

# Algorithms for Intersecting Parametric and Algebraic Curves II: Multiple Intersections

*Dinesh Manocha*<sup>1</sup>

Department of Computer Science  
Sitterson Hall, CB #3175  
University of North Carolina  
Chapel Hill, NC 27599-3175  
manocha@cs.unc.edu  
(919)962-1749.

*James Demmel*<sup>2</sup>

Computer Science Division and Mathematics Department  
University of California at Berkeley  
Berkeley, CA 94720

---

<sup>1</sup>Supported by IBM Fellowship, David and Lucile Packard Fellowship, National Science Foundation grant ASC-900593k and a Junior Faculty Award.

<sup>2</sup>Supported in part by National Science Foundation grant ASC-9005933 and Subcontract ORA4466.02 to the University of Tennessee (ARPA contract number DAAL03-91-C-0047).

## Abstract

The problem of computing the intersection of parametric and algebraic curves arises in many applications of computer graphics, geometric and solid modeling. Previous algorithms are based on techniques from elimination theory or subdivision and iteration and are typically limited to simple intersections of curves. Furthermore, algorithms based on elimination theory are restricted to low degree curves. This is mainly due to issues of efficiency and numerical stability. In this paper we use elimination theory and express the resultant of the equations of intersection as a matrix determinant. Using matrix computations the algorithm for intersection is reduced to computing eigenvalues and eigenvectors of matrices. We use techniques from linear algebra and numerical analysis to compute geometrically isolated higher order intersections of curves. Such intersections are obtained from tangential intersections, singular points etc. The main advantage of the algorithm lies in its *efficiency and robustness*. The numerical accuracy of the operations is well understood and we come up with tight bounds on the errors using 64 bit IEEE floating point arithmetic.

# 1 Introduction

Computing the intersection of parametric and algebraic curves is a fundamental problem in computer graphics and geometric and solid modeling. In particular, intersection is a primitive operation in the computation of a boundary representation from a CSG (constructive solid geometry) model in a CAD system. Other applications of intersection include hidden curve removal for free form surfaces, finding complex roots of polynomials, computing singular points etc. [1, 2, 3, 4, 5].

There are three major algorithms for computing the intersection of rational parametric curves. They are based on implicitization [6], Bézier subdivision [7] and interval arithmetic [8]. The implicitization approach is based on reducing the problem to finding roots of a univariate polynomials, while Bezier subdivision and interval arithmetic proceed by using the geometric properties of the control polygon of a Bezier curve. The relative performance and accuracy of these algorithms is highlighted in [2]. In particular, implicitization based approaches are considered faster than other intersection algorithms for curves of degree up to four. However, their relative performance degrades for higher degree curves. This is mainly due to issues of numerical stability and their effect on the choice of representation and algorithms for root finding. For curves of degree greater than three the resulting polynomial has degree at least 16. The problem of computing real roots of such high degree polynomials is frequently ill-conditioned [9]. As a result subdivision based approaches perform better.

The algorithms for algebraic curve intersection are analogous to those of intersecting parametric curves. Resultants are used to eliminate one variable from the two equations corresponding to the curves. The problem of intersection corresponds to computing roots of the resulting univariate polynomial. This approach causes numerical problems for higher degree curves (greater than four). A robust algorithm based on subdivision has been presented in [10]. However, resultant based algorithms are considered to be the fastest for lower degree curves.

The problems of computing the intersection of parametric and algebraic curves correspond to computing common solutions of a system of algebraic equations. Some general purpose methods to compute the system of algebraic equations can be used to solve intersection problems as well. These include Gröbner bases algorithms [11], homotopy methods [12] and interval arithmetic [13]. However, Gröbner bases algorithms need exact arithmetic and homotopy methods and interval arithmetic methods are slow as compared to algorithm highlighted above [14].

In many applications, the intersection may be of higher order involving tangencies and singular points. Such instances are rather common in industrial applications [15]. Most algorithms require special handling for tangencies and therefore require

additional computation for detecting them. In fact algorithms based on subdivision and Newton-type techniques often fail to accurately compute the intersections in such cases. Special techniques for computing first order tangential contacts of parametric curves are given in [15]. Sederberg presents a modification of his algorithm for computing all double points of an algebraic curve in a triangular domain [10]. However, no efficient and accurate techniques are known for computing higher order intersections.

An efficient and robust algorithm for intersecting parametric and algebraic curves using matrix computations is presented in [16, 17]. For parametric curve intersection, one of the curves is implicitized and represented as a matrix determinant. After substituting the other parametrization the problem of intersection is reduced to computing the eigenvalues and eigenvectors of a matrix. The advantages of this technique lie in its *efficiency, robustness and numerical accuracy*. The algorithm for algebraic curve intersection is similar.

In this paper we extend the algorithm of [16] to handle multiple intersections. Such intersections are obtained due to tangential intersections or multiple points on one of the curve. It is rather difficult to detect such cases using the geometric properties of the control polygon of the curves. Our algorithm is based on the algebraic and geometric multiplicities of the eigenvalues corresponding to intersections points. As such the problem of computing multiple roots of a polynomial or multiple eigenvalues of a matrix can be numerically ill-conditioned. As a result a small perturbation in the input data can result in significant order of change in the magnitudes of the resulting roots or eigenvalues. We represent the perturbed eigenvalues corresponding to the higher multiplicity eigenvalue as a cluster. Moreover, the arithmetic mean of the eigenvalues in the cluster is well conditioned as opposed to the individual eigenvalues in the cluster. We are able to accurately compute the mean using matrix computations. Given the higher multiplicity eigenvalue corresponding to the intersection we compute the intersection point, its multiplicity and the nature of intersection. The algorithm is implemented using routines from LAPACK and we illustrate the algorithm on various examples. The overall algorithm works well on geometrically isolated higher order intersections. In case the higher order intersections are not geometrically isolated, we present some heuristics to compute the higher order intersections. Their accuracy is a function of the sensitivity of the problem to perturbation. Some of the results presented in this paper have also appeared in [18].

The rest of the paper is organized in the following manner. In Section 2 we present our notation, review techniques from elimination theory and reduce the problems of intersecting parametric and algebraic curves to computing roots of polynomials expressed as matrix determinants. We also highlight a number of properties of the matrix determinants corresponding to the implicit representation and resultant for-

mulations. In Section 3, we review results from linear algebra and numerical analysis being used in the algorithm and show how the intersection problem can be expressed as an eigenvalue problem. Section 4 highlights the numerical problems in computing the higher multiplicity roots and eigenvalues and shows how the mean of the clusters of eigenvalues can be accurately computed. We also present our algorithm for computing the intersection multiplicity, intersection points and the nature of intersection.

## 2 Parametric and Algebraic Curves

A rational Bézier curve is of the form [19]:

$$\mathbf{P}(t) = (X(t), Y(t)) = \frac{\sum_{i=0}^n w_i \mathbf{P}_i B_{i,n}(t)}{\sum_{i=0}^n w_i B_{i,n}(t)}, \quad 0 \leq t \leq 1$$

where  $\mathbf{P}_i = (X_i, Y_i)$  are the coordinates of a control point,  $w_i$  is the weight of the control point and  $B_{i,n}(t)$  corresponds to the Bernstein polynomial

$$B_{i,n} = \binom{n}{i} (1-t)^{n-i} t^i.$$

For polynomial curves the denominator term is a constant. Other rational formulations like B-splines can be converted into a series of Bézier curves by knot insertion algorithms [19]. Thus, the problem of intersecting rational curves can be reduced to intersecting Bézier curves.

Algebraic plane curves are generally expressed in standard power basis:

$$F(x, y) = \sum_{i+j \leq n} c_{ij} x^i y^j = 0.$$

They can also be represented in Bernstein basis. The problem of intersection corresponds to computing the common points on such curves in a particular domain.

The set of rational parametric curves is a proper subset of algebraic plane curves [20]. In the following paragraphs we will highlight some algebraic properties of algebraic curves and they are applicable to parametric curves as well.

### 2.1 Multiple Points

A point on a curve is a *regular point* unless it satisfies the following definition [20]:

**Definition:** A *multiple point* of order  $k$  (or  $k$ -fold point,  $k > 1$ ) of a degree  $n$  curve, is a point  $\mathbf{p}$  of the curve such that a generic line through  $\mathbf{p}$  meets the curve in at most  $n - k$  further points.

Let us investigate the behavior of an algebraic curve at a multiple point. We can assume that the point under consideration is the origin, i.e.  $\mathbf{p} = (0, 0)$ , else we can bring it to the origin by a suitable linear transformation. The curve can be represented as

$$F(x, y) = U_0(x, y) + U_1(x, y) + \dots + U_{n-1}(x, y) + U_n(x, y) = 0,$$

where  $U_i(x, y)$  is a homogeneous polynomial of degree  $i$  in  $x$  and  $y$ . A generic line through the origin can be represented in the form  $x/a = y/b$ . The point of this line whose coordinates are  $(\gamma a, \gamma b)$ , where  $\gamma$  is a scalar, lies on the curve if  $\gamma$  is any of the roots of the equation

$$U_0(a, b) + \gamma U_1(a, b) + \gamma^2 U_2(a, b) + \dots + \gamma^i U_i(a, b) + \dots + \gamma^n U_n(a, b) = 0. \quad (1)$$

To make the curve have a  $k$ -fold point at the origin corresponds to making the equation, (1), have  $k$  nonzero roots for every value of the ratio  $a/b$ . This can happen, if and only if  $U_0(x, y), U_1(x, y), \dots, U_{k-1}(x, y)$  vanish identically. A line corresponds to a tangent at  $\mathbf{p}$ , if it has  $k + 1$  of its intersections with the curve at  $\mathbf{p}$  and the  $n - k - 1$  intersections at other points on the curve. All lines of the form  $x/a' = y/b'$ , where  $U_k(a', b') = 0$ , are tangent to the curve at  $\mathbf{p}$ . There can be at most  $k$  such lines.

This formulation of multiple points is constructive. It can be used to identify multiple points on the curve according to the following lemma:

**Lemma 2.1** *A point  $\mathbf{q} = (X_1, Y_1)$  is a point of multiplicity  $k$  on the curve  $F(x, y)$  if and only if every monomial of  $F(x - X_1, y - Y_1)$  has degree  $k$  or more.*

As far as parametric curves are concerned, multiple points can be defined in a similar fashion. In particular, a point  $\mathbf{q} = \mathbf{P}(t_1) = (X_1, Y_1) = (\frac{x(t_1)}{w(t_1)}, \frac{y(t_1)}{w(t_1)})$  has multiplicity  $k$  if any generic line passing through  $\mathbf{q}$  intersects  $\mathbf{P}(t)$  at  $n - k$  other points, where  $n$  is the degree of  $\mathbf{P}(t)$ . A generic line passing through  $\mathbf{q}$  is of the form  $aX + bY + c = 0$  such that  $aX_1 + bY_1 + c = 0$ .

Typical examples of multiple points includes cusps and loops. Every properly parametrized rational parametric curve has a finite number of singular points. Moreover the notion of regular and singular points on a curve can also be explained in terms of the *place* of a curve at a point.

In the neighborhood of a point  $\mathbf{q} = \mathbf{P}(t_1)$ , the curve can always be defined by a formal power series. For example,  $(X(t), Y(t))$  can be expressed as a power series representation in the neighborhood of  $\mathbf{q}$ . The formal power series or the local parametrization is called a *place* of  $\mathbf{P}(t)$  at  $\mathbf{q}$  and exists because of *Newton's* theorem [21]. The notion of a place is more specific than that of a curve point. Corresponding to every curve point the curve has a place. The curve may have more than one place

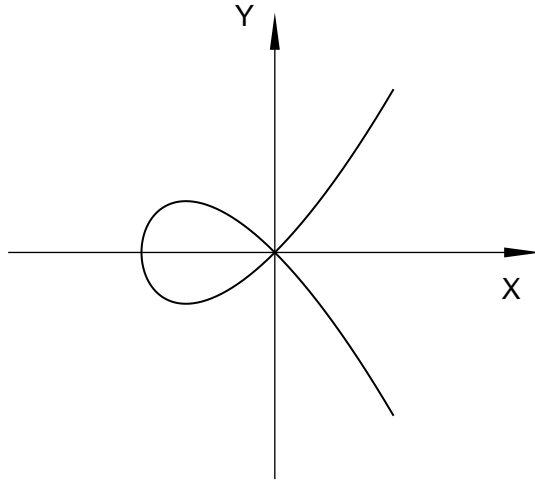


Figure 1: A cubic curve with a loop

at a singular point and has one place at every non-singular point. In particular, the curve has two or more places at a node or loop and one place at a cusp. More on places and their representation as branches is given in [22, 1, 23, 21].

A rational parametric curve  $\mathbf{P}(t)$  is *properly parametrized* if it has one-to-one relationship between the parameter  $t$  and points on the curve, except for a finite number of exceptional points. Let  $\mathbf{S}$  be one of these exceptional points. In other words, there is more than one value of the parameter  $t$ , which gives rise to the point  $\mathbf{S}$ . At such points, the curve has more than one place. The exact relationship between the number of parameter values corresponding to a point and the number of places at the same point is given by the following lemma [21]:

**Lemma 2.2** *The number of values of  $t$  that give rise to a point  $\mathbf{q}$  on a properly parametrized curve  $\mathbf{P}(t)$  is the number of places on the curve at  $\mathbf{q}$ .*

**Example 2.3** *Consider the cubic plane curve*

$$\mathbf{P}(t) = (x(t), y(t), w(t)) = (t^2 - 1, t^3 - t, 1)$$

*which is a parametrization of  $f(x, y, w) = y^2w - x^2w - x^3 = 0$ , a nodal cubic (as shown in Fig. 1). Since the degree of  $\mathbf{Q}(t)$  is equal to the degree of  $f(x, y, w)$  (which is three),  $\mathbf{Q}(t)$  is properly parametrized. The curve has one place at every point except at the origin, where it has two places corresponding to  $t = 1$  and  $t = -1$ .*

The singular point,  $\mathbf{q}$  on a rational curve can be classified according to the number of places the curve has at that point.

- The curve has one place at  $\mathbf{q}$ . These include simple cusps.
- The curve has more than one place at  $\mathbf{q}$ . These cases include loops.

For our algorithm we assume that the curve  $\mathbf{P}(t)$  is properly parametrized. As a result all the singular points having more than one place have more than one preimage (according to lemma 2.2). The cusps on a curve can be identified according to the following theorem from [24]:

**Theorem 2.4** *Given a rational curve  $\mathbf{P}(t) = (X(t), Y(t))$  with a proper parametrization, the curve has a cusp at  $\mathbf{q} = (X(t_1), Y(t_1))$  if and only if*

$$(X'(t_1), Y'(t_1)) = (0, 0).$$

The singular points on the rational curve can be classified according to the following theorem:

**Theorem 2.5** *A point  $\mathbf{q} = (X_1, Y_1) = (\frac{x(s_1)}{w(s_1)}, \frac{y(s_1)}{w(s_1)})$  is a singular point on  $\mathbf{P}(t)$  of multiplicity  $k$  if and only if the following equations have  $k$  common roots (counted with respect to multiplicity):*

$$X_1 w(t) - x(t) = 0,$$

$$Y_1 w(t) - y(t) = 0.$$

**Proof:** Let us consider the case when these equations have  $k$  common roots (counted properly). Let the roots be  $t_1, t_2, \dots, t_k$ . Some of them may be repeated roots. It turns out that for each  $t_i$ ,  $1 \leq i \leq k$ ,

$$x(t_i) = X_1 w(t_i), \quad y(t_i) = Y_1 w(t_i).$$

Lets consider the generic line passing through  $\mathbf{q}$  of the form  $aX + bY + c = 0$ , whose coefficients satisfy the equation  $aX_1 + bY_1 + c = 0$ . The intersections of this line and  $\mathbf{P}(t)$  are characterized by the roots of the equation  $f(t) = ax(t) + by(t) + cw(t) = 0$ . For each  $t_i$ ,

$$f(t_i) = ax(t_i) + by(t_i) + cw(t_i) = w(t_i)(aX_1 + bY_1 + c) = 0.$$

As a result, each  $t_i$  corresponds to a root of  $f(t)$  and therefore, the curve has multiplicity  $k$  at  $\mathbf{q}$ .

To prove the necessity we assume that  $\mathbf{q}$  is a point of multiplicity  $k$ , where  $k \geq 2$ . That implies that any line passing through  $\mathbf{q}$  intersects the curve at  $n - k$  points at



most. Let us represent the line as  $aX + bY + c = 0$  where  $a, b, c$  are chosen such that  $aX_1 + bY_1 + c = 0$ . After substituting the parametrization of the curve into the equation of the line we obtain  $f(t) = ax(t) + by(t) + cw(t) = 0$ , a polynomial of degree  $n$ . Its  $n$  roots correspond to the points of intersection. The fact  $\mathbf{q}$  is a point of multiplicity  $k$  implies that for  $k$  of the  $n$  roots, say  $t_1, \dots, t_k$ ,  $\mathbf{P}(t_i) = \mathbf{q}$ . Let us choose a line such that  $a = 0$ . Therefore,  $Y_1 = -\frac{c}{b}$  and  $t_i$  is a root of the equation  $y(t) - Y_1w(t) = 0$ . Similarly one can show that  $t_i$  is also a root of the equation  $x(t) - X_1w(t) = 0$ . Thus,  $t_1, t_2, \dots, t_k$  are the common roots of the two equations.

Q.E.D.

A simple version of *Bezout's theorem* is used for determining the number of intersections between a curve of degree  $m$  and that of degree  $n$  [20]. It is assumed that the curves have no component in common. That is:

*Two curves of degree  $m$  and  $n$  intersect at  $mn$  points, counted properly with respect to multiplicity.*

## 2.2 Elimination Theory

Elimination theory is a branch of classical algebraic geometry dealing with conditions under which sets of polynomials have common roots. Its results were known a century ago [25, 26]. The main result is the construction of a single resultant polynomial such that the vanishing of the resultant is the necessary and sufficient condition for the given system of equations to have a non-trivial solution. As far as geometric and solid modeling are concerned, the use of resultants was resurrected by Sederberg for implicitizing parametric curves and surfaces [6]. The problem of curve intersection is reduced to solving two polynomial equations simultaneously and therefore we use resultant formulations of two polynomials in one unknown. Surveys on various formulations of resultants are given in [6, 27] and effective techniques for computing and applying them are presented in [14].

Three methods are known in the literature for computing the resultant, owing to Bezout or Cayley and Sylvester [26]. Each of them expresses the resultant as the determinant of a matrix. The order of the matrix is different for different methods. We use Cayley's formulation of Bezout resultant as it results in a matrix of lower order. It has been explained in detail in [16] and specifically applied to the polynomials corresponding to the intersection of parametric and algebraic curves.

We are given two polynomials,  $F(x)$  and  $G(x)$ , of degree  $m$  and  $n$ , respectively.

Without loss of generality we assume that  $m \geq n$ . Consider the bivariate polynomial

$$P(x, \alpha) = \frac{F(x)G(\alpha) - F(\alpha)G(x)}{x - \alpha}.$$

$P(x, \alpha)$  is a polynomial of degree  $m - 1$  in  $x$  and also in  $\alpha$ . Let us represent it as

$$P(x, \alpha) = P_0(x) + P_1(x)\alpha + P_2(x)\alpha^2 + \dots + P_{m-1}(x)\alpha^{m-1}, \quad (2)$$

where  $P_i(x)$  is a polynomial of degree  $m - 1$  in  $x$ . The polynomials  $P_i(x)$  can be written as follows:

$$\begin{pmatrix} P_0(x) \\ P_1(x) \\ \vdots \\ P_{m-1}(x) \end{pmatrix} = \begin{pmatrix} P_{0,0} & P_{0,1} & \dots & P_{0,m-1} \\ P_{1,0} & P_{1,1} & \dots & P_{1,m-1} \\ \vdots & \vdots & \vdots & \vdots \\ P_{m-1,0} & P_{m-1,1} & \dots & P_{m-1,m-1} \end{pmatrix} \begin{pmatrix} 1 \\ x \\ x^2 \\ \vdots \\ x^{m-1} \end{pmatrix} \quad (3)$$

Let us denote the  $m \times m$  matrix by  $\mathbf{M}$ . Its determinant is the resultant of  $F(x)$  and  $G(x)$ . Let us assume that  $x = x_0$  is a common root of the two polynomials. As a result  $\mathbf{M}$  is singular and  $[1 \ x_0 \ x_0^2 \ \dots \ x_0^{m-1}]^T$  is a vector in the kernel of  $\mathbf{M}$ .

Let us consider the case when two polynomials,  $F(x)$  and  $G(x)$  have a root of multiplicity  $k$  at  $x = x_0$ . In other words,

$$F(x_0) = 0, \quad F'(x_0) = 0, \quad G(x_0) = 0, \quad G'(x_0) = 0, \quad \dots, \quad F^{k-1}(x_0) = 0, \quad G^{k-1}(x_0) = 0.$$

It turns out that the polynomial  $P(x, \alpha)$  and the matrix  $\mathbf{M}$  have some interesting properties. Later on we make use of these properties in computing the higher order intersections of curves.

**Lemma 2.6** *Given  $F(x)$  and  $G(x)$  with a root of multiplicity  $k$  at  $x = x_0$ , then the vectors*

$$\begin{aligned} & (1 \ x_0 \ x_0^2 \ \dots \ x_0^{m-1})^T, \\ & (0 \ 1 \ 2x_0 \ 3x_0^2 \ \dots \ (m-1)x_0^{m-2})^T, \\ & \vdots \\ & \left( 0 \ 0 \ \dots \ 0 \ (k-1)! \ k! \ x_0 \ \frac{(k+1)!}{2} x_0^2 \ \frac{(m-1)!}{(m-k)!} x_0^{m-k} \right)^T \end{aligned}$$

*are in the kernel of  $\mathbf{M}$ .*

**Proof:** We will highlight the proof for  $k = 2$  and it can be easily extended to arbitrary  $k$ . Let us consider the polynomial,  $P_x(x, \alpha)$ , which is the partial of derivative of  $P(x, \alpha)$  with respect to  $x$ .

$$P_x(x, \alpha) = \frac{(x - \alpha)(F'(x)G(\alpha) - F(\alpha)G'(x)) - (F(x)G(\alpha) - F(\alpha)G(x))}{(x - \alpha)^2} \quad (4)$$

Moreover it follows from (2)

$$P_x(x, \alpha) = P'_0(x) + P'_1(x)\alpha + P'_2(x)\alpha^2 + \dots + P'_{m-1}(x)\alpha^{m-1} \quad (5)$$

Since  $F(x_0) = 0$ ,  $G(x_0) = 0$ ,  $F'(x_0) = 0$  and  $G'(x_0) = 0$  it follows that  $P_x(x_0, \alpha) = 0$ . Equating (4) and (5) and substituting  $x = x_0$  results in

$$P_x(x_0, \alpha) = P'_0(x_0) + P'_1(x_0)\alpha + P'_2(x_0)\alpha^2 + \dots + P'_{m-1}(x_0)\alpha^{m-1} = 0.$$

This relationship is true for any  $\alpha$ . That implies that

$$P'_0(x_0) = 0, P'_1(x_0) = 0, \dots, P'_{m-1}(x_0) = 0.$$

This can be expressed in a matrix formulation in the following manner:

$$\begin{pmatrix} P_{0,0} & P_{0,1} & \dots & P_{0,m-1} \\ P_{1,0} & P_{1,1} & \dots & P_{1,m-1} \\ \vdots & \vdots & \vdots & \vdots \\ P_{m-1,0} & P_{m-1,1} & \dots & P_{m-1,m-1} \end{pmatrix} \begin{pmatrix} 0 \\ 1 \\ 2x_0 \\ \vdots \\ (m-1)x_0^{m-2} \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ \vdots \\ 0 \end{pmatrix}$$

For arbitrary  $k$ , we know that  $F(x_0) = 0, F'(x_0) = 0, \dots, F^{k-1}(x_0) = 0$  and similarly for  $G(x)$ . As a result we can show that the first  $k - 1$  partial derivative of  $P(x, \alpha)$  with respect to  $x$  vanish at  $x = x_0$ . Therefore the  $k$  vectors corresponding to the partials of  $(1 \ x \ x^2 \ \dots \ x^{m-1})$  at  $x = x_0$  vectors lie in the kernel of  $\mathbf{M}$ .

Q.E.D.

## 2.3 Implicitizing Parametric Curves

Given a rational Bézier curve,  $\mathbf{P}(t)$ , we express it in homogeneous form as

$$\mathbf{p}(t) = (x(t), y(t), w(t)) = (\sum_{i=0}^m w_i X_i B_{i,m}(t), \sum_{i=0}^m w_i Y_i B_{i,m}(t), \sum_{i=0}^m w_i B_{i,m}(t)).$$

We assume that the curve,  $\mathbf{P}(t)$  has a *proper* parametrization and that moreover

$$GCD(x(t), y(t), w(t))$$

is a constant. Algorithms to compute the proper parametrizations of curves have been described in [28, 24, 29].

To implicitize the curve we consider the following system of equations

$$\begin{aligned} F_1(t) : Xw(t) - x(t) &= 0 \\ F_2(t) : Yw(t) - y(t) &= 0. \end{aligned} \tag{6}$$

Consider them as polynomials in  $t$  and  $X, Y$  are treated as symbolic coefficients. The implicit representation corresponds to the resultant of (6). An efficient algorithm to compute the entries of the resultant matrix,  $\mathbf{M}$ , is presented in [16]. It involves purely numeric computation on the coefficients of the parametric curves. The entries of  $\mathbf{M}$  are linear polynomials in  $X$  and  $Y$ .

The algorithm for computing the entries of the matrix assumes that the polynomials  $x(t), y(t), w(t)$  are expressed in power basis. However, converting from Bézier to power basis can introduce numerical errors [30]. To circumvent this problem we perform a reparametrization.

We are given

$$\mathbf{p}(t) = \left( \sum_{i=0}^m w_i X_i \binom{m}{i} (1-t)^{m-i} t^i, \sum_{i=0}^m w_i Y_i \binom{m}{i} (1-t)^{m-i} t^i, \sum_{i=0}^m w_i \binom{m}{i} (1-t)^{m-i} t^i \right).$$

On dividing by  $(1-t)^m$ , we obtain

$$\mathbf{p}(t) = \left( \sum_{i=0}^m w_i X_i \binom{m}{i} \frac{t^i}{(1-t)^i}, \sum_{i=0}^m w_i Y_i \binom{m}{i} \frac{t^i}{(1-t)^i}, \sum_{i=0}^m w_i \binom{m}{i} \frac{t^i}{(1-t)^i} \right)$$

Let  $s = \frac{t}{(1-t)}$  and the resulting parametrization is

$$\bar{\mathbf{p}}(s) = \left( \sum_{i=0}^m w_i X_i \binom{m}{i} s^i, \sum_{i=0}^m w_i Y_i \binom{m}{i} s^i, \sum_{i=0}^m w_i \binom{m}{i} s^i \right).$$

The rest of the algorithm proceeds by computing the implicit representation of  $\bar{\mathbf{p}}(s)$  and computing a matrix formulation by Cayley's method as

$$\mathbf{M} \begin{pmatrix} 1 \\ s \\ s^2 \\ \vdots \\ s^{m-1} \end{pmatrix}.$$

Substitute  $s = \frac{t}{(1-t)}$  and multiply the right hand side vector by  $(1-t)^{m-1}$ . The resulting linear system has the form

$$\mathbf{M} \begin{pmatrix} (1-t)^{m-1} \\ t(1-t)^{m-2} \\ t^2(1-t)^{m-3} \\ \vdots \\ t^{m-1} \end{pmatrix}. \quad (7)$$

This relationship is used to compute the inverse coordinates of the intersection points.

## 2.4 Properties of Implicit Representation

In the previous section we have presented a matrix determinant formulation for the implicit representation of the curve. In this section we highlight some properties of this formulation, denoted as  $\mathbf{M} = \mathbf{F}(X, Y)$ .

It follows from the properties of the implicit representation that  $\mathbf{F}(X_1, Y_1)$  is a singular matrix if and only if  $(X_1, Y_1)$  is a point lying on the curve. Furthermore, let us assume that  $(X_1, Y_1)$  is a *regular point* and not a *singular point* on the curve. Corresponding to a regular point  $(X_1, Y_1)$  on the curve  $\mathbf{P}(t)$ , there exists a unique preimage  $t_1$  such that  $\mathbf{P}(t_1) = (X_1, Y_1)$ . Since  $\mathbf{F}(X_1, Y_1)$  is a singular matrix, it has a vector in the kernel of the form

$$\begin{pmatrix} 1 & t_1 & t_1^2 & \dots & t_1^{m-1} \end{pmatrix}^T.$$

Moreover,  $\mathbf{F}(X_1, Y_1)$  has a kernel of dimension one.

Let us consider the case when  $(X_1, Y_1)$  corresponds to a singular point on a curve.

**Theorem 2.7** *Let  $\mathbf{q} = (X_1, Y_1) = \mathbf{P}(t_1)$  correspond to a point of multiplicity  $k$  on the curve. The kernel of the matrix  $\mathbf{F}(X_1, Y_1)$  has dimension  $k$ .*

**Proof:** The fact that  $\mathbf{q}$  is a point of multiplicity  $k$  implies that the following equations have  $k$  common roots: (according to Theorem 2.5)

$$X_1 w(t) - x(t) = 0,$$

$$Y_1 w(t) - y(t) = 0.$$

Let the common roots be  $t_1, t_2, \dots, t_l$ , where  $t_i$  is a root of multiplicity  $m_i$ . Therefore

$$m_1 + m_2 + \dots + m_l = k$$

and each  $t_i$  corresponds to a distinct root.

The equations mentioned above correspond exactly to the parametric equations (6), which are used for computing the implicit representation of the parametric curve. Let  $t_i$  be one of the roots of these equations. The derivation of Cayley's resultant implies that

$$\mathbf{F}(X_1, Y_1) \begin{pmatrix} 1 \\ t_i \\ t_i^2 \\ \vdots \\ t_i^{m-1} \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{pmatrix}.$$

As a result  $\mathbf{v}_i = (1 \ t_i \ t_i^2 \ \dots \ t_i^m)^T$  is a vector in the kernel of  $\mathbf{F}(X_1, Y_1)$ . Since each  $t_i$  is distinct it follows that  $\mathbf{v}_1, \mathbf{v}_2, \dots, \mathbf{v}_l$  are independent vectors in the kernel of  $\mathbf{F}(X_1, Y_1)$ .

Given that  $t_i$  is a root of multiplicity  $m_i$  of the above equations. According to Lemma 2.6 the following vectors also belong to the kernel of  $\mathbf{F}(X_1, Y_1)$ :

$$\begin{aligned} \mathbf{v}_{i_1} &= (1 \ t_i \ t_i^2 \ \dots \ t_i^{m-1})^T, \\ \mathbf{v}_{i_2} &= (0 \ 1 \ 2t_i \ 3t_i^2 \ \dots \ (m-1)t_i^{m-2})^T, \\ &\vdots \\ \mathbf{v}_{i_{m_i}} &= \left( 0 \ 0 \ \dots \ 0 \ (m_i-1)! \ m_i!t_i \ \frac{(m_i+1)!}{2}t_i^2 \ \frac{(m-1)!}{(m-m_i-2)!}t_i^{m-m_i} \right)^T. \end{aligned}$$

For  $1 \leq i \leq l$  these vectors constitute a  $k$  dimensional kernel of  $\mathbf{F}(X_1, Y_1)$ .

Q.E.D.

## 2.5 Intersecting Parametric Curves

Given two rational Bézier curves,  $\mathbf{P}(t)$  and  $\mathbf{Q}(u)$  of degree  $m$  and  $n$  respectively, the intersection algorithm proceeds by implicitizing  $\mathbf{P}(t)$  and obtaining a  $m \times m$  matrix  $\mathbf{M}$ , whose entries are linear combinations of symbolic coefficients  $X, Y$ . The second parametrization  $\mathbf{Q}(u) = (\bar{x}(u), \bar{y}(u), \bar{w}(u))$  is substituted into the matrix formulation. It results in a matrix polynomial  $\mathbf{M}(u)$  such that each of its entries is a linear combination of  $\bar{x}(u), \bar{y}(u)$  and  $\bar{w}(u)$ . The intersection points correspond to the roots of

$$\text{Determinant}(\mathbf{M}(u)) = 0. \tag{8}$$

This completes the reduction of the problem of intersecting parametric curves to a *nonlinear eigenvalue problem* (8). The reduction to a standard eigenvalue problem will be complete in section 3.8.

## 2.6 Intersecting Algebraic Curves

In this section we consider the intersection of two algebraic plane curves, represented as zeros of  $F(x, y)$  and  $G(x, y)$ , polynomials of degree  $m$  and  $n$ , respectively. The polynomials may be represented in power basis or Bernstein basis. Let the points of intersection be  $(x_1, y_1), \dots, (x_{mn}, y_{mn})$ . To simplify the problem we compute the projection of these points on the x-axis. Algebraically projection corresponds to computing the resultant of  $F(x, y)$  and  $G(x, y)$  by treating them as polynomials in  $y$  and expressing the coefficients as polynomials in  $x$ . The resultant  $R(x)$  is a polynomial of degree  $mn$ . The resultant is computed using Cayley's formulation in (3). Let us denote the  $m \times m$  matrix by  $\mathbf{M}(x)$ . The problem of intersection corresponds to computing roots of

$$\text{Determinant}(\mathbf{M}(x)) = 0. \tag{9}$$

More details on computing the resultant corresponding to the intersection of algebraic curves are given in [16]. The kernel of  $\mathbf{M}(x_0)$ , where  $x_0$  is a root of the polynomial given above, has properties similar to that of the matrix corresponding to the implicit representation of the parametric equations. For simple intersections, its entries are functions of  $y_0$ , the  $y$ -coordinate of the intersection.

This completes the reduction of the problem of intersecting algebraic curves to a *nonlinear eigenvalue problem* (9). The reduction to a standard eigenvalue problem will be complete in section 3.8.

## 3 Matrix Computations

In this section we review some techniques from linear algebra and numerical analysis [31, 35, 34]. We also reduce the problem of computing the intersection of parametric and algebraic curves to eigenvalue problems. We also discuss the numerical accuracy of the problems in terms of their condition number and the algorithms used to solve those problems. In particular we highlight some features of these techniques used in our algorithm for intersection in Section 4.

### 3.1 Singular Value Decomposition

The singular value decomposition (SVD) is a powerful tool which gives us accurate information about matrix rank in the presence of round off errors. The rank of a matrix can also be computed by Gauss elimination. However, there arise many situations where near rank deficiency prevails. Rounding errors and fuzzy data make rank

determination a nontrivial exercise. In these situations, the numerical rank is easily characterized in terms of the SVD.

Given a real  $m \times n$  matrix  $\mathbf{A}$ , where without loss of generality we assume  $m \geq n$ , one can write

$$\mathbf{A} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T$$

where  $\mathbf{U}$  is an  $m \times n$  orthogonal matrix,  $\mathbf{V}$  is an  $n \times n$  orthogonal matrix and  $\mathbf{\Sigma}$  is an  $n \times n$  diagonal matrix of the form

$$\mathbf{\Sigma} = \text{diag}(\sigma_1, \sigma_2, \dots, \sigma_n).$$

Moreover,  $\sigma_1 \geq \sigma_2 \geq \dots \geq \sigma_n \geq 0$ . The  $\sigma_i$ 's are called the *singular values* and the columns of  $\mathbf{U}$  and  $\mathbf{V}$ , denoted as  $u_i$ 's and  $v_j$ 's, are known as the left and right singular vectors, respectively [31].

The singular values give accurate information about the rank of the matrix. The matrix  $\mathbf{A}$  has rank  $k < n$  if  $\sigma_k \neq 0$  and  $\sigma_{k+1} = \dots = \sigma_n = 0$ . Furthermore, the singular values tell exactly how close  $\mathbf{A}$  is to a rank deficient matrix: the closest matrix to  $\mathbf{A}$  of rank at most  $k$  is  $\mathbf{A}_k \equiv \sum_{i=1}^k \sigma_i u_i v_i^T$ , and  $\|\mathbf{A} - \mathbf{A}_k\|_2 = \sigma_{k+1}$  ( $\|\cdot\|_2$  is the spectral norm of a matrix) [31]. The cost of the algorithm (for  $m = n$ ) ranges from  $8n^3/3$  if only  $\mathbf{\Sigma}$  is desired to  $22n^3$  if  $\mathbf{U}$ ,  $\mathbf{\Sigma}$  and  $\mathbf{V}$  are desired.

## 3.2 Eigenvalues and Eigenvectors

Given a  $n \times n$  matrix  $\mathbf{A}$ , its eigenvalues and eigenvectors are the solutions to the equation

$$\mathbf{A}\mathbf{x} = s\mathbf{x},$$

where  $s$  is the eigenvalue and  $\mathbf{x} \neq \mathbf{0}$  is the eigenvector. The eigenvalues of a matrix are the roots of its characteristic polynomial,  $\text{Determinant}(\mathbf{A} - s\mathbf{I})$ . Efficient serial algorithms for computing all the eigenvalues and eigenvectors are well known [31], and their implementations are available as part of packages EISPACK [32] and LAPACK [34]. Most algorithms use similarity transformations of the form  $\mathbf{A}' = \mathbf{Q}\mathbf{A}\mathbf{Q}^{-1}$ , where  $\mathbf{Q}$  is a nonsingular  $n \times n$  matrix. This transformation leaves the eigenvalues of  $\mathbf{A}$  unchanged, and changes eigenvectors  $\mathbf{y}$  of  $\mathbf{A}$  into eigenvectors  $\mathbf{Q}\mathbf{y}$  of  $\mathbf{A}'$ . Algorithms choose  $\mathbf{Q}$  so that  $\mathbf{A}'$  is (nearly) upper triangular, which makes its eigenvalues and eigenvectors easy to compute. The standard algorithm is *QR* iteration, which chooses  $\mathbf{Q}$  to be an orthogonal matrix, which guarantees that the entire algorithm is numerically stable [31]. One *QR* step of iteration goes as follows. We first (implicitly) factorize

$$\mathbf{A} - \sigma\mathbf{I} = \mathbf{U}\mathbf{R}, \tag{10}$$



where  $\sigma$  is a scalar referred to as a *shift*,  $\mathbf{U}$  is an orthogonal matrix and  $\mathbf{R}$  is an upper triangular matrix. The next iterate is

$$\mathbf{A}' = \mathbf{R}\mathbf{U} + \sigma\mathbf{I} = \mathbf{U}^T\mathbf{A}\mathbf{U}.$$

The shift  $\sigma$  is chosen as an approximate eigenvalue in such a way that the bottom right entry of  $\mathbf{A}$  converges quadratically to an eigenvalue. To make this algorithm practical  $\mathbf{A}$  is initially reduced to Hessenberg form, which cuts the cost of an iteration, and a shift by a complex  $\sigma$  is replaced by a combined shift by  $\sigma$  and its complex conjugate  $\bar{\sigma}$ , to avoid complex arithmetic. Ultimately, the sequence of matrix iterates converges to its *real Schur decomposition* of the form

$$\mathbf{Q}\mathbf{A}\mathbf{Q}^{-1} = \begin{pmatrix} R_{11} & R_{12} & \cdots & R_{1m} \\ 0 & R_{22} & \cdots & R_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & R_{mm} \end{pmatrix}, \quad (11)$$

where each  $R_{ii}$  is either a  $1 \times 1$  matrix or a  $2 \times 2$  matrix having complex conjugate eigenvalues. Given the real Schur decomposition, computing the eigenvalues is a trivial operation. More details are given in [31]. The cost of the algorithm ranges from  $10n^3$  if only eigenvalues are desired, to about  $26n^3$  for all eigenvalues and eigenvectors.

### 3.3 Generalized Eigenvalue Problem

Given  $n \times n$  matrices,  $\mathbf{A}$  and  $\mathbf{B}$ , the generalized eigenvalue problem corresponds to solving

$$\mathbf{A}\mathbf{x} = s\mathbf{B}\mathbf{x}.$$

We also refer to the generalized eigenproblem for the *matrix pencil* (or just *pencil*)  $\mathbf{A} - \lambda\mathbf{B}$ . The scalar  $s$  is the eigenvalue and the vector  $\mathbf{x} \neq \mathbf{0}$  is the eigenvector. If  $\mathbf{B}$  is nonsingular, the problem can be reduced to a standard eigenvalue problem by multiplying both sides of the equation by  $\mathbf{B}^{-1}$  and thereby obtaining:

$$\mathbf{B}^{-1}\mathbf{A}\mathbf{x} = s\mathbf{x}.$$

However, if  $\mathbf{B}$  is close to singular, such a reduction can cause numerical problems (see the next section). Algorithms for the generalized eigenvalue problem implicitly apply the algorithm of the last section to  $\mathbf{B}^{-1}\mathbf{A}$  without forming  $\mathbf{B}^{-1}\mathbf{A}$ . In particular, we use the *QZ* algorithm for computing the eigenvalues and eigenvectors for this problem [31]. Its running time is still  $O(n^3)$ , but the constant can be as high as 75. Generally, it is 2.5 to 3 times slower than the *QR* algorithm of the last section.

### 3.4 Condition Numbers

The condition number of a problem measures the sensitivity of a solution to small changes in the input. A problem is *ill-conditioned* if its condition number is large, and *ill-posed* if its condition number is infinite. These condition numbers are used to bound errors in computed solutions of numerical problems. More details on condition numbers are given in [31, 35]. The implementations of these condition number computations are available as part of LAPACK [36].

In our intersection algorithm, we will be performing computations like matrix inversion and computing eigenvalues and eigenvectors of a matrix. Therefore, we will be concerned with the numerical accuracy of these operations.

### 3.5 Condition Number of a Square Matrix

The *condition number of a square matrix* is  $\frac{\sigma_1(\mathbf{A})}{\sigma_n(\mathbf{A})}$ , where  $\sigma_1$  and  $\sigma_n$  are the largest and smallest singular values. Thus, the condition number is large precisely when the distance to the nearest rank deficient matrix,  $\sigma_n(\mathbf{A})$ , is small. This condition number determines the accuracy of computing  $\mathbf{A}^{-1}$  or solving  $\mathbf{Ax} = \mathbf{b}$ . Computing the singular values takes  $O(n^3)$  time, which is rather expensive. Good estimators for the condition number of  $O(n^2)$  complexity, once  $\mathbf{Ax} = \mathbf{b}$  has been solved via Gaussian elimination, are available in LINPACK and LAPACK and we use them in our algorithm.

### 3.6 Condition Number of Simple Eigenvalues

Let  $s$  be a simple<sup>3</sup> eigenvalue of the  $n \times n$  matrix,  $\mathbf{A}$ , with unit right eigenvector  $\mathbf{x}$  and unit left eigenvector  $\mathbf{y}$ . That is,  $\mathbf{Ax} = s\mathbf{x}$ ,  $\mathbf{y}^H \mathbf{A} = s\mathbf{y}^H$  and  $\|\mathbf{x}\|_2 = \|\mathbf{y}\|_2 = 1$ . Here  $\|\mathbf{v}\|_2$  stands for the 2-norm of a vector and  $\mathbf{y}^H$  is the conjugate-transpose of  $\mathbf{y}$ . Let  $\mathbf{P} = (\mathbf{x} \cdot \mathbf{y}^H) / (\mathbf{y}^H \cdot \mathbf{x})$  be the spectral projector. Therefore,  $\|\mathbf{P}\|_2 = \frac{1}{|\mathbf{y}^H \mathbf{x}|}$ . Let  $\mathbf{E}$  be a perturbation of  $\mathbf{A}$ , and  $\epsilon_2 = \|\mathbf{E}\|_2$ . Moreover, let  $s'$  be the perturbed eigenvalue of  $\mathbf{A} + \mathbf{E}$ . Then

$$|s' - s| \leq \epsilon_2 \|\mathbf{P}\|_2 + O(\epsilon_2^2). \quad (12)$$

Thus, we call  $\|\mathbf{P}\|_2$  the condition number of the eigenvalue  $s$ .

### 3.7 Condition Number of Clustered Eigenvalues

To compute the higher order intersections we are interested in computing the condition numbers of a cluster of eigenvalues. We highlight the need for using clusters in Section

---

<sup>3</sup>A simple eigenvalue is an eigenvalue of multiplicity one.

4. We use these condition numbers in determining the accuracy of eigenvalues with multiplicity greater than one. We represent the real Schur decomposition as

$$\mathbf{A}' = \begin{pmatrix} \mathbf{A}_{11} & \mathbf{A}_{12} \\ 0 & \mathbf{A}_{22} \end{pmatrix}$$

where the eigenvalues of the  $m \times m$  matrix  $\mathbf{A}_{11}$  are precisely those we are interested in. In particular we are interested in bounding the perturbation in the average of the eigenvalues of the cluster,  $\bar{\sigma} \equiv \text{trace}(\mathbf{A}_{11})/m$ .

To compute the error bound, we define the spectral projector

$$\mathbf{P} = \begin{pmatrix} \mathbf{I}_m & \mathbf{R} \\ \mathbf{0} & \mathbf{0} \end{pmatrix},$$

where  $\mathbf{R}$  satisfies the system of linear equations

$$\mathbf{A}_{11}\mathbf{R} - \mathbf{R}\mathbf{A}_{22} = \mathbf{A}_{12}.$$

Thus,  $\|\mathbf{P}\|_2 = (1 + \|\mathbf{R}\|_2^2)^{1/2}$ . Computing  $\|\mathbf{P}\|_2$  is expensive and a cheaper overestimate is obtained as  $\|\mathbf{P}\|' = (1 + \|\mathbf{R}\|_F^2)^{1/2}$ , where  $\|\mathbf{R}\|_F$  is the Frobenius norm  $\mathbf{R}$  [31].

Let  $\mathbf{E}$  be the perturbation of  $\mathbf{A}$  and  $\epsilon_2 = \|\mathbf{E}\|_2$ . Let  $\bar{\sigma}'$  be the average of the perturbed eigenvalues. Then

$$|\bar{\sigma} - \bar{\sigma}'| \leq \epsilon_2 \|\mathbf{P}\|_2 + O(\epsilon_2^2). \quad (13)$$

We substitute  $\|\mathbf{P}\|'$  to obtain a slightly weaker bound on the perturbation in  $\bar{\sigma}$  for sufficiently small  $\epsilon_2$ . The average of a cluster is often much better conditioned than individual eigenvalues in the cluster. This technique works well once we can choose the right set of eigenvalues comprising a cluster. More details are presented in Section 4.

### 3.8 Reduction to an Eigenvalue Problem

In this section we consider the problem of intersecting parametric curves and reduce it to computing the eigendecomposition of a matrix. The same reduction is applicable to the intersection of algebraic plane curves.

In Section 2 we had reduced the problem of intersecting parametric curves  $\mathbf{P}(t)$  and  $\mathbf{Q}(u)$ , of degree  $m$  and  $n$  respectively, to finding roots of the matrix determinant as shown in (8). Each entry of the  $m \times m$  matrix  $\mathbf{M}(u)$  is a linear combination of Bernstein polynomials of degree  $n$  in  $u$ . A similar formulation has been obtained for

the intersection of algebraic plane curves  $F(x, y)$  and  $G(x, y)$ , of degree  $m$  and  $n$ , respectively, as shown in (9). Let us represent  $\mathbf{M}(u)$  as a matrix polynomial

$$\mathbf{M}(u) = \mathbf{M}_n u^n + \mathbf{M}_{n-1} u^{n-1} (1-u) + \mathbf{M}_{n-2} u^{n-2} (1-u)^2 + \dots + \mathbf{M}_0 (1-u)^n,$$

where  $\mathbf{M}_i$  is a matrix with numeric entries. On dividing this equation by  $(1-u)^n$  we obtain the polynomial

$$\mathbf{M}_n \left(\frac{u}{1-u}\right)^n + \mathbf{M}_{n-1} \left(\frac{u}{1-u}\right)^{n-1} + \mathbf{M}_{n-2} \left(\frac{u}{1-u}\right)^{n-2} + \dots + \mathbf{M}_0.$$

Substitute  $s = \frac{u}{1-u}$  and to get the new polynomial

$$\overline{\mathbf{M}}(s) = \mathbf{M}_n s^n + \mathbf{M}_{n-1} s^{n-1} + \dots + \mathbf{M}_0. \quad (14)$$

In the original problem we were interested in the roots of  $\text{Determinant}(\mathbf{M}(u)) = 0$  in the range  $[0, 1]$ . However, after reparametrizing we want to compute the roots of  $P(s) \equiv \text{Determinant}(\overline{\mathbf{M}}(s)) = 0$  in the domain  $[0, \infty]$ . This can result in scaling or overflow problems if the original system has a root  $u \approx 1$ . In such cases  $\mathbf{M}_n$  is nearly singular or ill-conditioned. Our algorithm takes care of such cases by performing linear transformations or using projective coordinates [16].

Let us consider the case when  $\mathbf{M}_n$  is a nonsingular and well conditioned matrix. As a result computation of  $\mathbf{M}_n^{-1}$  does not introduce severe numerical errors. Let

$$\mathbf{L}(s) = \mathbf{M}_n^{-1} \overline{\mathbf{M}}(s), \text{ and } \overline{\mathbf{M}}_i = \mathbf{M}_n^{-1} \mathbf{M}_i, \quad 0 \leq i < n.$$

$\overline{\mathbf{L}}(s)$  is a monic matrix polynomial. Its determinant has the same roots as that of  $P(s)$ . Let  $s = s_0$  be a root of the equation

$$\text{Determinant}(\overline{\mathbf{L}}(s)) = 0.$$

As a result  $\overline{\mathbf{L}}(s_0)$  is a singular matrix and there is at least one non trivial vector in its kernel. Let us denote that  $m \times 1$  vector as  $\mathbf{v}$ . That is

$$\overline{\mathbf{L}}(s_0) \mathbf{v} = \mathbf{0}, \quad (15)$$

where  $\mathbf{0}$  is a  $m \times 1$  null vector.

**Theorem 3.1** *Given the matrix polynomial,  $\overline{\mathbf{L}}(s)$  the roots of the polynomial corresponding to its determinant are the eigenvalues of the matrix*

$$\mathbf{C} = \begin{bmatrix} \mathbf{0} & \mathbf{I}_m & \mathbf{0} & \dots & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{I}_m & \dots & \mathbf{0} \\ \vdots & \vdots & \dots & \vdots & \vdots \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \dots & \mathbf{I}_m \\ -\overline{\mathbf{M}}_0 & -\overline{\mathbf{M}}_1 & -\overline{\mathbf{M}}_2 & \dots & -\overline{\mathbf{M}}_{n-1} \end{bmatrix}, \quad (16)$$

where  $\mathbf{0}$  and  $\mathbf{I}_m$  are  $m \times m$  null and identity matrices, respectively. Furthermore, the eigenvector of  $\mathbf{C}$  corresponding to the eigenvalue  $s = s_0$  are of the form:

$$[\mathbf{v} \ s_0 \mathbf{v} \ s_0^2 \mathbf{v} \ \dots \ s_0^{n-1} \mathbf{v}]^T,$$

where  $\mathbf{v}$  is the vector in the kernel of  $\bar{\mathbf{L}}(s_0)$  as shown in (15).

**Proof:** [16].

The relationship between the eigenvalues of  $\mathbf{C}$  and the roots of  $P(s)$  has been proved using similarity transformations in [37]. Many times the leading matrix  $\mathbf{M}_n$  is singular or close to being singular (due to high condition number). Some techniques based on linear transformations are highlighted in [16] such that the problem of finding roots of determinant of matrix polynomial can be reduced to an eigenvalue problem. However, there are cases where they do not work. In such cases, we reduce the intersection problem to a generalized eigenvalue problem.

**Theorem 3.2** *Given the matrix polynomial,  $\bar{\mathbf{L}}(s)$  the roots of the polynomial corresponding to its determinant are the eigenvalues of the generalized system  $\mathbf{C}_2 - \mathbf{C}_1 s$ , where*

$$\mathbf{C}_1 = \begin{bmatrix} \mathbf{I}_m & \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} \\ \mathbf{0} & \mathbf{I}_m & \mathbf{0} & \dots & \mathbf{0} \\ \vdots & \vdots & \dots & \vdots & \vdots \\ \mathbf{0} & \mathbf{0} & \dots & \mathbf{I}_m & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} & \mathbf{M}_n \end{bmatrix} \quad \mathbf{C}_2 = \begin{bmatrix} \mathbf{0} & \mathbf{I}_m & \mathbf{0} & \dots & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{I}_m & \dots & \mathbf{0} \\ \vdots & \vdots & \dots & \vdots & \vdots \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \dots & \mathbf{I}_m \\ -\mathbf{M}_0 & -\mathbf{M}_1 & -\mathbf{M}_2 & \dots & -\mathbf{M}_{n-1} \end{bmatrix}, \quad (17)$$

where  $\mathbf{0}$  and  $\mathbf{I}_m$  are  $m \times m$  null and identity matrices, respectively.

**Proof:** [16, 37].

### 3.9 Eigenvalue Formulation

In the previous section, we have shown how the problem of intersection can be reduced to an eigenvalue problem. This is based on eliminating variables and matrix computations. For higher degree curve intersections, the roots can also be computed by expanding the determinant and finding the roots of the resulting problem. However, the roots of high degree polynomials can be very ill-conditioned and hard to find [38]. Therefore, we use more reliable matrix computations to compute the roots. In particular, we use the QR algorithm, which computes all the eigenvalues using orthogonal similarity transformations (and optionally eigenvectors), and is guaranteed to find one

copy of each root of a matrix differing very slightly from the input matrix. One can show this is equivalent to finding one copy of each root of a polynomial differing very slightly from the polynomial in the bottom row [47, 48].

In most applications we are only interested in eigenvalues in a given domain. Until recently, there were no reliable algorithms other than QR, which must find all the eigenvalues, wanted or not. These recent algorithms [42] generally take more floating point operation than QR, but have the advantage of being easy to parallelize, which QR is not. Another class of parallel eigenvalue algorithm applies homotopy continuation to  $\text{Determinant}(\bar{\mathbf{L}}(s))$  [43, 44]. These algorithms, while parallelizable and often accurate, have the same drawbacks as homotopy methods applied directly to the polynomial systems: the difficulty of guaranteeing that all desired roots are computed exactly once, without taking arbitrarily small steps. See [49] for a discussion of parallel eigenvalue algorithms.

## 4 Higher Order Intersections

In the previous sections, we have shown how the problem of curve intersection can be reduced to eigendecomposition. Simple eigenvalues of the resulting matrix correspond to simple intersections. In this section, we present algorithms to robustly compute geometrically isolated higher order intersections.

According to Bezout's theorem two rational or algebraic curves of degree  $m$  and  $n$  intersect in  $mn$  points (counted properly) [20]. The *intersection multiplicity* of a point,  $\mathbf{p}$ , is defined in the following manner:

*An intersection point,  $\mathbf{p}$ , has multiplicity  $m$ , if a small perturbation in the coefficients of the curve(s) (the coefficients of the control polygon for rational parametric curves) results in at most  $m$  distinct intersection points in the neighborhood of  $\mathbf{p}$ .*

Such intersections arise due to the tangential intersection of the two curves at  $\mathbf{p}$  or if  $\mathbf{p}$  is a singular point on one of the curves. Some examples are highlighted in Fig. 2. They are:

- (a) Tangential intersection of two ellipses. The intersection multiplicity is 2.
- (b) Second order intersection of a parabola with a curve having a loop.
- (c) Intersection of an ellipse and a curve with a cusp. The intersection multiplicity is 2.
- (d) Tangential intersection of a parabola with a cubic curve having a loop. The multiplicity of intersection is 3.

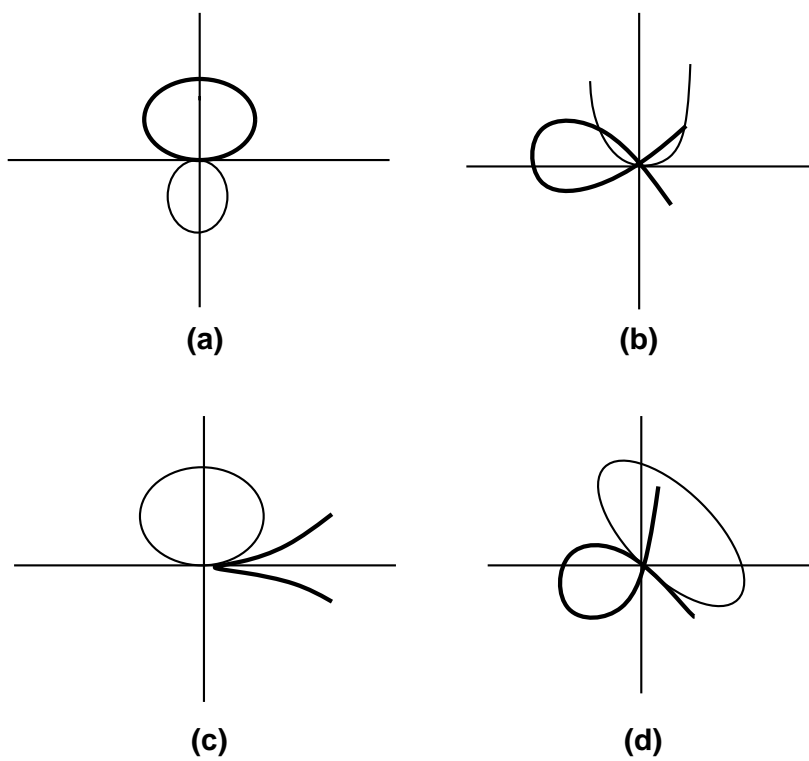


Figure 2: Higher order intersections

In the previous section we showed that the intersection points correspond to the  $mn$  eigenvalues of  $\mathbf{C}$  in (16) or the generalized eigenvalue problem,  $\mathbf{C}_2 - \mathbf{C}_1s$  in (17). As a result there is a direct relationship between the multiplicities of the eigenvalues and the intersection multiplicity of the point. Eigenvalues of multiplicity greater than one are the multiple roots of

$$\text{Determinant}(\mathbf{C} - s\mathbf{I}) = 0 \quad \text{or} \quad \text{Determinant}(\mathbf{C}_2 - \mathbf{C}_1s) = 0$$

and their multiplicity corresponds to the multiplicity of the roots. This multiplicity is also referred to as the *algebraic multiplicity* of the eigenvalue. The *geometric multiplicity* of an eigenvalue,  $s_0$ , is the dimension the vector space corresponding to the kernel of  $\mathbf{C} - s_0\mathbf{I}$  or  $\mathbf{C}_2 - \mathbf{C}_1s_0$ . It is bounded by the algebraic multiplicity of  $s_0$ . The relationship between algebraic and geometric multiplicities of the eigenvalue and that of the intersection point is different for intersection of parametric and algebraic curves.

- *Parametric Curve Intersection:* The eigenvalues correspond to the roots of (8), except in some cases we may have used a linear transformation and therefore they are the roots of a transformed equation. Each eigenvalue,  $s_0$ , corresponds to an intersection point,  $\mathbf{Q}(s_0)$ . As a result, higher multiplicity eigenvalues correspond to higher order intersections. In fact, the intersection multiplicity of  $\mathbf{Q}(s_0)$  is exactly equal to the algebraic multiplicity of the eigenvalue  $s_0$ . The nature of intersection, whether it is a singular point or tangential intersection, is determined from the algebraic and geometric multiplicities of the eigenvalue.
- *Algebraic Curve Intersection:* The eigenvalues correspond to the roots of (9). As a result, each eigenvalue  $s_0$  corresponds to the  $x$ -coordinate of the intersection point of the form  $(s_0, y)$ . It is possible that the curves intersect in two points of the form  $\mathbf{p}_1 = (s_0, y_1)$  and  $\mathbf{p}_2 = (s_0, y_2)$ . In such cases  $\mathbf{p}_1$  and  $\mathbf{p}_2$  may be simple points of intersection (having intersection multiplicity equal to one), whereas the  $s_0$  is an eigenvalue of multiplicity two at least. Thus, eigenvalues of multiplicity greater than one may or may not correspond to an intersection point of multiplicity greater than one. Such cases are again distinguished by the algebraic and geometric multiplicities of the eigenvalues.

We initially describe the algorithm for computing higher multiplicity eigenvalues and later on present a separate algorithm for computing the corresponding intersection points of parametric and algebraic curves.



## 4.1 Higher Multiplicity Eigenvalues

As such the problem of computing higher order roots of polynomial equations or eigenvalues can be numerically ill-conditioned. We highlight the problem with a classic example from matrix computations [36]. Let  $\mathbf{A}_\eta$  be a  $11 \times 11$  matrix of the form:

$$\mathbf{A}_\eta = \begin{pmatrix} 0 & 1 & 0 & \dots & 0 & 0 \\ 0 & 0 & 1 & \dots & 0 & 0 \\ \vdots & \vdots & \dots & \dots & \vdots & \vdots \\ 0 & 0 & \dots & 0 & 1 & 0 \\ \eta & 0 & 0 & \dots & 0 & 0 \\ 0 & 0 & \dots & 0 & 0 & 0.5 \end{pmatrix}. \quad (18)$$

$\mathbf{A}_\eta$  is a block diagonal matrix with a  $10 \times 10$  block at the upper left and a  $1 \times 1$  block at the lower right. If  $\eta = 0$ , the upper left block is a  $10 \times 10$  Jordan block [35]. As a result, it has a single eigenvalue,  $s = 0$ , of algebraic multiplicity 10 and geometric multiplicity 1. The corresponding eigenvector is  $\mathbf{v} = [1 \ 0 \ 0 \ \dots \ 0]^T$ . For small values of  $\eta$  the eigenvalues become distinct numbers all with absolute value  $\eta^{-1}$  and the eigenvectors which have rotated away from  $\mathbf{v}$  by about  $\eta^{-1}$  radians. For  $\eta = 10^{-10}$ ,  $\eta^{-1} = 0.1$ , which is a much larger change. As a result the eigenvalue of  $\mathbf{A}_\eta$  at  $\eta = 0$  and the corresponding eigenvector is an ill-posed problem<sup>4</sup>. Such ill-posed problems arise frequently whenever the curves have higher order intersections.

A better solution to this problem is obtained by considering the matrix to have a cluster of 10 eigenvalues near zero, and computing the *arithmetic mean* of this cluster instead of the individual eigenvalues. The arithmetic mean of a cluster of eigenvalues can be much less sensitive to small perturbations than the individual eigenvalues [39]; explicit expressions are given in equation (13). In particular, if a cluster arises from the perturbation of a single multiple eigenvalue, the mean may be a much more accurate approximation of the multiple eigenvalue than any of the computed eigenvalues.

Consider example (18). The mean of the eigenvalues of the leading 10-by-10 submatrix is exactly zero for any  $\eta$ , because this mean is .1 times the trace of this submatrix, which is clearly zero. The eigenvalues of this block lie evenly spaced on a circle of radius  $\eta^{-1}$  in just such a way that their sum cancels to zero.

Similar properties hold for the higher multiplicity roots of polynomial equations. That is, although the higher multiplicity roots are ill-conditioned, the average of a cluster of roots of the perturbed polynomial is stable [38].

---

<sup>4</sup>This is due to the fact that the sensitivity of the eigenvalue is not proportional to the norm of the perturbation  $\eta$ , but the tenth root of  $\eta$ .

We may apply this idea to computing accurate multiple curve intersections as follows. Suppose we are given two curves that intersect with multiplicity  $k$  at  $\mathbf{p}$ . We compute the implicit representation and reduce the problem to an eigenvalue problem. Let  $s$  be the multiple eigenvalue corresponding to  $\mathbf{p}$ . Since the reduction involves rounding errors, the resulting eigenproblem will not have a multiple eigenvalue at  $s$ , but rather a cluster of  $k$  eigenvalues near  $s$ . Often these clustered eigenvalues will lie on a circle centered at the original multiple eigenvalue. If we can correctly identify these clustered eigenvalues and compute their mean, this will be an accurate approximation of the multiple eigenvalue.

So the underlying problem is to cluster eigenvalues. This problem has been worked on for a long time, and is known to be very difficult to solve in general [45, 46]. For example, in example (18), setting  $\eta = 1/1024$  means that one of the eigenvalues in the “cluster” around zero equals .5. Should the “other” eigenvalue at .5 be part of the cluster or not? The answer is no, but that is hard to tell just from looking at the eigenvalues as points in the complex plane with no further information. We rely on the fact that the pathologies which make this problem hard in general are rare enough in practice that the following simple heuristic often works (otherwise, human intervention might be required at this point):

1. let  $\mathbf{C}$  (or  $\mathbf{C}_2 - \mathbf{C}_1\mathbf{s}$ ) be the matrix (or pencil) whose eigenvalues we want to cluster.
2. Take each computed eigenvalue  $s_i$ , and compute its condition number  $\|P_i\|$ , as in (12).
3. Round  $s_i$  to  $s'_i$ , where all decimal places of  $s'_i$  smaller than the error bound  $\epsilon\|P_i\| \cdot \|\mathbf{C}\|$  (or  $\epsilon\|P_i\| \cdot \|\mathbf{C}_1, \mathbf{C}_2\|$ ) are set to zero ( $\epsilon$  is the machine precision).
4. Put those  $s'_i$  that equal one another in their nonzero digits in the same cluster.
5. Take the arithmetic mean of the corresponding  $s_i$ .

To speed up the cluster identification process we sort the eigenvalues based on their magnitudes. The eigenvalues corresponding to the same cluster would appear next to each other in the sorted list. In the rest of this section we demonstrate this method on the intersection of parametric curves.

**Example 4.1** Consider the intersection of cubic curve,  $\mathbf{P}(t) = (x(t), y(t)) = (t^2 - 1, t^3 - t)$  with the parabola  $\mathbf{Q}(u) = (\bar{x}(u), \bar{y}(u)) = (u^2 + u, u^2 - u)$  (as shown in Fig. 3). The cubic curve has a loop at the origin. We are interesting in all intersection points corresponding to the domain  $t \times u = [-2, 2] \times [-1, 1]$ .

The implicit representation of  $\mathbf{P}(t)$  is a matrix determinant of the form

$$\mathbf{M} = \begin{pmatrix} -1 - x & -y & 1 + x \\ -y & x & 0 \\ 1 + x & 0 & -1 \end{pmatrix}.$$

After substituting and reducing the problem to an eigenvalue problem we obtain a  $6 \times 6$  matrix

$$\mathbf{C} = \begin{pmatrix} 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ -1 & 0 & 1 & -1 & 0 & 0 \\ -1 & 0 & 1 & -2 & -1 & 0 \\ -1 & 0 & 1 & -2 & -2 & -1 \end{pmatrix} \quad (19)$$

The eigenvalues of this matrix and their error estimates are:

Intersection Number ( $i$ )	Eigenvalue $s_i$	Cond. Number $C_i$	Accurate Digits $d_i$	Truncated Eigenvalue ( $s'_i$ )
1.	-2.58740105196	1.6402	15	-2.58740105196
2.	-0.20629947401 + j 1.374729636	1.38682	15	-0.20629947401 + j 1.374729636
3.	-0.20629947401 - - j 1.374729636	1.386829	15	-0.20629947401 - - j 1.374729636
4.	-8.8380412e-17 + j 2.7275325842e-9	1.79612e08	8	0.0
5.	-8.8380412e-17 - j 2.7275325842e-9	1.79612e08	8	0.0
6.	0.0	56.35	14	0.0

Thus, the fourth and fifth eigenvalues have a high condition number. They are represented as  $s_4 = j0.00000000272$  and  $s_5 = -j0.00000000272$ . However, they are accurate up to 8 digits only. As a result we find that  $s'_4 = s'_5 = s'_6 = 0$  belong to the same cluster. The arithmetic mean  $(s_4 + s_5 + s_6)/3 = 0$ , and the condition number of this mean is 26.82. As a result, the mean is correct up to about  $\pm 10^{-14}$  and therefore we take  $s' = 0.0$  to be an eigenvalue of (algebraic) multiplicity three.

The procedure highlighted above is used for computing the intersection multiplicity of the point. However, the higher order intersection can be due to a tangential intersection, one of the curves having a singular point at the point of intersection or their

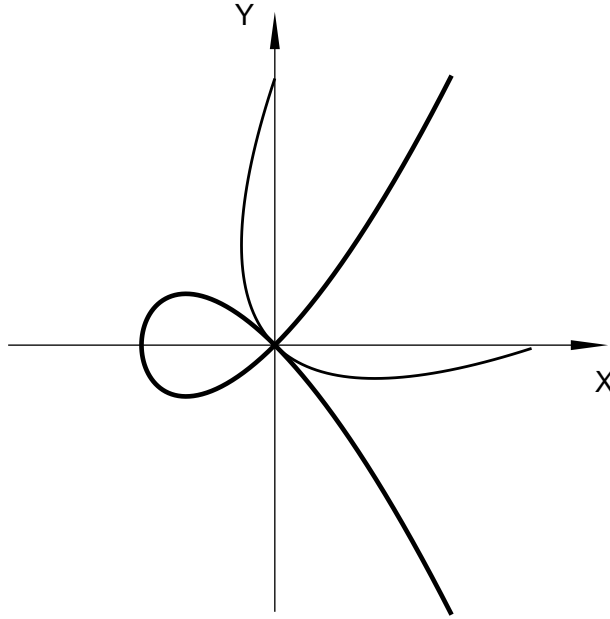


Figure 3: Higher order intersection of a cubic curve and a parabola

combinations. In the following sections we present techniques to compute the higher order intersection points and determine the nature of intersection. The algorithm is similar for parametric and algebraic curves.

## 4.2 Roots of Univariate Polynomials

In this section we present an algorithm which computes the common roots of two univariate polynomials. The algorithm is used for finding the intersection points corresponding to multiple intersections.

Given two polynomials,  $f(t)$  and  $g(t)$ , the number of common roots of these polynomials correspond to the roots of the polynomial corresponding to their greatest common divisor (GCD). However, computing the GCD is rather difficult when the coefficients of the polynomials are expressed in floating point. To circumvent the problem we propose finding all roots of  $f(t) = 0$  and determine which of them correspond to the roots of  $g(t)$ . We perform these procedures using matrix computations. Let us assume that  $f(t)$  and  $g(t)$  are expressed in power basis.

Given  $f(t) = f_n t^n + f_{n-1} t^{n-1} + \dots + f_1 t + f_0$ . Without loss of generality we assume that  $f_n = 1$ . It is a well known fact in linear algebra that the roots of  $f(t)$  are exactly

the eigenvalues of the companion matrix [35]:

$$\mathbf{F} = \begin{pmatrix} 0 & 1 & 0 & \dots & 0 & 0 \\ 0 & 0 & 1 & 0 & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \dots & 0 & 1 \\ -f_0 & -f_1 & -f_2 & \dots & -f_{n-2} & -f_{n-1} \end{pmatrix}.$$

A similar matrix  $\mathbf{G}$  can be formulated corresponding to  $g(t)$ . To compute the roots of  $f(t)$  we compute the eigenvalues of  $\mathbf{F}$  using the QR algorithm. It can also be used to accurately compute the roots of multiplicity greater than one by identifying the corresponding clusters and computing their arithmetic mean. Given an eigenvalue  $t = t_1$  of  $\mathbf{F}$  we might verify whether it is a root of  $g(t)$  by computing the rank of  $\mathbf{G} - t_1\mathbf{I}$  using singular value decompositions. However, it is much cheaper to use the following simple and nearly equivalent formula:

$$\bar{q} = \begin{cases} \frac{(1-|t_1|)|f(t_1)|}{1-|t_1|^{n+1}} & \text{if } |t_1| \neq 1 \\ \frac{|f(t_1)|}{n+1} & \text{otherwise.} \end{cases}.$$

It is easy to verify that

$$\bar{q} = \min\left\{\max_{0 \leq i \leq n} |q_i| : \sum_{i=0}^n (f_i + q_i)t_1^i = 0\right\},$$

i.e.  $\bar{q}$  is the smallest absolute perturbation to the coefficients of  $f(t)$  so that the perturbed polynomial has a zero at  $t_1$ .

To verify whether  $t_1$  is a root of multiplicity greater than one, we may compute the companion matrix  $\mathbf{G}'$  corresponding to the polynomial  $g'(t)$  and compute the rank of  $\mathbf{G}' - t_1\mathbf{I}$ . The order of  $\mathbf{G}'$  is one less than the order of  $\mathbf{G}$ .

### 4.3 Parametric Curve Intersection

Given the higher multiplicity eigenvalue  $s$  and the corresponding intersection point,  $\mathbf{p} = \mathbf{Q}(s) = (\bar{x}(s), \bar{y}(s), \bar{w}(s))$ , we are interested in computing the preimage of  $\mathbf{p}$  with respect to  $\mathbf{P}(t)$  and knowing the nature of intersection. The resulting algorithm depends on the geometric multiplicity of the eigenvalue. We are mainly interested in knowing whether the geometric multiplicity of the eigenvalue is one. The geometric multiplicity corresponds to the rank of  $\mathbf{C} - s\mathbf{I}$  or  $\mathbf{C}_2 - \mathbf{C}_1s$ . This can be accurately computed using singular value decompositions (SVD) of  $\mathbf{C} - s\mathbf{I}$  or  $\mathbf{C}_2 - \mathbf{C}_1s$ . The number of singular values equal to zero (or very close to zero) correspond to the geometric multiplicity of the eigenvalue.

Let us consider the case when the geometric multiplicity of  $s$  is one. According to Theorem 2.7,  $\mathbf{p}$  is not a singular point on  $\mathbf{P}(t)$ . The multiple intersection corresponds either to tangential intersection or to the fact that  $\mathbf{p}$  is a singular point on  $\mathbf{Q}(u)$ . The unique eigenvector corresponding to  $s$  is of the form  $k((1 - t_0)^{m-1} t_0(1 - t_0)^{m-2} \dots t_0^{m-1})^T$  and therefore, we can compute the preimage  $t_0$  from the eigenvector.

Theorem 2.5 can be used to determine whether  $\mathbf{p}$  is a singular point on  $\mathbf{Q}(u)$  or not. In particular we formulate the equations:

$$X_1 \bar{w}(u) - \bar{x}(u) = 0$$

$$Y_1 \bar{w}(u) - \bar{y}(u) = 0,$$

where  $(X_1, Y_1) = (\frac{\bar{x}(s)}{\bar{w}(s)}, \frac{\bar{y}(s)}{\bar{w}(s)})$ . The number of common roots of these two equations is computed by the algorithm described in the previous section.

If  $\mathbf{p}$  is not a singular point with respect to either of the curves, the two curves intersect tangentially (or with higher order continuity) [40].

Let us now consider the case when the geometric multiplicity of  $s$  is greater than one (say  $k$ ). As a result there are  $k$  linearly independent vectors spanning the eigenspace corresponding to  $s$ . The eigendecomposition routines compute a basis for this space which may or may not have a structure of the form  $k((1 - t_0)^{m-1} t_0(1 - t_0)^{m-2} \dots t_0^{m-1})^T$ . In fact, the eigenspace is spanned by vectors highlighted in Theorem 2.7.

To compute the  $t$  parameters we compute the roots of the equations

$$X_1 w(t) - x(t) = 0,$$

$$Y_1 w(t) - y(t) = 0$$

using the algorithm for univariate polynomials highlighted in the previous section.

**Example 4.2** *Let us again consider the intersection of a parabola and a cubic curve with a loop shown in Fig. 3. The curves are*

$$\mathbf{P}(t) = (t^2 - 1, t^3 - t), \quad \mathbf{Q}(u) = (u^2 + u, u^2 - u).$$

*In Example 4.1 we computed the entries of the matrix  $\mathbf{C}$ , as shown in (19), whose eigenvalues correspond to the intersection points. Based on the condition number of the eigenvalues we concluded that  $\lambda = 0$  is an eigenvalue of multiplicity three. Furthermore the intersection point is  $\mathbf{Q}(0) = (0.0, 0.0)$ . Let us compute the  $t$  values corresponding to this point.*

The singular value decomposition of  $\mathbf{C}$  results in two singular values equal to 0.0. As a result the geometric multiplicity of  $s = 0$  is two. The  $t$  values correspond to the roots of the equations

$$f(t) : t^2 - 1 = 0$$

$$g(t) : t^3 - t = 0.$$

The roots of  $f(t)$  are computed using the eigenvalues of the companion matrix

$$\mathbf{F} = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}.$$

The eigenvalues are  $-1.0$  and  $1.0$ . We verify whether they are roots of  $g(t)$  by computing the ranks of  $\mathbf{G} - \mathbf{I}$  and  $\mathbf{G} + \mathbf{I}$ , where  $\mathbf{G}$  is the companion matrix of  $g(t)$ ,

$$\mathbf{G} = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix}.$$

It turns out that both the matrices mentioned above are rank deficient and therefore,  $-1.0$  and  $1.0$  are the roots of  $f(t)$  as well as  $g(t)$ . Furthermore, they are roots of multiplicity one. As a result,  $\mathbf{P}(t)$  has a loop at the origin. The fact that the curves intersect with multiplicity 3 implies that  $\mathbf{Q}(u)$  is tangential to one of the branches of  $\mathbf{P}(t)$  at the origin.

## 4.4 Algebraic Curves

Given an eigenvalue  $s$  of multiplicity  $k$ . The eigenvalue corresponds to the  $X$  coordinate of intersection. In this section we present techniques to compute the  $Y$  coordinates of the point of intersection of the algebraic curves  $F(x, y)$  and  $G(x, y)$ . The analysis is very similar to that of intersecting parametric curve.

Let us consider the case when the geometric multiplicity of  $s$  is one. As a result the unique vector lying in the kernel is of the form  $k(1 \ y_0 \ y_0^2 \ \dots \ y_0^{m-1})^T$ . Therefore, we can compute the  $Y$  coordinate from the eigenvector. Furthermore the intersection multiplicity of the point  $(s, y_0)$  on each curve can be computed by the degrees of the lowest degree monomials of  $F(x - s, y - y_0)$  and  $G(x - s, y - y_0)$  (according to Lemma 2.1).

In case the geometric multiplicity of  $s$  is greater than one, the  $Y$  coordinates are computed by solving the univariate equations  $F(s, y) = 0$  and  $G(s, y) = 0$ . The intersection multiplicity of each point is computed in a similar fashion.

We illustrate the algorithm on the intersection of quartic algebraic curves.

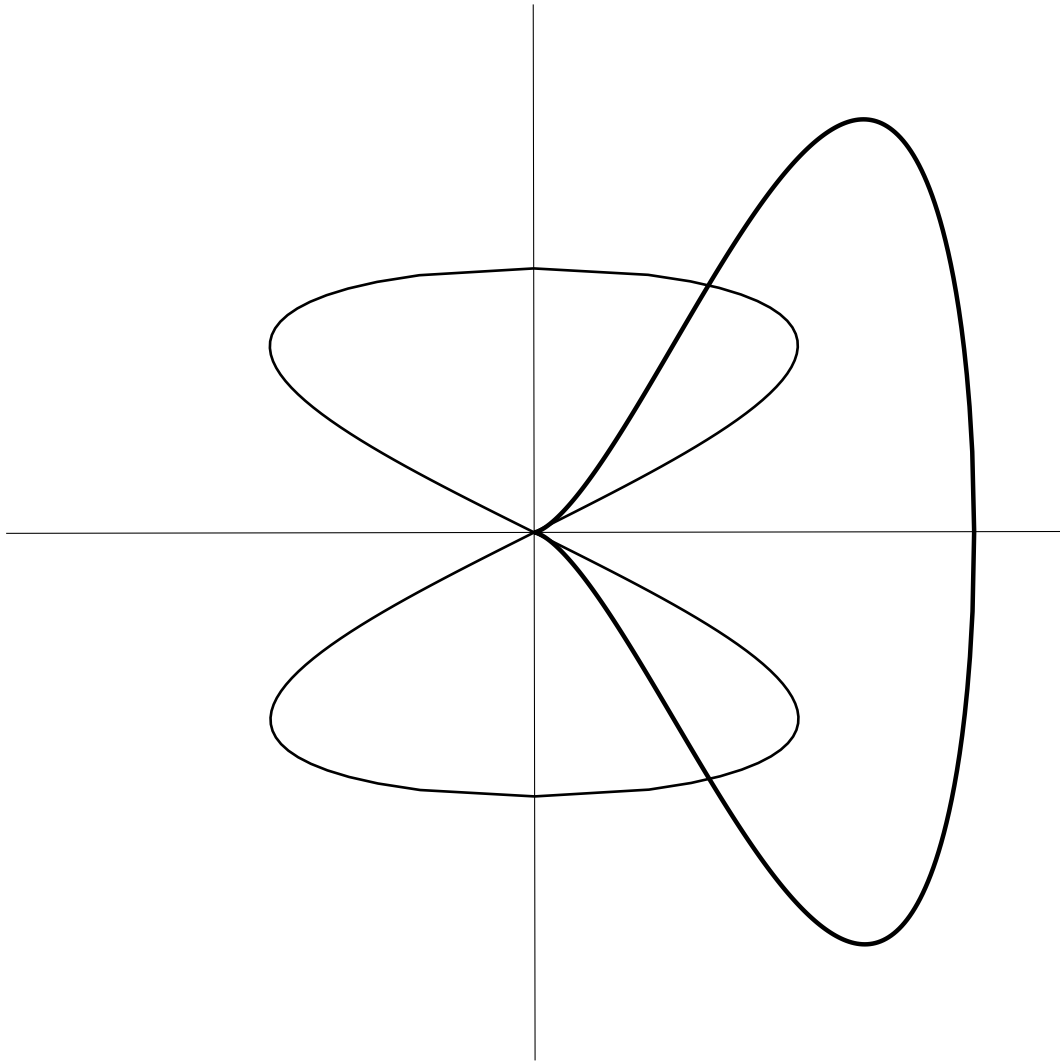


Figure 4: Intersection of quartic algebraic curves



**Example 4.3** Given

$$F(x, y) = x^2 + 4y^4 - 4y^2 = 0$$

and

$$G(x, y) = 3x^4 - 5x^3 + y^2 = 0.$$

These curves are shown in Fig. 4.  $P(x, y)$  is a nodal quartic having a self intersection at the origin, whereas  $Q(x, y)$  has a cusp at the origin. The intersection algorithm proceeds by eliminating  $x$  from these two equations using Cayley's resultant. In other words we treat them as polynomials in  $y$  and  $x$  is as a symbolic constant. The resulting matrix is:

$$\mathbf{M}(x) = \begin{pmatrix} 0 & -x^2 + 20x^3 - 12x^4 & 0 & -20x^3 + 12x^4 \\ -x^2 + 20x^3 - 12x^4 & 0 & -20x^3 + 12x^4 & 0 \\ 0 & -20x^3 + 12x^4 & 0 & 4 \\ -20x^3 + 12x^4 & 0 & 4 & 0 \end{pmatrix}.$$

This is expressed as a matrix polynomial

$$\mathbf{M}(x) = \mathbf{M}_0 + \mathbf{M}_1x + \mathbf{M}_2x^2 + \mathbf{M}_3x^3 + \mathbf{M}_4x^4,$$

where the leading matrix is

$$\mathbf{M}_4 = \begin{pmatrix} 0 & -12 & 0 & 12 \\ -12 & 0 & 12 & 0 \\ 0 & 12 & 0 & 0 \\ 12 & 0 & 0 & 0 \end{pmatrix}.$$

The condition number of  $\mathbf{M}_4$  is 2.6180. As a result we are able to reduce the problem of intersection to an eigenvalue problem. The resulting  $16 \times 16$  matrix is

$$\mathbf{C} = \begin{pmatrix} 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & -0.33 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1.66 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -0.33 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1.66 & 0 & 0 & 0 \\ 0 & 0 & -0.33 & 0 & 0 & 0 & 0 & 0 & 0.08 & 0 & 0 & 0 & 0 & 1.66 & 0 \\ 0 & 0 & 0 & -0.33 & 0 & 0 & 0 & 0 & 0 & 0.08 & 0 & 0 & 0 & 0 & 1.66 \end{pmatrix}.$$

(The nonzero entries are actually  $-1/3$ ,  $1/12$  and  $5/3$  rounded to machine precision.)  $\mathbf{C}$  has 3 distinct real eigenvalues. However, they are eigenvalues of multiplicities greater than one and account for 8 of the 16 eigenvalues of the matrix. In this case the QR algorithm computes the higher multiplicity eigenvalues accurately. They are:

- $s_1 = 0$  of multiplicity 4.
- $s_2 = 0.6620955$  of multiplicity 2.
- $s_3 = 0.051632$  of multiplicity 2,

Let us consider each one of them and compute the corresponding points of intersections. We start with  $s_1$ . The geometric multiplicity of  $s_1$  is two, computed using singular values of  $\mathbf{C}$ . To compute the corresponding  $y$  coordinate we substitute  $x = s_1 = 0$  in both the equations and solve for  $y$ .

$$F(0, y) = 4y^4 - 4y^2 = 0$$

and

$$G(0, y) = y^2 = 0.$$

Using the root finding algorithm (based on companion matrices) we find that  $y = 0$  is a root of multiplicity 2. Furthermore we find at the origin both  $F(x, y)$  and  $G(x, y)$  have multiplicity 2. Therefore, the two curves intersect with multiplicity four at the origin.

Let us consider  $s_2$ . The geometric multiplicity of  $s_2$  is 2, computed using the singular value decomposition of  $\mathbf{C} - s_2\mathbf{I}$ . To compute the  $Y$  coordinates we solve for

$$F(s_2, y) = 4y^4 - 4y^2 + 0.43837051834 = 0$$

and

$$G(s_2, y) = -0.87470971486 + y^2 = 0.$$

The roots of  $F(s_2, y)$  are the eigenvalues of

$$\begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ -0.109592 & 0 & 1 & 0 \end{pmatrix}.$$

The eigenvalues are  $0.9352589, -0.9352589, 0.35396437, -0.35396437$ . Among these four eigenvalues the first two are also the roots of  $G(s_2, y)$  and therefore the two points of intersection are

$$(x, y) = (0.66209555, 0.935259169)$$

and

$$(x, y) = (0.66209555, -0.935259169).$$

Each of them is a regular point on  $F(x, y)$  and  $G(x, y)$

A similar analysis hold for  $s_3$ . The two points of intersection are

$$(x, y) = (0.516329456, 0.0258250860)$$

and

$$(x, y) = (0.516329456, -0.0258250860).$$

They are also regular points.

We highlight the performance of the algorithm on some high degree curves.

**Example 4.4**<sup>5</sup> Let us consider the curve  $F(x, y) = 0$  defined in [41]:

$$F(x, y) = -2 - 7x + 14x^3 - 7x^5 + x^7 + (7 - 42x^2 + 35x^4 - 7x^6)y + (16 + 42x - 70x^3 + 21x^5)y^2 + (-14 + 70x^2 - 35x^4)y^3 + (-20 - 35x + 35x^3)y^4 + (7 - 21x^2)y^5 + (8 + 7x)y^6 - y^7 - y^8.$$

and let  $G(x, y) = F_y(x, y)$  be:

$$G(x, y) = 7 - 42x^2 + 35x^4 - 7x^6 + 2(16 + 42x - 70x^3 + 21x^5)y + 3(-14 + 70x^2 - 35x^4)y^2 + 4(-20 - 35x + 35x^3)y^3 + 5(7 - 21x^2)y^4 + 6(8 + 7x)y^5 - 7y^6 - 8y^7.$$

Computing the common roots of these two curves is one step in computing the singularities of  $F(x, y) = 0$ . The curves are shown in Fig. 5.

To compute their intersections, we consider them as polynomials in  $y$  and compute the resultant. After eliminating, we express as a matrix polynomial in  $x$  as:

$$\mathbf{M}(x) = \mathbf{M}_0 + \mathbf{M}_1x + \mathbf{M}_2x^2 + \mathbf{M}_3x^3 + \mathbf{M}_4x^4 + \mathbf{M}_5x^5 + \mathbf{M}_6x^6 + \mathbf{M}_7x^7 + \mathbf{M}_8x^8 + \mathbf{M}_9x^9 + \mathbf{M}_{10}x^{10} + \mathbf{M}_{11}x^{11} + \mathbf{M}_{12}x^{12},$$

where

$$\mathbf{M}_{12} = \begin{pmatrix} 7 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}, \mathbf{M}_{11} = \begin{pmatrix} 0 & -42 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -42 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

---

<sup>5</sup>This example was suggested to us by one of the reviewers.

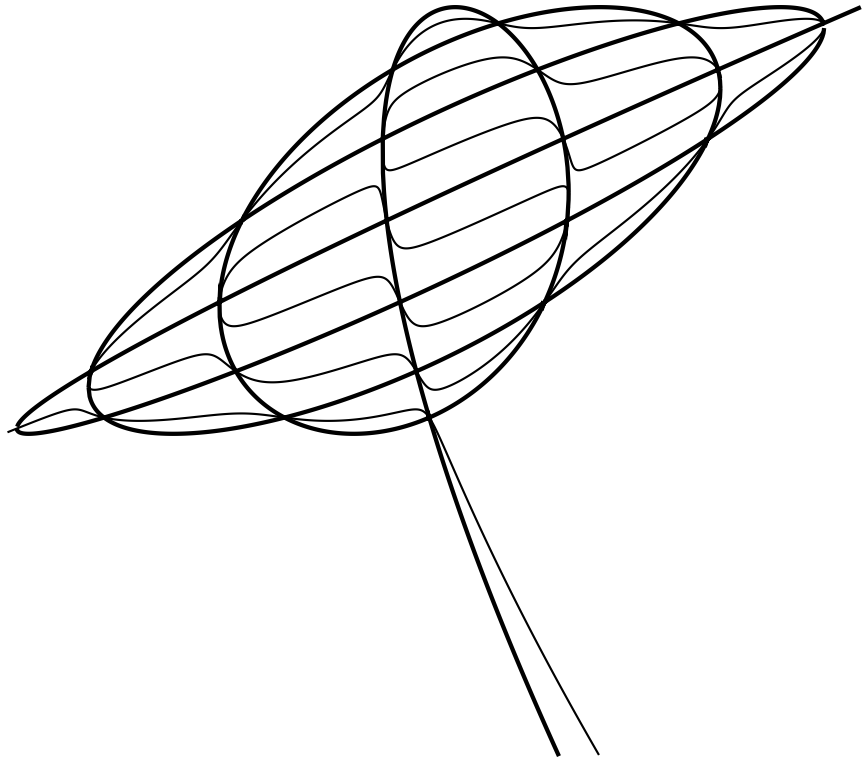


Figure 5: Intersection of high degree curves (of degree eight and seven)

$$\begin{aligned}
\mathbf{M}_{10} &= \begin{pmatrix} -56 & 0 & 105 & 0 & 0 & 0 & 0 & 0 \\ 0 & 252 & 0 & 0 & 0 & 0 & 0 & 0 \\ 105 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}, \quad \mathbf{M}_9 = \begin{pmatrix} 0 & 280 & 0 & -140 & 0 & 0 & 0 & 0 \\ 280 & 0 & -630 & 0 & 0 & 0 & 0 & 0 \\ 0 & -630 & 0 & 0 & 0 & 0 & 0 & 0 \\ -140 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \\
\mathbf{M}_8 &= \begin{pmatrix} 161 & 0 & -595 & 0 & 105 & 0 & 0 & 0 \\ 0 & -1330 & 0 & 840 & 0 & 0 & 0 & 0 \\ -595 & 0 & 1575 & 0 & 0 & 0 & 0 & 0 \\ 0 & 840 & 0 & 0 & 0 & 0 & 0 & 0 \\ 105 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}, \quad \mathbf{M}_7 = \begin{pmatrix} -32 & -644 & 80 & 700 & -48 & -42 & 8 & 0 \\ -644 & 80 & 2660 & -48 & -630 & 8 & 0 & 0 \\ 80 & 2660 & -48 & -2100 & 8 & 0 & 0 & 0 \\ 700 & -48 & -2100 & 8 & 0 & 0 & 0 & 0 \\ -48 & -630 & 8 & 0 & 0 & 0 & 0 & 0 \\ -42 & 8 & 0 & 0 & 0 & 0 & 0 & 0 \\ 8 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \\
\mathbf{M}_6 &= \begin{pmatrix} -196 & -112 & 1078 & 140 & -490 & -56 & 7 & 7 \\ -112 & 2352 & -420 & -2940 & 280 & 252 & -49 & 0 \\ 1078 & -420 & -4900 & 280 & 1575 & -49 & 0 & 0 \\ 140 & -2940 & 280 & 2800 & -49 & 0 & 0 & 0 \\ -490 & 280 & 1575 & -49 & 0 & 0 & 0 & 0 \\ -56 & 252 & -49 & 0 & 0 & 0 & 0 & 0 \\ 7 & -49 & 0 & 0 & 0 & 0 & 0 & 0 \\ 7 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \\
\mathbf{M}_5 &= \begin{pmatrix} 308 & 588 & -560 & -980 & 336 & 196 & -56 & 0 \\ 588 & 784 & -3528 & -504 & 1960 & 280 & -42 & -42 \\ -560 & -3528 & 1176 & 4900 & -728 & -630 & 126 & 0 \\ -980 & -504 & 4900 & -728 & -2100 & 126 & 0 & 0 \\ 336 & 1960 & -728 & -2100 & 126 & 0 & 0 & 0 \\ 196 & 280 & -630 & 126 & 0 & 0 & 0 & 0 \\ -56 & -42 & 126 & 0 & 0 & 0 & 0 & 0 \\ 0 & -42 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \\
\mathbf{M}_4 &= \begin{pmatrix} 98 & 350 & -735 & -700 & 539 & 280 & -35 & -35 \\ 350 & -1470 & 980 & 2744 & -1400 & -770 & 245 & 0 \\ -735 & 980 & 4704 & 700 & -2975 & -595 & 105 & 105 \\ -700 & 2744 & 700 & -4200 & 1085 & 840 & -175 & 0 \\ 539 & -1400 & -2975 & 1085 & 1575 & -175 & 0 & 0 \\ 280 & -770 & -595 & 840 & -175 & 0 & 0 & 0 \\ -35 & 245 & 105 & -175 & 0 & 0 & 0 & 0 \\ -35 & 0 & 105 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \\
\mathbf{M}_3 &= \begin{pmatrix} -728 & -196 & 1400 & 490 & -672 & -196 & 112 & 0 \\ -196 & -3080 & 1470 & 3248 & -1176 & -1008 & 140 & 140 \\ 1400 & 1470 & -4592 & -3136 & 2352 & 1120 & -420 & 0 \\ 490 & 3248 & -3136 & -3248 & 2100 & 700 & -140 & -140 \\ -672 & -1176 & 2352 & 2100 & -980 & -630 & 140 & 0 \\ -196 & -1008 & 1120 & 700 & -630 & 140 & 0 & 0 \\ 112 & 140 & -420 & -140 & 140 & 0 & 0 & 0 \\ 0 & 140 & 0 & -140 & 0 & 0 & 0 & 0 \end{pmatrix}
\end{aligned}$$

$$\mathbf{M}_2 = \begin{pmatrix} 0 & -252 & 98 & 630 & -147 & -336 & 42 & 42 \\ -252 & 392 & -490 & -882 & 1008 & 336 & -294 & 0 \\ 98 & -490 & -882 & -1512 & 924 & 1386 & -210 & -210 \\ 630 & -882 & -1512 & 1904 & -294 & -700 & 350 & 0 \\ -147 & 1008 & 924 & -294 & -700 & -490 & 105 & 105 \\ -336 & 336 & 1386 & -700 & -490 & 252 & -63 & 0 \\ 42 & -294 & -210 & 350 & 105 & -63 & 0 & 0 \\ 42 & 0 & -210 & 0 & 105 & 0 & 0 & 0 \end{pmatrix}$$

$$\mathbf{M}_1 = \begin{pmatrix} 392 & 0 & -840 & 0 & 420 & 0 & -56 & 0 \\ 0 & 1848 & -196 & -2380 & 294 & 840 & -84 & -84 \\ -840 & -196 & 3220 & 294 & -1848 & -84 & 252 & 0 \\ 0 & -2380 & 294 & 3752 & -476 & -1428 & 140 & 140 \\ 420 & 294 & -1848 & -476 & 1092 & 140 & -140 & 0 \\ 0 & 840 & -84 & -1428 & 140 & 532 & -42 & -42 \\ -56 & -84 & 252 & 140 & -140 & -42 & 14 & 0 \\ 0 & -84 & 0 & 140 & 0 & -42 & 0 & 0 \end{pmatrix}$$

$$\mathbf{M}_0 = \begin{pmatrix} 113 & 28 & -258 & -70 & 145 & 42 & -23 & -7 \\ 28 & 548 & 42 & -740 & -70 & 282 & 17 & -32 \\ -258 & 42 & 1128 & 210 & -780 & -207 & 138 & 42 \\ -70 & -740 & 210 & 1310 & -95 & -600 & 10 & 80 \\ 145 & -70 & -780 & -95 & 605 & 150 & -115 & -35 \\ 42 & 282 & -207 & -600 & 150 & 318 & -27 & -48 \\ -23 & 17 & 138 & 10 & -115 & -27 & 23 & 7 \\ -7 & -32 & 42 & 80 & -35 & -48 & 7 & 8 \end{pmatrix}$$

It turns out that  $\mathbf{M}_{12}$  is singular. So we perform a transformation  $x = 1/z$  and express the resulting matrix polynomial in  $z$  with the leading matrix being  $\mathbf{M}_0$ . Its estimated condition number is  $1.416e03$  and we reduce the problem to a  $96 \times 96$  eigenvalue problem. The computed eigenvalues of the resulting matrix together with their accuracy are listed in Table 1.

Based on accurate digits we formulate clusters, verify the condition numbers of the arithmetic mean of the clusters and the geometric multiplicity of the resulting eigenvalue. The geometric multiplicities are computed by SVD of the resulting  $8 \times 8$  matrices,  $\mathbf{M}(x')$ . The common intersections of  $F(x, y)$  and  $G(x, y)$  along with their algebraic and geometric multiplicities are listed in Table 2.

The overall algorithm takes about 3.0 seconds on a DEC 5000/25. It takes about one second for eigendecomposition and the rest of the time is spent in computing the geometric multiplicities.

## 4.5 Robustness of the Algorithm

The robustness and accuracy of the algorithm is a function of the accuracy of the eigendecomposition algorithms and the identification of the right "cluster". In practice, the algorithm works well on geometrically isolated higher order intersections. However, in theory it is possible that two higher order intersections are close to each other. Moreover, the individual eigenvalues can be perturbed to an extent that the

Number	Real(z)	Imag(z)	Accurate Digits
1	-21.8239255889146	0.0000000000000	8
2	-21.8238697532316	0.0000000000000	8
3	-5.9796467657532	0.0000075722219	5
4	-5.9796467657532	-0.0000075722219	5
5	-3.1218320398262	0.0000000000000	9
6	-3.1218268062641	0.0000000000000	9
7	1.6645043055005	0.0000018167478	5
8	1.6645043055005	-0.0000018167478	5
9	-2.5791530900720	0.0000003108990	6
10	-2.5791530900720	-0.0000003108990	6
11	-2.2469801054210	0.0000000000000	9
12	-2.2469791020081	0.0000000000000	9
13	-2.0763570376449	0.0000019668920	5
14	-2.0763570376449	-0.0000019668920	5
15	-2.0670627378248	0.0000000000000	7
16	1.0318107362316	0.0000000000000	9
17	1.0318071051056	0.0000000000000	9
18	0.9647193722165	0.0000007378748	6
19	0.9647193722165	-0.0000007378748	6
20	-0.7129033739529	0.0000000000000	8
21	-0.7129007753810	0.0000000000000	8
22	0.8261672053897	0.0000013735269	5
23	0.8261672053897	-0.0000013735269	5
24	0.8019380319127	0.0000000000000	7
25	0.8019374396994	0.0000000000000	7
26	0.7929861451499	0.0000000000000	7
27	-0.5378497133300	0.0000026805337	5
28	-0.5378497133300	-0.0000026805337	5
29	-0.5549586052974	0.0000000000000	6
30	-0.5549576588689	0.0000000000000	6
31	-0.4969323219612	0.0000007195077	6
32	-0.4969323219612	-0.0000007195077	6
33	-0.4967419296039	0.0000000000000	7
34	0.4361477639814	0.0000045890473	5
35	0.4361477639814	-0.0000045890473	5
36	0.3895135783076	0.0000048132530	5
37	0.3895135783076	-0.0000048132530	5
38	0.3757626012196	0.0000000000000	7
39	0.3757799840033	0.0000000000000	7
40	0.3726073401953	0.0000000000000	8
41	-0.2555596084196	0.0000000000000	8
42	-0.3231347455170	0.0000000000000	9
42	-0.3231233746709	0.0000000000000	9
44	-0.3109306454191	0.0000038268042	5
45	-0.3109306454191	-0.0000038268042	5
46	0.2739953662842	0.0000064454115	5
47	0.2739953662842	-0.0000064454115	5
48	0.2735391968366	0.0000000000000	9
49	-0.3086572987647	0.0000000000000	9

Table 1: Computed eigenvalues in the domain

Number	X	Y	Alg. Mult.	Geom. Mult.
1	-0.0458212706017	-1.8477591513492	2	1
2	-0.1672339586558	-1.4142135624500	2	1
3	-0.3203247283142	-0.7653674555782	2	1
4	0.6007794613059	1.8477592095096	2	1
5	-0.3877241734309	1.4142136289805	2	1
6	-0.4450417685441	-0.0000004115470	2	1
7	-0.4816127389797	0.7653667938940	2	1
8	-0.4837782529292	0.6306924292553	1	1
9	0.9691699891128	1.4142124123740	2	1
10	1.0365708710736	-0.7653668640997	2	1
11	-1.4027197536228	-1.8477587227966	2	1
12	1.210408732610	0.7653668539221	2	1
13	1.2469791432823	-0.0000022840601	2	1
14	1.2610560803820	0.2653593693027	1	1
15	-1.8592554299390	-1.4142047410167	2	1
16	-1.8019361993030	0.0000020353240	2	1
17	2.0123464620966	-0.7653699775488	2	1
18	-2.0131177587473	-0.8121033532967	1	1
19	2.2928009325820	1.8477590518994	2	1
20	2.5673045965298	0.7653673197271	2	1
21	2.6612547303920	1.4141103483404	2	1
22	2.6837903930604	1.2336947226103	1	1
23	-3.9129814221585	-1.9506520785163	1	1
24	-3.0947931297711	-1.8477468361254	2	1
25	-3.2161513016901	-1.4142157538389	2	1
26	3.6496967578744	1.8478073904536	2	1
27	3.6557831987693	1.8039829438812	1	1
28	-3.2398391484735	-1.5636774950376	1	1

Table 2: Intersections and their multiplicities



algorithm cannot identify it in the right cluster. However, based on the condition numbers of the mean and the algebraic and geometric multiplicities, we can identify such cases. One possible solution is to enumerate all possible combinations of clusters (by partitioning the perturbed eigenvalues into different sets) and compute the condition numbers of their arithmetic mean. The complexity of this approach is an exponential function of the number of eigenvalues in the cluster and it is therefore, useful for low multiplicity cases only.

## 5 Conclusion

In this paper we have highlighted a new algorithm for computing the higher order intersections of parametric and algebraic curves. Earlier algorithms were robust for simple intersections only. The algorithm involves use of resultants to represent the implicit representation of a parametric curve as a matrix determinant. The intersection problem is being reduced to an eigenvalue problem. There is a nice relationship between the algebraic and geometric multiplicities of the eigenvalues and the order of intersection. We used this relationship in accurately computing such intersections. Efficient implementations of eigenvalue routines are available as part of linear algebra packages and we used them in our implementations. The approach presented here is also useful for computing higher order curve-surface intersections.

## 6 Acknowledgements

We are grateful to the reviewers for their comments on the paper. The example 4.4 was recommended by one of the reviewers as well.

## References

- [1] C.M. Hoffmann. *Geometric and Solid Modeling*. Morgan Kaufmann, San Mateo, California, 1989.
- [2] T.W. Sederberg and S.R. Parry. Comparison of three curve intersection algorithms. *Computer-Aided Design*, 18(1):58–63, 1986.
- [3] G. Elber and E. Cohen. Hidden curve removal for free form surfaces. *Computer Graphics*, 24(4):95–104, 1990.

- [4] G.E. Collins and W. Krandick. An efficient algorithm for infallible polynomial complex root isolation. In *Proceedings of International Symposium on Symbolic and Algebraic Computation*, pages 189–194, Berkeley, California, 1992.
- [5] R.T. Farouki and T. Sakkalis. Singular points of algebraic curves. *Journal of Symbolic Computation*, 9(4):457–483, 1990.
- [6] T.W. Sederberg. *Implicit and Parametric Curves and Surfaces*. PhD thesis, Purdue University, 1983.
- [7] J.M. Lane and R.F. Riesenfeld. A theoretical development for the computer generation and display of piecewise polynomial surfaces. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2(1):150–159, 1980.
- [8] P.A. Koparkar and S. P. Mudur. A new class of algorithms for the processing of parametric curves. *Computer-Aided Design*, 15(1):41–45, 1983.
- [9] J.H. Wilkinson. The evaluation of the zeros of ill-conditioned polynomials. parts i and ii. *Numer. Math.*, 1:150–166 and 167–180, 1959.
- [10] T.W. Sederberg. Algorithms for algebraic curve intersection. *Computer-Aided Design*, 21(9):547–555, 1989.
- [11] B. Buchberger. Groebner bases: An algorithmic method in ideal theory. In N.K. Bose, editor, *Multidimensional Systems Theory*, pages 184–232. D. Reidel Publishing Co., 1985.
- [12] C.B. Garcia and W.I. Zangwill. Finding all solutions to polynomial systems and other systems of equations. *Math. Prog.*, 16:159–176, 1979.
- [13] J. Snyder. Interval arithmetic for computer graphics. In *Proceedings of ACM Siggraph*, pages 121–130, 1992.
- [14] D. Manocha. *Algebraic and Numeric Techniques for Modeling and Robotics*. PhD thesis, Computer Science Division, Department of Electrical Engineering and Computer Science, University of California, Berkeley, May 1992.
- [15] R.P. Markot and R. L. Magedson. Solutions of tangential surface and curve intersections. *Computer-Aided Design*, 21(7):421–427, 1989.
- [16] D. Manocha and J. Demmel. Algorithms for intersecting parametric and algebraic curves. Technical Report UCB/CSD 92/698, Computer Science Division, University of California at Berkeley, 1992.

- [17] D. Manocha and J. Demmel. Algorithms for intersecting parametric and algebraic curves. In *Graphics Interface '92*, pages 232–241, 1992.
- [18] D. Manocha. Robust techniques for curve and surface intersections. In *Proceedings of SPIE Conference on Curves and Surfaces in Computer Vision and Graphics III*, pages 58–69, 1992.
- [19] R.H. Bartels, J.C. Beatty, and B.A. Barsky. *An Introduction to Splines for use in Computer Graphics & Geometric Modeling*. Morgan Kaufmann, San Mateo, California, 1987.
- [20] R.J. Walker. *Algebraic Curves*. Princeton University Press, New Jersey, 1950.
- [21] J.G. Semple and G.T. Kneebone. *Algebraic Curves*. Oxford University Press, London, 1959.
- [22] S. S. Abhyankar. What is the difference between a parabola and a hyperbola. *The Mathematical Intelligencer*, 10(4), 1988.
- [23] D. Manocha and J.F. Canny. Rational curves with polynomial parametrization. *Computer-Aided Design*, 23(9):645–652, 1991.
- [24] D. Manocha and J.F. Canny. Detecting cusps and inflection points in curves. *Computer Aided Geometric Design*, 9:1–24, 1992.
- [25] F.S. Macaulay. On some formula in elimination. *Proceedings of London Mathematical Society*, pages 3–27, May 1902.
- [26] G. Salmon. *Lessons Introductory to the Modern Higher Algebra*. G.E. Stechert & Co., New York, 1885.
- [27] B. Sturmfels. Sparse elimination theory. In D. Eisenbud and L. Robbiano, editors, *Computational Algebraic Geometry and Commutative Algebra*. Cambridge University Press, 1991.
- [28] D. Manocha. Regular curves and proper parametrizations. In *Proceedings of International Symposium on Symbolic and Algebraic Computations*, pages 271–276, 1990.
- [29] T.W. Sederberg. Improperly parametrized rational curves. *Computer Aided Geometric Design*, 3:67–75, 1986.
- [30] R.T. Farouki and V.T. Rajan. On the numerical condition of polynomials in bernstein form. *Computer Aided Geometric Design*, 4:191–216, 1987.

- [31] G.H. Golub and C.F. Van Loan. *Matrix Computations*. John Hopkins Press, Baltimore, 1989.
- [32] B.S. Garbow, J.M. Boyle, J. Dongarra, and C.B. Moler. *Matrix Eigensystem Routines – EISPACK Guide Extension*, volume 51 of *Lecture Notes in Computer Science*. Springer-Verlag, Berlin, 1977.
- [33] J. Demmel. LAPACK: A portable linear algebra library for supercomputers. In *Proceedings of the 1989 IEEE Control Systems Society Workshop on Computer-Aided Control System Design*, Tampa, FL, Dec 1989. IEEE.
- [34] E. Anderson, Z. Bai, C. Bischof, J. Demmel, J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, and D. Sorensen. *LAPACK User's Guide, Release 1.0*. SIAM, Philadelphia, 1992.
- [35] J.H. Wilkinson. *The algebraic eigenvalue problem*. Oxford University Press, Oxford, 1965.
- [36] Z. Bai, J. Demmel, and A. McKenney. On the conditioning of the nonsymmetric eigenproblem: Theory and software. Computer Science Dept. Technical Report 469, Courant Institute, New York, NY, October 1989. (LAPACK Working Note #13).
- [37] I. Gohberg, P. Lancaster, and L. Rodman. *Matrix Polynomials*. Academic Press, New York, 1982.
- [38] J.H. Wilkinson. *Rounding Errors in Algebraic Processes*. Prentice-Hall, Englewood Cliffs, New Jersey, 1963.
- [39] T. Kato. *Perturbation Theory for Linear Operators*. Springer Verlag, Berlin, 2 edition, 1980.
- [40] Tony D. Derosé. *Geometric Continuity: A Parametrization independent measure of continuity for computer aided geometric design*. PhD thesis, University of California at Berkeley, 1985.
- [41] T. Riabokin. Technical Report Research Report #86, Department of Aero Engineering, Univ. of Minnesota, Institute of Technology, 1952. Dewey Decimal Call Number 516.54 R35 O.
- [42] Z. Bai and J. Demmel. *Design of a parallel nonsymmetric eigenroutine toolbox, Part I*. in *Proceedings of the Sixth SIAM Conference on Parallel Processing for Scientific Computing*, SIAM, 1993.

- [43] T.-Y. Li and Z. Zeng. Homotopy-determinant algorithm for solving nonsymmetric eigenvalue problems. *Math. Comp.* 59:483–502, 1992.
- [44] T.-Y. Li, Z. Zeng and L. Cong. Solving eigenvalue problems of nonsymmetric matrices with real homotopies. *SIAM J. Num. Anal.* 29:229–248, 1992.
- [45] J. Demmel. *Nearest Defective Matrices and the Geometry of Ill-Conditioning*. in Reliable Numerical Computation, M. Cox and S. Hammarling eds., Clarendon Press, 1990.
- [46] L. Reichel and L. N. Trefethen. Eigenvalues and pseudo-eigenvalues of Toeplitz matrices. *Lin. Alg. Appl.*, 162–164:153–185, 1992.
- [47] P. Van Dooren and P. Dewilde. The eigenstructure of an arbitrary polynomial matrix: computational aspects. *Lin. Alg. Appl.*, 50:545–579, 1983.
- [48] A. Edelman and H. Murakami. Polynomial roots from companion matrix eigenvalues. submitted to *Math. Comp.*, 1993.
- [49] J. Demmel, M. Heath and H. van der Vorst. *Parallel Numerical Linear Algebra*. in Acta Numerica, volume 2, ed. A. Iserles, Cambridge University Press, 1993.