Polished Algebraic Path. Should the resultant construction itself degenerate — for instance under sparse Fourier coefficients — the algorithm instead falls back to the Numerical Fallback Path, substituting a full grid search for the failed algebraic step while retaining the same Newton-polishing and verification stages downstream in rare cases. This branching ensures that a single degenerate configuration does not cause the detection procedure to fail outright, at the cost of the fallback path's higher computational expense relative to the algebraic route. The proposed self-intersection detection algorithm (full pipeline) comprises twelve steps. For each step, its purpose, thresholds, safety margins, potential failure modes, and corresponding mitigation strategies are described. A MATLAB implementation of the root-finding pipeline is provided in Online Resource 2.

**Step 1:** *Input and Coefficient Normalization*

The Fourier coefficients $a_{x_n}, b_{x_n}, a_{y_n}, b_{y_n}$ are each normalized by their maximum absolute value.

**Step 2:** *Constructing the Bivariate System*

Sum-to-product identities and a Chebyshev substitution construct $\mathrm{A}_x, \mathrm{B}_x, \mathrm{A}_y, \mathrm{B}_y$ and eliminate the auxiliary variable $w$, yielding $\mathrm{P}_1$ and $\mathrm{P}_2$ in Equation (A4). If a coordinate's Fourier series has a single dominant harmonic, one or more of these matrices can collapse to single-term polynomials, causing $p_1(u, v)$ and $p_2(u, v)$ to share a common factor — a degeneracy caught numerically in Step 4.

**Step 3:** *Chebyshev Basis Conversion and Bézout Matrix Construction via FFT*

The Bézout matrix polynomial $\mathrm{B}(v) = \sum \mathrm{A}_k\, T_k(v)$ is assembled via FFT on Chebyshev–Lobatto nodes (Trefethen 2000; Trefethen 2013), avoiding the ill-conditioned Vandermonde matrices used by Gnazzo (2020). Since $u$ and $v$ enter symmetrically, either could be eliminated first; the one with lower degree in $p_1(u, v)$ and $p_2(u, v)$ is chosen, giving a smaller colleague-pencil problem.

**Step 4:** *Degeneracy Screening of the Resultant Tensor*
The coefficient tensor $b_{ijk} = \mathrm{A}_k$ is checked for degeneracy: $\max|b_{ijk}| < 10^{-9}$ (Cox et al. 2015). This threshold sits between the $10^{-13}$ to $10^{-14}$ floating-point noise floor and the $10^{-2}$ to $10^{0}$ magnitude of a genuinely non-degenerate tensor, so it correctly flags structural degeneracy from Step 2 rather than noise. If triggered, the algebraic branch is bypassed in favor of the numerical safety net (Step 11), avoiding a silently empty or erroneous result.
**Step 5:** *Colleague-Pencil Eigenvalue Extraction*
The equation $\det \mathrm{B}(v) = 0$ becomes a generalized eigenvalue problem via the colleague-pencil method ($\mathcal{L}_0 - \lambda\mathcal{L}_1$). Real eigenvalues within $[-1.01, 1.01]$ are retained — a 1% margin absorbing ordinary QZ rounding error without admitting genuinely out-of-range roots. An eigenvalue counts as real if $|\mathrm{Im}(v)| < 10^{-6}$, looser than machine precision because colleague-pencil conditioning worsens as the harmonic cutoff $N$ grows (Noferini and Townsend 2016; Lawrence and Corless 2014; Bini and Fiorentino 2000), which can give genuinely real roots a small spurious imaginary part. The main failure mode follows directly: for large $N$, genuine real roots may be misclassified as complex or fall outside the inclusion disk of Equation (10). Steps 10–11 guard against this.
**Step 6:** *Per-Eigenvalue Diagnostics — Forward-Error Bound and Nullspace Dimension*
For each candidate $v^*$, $\mathrm{B}(v^*)$ is reassembled and its SVD computed, giving two diagnostics: the forward-error bound $r_{\widehat{v^*}}$ (Gnazzo 2020; Tisseur 2000), and the nullspace dimension $d$ — the number of trailing singular values within a factor of 100 of the smallest — a practical heuristic for separating simple roots from repeated ones. A large $r_{\widehat{v^*}}$ flags an intrinsically sensitive, typically near-tangential root rather than a numerical error. Both diagnostics are reported per root as reliability and multiplicity measures, not merely internal checks.
**Step 7:** *Extraction of u — Eigenvector Path and Root-Finding Fallback*
When $d = 1$, $u^*$ is read directly from the right singular vector of smallest singular value, normalized so its first entry is 1 (its second entry gives $u^*$), then verified by requiring both $p_1(u^*, v^*)$ and $p_2(u^*, v^*)$ below $10^{-6}$ — excluding any polynomial whose coefficients are all below $10^{-8}$ as uninformative. When $d > 1$ (a repeated eigenvalue, indicating a higher-order self-intersection), no single eigenvector determines $u^*$ uniquely; root-finding is instead applied to the non-trivial polynomial and validated with the same $10^{-6}$ test. This threshold is tighter than Step 4's since it checks one evaluation rather than accumulated tensor noise, sitting above the $10^{-14}$ noise floor and well below any genuine near-miss scale.
**Step 8:** *Recovery of w and the Auxiliary 0/0 Degeneracy*
$w$ is recovered by calculating $(A_x B_x + A_y B_y)/(A_x^2 + A_y^2)$ at $(u^*, v^*)$, and only accepted if it satisfies $u^2 + w^2 = 1$ within $10^{-3}$ — looser than other checks since division amplifies upstream error, making this only a preliminary test ahead of Step 9's stricter arbitration.
The main failure mode is division by zero: when $A_x = A_y = 0$ ($A_x^2 + A_y^2 < 10^{-8}$, component magnitudes under $\sim 5 \times 10^{-4}$), the formula is undefined. Rather than discarding the root, both signs $w = \pm(1 - u^{*2})^{0.5}$ are retained and passed to Step 9 for final arbitration.
**Step 9:** *Recovery of $s_1$, $s_2$ and Curve Validation*
Equations (8) and (9) convert each root $(u^*, v^*, w^*)$ back to curve parameters $s_1$ and $s_2$, accepted only if $|x(s_1) - x(s_2)|$ and $|y(s_1) - y(s_2)|$ both below $5 \times 10^{-3}$. This is the loosest tolerance in the pipeline, deliberately: loose enough to retain candidates with moderate upstream error for Step 10's Newton polishing to correct, yet tight enough to reject near-misses and extraneous roots left over from Step 2's elimination.
**Step 10:** *Newton Polishing*
Each candidate passing Step 9 is refined via ten Newton–Raphson iterations (B1) with an analytically computed Jacobian.

$$\begin{bmatrix} s_1^{i+1} \\ s_2^{i+1} \end{bmatrix} = \begin{bmatrix} s_1^{i} \\ s_2^{i} \end{bmatrix} - \begin{bmatrix} x'(s_1) & -x'(s_2) \\ y'(s_1) & -y'(s_2) \end{bmatrix}^{-1} \begin{bmatrix} x(s_1^i) - x(s_2^i) \\ y(s_1^i) - y(s_2^i) \end{bmatrix}$$

$$x'(s) = 2\pi \sum_{n=1}^{N} n\left(b_{x_n} \cos 2\pi n s - a_{x_n} \sin 2\pi n s\right) \tag{B1}$$

$$y'(s) = 2\pi \sum_{n=1}^{N} n\left(b_{y_n} \cos 2\pi n s - a_{y_n} \sin 2\pi n s\right)$$

A singular Jacobian — occurring at genuinely tangential self-intersections — is not a failure: the pre-polishing candidate is retained, since the singularity itself confirms tangency. Convergence otherwise confirms a genuine self-intersection, while divergence or oscillation flags a likely near-miss. For closely spaced intersections, this step doubles as a diagnostic: distinct candidates converging to distinct roots confirm nearby intersections, while collapse to a single root attributes their apparent separation to noise.

**Step 11:** *Numeric Safety Net*

A coarse grid search over $s \in [0,1)$ finds closely spaced parameter pairs, clusters nearby seeds, and applies the same Newton polishing (Step 10) to each cluster centroid. Its only limitation is grid resolution: two genuine intersections closer than the grid spacing may merge into one cluster or be missed entirely. This provides a fallback route when the algebraic pipeline fails outright — through complete degeneracy (Step 4) or ill-conditioning (Steps 5–7).

**Step 12:** *Merging and Deduplication*

Candidates from both paths are pooled and deduplicated using two thresholds — $10^{-5}$ in $(u, v, w)$ space and $10^{-3}$ in $(s_1, s_2)$ space — since the nonlinear mapping between them doesn't preserve distances uniformly. Both are large enough to absorb cross-path numerical discrepancies yet smaller than the minimum intersection spacing observed in the validated test cases; curves with more closely spaced intersections would need tighter tolerances.

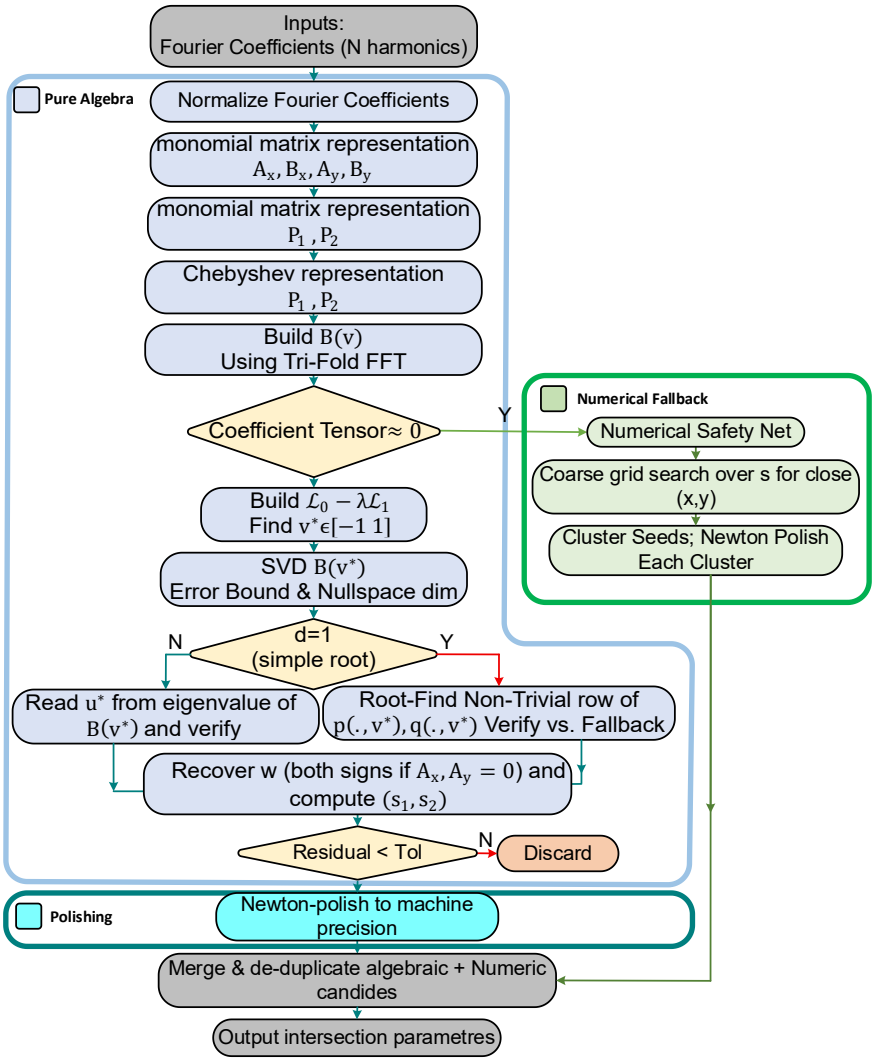


**Fig. 10** Flowchart of the proposed self-intersection detection algorithm (full pipeline), the Pure Algebraic Path (PAP, blue), the Newton Polished Algebraic Path (NPAP, Pure Algebraic Path+ Newton polishing), the Numerical Fallback Path (NFBP, green)