Algorithms for Intersecting Parametric and Algebraic Curves

Dinesh Manocha[1]

Computer Science Division
University of California at Berkeley
Berkeley, CA 94720

James Demmel[2]

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

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. The former is however, 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. The matrix itself rather than its symbolic determinant, a polynomial, is used as the representation. The algorithm for intersection corresponds to substituting the other equation to construct an equivalent matrix such that the intersection points can be extracted from the eigenvalues and eigenvectors of the latter. Moreover, the algebraic and geometric multiplicities of the eigenvalues give us information about the intersection (mutiplicity, tangential intersection etc.). As a result we are able to accurately compute higher order intersections in most cases. The main advantage of this approach lies in its *efficiency and robustness.* Moreover, the numerical accuracy of these operations is well understood. For almost all cases we are able to compute accurate answers in 64 bit IEEE floating point arithmetic.

[1]Supported by IBM Fellowship, David and Lucile Packard Fellowship and National Science Foundation grant ASC-9005933.

[2]Supported in part by National Science Foundation grants DCR-8552474, ASC-8715728 and ASC-9005933

1 Introduction

The problems of computing the intersection of parametric and algebraic curves are fundamental to geometric and solid modeling. Parametric curves, like B-splines and Bézier curves, are extensively used in the modeling systems and algebraic plane curves are becoming popular as well [Hof89, MM89, SP86, Sed89]. 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 etc. [EC90, CK92]. Algorithms for computing the intersection of these curves have been extensively studied in the literature.

As far as computing the intersection of rational parametric curves is concerned, algorithms based on implicitization [Sed83], Bézier subdivision [LR80] and interval arithmetic [KM83] are well known. The implicitization approach is based on the fact that every rational parametric curve can be implicitized into an algebraic plane curve of the form $F(x,y) = 0$, where $F(x,y)$ is a bivariate polynomial. Algorithms for implicitization make use of resultants and the computation involves expanding a symbolic determinant [Sed83]. Given the implicit representation of one curve, substitute the second parametrization and obtain a univariate polynomial in its parameter. The problem of intersection corresponds to computing the roots of the resulting polynomial. The Bézier subdivision relies on the convex hull property of Bézier curves and de Casteljau algorithm for subdividing Bézier curves. The intersection algorithm proceeds by comparing the convex hulls of the two curves. If they do not overlap, the curves do not intersect. Otherwise the curves are subdivided and the resulting convex hulls are checked for intersection. At each iteration the algorithm rejects regions of the curve that do not contain intersection points. With each subdivision, the new curve segments become increasingly better approximated by a straight line. After the two curves segments are approximated by straight lines up to certain tolerance, their intersection point is accepted as the intersection of two curves. The algorithm performs a linearly converging binary search. It has been improved by [SWZ89] by more effective use of the convex hull property. The resulting algorithm has the flavor of a geometrically based interval Newton method and has better convergence behavior. The interval arithmetic approach uses an idea similar to subdivision. Each curve is preprocessed to determine its vertical and horizontal tangents, and the curve is divided into 'intervals' which have vertical or horizontal tangents only at the endpoints. Thus, the convex hull is a rectangular bounding box and the subdivision amounts to evaluating the coordinate of the midpoint of the interval and defining the resulting rectangles. The rest is similar to subdivision.

The relative performance and accuracy of these algorithms is highlighted in [SP86]. In particular, implicitization based approaches are considered faster than other intersection algorithms for curves of degree up to four. This includes faster subdivision based algorithms [SWZ89]. 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. As far as computation of implicit representation is concerned, stable algorithms are available for curves of degree up to three [Hob91]. Furthermore, for curves of degree up to three, the entries of the matrix are represented as polynomials in power basis and the roots of its determinant are computed using a standard polynomial solver, such as Jenkins–Traub [SP86]. For curves of degree greater than three, the resulting univariate polynomial has degree 16 or higher. The problem of computing real roots of such high degree polynomials is frequently ill–conditioned [Wil59]. Furthermore, converting from power to Bernstein basis introduces numerical errors [FR87]. As a result the algorithm involves representing matrix entries as linear combinations of Bernstein polynomials, multiplying Bernstein polynomials for expanding the determinant and using subdivision for computing the roots of the resulting polynomial. These have a considerable effect on the efficiency of the resulting algorithms and therefore, algorithms based on subdivision perform better.

The algorithms for algebraic curve intersection are analogous to those of intersecting parametric curves. Resultants can be 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 [Sed89]. However, resultant based algorithms are considered to be the fastest for lower degree curves.

In many applications, the intersection may be of higher order involving tangencies and singular points. Such instance are rather common in industrial applications [MM89]. Most algorithms require special handling for tangencies and thereby requiring 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 [MM89]. [Sed89] presents a modification of his algorithm for computing all double points of an algebraic curve in a triangular domain. However, no efficient and accurate techniques are known for computing higher order intersections.

In this paper we present efficient and robust algorithms for intersecting parametric and algebraic curves. For parametric curves we implicitize one of the curves and represent the implicit form as a matrix determinant. However, we do not compute the symbolic determinant and express the implicit formulation as a matrix. Given the implicit form, we substitute the other parametrization into the matrix formulation and use the resulting matrix to construct a numerical matrix such that the intersection points can be computed from its eigendecomposition. This is in contrast with expanding the symbolic determinant and finding the roots of the resulting polynomial. The advantages of this technique lie in *efficiency, robustness and numerical accuracy.* The algorithms for computing eigenvalues and eigenvectors of a matrix are *backward stable*[3] and fast implementations are available as part of packages like EISPACK and LAPACK [GL89, ABB+92]. Furthermore, we effectively use the algebraic and geometric multiplicities of the eigenvalues to determine the exact multiplicity of the intersection. The exact multiplicity of the eigenvalues is computed using techniques from matrix computations and their implementation is available in LAPACK [BDM89]. As a result the algorithm involves no special handling to deal with higher order intersections. The algorithm for intersecting algebraic curves is rather similar, except the relationship between algebraic and geometric multiplicities of the eigenvalue and the multiplicity of intersection is different.

The rest of the paper is organized in the following manner. In Section 2 we present our notation and review techniques from Elimination theory for implicitizing parametric curves. Furthermore, we show that the problems of intersecting parametric and algebraic curves can be reduced 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 obtained by computing the resultant of the polynomials. In Section 3, we review results from linear algebra and numerical analysis being used in the algorithm. Section 4 deals with reducing the problem of root finding to computing the eigendecomposition. Given the eigenvalues and eigenvectors, we compute the intersection points of parametric curves in the domain of interest. We also discuss the performance and robustness of the resulting algorithm. Section 5 deals with higher order intersections and illustrates the technique for parametric and algebraic curves with some examples.

2 Parametric and Algebraic Curves

A rational Bézier curve is of the form [BBB87]:

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

[3]An eigendecomposition algorithm is backward stable if it computes the exact eigendecomposition of a slightly perturbed matrix.

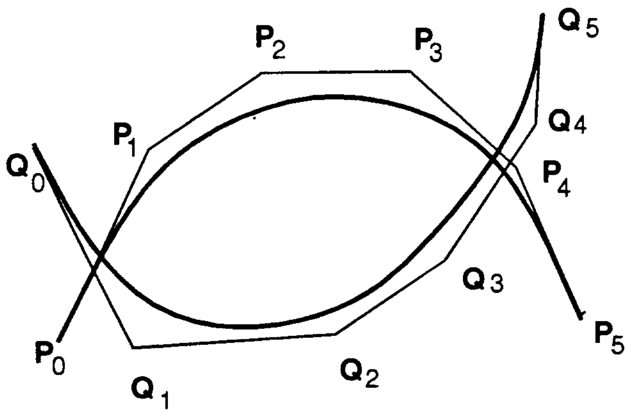

Figure 1: Intersection of Bézier curves

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 [BBB87]. Thus, the problem of intersecting rational curves can be reduced to intersecting Bézier curves. Each of these curves is described by its corresponding control polygon and the curve is always contained in the convex hull of the control points. Therefore, the intersection of the convex hull of two such curves is a necessary condition for the intersection of curves. One such instance has been highlighted in Fig. 1.

Algebraic plane curves are generally expressed in standard power basis:

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

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 [Wal50]. 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

Any point on the curve in general is a *regular point.* A few points on the curve are *multiple or singular points* [Wal50].

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 only $n-k$ 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. $\mathbf{p} = (0,0,1)$, else we can bring it to the origin by a suitable linear transformation. The curve can be represented as

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

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 (ka, kb), where k is a scalar, lies on the curve if k is any of the roots of the equation

$$U_0(a,b) + kU_1(a,b) + U_2(a,b) + \ldots + k^iU_i(a,b) + \ldots + k^nU_n(a,b) = 0. \qquad (1)$$

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 $U_0(x,y)$, $U_1(x,y)$, $\ldots$, $U_{k-1}(x,y)$ 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 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 [SK59]. 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 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 [Abh88, Hof89, MC91, SK59].

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 [SK59]:

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

For our algorithm we assume that the curve $\mathbf{P}(t)$ is properly parametrization. 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 classified according to the following theorem from [MC92]:

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*

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

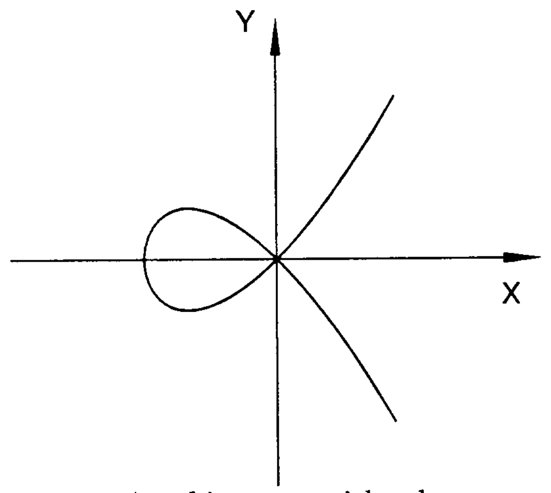

Figure 2: A cubic curve with a loop

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

Theorem 2.5 *A point* $\mathbf{q} = (X_1, Y_1) = (\frac{x(t_1)}{w(t_1)}, \frac{y(t_1)}{w(t_1)})$ *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 $t_1, t_2, \ldots, t_k$. Some of them may be repeated roots. It turns out that for each t_i, $1 \leq i \leq k$,

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

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 other part of the theorem 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 $t_1, \ldots, t_k$, $\mathbf{P}(t_i) = \mathbf{q}$. Let us choose a line such that $a = 0$. Therefore, $Y_1 = -\frac{c}{b}$ and t_i is a root of the equation $y(t) - Y_1 w(t) = 0$. Similarly one can show that t_i is also a root of the equation $x(t) - X_1 w(t) = 0$. Thus, $t_1, t_2, \ldots, t_k$ 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 [Wal50]. 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 have been known a century ago [Mac02, Sal85]. 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 [Sed83]. In this paper we will be dealing with resultants of two polynomials in one unknown. Surveys on various formulations of resultants are given in [Chi90, Sed83, Stu91] and effective techniques for computing and applying them are presented in [Man92].

Given two polynomials in one unknown, their resultant is a polynomial in their coefficients. Moreover, the vanishing of the resultant is a necessary and sufficient condition for the two polynomials to have a common root. Three methods are known in the literature for computing the resultant, owing to Sylvester, Bezout and Cayley [Sal85]. Each of them expresses the resultant as determinant of a matrix. The order of the matrix is different for different methods. We use Cayley's formulation as it results in a matrix of lower order.

Given two polynomials, $F(x)$ and $G(x)$ of degree m and n, respectively. Without loss of generality we assume that $m \geq n$. Let 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 α. Let us represent it as

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

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

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

Let us denote the $m \times m$ matrix by $\mathbf{M}$. The determinant of $\mathbf{M}$ is the resultant of $F(x)$ and $G(x)$ [Sal85]. Let us assume that $x = x_0$ is a common root of the two polynomials. Therefore, $P(x_0, \alpha) = 0$ for all α. As a result $P_i(x_0) = 0$ for $0 \leq i < m$. This condition corresponds to the fact that $\mathbf{M}$ is singular and $[1 \; x_0 \; x_0^2 \; \ldots \; x_0^{m-1}]^T$ is a vector in the kernel of $\mathbf{M}$.

The Cayley's formulation highlighted above is used for implicitizing parametric curves and eliminating a variable from a pair of bivariate algebraic equations, representing algebraic plane curves.

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,

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

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 higher order intersections of curves.

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

$$\begin{pmatrix} 1 & x_0 & x_0^2 & \ldots & x_0^{m-1} \end{pmatrix}^T,$$

$$\begin{pmatrix} 0 & 1 & 2x_0 & 3x_0^2 & \ldots & (m-1)x_0^{m-2} \end{pmatrix}^T,$$

$$\vdots$$

$$\begin{pmatrix} 0 & 0 & \ldots & 0 & (k-1)! & k!\,x_0 & \frac{(k+1)!}{2}x_0^2 & \frac{(m-1)!}{(m-k-2)!}x_0^{m-k} \end{pmatrix}^T$$

are in the kernel of **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.

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

Moreover it follows from (2)

$$P_x(x,\alpha) = P_0'(x) + P_1'(x)\alpha + P_2'(x)\alpha^2 + \ldots + P_{m-1}'(x)\alpha^{m-1} \qquad (5)$$

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

$$P_x(x_0,\alpha) = P_0'(x_0) + P_1'(x_0)\alpha + P_2'(x_0)\alpha^2 + \ldots + P_{m-1}'(x_0)\alpha^{m-1} = 0.$$

This relationship is true for α. That implies that

$$P_0'(x_0) = 0,\ P_1'(x_0) = 0,\ \ldots,\ P_{m-1}'(x_0) = 0.$$

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

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

For arbitrary k, we know that $F(x_0) = 0, F'(x_0) = 0, \ldots, F^{k-1}(x_0) = 0$ 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 $(1 \;\; x \;\; x^2 \;\; \ldots \;\; x^{m-1})$ at $x = x_0$ vectors lie in the kernel of **M**.

Q.E.D.

2.3 Implicitizing Parametric Curves

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

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

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 [Man90, MC92, Sed86].

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).

The computation of the entries of $\mathbf{M}$ involves symbolic computation. We minimize the symbolic computation in the following manner. Let

$$P(t,\alpha) = \frac{F_1(t)F_2(\alpha) - F_1(\alpha)F_2(t)}{t - \alpha},$$

$$\Rightarrow P(t,\alpha) = X\frac{w(\alpha)y(t) - w(t)y(\alpha)}{t-\alpha} + Y\frac{x(\alpha)w(t) - x(t)w(\alpha)}{t-\alpha} + \frac{x(t)y(\alpha) - x(\alpha)y(t)}{t-\alpha}.$$

Each term of the form $\frac{f(t)g(\alpha)-f(\alpha)g(t)}{t-\alpha}$ corresponds to a polynomial and can be expressed as product of matrices and vectors, as shown in (3). In other words,

$$P(t,\alpha) = \begin{bmatrix} 1 & \alpha & \alpha^2 & \dots & \alpha^{m-1} \end{bmatrix}^T (X\mathbf{M}_1 + Y\mathbf{M}_2 + \mathbf{M}_3) \begin{pmatrix} 1 \\ t \\ t^2 \\ \vdots \\ t^{m-1} \end{pmatrix},$$

where $\mathbf{M}_1, \mathbf{M}_2$ and $\mathbf{M}_3$ are $m \times m$ matrices whose entries are numeric constants. Let us call them the *coefficient matrices.* The implicit representation of the curve is given as

$$\mathbf{M} = X\mathbf{M}_1 + Y\mathbf{M}_2 + \mathbf{M}_3. \tag{7}$$

Given $f(t), g(t)$, polynomials of degree m, we compute the $m \times m$ coefficient matrix, $\mathbf{P}$ corresponding to $\frac{f(s)g(t)-f(t)g(s)}{s-t}$ in the following manner. Let $F[0] \dots F[m]$ and

$G[0] \ldots G[m]$ correspond to the coefficients of the polynomials, $f(s)$ and $g(s)$. Moreover, let a monomial in two variables s and t be denoted as $a_{i,j}s^i t^j$, where $a_{i,j}$ is the coefficient. We represent a monomial data structure by denoting the coefficient and the exponents of s and t as *monomial.a*, *monomial.s* and *monomial.t*. Furthermore, let $Poly[0] \ldots Poly[max]$ denote a collection of *monomials*. The algorithm is:

Algorithm I

```
1. $index = 0$.
2. $for\ (\ 0 \le i \le m\ )$ do
   (a) $for\ (\ 0 \le j \le i\ )$ do
       i. $monomial.a = F[i] * G[j]$.
       ii. $bound = i - j - 1$.
       iii. $for\ (\ 0 \le k \le bound\ )$ do
            A. $monomial.s = i - 1 - k$.
            B. $monomial.t = k + j$.
            C. $Poly[index++] = monomial$.
   (b) $for\ (\ i+1 \le j \le m\ )$ do
       i. $monomial.a = -F[i] * G[j]$.
       ii. $bound = j - i - 1$.
       iii. $for\ (\ 0 \le k \le bound\ )$ do
            A. $monomial.s = j - 1 - k$.
            B. $monomial.t = k + i$.
            C. $Poly[index++] = monomial$.
3. $for\ (\ 0 < i < index\ )$ do
   (a) $monomial = Poly[i]$.
   (b) $j = monomial.s$.
   (c) $k = monomial.t$.
   (d) $\mathbf{P}[j][k] = \mathbf{P}[j][k] + monomial.a$.
```

One advantage of the algorithm lies in the fact that the computation involved is purely numeric. As a result, we can use error analysis techniques and come up a with a tight bound on the accuracy of each entry of **P**.

We express the resultant of $F_1(t)$ and $F_2(t)$, **M**, as a matrix determinant. In this case the matrix has order m. In fact we use the matrix formulation, **M**, to represent the implicit form and do not compute its symbolic determinant.

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 [FR87]. To circumvent this problem we perform a reparametrization.

Given

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

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

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

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

$$\overline{\mathbf{p}}(s) = (\Sigma_{i=0}^{m} w_i X_i \binom{m}{i} s^i, \Sigma_{i=0}^{m} w_i Y_i \binom{m}{i} s^i, \Sigma_{i=0}^{m} w_i \binom{m}{i} s^i).$$

The rest of the algorithm proceeds by computing the implicit representation of $\overline{\mathbf{p}}(s)$ 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

$$\mathbf{M} \begin{pmatrix} (1-t)^{m-1} \\ t(1-t)^{m-2} \\ t^2(1-t)^{m-3} \\ \vdots \\ t^{m-1} \end{pmatrix}. \tag{8}$$

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 proposed a matrix determinant formulation for the implicit representation of the curve. In this section we highlight some properties of this formulation, denoted as $\mathbf{M} = \mathbf{F}(X, Y)$ in (7).

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.

Lets 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 $\mathbf{v}_i = (1 \;\; t_i \;\; t_i^2 \;\; \ldots \;\; t_i^m)^T$ is a vector in the kernel of $\mathbf{F}(X_1, Y_1)$. Since each t_i is distinct it follows that $\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_l$ 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 $\mathbf{F}(X_1, Y_1)$:

$$\mathbf{v}_{\mathbf{i}_1} = \left(1 \;\; t_i \;\; t_i^2 \;\; \ldots \;\; t_i^{m-1}\right)^T,$$

$$\mathbf{v}_{\mathbf{i}_2} = \left(0 \;\; 1 \;\; 2t_i \;\; 3t_i^2 \;\; \ldots \;\; (m-1)t_i^{m-2}\right)^T,$$

$$\vdots$$

$$\mathbf{v}_{\mathbf{i}\mathbf{m}_\mathbf{i}} = \left(0 \;\; 0 \;\; \ldots \;\; 0 \;\; (m_i-1)! \;\; m_i!t_i \;\; \frac{(m_i+1)!}{2}t_i^2 \;\; \frac{(m-1)!}{(m-m_i-2)!}t_i^{m-m_i}\right)^T.$$

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

Q.E.D.

Example 2.8 *Consider the parametric curve:*

$$\mathbf{p}(t) = (x(t), y(t), w(t)) = (4 - 3t + 6t^2 + 6t^3, 1 + 6t - 4t^3, 1 + 6t + 6t^2 + t^3).$$

Using the algorithm we obtain its implicit representation as

$$\mathbf{M} = \begin{pmatrix} -27 + 27y & 6 - 6x + 18y & 22 - 5x - 2y \\ 6 - 6x18y & 58 - 41x - 56y & 24 - 30x - 39y \\ 22 - 5x - 2y & 24 - 30x - 39y & 24 - 24x - 30y \end{pmatrix}.$$

Given $(x, y) = (4, 1)$, we see that the matrix obtained after substituting these values is singular and the vector in its kernel is $[1 \;\; 0 \;\; 0]^T$. As a result, preimage of $(4, 1)$ is $t = 0$.

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 $\mathbf{Q}(u) = (\overline{x}(u), \overline{y}(u), \overline{w}(u))$ 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 $\overline{x}(u), \overline{y}(u)$ and $\overline{w}(u)$. The intersection points correspond to the roots of

$$\text{Determinant}(\mathbf{M}(u)) = 0. \qquad (9)$$

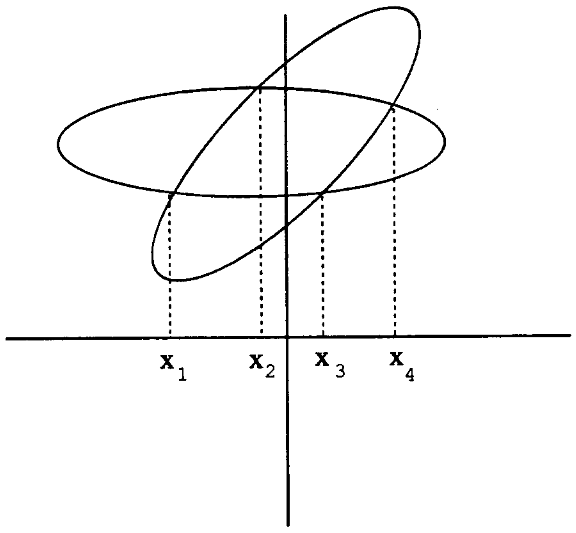

Figure 3: Intersection and projection of two ellipses

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 $(x_1, y_1), \ldots, (x_{mn}, y_{mn})$. 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. One such case corresponding to the intersection of two ellipses has been shown in Fig. 3. In this case the resultant is a polynomial of degree 4 in x, say $H(x)$, such that

$$H(x_1) = 0, \quad H(x_2) = 0, \quad H(x_3) = 0, \quad H(x_4) = 0.$$

Thus, given $H(x)$, the problem of intersection reduces to finding its roots.

Let us express $F(x, y)$ and $G(x, y)$ as polynomials in y and the coefficients are polynomials in x. That is,

$$F(x, y) = F_0(x) + F_1(x)y + \ldots + F_m(x)y^m$$

and

$$G(x, y) = G_0(x) + G_1(x)y + \ldots + G_n(x)y^n,$$

where $F_i(x)$ is a polynomial of degree $m - i$ and $G_j(x)$ is a polynomial of degree $n - j$. Without loss of generality we assume that $m \geq n$. We compute the resultant using Cayley's formulation. In case, the curves are expressed in Bernstein basis, we use the reparametrization highlighted in the previous section for implicitization. The algorithm for computing the resultant matrix is similar to Algorithm I. In particular, let

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

This can be expressed as

$$P(x,y,\alpha) = \Sigma_{i=0}^{m}\Sigma_{j=0}^{i}(F_i(x)G_j(x)y^j\alpha^j\frac{y^{i-j}-\alpha^{i-j}}{y-\alpha}) + \Sigma_{i=0}^{m}\Sigma_{j=i+1}^{n}(F_i(x)G_j(x)y^i\alpha^i\frac{y^{j-i}-\alpha^{j-i}}{y-\alpha}).$$

The main difference between Algorithm I and computation of $P(x,y,\alpha)$ lies in the fact that the coefficients $F_i(x)$ and $G_j(x)$ are polynomials in x, whereas in Algorithm I they are constants. Let us denote the $m \times m$ matrix by $\mathbf{M}(x)$. The problem of intersection corresponds to computing roots of

$$\text{Determinant}(\mathbf{M}(x)) = 0. \tag{10}$$

Example 2.9 *Consider the algebraic curves:*

$$F(x,y) = (x^2-2y)^2 + (x-y)^4 - 5,$$

$$G(x,y) = x^3 - y^3 - 2x^2 + y^2 - 1$$

We treat them as polynomials in y and the resulting curves are

$$F(x,y) = (-5+2x^4) + (-4x^2-4x^3)y + (4+6x^2)y^2 - 4xy^3 + y^4$$

$$G(x,y) = -1 - 2x^2 + x^3 + y^2 - y^3$$

The matrix $\mathbf{M}(x)$ obtained after eliminating y is

$$\mathbf{M}(x) = \begin{pmatrix} 4x^2+4x^3+8x^4+4x^5-4x^6 & -4-14x^2+4x^3-14x^4+6x^5 & 4x+8x^3-2x^4 & -1-2x^2+x^3 \\ -4-14x^2+4x^3-14x^4+6x^5 & 4x+4x^2+12x^3-2x^4 & -1-6x^2-3x^3 & 0 \\ 4x+8x^3-2x^4 & -1-6x^2-3x^3 & 4-4x+6x^2 & 1 \\ -1-2x^2+x^3 & 0 & 1 & -1 \end{pmatrix}.$$

The kernel of $\mathbf{M}(x_0)$ has properties similar to that of the matrix corresponding to the implicit representation of the parametric equations. We highlight some of them in Section 5.

3 Matrix Computations

In this section we review some techniques from linear algebra and numerical analysis. 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 and Section 5.

3.1 Hessenberg Matrix

A Hessenberg matrix is of the form

$$\mathbf{H} = \begin{pmatrix} h_{11} & h_{12} & h_{13} & \dots & h_{1n} \\ h_{21} & h_{22} & h_{23} & \dots & h_{2n} \\ 0 & h_{32} & h_{33} & \dots & h_{3n} \\ 0 & 0 & h_{43} & \dots & h_{4n} \\ \vdots & \vdots & \vdots & \dots & \vdots \\ 0 & \dots & 0 & h_{n,n-1} & h_{nn} \end{pmatrix}.$$

In other words it is like an upper triangular matrix, except all that the subdiagonal elements may be non-zero. Given a matrix $\mathbf{A}$, it can be converted into a Hessenberg matrix using similarity transformations of the form $\mathbf{QAQ}^{-1}$, where $\mathbf{Q}$ is an orthogonal matrix. $\mathbf{Q}$ is an orthogonal matrix if $\mathbf{QQ}^T = \mathbf{I}$.

3.2 QR Factorization

The QR factorization of an $m \times n$ matrix $\mathbf{A}$ is given by

$$\mathbf{A} = \mathbf{QR},$$

where $\mathbf{Q}$ is an $m \times m$ orthogonal matrix and $\mathbf{R}$ is an $m \times n$ upper triangular matrix. More details on its computations are given in [GL89].

3.3 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 non-trivial exercise. In these situations, the numerical rank is easily characterized in terms of the SVD.

Given $\mathbf{A}$, a $m \times n$ real matrix then there exist orthogonal matrices $\mathbf{U}$ and $\mathbf{V}$ such that

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

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

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

Moreover, $\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_n \geq 0$. The σ_i's are called the *singular values* and 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 [GL89]. The relationship between the elements of $\mathbf{A}$, singular values and singular vectors can be expressed as:

$$\mathbf{A}_{ij} = \Sigma_{k=1}^{n} \sigma_k \mathbf{U}_{ik} \mathbf{V}_{jk},$$

where $\mathbf{A}_{ij}, \mathbf{U}_{ij}, \mathbf{V}_{ij}$ represent the element in the ith row and jth column of $\mathbf{A}, \mathbf{U}$ and $\mathbf{V}$, respectively.

The singular values give accurate information about the rank of the matrix. The matrix $\mathbf{A}$ has rank $k < n$, if $\sigma_{k+1} = 0$, $\sigma_{k+2} = 0, \ldots, \sigma_n = 0$. Furthermore, the smallest positive singular value gives us information about the closeness to a rank deficient matrix [GL89].

3.4 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, corresponding to determinant($\mathbf{A} - s\mathbf{I}$). As a result, the eigenvalues of a diagonal matrix, upper triangular matrix or a lower triangular matrix correspond to the elements on its diagonal. Efficient algorithms for computing eigenvalues and eigenvectors are well known, [GL89], and their implementations are available as part of packages EISPACK, [GBDM77], and LAPACK [Dem89, ABB+92]. Most algorithms make use of the similarity transformations of the form $\mathbf{A}' = \mathbf{Q}\mathbf{A}\mathbf{Q}^{-1}$, where $\mathbf{Q}$ is any non-singular $n \times n$ matrix. This transformation has the characteristic that the eigenvalues of $\mathbf{A}$ and $\mathbf{A}'$ are identical. Furthermore, if $\mathbf{y}$ is an eigenvector of $\mathbf{A}'$, $\mathbf{Q}^{-1}\mathbf{y}$ is an eigenvector of $\mathbf{A}$. Standard algorithms for eigenvalue computations, like the QR algorithm, choose $\mathbf{Q}$ to be an orthogonal matrix, since similarity transformation by an orthogonal matrix is a numerically stable operation [GL89]. Given **A** the eigendecomposition algorithm converts it into a Hessenberg matrix using a sequence of similarity transformations by orthogonal matrices. That is,

$$\mathbf{H} = \mathbf{Q}^T \mathbf{A} \mathbf{Q},$$

where **Q** is an orthogonal matrix and **H** is an Hessenberg matrix. Given **H**, the eigendecomposition algorithm proceeds by similarity transformations by orthogonal matrices. Each of these similarity transformation corresponds to a QR iteration of the form:

$$\mathbf{H} - s\mathbf{I} = \mathbf{U}\mathbf{R}, \tag{11}$$

where s is a scalar referred to as a shift, **U** is an orthogonal matrix and **R** is an upper triangular matrix. This step corresponds to QR factorization of the matrix $\mathbf{H} - s\mathbf{I}$. Given **U** and **R**, the next step of the iteration computes a modified Hessenberg matrix given by

$$\mathbf{H} = \mathbf{R}\mathbf{U} + s\mathbf{I}.$$

The shifts are chosen appropriately such that the matrix converges to its to its *real Schur decomposition* of the form [GL89, Wil65]:

$$\mathbf{Q}\mathbf{A}\mathbf{Q}^{-1} = \begin{pmatrix} R_{11} & R_{12} & \dots & R_{1m} \\ 0 & R_{22} & \dots & R_{2m} \\ \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \dots & R_{mm} \end{pmatrix}, \tag{12}$$

where each R_{ii} is either a 1×1 matrix or a 2×2 matrix having complex conjugate eigenvalues. Given the real Schur decomposition, computing the eigenvalues is a trivial operation. Many a times a matrix has complex eigenvalues, the above algorithm is modified to double shift consisting of a complex number and its conjugate. More details are given in [GL89]. We will use the QR algorithm with *double implicit shift strategy* to compute the real Schur decomposition. Given the matrix eigenvalues, real Schur decomposition and matrix **Q**, computing eigenvectors corresponds to solving quasi triangular systems [GL89, Wil65]. The running time of these algorithms is $O(n^3)$. However, the constant in front of n^3 can be as high as 25 for computing all the eigenvalues and eigenvectors. In many cases we may a priori know some of the eigenvalues of the given matrix. We use that information in choosing the appropriate shifts.

For example if s' is an eigenvalue and $\mathbf{H} - s'\mathbf{I}$ can be decomposed as

$$\mathbf{H} - s'\mathbf{I} = \mathbf{U}'\mathbf{R}'. \tag{13}$$

Since $\mathbf{H} - s'\mathbf{I}$ is singular, $\mathbf{R}'$ is singular as well and a zero appears on its diagonal. As a result the problem reduces to finding the real Schur form corresponding to a Hessenberg matrix of lower order.

3.5 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 represent this problem as eigenvalues of $\mathbf{A} - s\mathbf{B}$. The vectors $\mathbf{x} \neq \mathbf{0}$ correspond to the eigenvectors of this equation. If $\mathbf{B}$ is non–singular and its condition number (defined in the next section) is low, the problem can be reduced to an 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, $\mathbf{B}$ may have a high condition number and such a reduction can cause numerical problems. Algorithms for the generalized eigenvalue problems apply orthogonal transformations to $\mathbf{A}$ and $\mathbf{B}$. In particular, we use the QZ algorithm for computing the eigenvalues and eigenvectors for this problem [GL89]. Its running time is $O(n^3)$. However, the constant can be as high as 75. Generally, it is slower by a factor of 2.5 to 3 as compared to QR algorithm for computing eigenvalues and eigenvectors of a matrix.

3.6 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 [GL89, Wil65]. The implementations of these condition number computations are available as part of LAPACK [BDM89].

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.

3.7 Condition Number of a Square Matrix

The *condition number of a square matrix* corresponds to $\frac{\sigma_1(\mathbf{A})}{\sigma_n(\mathbf{A})}$, where σ_1 and σ_n are the largest and smallest singular values. This condition number is used in determining the accuracy of $\mathbf{A}^{-1}$ computation or solving linear systems of the form $\mathbf{Ax} = \mathbf{b}$. Computing the singular values takes $O(n^3)$ time, which is rather expensive. Good estimators 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.

3.8 Condition Number of Simple Eigenvalues

Let s be a simple[4] eigenvalue of the $n \times n$ matrix, $\mathbf{A}$, with unit right eigenvector $\mathbf{x}$ and unit left eigenvector $\mathbf{y}$. That is, $\mathbf{Ax} = s\mathbf{x}$, $\mathbf{y}^T\mathbf{A} = s\mathbf{y}^T$ and $\| \mathbf{x} \|_2 = \| \mathbf{y} \|_2 = 1$. Here $\| \mathbf{v} \|_2$ stands for the 2–norm of a vector. Let $\mathbf{P} = (\mathbf{x} \cdot \mathbf{y}^T)/(\mathbf{y}^T \cdot \mathbf{x})$ be the spectral projector. Therefore, $\| P \|_2 = \frac{1}{|\mathbf{y}^T\mathbf{x}|}$. 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

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

Thus, for sufficiently small perturbations in the matrix, the perturbation in the eigenvalues is a function of $\| \mathbf{P} \|_2$.

3.9 Condition Number of Clustered Eigenvalues

In many cases we are interested in computing the condition numbers of a cluster of eigenvalues. We highlight the need for using clusters in Section 5. We use these condition numbers in determining the accuracy of eigenvalues with multiplicity greater than one. We represent the real Schur decomposition as

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

and the eigenvalues of the $m \times m$ matrix $\mathbf{A}_{11}$ are exactly those we are interested in. In particular we are interested in bounding the perturbation in the average of the eigenvalues of the cluster, represented as $\bar{s} = \text{trace}(\mathbf{A}_{11})/m$.

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

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

[4] A simple eigenvalue is an eigenvalue of multiplicity one.

Thus, $\| \mathbf{P} \|_2 = (1 + \| \mathbf{R} \|_2^2)^{1/2}$. Computing $\| \mathbf{P} \|_2$ is expensive and a cheaper overestimate is obtained as

$$\| \mathbf{P} \|' = (1 + \| \mathbf{R} \|_F^2)^{1/2}.$$

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

$$| \overline{s} - \overline{s}' | \leq \epsilon_2 \| \mathbf{P} \|_2 + O(\epsilon_2^2). \tag{14}$$

We substitute $\| \mathbf{P} \|'$ to obtain a slightly weaker bound on the perturbation in $\overline{s}$ for sufficiently small ϵ_2. The average of a cluster is often much better conditioned than individual eigenvalues in the cluster.

3.10 Accuracy of Right Eigenvectors

As far as eigenvectors are concerned, bounds for their accuracy are given in detail in [Wil65, ABB+92]. However, we will not be computing these bounds to analyze the accuracy of our computation. The actual bounds tell us about the maximum error in any term of the eigenvector. We only assume that each term of the eigenvector has a similar bound on the absolute error. Thus, the terms of eigenvectors of maximum magnitude have the smallest bound on their relative error.

4 Reduction to 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 a matrix determinant as shown in (9). 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 (10). Let us represent it as a matrix polynomial

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

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

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

Substitute $s = \frac{u}{1-u}$ and the new polynomial is of the form

$$\overline{\mathbf{M}}(s) = \mathbf{M}_n s^n + \mathbf{M}_{n-1} s^{n-1} + \ldots + \mathbf{M}_0. \tag{15}$$

In the original problem we were interested in the roots of Determinant($\mathbf{M}(u)$) $= 0$ in the range $[0,1]$. However, after reparametrizing we want to compute the roots of Determinant($\overline{\mathbf{M}}(s)$) $= 0$ in the domain $[0,\infty]$. This can result in 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.

If $\mathbf{M}_0, \mathbf{M}_1, \ldots, \mathbf{M}_n$ are $m \times m$ numeric matrices, then the matrix–valued function defined by

$$\mathbf{L}(s) = \Sigma_{i=0}^{n} \mathbf{M}_i s^i$$

is called a *matrix polynomial* of degree n. When $\mathbf{M}_n = \mathbf{I}$, the identity matrix, the matrix polynomial is said to be *monic.* More details on matrix polynomials and their properties are given in [GLR82]. In our application we will be dealing with matrix polynomials in the context of finding the points of intersection. Our main interest is in finding roots of the polynomial equation

$$P(s) = \text{Determinant}(\mathbf{L}(s)) = 0. \tag{16}$$

A simple solution to this problem is expand the determinant and compute the roots of the resulting polynomial. However, the resulting approach is numerically unstable and expensive in practice.

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

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

$\overline{\mathbf{L}}(s)$ 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

$$\overline{\mathbf{L}}(s_0)\mathbf{v} = \mathbf{0}, \tag{17}$$

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

Theorem 4.1 *Given the matrix polynomial,* $\overline{\mathbf{L}}(s)$ *the roots of the polynomial corresponding to its determinant are the eigenvalues of the matrix*

$$\mathbf{C} = \begin{bmatrix} \mathbf{0} & \mathbf{I}_m & \mathbf{0} & \dots & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{I}_m & \dots & \mathbf{0} \\ \vdots & \vdots & \dots & \vdots & \vdots \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \dots & \mathbf{I}_m \\ -\overline{\mathbf{M}}_0 & -\overline{\mathbf{M}}_1 & -\overline{\mathbf{M}}_2 & \dots & -\overline{\mathbf{M}}_{n-1} \end{bmatrix}, \tag{18}$$

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* $\overline{\mathbf{L}}(s_0)$ *as highlighted in (17).*

Proof: The eigenvalues of $\mathbf{C}$ correspond to the roots of

$$\text{Determinant}(\mathbf{C} - s\mathbf{I}) = 0.$$

$\mathbf{C}$ is a matrix of order m^2. Let $s = s_0$ be an eigenvalue of $\mathbf{C}$. As a result there is a non-trivial vector $\mathbf{V}$ in the kernel of $\mathbf{C} - s_0\mathbf{I}$. Furthermore, we represent $\mathbf{V}$ as

$$\mathbf{V} = [\mathbf{v}_1^T\ \mathbf{v}_2^T\ \dots\ \mathbf{v}_m^T]$$

and each $\mathbf{v}_i$ is an $m \times 1$ vector. The relationship between $\mathbf{C}$, s_0 and $\mathbf{V}$ can be represented as

$$\begin{bmatrix} \mathbf{0} & \mathbf{I}_m & \mathbf{0} & \dots & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{I}_m & \dots & \mathbf{0} \\ \vdots & \vdots & \dots & \vdots & \vdots \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \dots & \mathbf{I}_m \\ -\overline{\mathbf{M}}_0 & -\overline{\mathbf{M}}_1 & -\overline{\mathbf{M}}_2 & \dots & -\overline{\mathbf{M}}_{n-1} \end{bmatrix} \begin{bmatrix} \mathbf{v}_1 \\ \mathbf{v}_2 \\ \vdots \\ \mathbf{v}_{m-1} \\ \mathbf{v}_m \end{bmatrix} = s_0 \begin{bmatrix} \mathbf{v}_1 \\ \mathbf{v}_2 \\ \vdots \\ \mathbf{v}_{m-1} \\ \mathbf{v}_m \end{bmatrix}. \tag{19}$$

Multiplying the submatrices of $\mathbf{C}$ with the vectors in $\mathbf{V}$ and equating them with the vectors on the right hand side results in:

$$\mathbf{v}_2 = s_0\mathbf{v}_1,$$

$$\mathbf{v}_3 = s_0\mathbf{v}_2,$$

$$\vdots$$

$$\mathbf{v}_m = s_0 \mathbf{v}_{m-1}$$

and

$$-\overline{\mathbf{M}}_0\mathbf{v}_1 - \overline{\mathbf{M}}_1\mathbf{v}_2 - \overline{\mathbf{M}}_2\mathbf{v}_3 - \ldots - \overline{\mathbf{M}}_{m-1}\mathbf{v}_m = s_0\mathbf{v}_m.$$

These relations imply

$$\mathbf{v}_i = s_0^{i-1}\mathbf{v}_1, \quad \text{for } 1 <= i <= m.$$

and

$$-(\overline{\mathbf{M}}_0 + s_0\overline{\mathbf{M}}_1 + s_0^2\overline{\mathbf{M}}_2 + \ldots + s_0^{m-1}\overline{\mathbf{M}}_{m-1} + s_0^m\mathbf{I}_m)\mathbf{v}_1 = \mathbf{0}.$$

Equating the above relation with (17) results in the fact that s_0 is a solution of $\overline{\mathbf{L}}(s) = 0$ and $\mathbf{v}_1$ is a vector in the kernel of $\overline{\mathbf{L}}(s_0) = 0$. Thus, every eigenvalue of $\mathbf{C}$ is a root of $P(s)$. Since the leading matrix of $\mathbf{L}(s)$ is non-singular, $P(s)$ is a polynomial of degree mn. Furthermore, $\mathbf{C}$ is a matrix of order mn and therefore, has mn eigenvalues. Thus, all the roots of $P(s)$ correspond to the eigenvalues of $\mathbf{C}$.

Q.E.D.

The matrix polynomials have been used to solve general systems of non-linear polynomial equations. More details are highlighted in [Man92]. The relationship between the eigenvalues of $\mathbf{C}$ the roots of $P(s)$ has been proved using similarity transformations in [GLR82]. Many a 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 later part of this section, 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. For example, when the matrices have singular pencils. In such cases, we reduce the intersection problem to a generalized eigenvalue problem.

Theorem 4.2 *Given the matrix polynomial, $\overline{\mathbf{L}}(s)$ the roots of the polynomial corresponding to its determinant are the eigenvalues of the generalized system $\mathbf{C}_1 s + \mathbf{C}_2$, where*

$$\mathbf{C}_1 = \begin{bmatrix} \mathbf{I}_m & \mathbf{0} & \mathbf{0} & \ldots & \mathbf{0} \\ \mathbf{0} & \mathbf{I}_m & \mathbf{0} & \ldots & \mathbf{0} \\ \vdots & \vdots & \ldots & \vdots & \vdots \\ \mathbf{0} & \mathbf{0} & \ldots & \mathbf{I}_m & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \ldots & \mathbf{0} & \mathbf{M}_n \end{bmatrix} \quad \mathbf{C}_2 = \begin{bmatrix} \mathbf{0} & \mathbf{I}_m & \mathbf{0} & \ldots & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{I}_m & \ldots & \mathbf{0} \\ \vdots & \vdots & \ldots & \vdots & \vdots \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \ldots & \mathbf{I}_m \\ -\mathbf{M}_0 & -\mathbf{M}_1 & -\mathbf{M}_2 & \ldots & -\mathbf{M}_{n-1} \end{bmatrix}, \tag{20}$$

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

The proof of this theorem is similar to that of Theorem 2.4 and can also be proved using similarity transformations as highlighted in [GLR82].

It follows from Theorem 4.1 that the eigenvalues of $\mathbf{C}$ correspond exactly to the preimages of intersection points on $\mathbf{Q}(u)$. However, we are only interested in the eigenvalues in the range $s_0 \in [0, \infty]$ and the preimages on the curve are obtained by substituting $u_0 = \frac{s_0}{1+s_0}$. This gives us a list of all the intersection points on $\mathbf{Q}(u_0)$ such that $u_0 \in [0,1]$. However, these points on $\mathbf{P}(t)$ may not lie in the range $t \in [0,1]$. As a result it is important for us to compute the preimage of the intersection point $(x_0, y_0, w_0) = \mathbf{Q}(u_0)$ with respect to $\mathbf{P}(t)$. We use the property of the linear system of equations (8) and Theorem 4.1.

Let us assume that (x_0, y_0, w_0) is a simple point on $\mathbf{P}(t)$. Points of higher multiplicity are accounted for in the next section. Substitute for $(X, Y, W) = (x_0, y_0, w_0)$ in the matrix, $\mathbf{M}$ as shown in (8), corresponding to the implicit representation of $P(t)$. The resulting matrix is singular and let us assume that its kernel has dimension one. Kernels of higher dimension are handled in the next section. The vector in the kernel corresponds to $\mathbf{v}$ shown in (17). Given the eigenvector of $\mathbf{C}$ corresponding to the eigenvalue s_0, we use Theorem 4.1 to compute the eigenvector $\mathbf{v}$. Given $\mathbf{v}$ we use the structure of the linear system to compute the preimage of the point (x_0, y_0, w_0) by using the relation

$$\begin{pmatrix} (1-t_0)^{m-1} \\ t_0(1-t_0)^{m-2} \\ \vdots \\ t_0^{m-1} \end{pmatrix} = k \begin{pmatrix} v_1 \\ v_2 \\ \vdots \\ v_m \end{pmatrix},$$

where $k \neq 0$ is a constant. Thus, $t_0 = \frac{v_2}{v_1+v_2}$. The relationship between the eigenvalue s_0 of C, elements v_1, v_2 of the eigenvector $\mathbf{V}$ corresponding to s_0 and the point of intersection (x_0, y_0, w_0) can be expressed as

$$(x_0, y_0, w_0) = \mathbf{Q}(\frac{s_0}{1+s_0}) = \mathbf{P}(\frac{v_2}{v_1+v_2}). \tag{21}$$

As a result we are able to compute all the points of intersection in the domain of interest by computing the eigendecomposition of $\mathbf{C}$.

Solving a generalized eigenvalue system is more expensive than the normal eigenvalue system (almost a factor of 3). In many cases, we can perform a linear transformation on the coordinate of the matrix polynomial and reduce the resulting problem to an eigenvalue problem. The basic idea involves transforming

$$s = \frac{a\bar{s}+b}{c\bar{s}+d},$$

where a, b, c and d are random numbers. The matrix polynomial $\overline{\mathbf{M}}(s)$ in (15) is transformed into

$$\mathbf{P}(\overline{s}) = (c\overline{s}+d)^n \mathbf{M}(\frac{a\overline{s}+b}{c\overline{s}+d})$$

$$= \mathbf{M}_n(a\overline{s}+b)^n + \mathbf{M}_{n-1}(a\overline{s}+b)^{n-1}(c\overline{s}+d) + \ldots + \mathbf{M}_1(a\overline{s}+b)(c\overline{s}+d)^{n-1} + \mathbf{M}_0(c\overline{s}+d)^n$$

$$\Rightarrow \mathbf{P}(\overline{s}) = \mathbf{P}_n\overline{s}^n + \mathbf{P}_{n-1}\overline{s}^{n-1} + \ldots + \mathbf{P}_1\overline{s} + \mathbf{P}_0,$$

where $\mathbf{P}_i$'s are computed from the $\mathbf{M}_j$'s. If $\mathbf{P}_n$ is a well-conditioned matrix then the problem of intersection is reduced to an eigenvalue problem, otherwise use a different transformation (by a different choice of a,b,c and d). The linear transformation is performed up to four or five times. If all the resulting leading matrices, $\mathbf{P}_n$, are ill-conditioned, the intersection problem is reduced to a generalized eigenvalue problem. There are cases when any linear transformation can result in an ill-conditioned leading matrix. Furthermore, the domain of the eigenvalue system obtained after transformation is $[s_1, s_2]$ or $[s_2, s_1]$ depending upon the signs of a, b, c and d, where $s_1 = -\frac{b}{a}$ and $s_2 = -\frac{d}{c}$.

4.1 Implementation and Performance

The reduction to an eigenvalue or a generalized eigenvalue system involves estimating the condition number of a matrix, linear equation solving and finding the eigenvalues of a matrix. The eigenvalue algorithm reduces the matrix to its real Schur form using similarity transformations. For eigenvalues lying in the domain of interest, we compute the corresponding eigenvectors. These eigenvectors are obtained by solving quasi upper triangular systems and multiplying by an orthogonal matrix corresponding to similarity transformations. Furthermore, we also compute the condition number of each eigenvalue in the domain of interest. The condition number computation requires the left as well right eigenvectors of the matrix.

We used LAPACK implementation of QR algorithms. Some of the routines were modified to compute the eigenvalues in the domain of interest. Furthermore, the domain was specified as $\alpha + j\beta$, where $\alpha > -\epsilon$, $|\beta| < \epsilon$ and $j = \sqrt{-1}$. ϵ is a small positive constant used to account for the numerical errors. In particular, we make ϵ a function of the condition number of $\mathbf{M}_n$ or $\mathbf{P}_n$ for eigenvalue problems. To compute the inverse coordinate of the intersection point, the right eigenvector $\mathbf{V}$ corresponding to the eigenvalue s_0 is computed. Let

$$\mathbf{V} = [v_{1,1}\ v_{1,2}\ \ldots\ v_{1,m}\ v_{2,1}\ \ldots\ v_{2,m}\ \ldots\ v_{n,1}\ \ldots\ v_{n,m}]^T.$$

Analysis of the accuracy of eigenvector computation indicates that each term of the eigenvector has a similar bound on its absolute error. As a result we tend to use terms of maximum magnitude to minimize the relative error in the computation. In this case we compute the entries of $\mathbf{v} = [v_1\ v_2\ \ldots\ v_m]^T$ as:

If $s_0 \geq 1$

$$[v_1\ v_2\ \ldots v_m]^T = \frac{1}{(s_0)^n}[v_{n,1}\ v_{n,2}\ \ldots\ v_{n,m}]^T$$

otherwise

$$[v_1\ v_2\ \ldots v_m]^T = [v_{1,1}\ v_{1,2}\ \ldots\ v_{1,m}]^T$$

Given $\mathbf{v}$ the inverse coordinate, t_0, is computed using v_1, v_2 or v_{m-1}, v_m by making use of similar numerical properties.

The performance of the algorithm is largely governed by the eigendecomposition routines. Roughly $80 - 85\%$ of the time is spent in these routines. The eigenvalue algorithms compute all the eigenvalues of the given matrix. It is difficult to restrict them to computing eigenvalues in the domain of interest without using any heuristics. The order of the matrix, say p, corresponds to the degree of the two curves and the number of eigenvalues is equal to the order. The running time of the algorithm is a cubic function of p. However, eigenvalue algorithms have good convergence. Each iteration of the algorithm corresponds to a similarity transformation, whose complexity is a quadratic function of the matrix order. The double shifted QR algorithm has quadratic convergence for each eigenvalue. This is true for almost all instances of the problem. Moreover, it is a long observed fact that the algorithm requires two iterations per eigenvalue. As a result it is possible to bound the actual running time of the eigenvalue computation by $10p^3$ for most cases. Furthermore the eigendecomposition algorithms are backward stable. We have been to able accurately compute the intersections of curves of degree up to seven. In practice it is possible to obtain accurate solutions for matrices of order 100 or more. This is in contrast with computing roots of high degree univariate polynomials (which may be an ill-conditioned problems) or using symbolic computation for determinant computation and finding the roots of the resulting polynomial expressed in Bernstein basis using subdivision and iteration (which is relatively expensive and has slow convergence).

The next step of the algorithm reorders the real Schur form (by similarity transformations) such that the eigenvalues lying in the domain of interest are at the top left of the Schur form. Let there be q eigenvalues in the domain of interest. For each such eigenvalue the eigenvectors are computed by solving the quasi upper triangular systems. The resulting vectors are multiplied by an orthogonal matrix to obtain the eigenvectors of C. The running time of these operations is $O(qp^2)$.

Example 4.3 *We illustrate the algorithm by considering the intersection of two rational cubic Bézier curves. The example is taken from [Sed83]. The control points of*

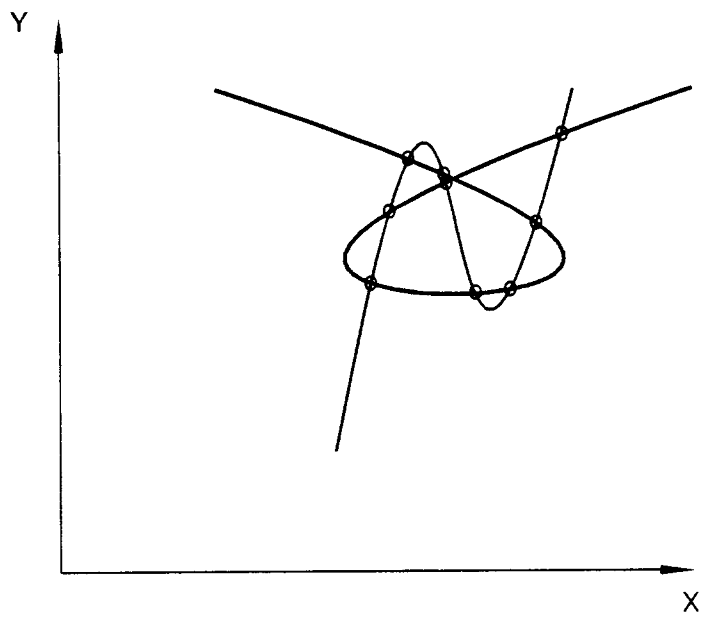

Figure 4: Intersection of rational cubic Bézier curves

two Bézier curves (as shown in Fig. 4), expressed in homogeneous coordinates, are (4,1,1), (5,6,2), (5,0,2), (6,4,1) *and* (7,4,1), (1,2,2),(9,2,2),(3,4,1). *Thus,*

$$\mathbf{P}(t) = (x(t), y(t), w(t)) = (4(1-t)^3 + 30(1-t)^2 t + 30(1-t)t^2 + 6t^3,$$

$$(1-t)^3 + 36(1-t)^2 t + 4t^3, (1-t)^3 + 6(1-t)^2 t + 6(1-t)t^2 + t^3).$$

The implicit representation has a matrix determinant formulation given as

$$\mathbf{M} = \begin{pmatrix} -114 + 30x - 6y & 30 - 6x - 6y & -10 + 3x - 2y \\ 30 - 6x - 6y & 1070 - 213x - 2y & 96 - 12x - 6y \\ -10 + 3x - 2y & 96 - 12x - 6y & -120 + 24x - 6y \end{pmatrix}.$$

The second parametrization, $\mathbf{Q}(u)$ *is substituted into the matrix formulation (after a reparametrization of the form* $s = \frac{u}{1-u}$*). The resulting matrix polynomial has the form*

$$\begin{aligned} \overline{M}(s) \;&=\; \begin{pmatrix} -48 & -12 & -9 \\ -12 & 423 & 36 \\ -9 & 36 & -72 \end{pmatrix} s^3 + \begin{pmatrix} 864 & -216 & 78 \\ -216 & -5106 & -144 \\ 78 & -144 & 504 \end{pmatrix} s^2 \\ &+ \begin{pmatrix} -576 & 72 & -66 \\ 72 & 5118 & 432 \\ -66 & 432 & -648 \end{pmatrix} s + \begin{pmatrix} 72 & -36 & 3 \\ -36 & -429 & -12 \\ 3 & -12 & 24 \end{pmatrix}. \end{aligned}$$

The exact condition number of the leading matrix is 9.525. *The LINPACK's estimator returns the approximate value as* 7.0621 [5].

Multiplying $\mathbf{M}(s)$ *with the inverse of the leading matrix and constructing the equivalent companion matrix results in*

$$\mathbf{C} = \begin{pmatrix} 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 1.481 & -1.07 & 0 & -11.92 & 4.581 & 0.401 & 17.84 & -8.248 & 0.401 \\ 0.133 & 0.946 & 0 & -0.534 & -11.923 & -0.229 & 1.069 & 11.43 & -0.23 \\ -0.076 & 0.440 & 0.333 & 0.306 & -0.534 & -9.165 & -0.613 & 4.747 & 6.835 \end{pmatrix}.$$

The eigendecomposition of $\mathbf{C}$ results in 9 points of intersection. The intersections points are computed using the relationship highlighted in (21). They are:

Num.	s_0	E_i	$u_0 = \frac{s_0}{1+s_0}$	$\alpha = \frac{v_1}{v_8}$	$\beta = \frac{v_2}{v_9}$	$t_0 = \frac{\beta}{\alpha+\beta}$	(X,Y)
1.	15.369	2.32e-14	0.9389	0.2173	0.0472	0.1785	(4.619, 3.412)
2.	11.802	2.85e-14	0.9219	0.6657	0.4432	0.3997	(4.911, 3.289)
3.	5.507	2.71e-14	0.8463	0.0703	1.000	0.9343	(5.688, 2.877)
4.	1.4654	1.27e-13	0.5944	0.1614	1.000	0.8610	(5.467, 2.321)
5.	0.5361	2.32e-14	0.3490	1.00	0.066	0.0622	(4.298, 2.378)
6.	0.1534	2.98e-14	0.133	1.00	0.1233	0.1099	(4.455, 2.971)
7.	0.0974	2.38e-14	0.0888	1.00	0.7277	0.4212	(4.931, 3.218)
8.	1.145	1.18e-13	0.534	1.00	0.4644	0.683	(4.174, 2.290)
9.	0.0382	1.14e-14	0.0369	0179	0.00032	0.9823	(5.901, 3.615)

Eigendecomposition and Intersection Points

where

- Num. is the number of intersection.
- s_0 is the eigenvalue of the matrix.
- E_i is the maximum error in the eigenvalue computation.
- u_0 is the parameter in the first curve obtained after reparametrization.
- v_1, v_2, v_8 and v_9 are the elements of the eigenvector corresponding to s_0.

[5] Using double precision arithmetic we have reduced matrix polynomials with leading matrices of condition numbers up to 100000 to eigenvalue problems.

- t_0 is the parameter in the domain of other curve obtained after reparametrization.
- (X, Y) is the intersection point on the curve.

In the columns corresponding to the components of the eigenvectors we choose the elements v_1 or v_8 depending upon their relative magnitudes. The error bounds in the third column are obtained by using the condition number of the eigenvalues (as explained in section 3) and matrix norm as

$$E_i = \epsilon \parallel \mathbf{C} \parallel \text{cond}_i, \tag{22}$$

where $\epsilon = 2.2204e - 16$ is the machine precision for 64 bit IEEE floating point arithmetic and cond_i is the condition number of the ith eigenvalue. As a result, the eigendecomposition algorithms computes the eigenvalues of C up to 12 digits of accuracy. The other sources of error arise from the computation of the entries of $\mathbf{M}$, the matrix corresponding to the implicit representation, and inverting the leading matrix of the matrix polynomial $\overline{\mathbf{M}}(s)$. In our case, this can account for inaccuracy of one digit (due to condition number of the matrix to be inverted). As a result, the intersection points are computed up to 11 digits of accuracy.

Example 4.4 *Lets again consider the intersection of algebraic plane curves highlighted in Example 2.9. In this case we are given two algebraic plane curves, $F(x,y)$ and $G(x,y)$ of degrees four and three, respectively. The resultant matrix obtained after eliminating y is:*

$$\mathbf{M}(x) = \mathbf{M}_0 + \mathbf{M}_1 x + \mathbf{M}_2 x^2 + \mathbf{M}_3 x^3 + \mathbf{M}_4 x^4 + \mathbf{M}_5 x^5 + \mathbf{M}_6 x^6$$

$$= \begin{pmatrix} 4x^2+4x^3+8x^4+4x^5-4x^6 & -4-14x^2+4x^3-14x^4+6x^5 & 4x+8x^3-2x^4 & -1-2x^2+x^3 \\ -4-14x^2+4x^3-14x^4+6x^5 & 4x+4x^2+12x^3-2x^4 & -1-6x^2-3x^3 & 0 \\ 4x+8x^3-2x^4 & -1-6x^2-3x^3 & 4-4x+6x^2 & 1 \\ -1-2x^2+x^3 & 0 & 1 & -1 \end{pmatrix}.$$

The entries of the matrix are polynomials of degree 6. However, the determinant of $\mathbf{M}(x)$ is a polynomial of degree 12 as the two curves intersect in 12 points, according to Bezout theorem. Therefore, the leading matrix, $\mathbf{M}_6$, is singular and it is not possible to reduce the matrix polynomial $\mathbf{M}(x)$ to an eigenvalue problem using Theorem 4.1. It is however, possible to reduce it to a 24×24 generalized eigenvalue problem using Theorem 4.2. This is relatively expensive as we are only interested in 12 roots.

Let us use the transformation, $x = 1/z$ and multiply $\mathbf{M}(1/z)$ by z^6. The new matrix polynomial is $\overline{\mathbf{M}}(z) = z^6\mathbf{M}(1/z)$. This implies that

$$\text{Determinant}\overline{\mathbf{M}}(z) = z^{24} \ \text{Determinant}\mathbf{M}(1/z).$$

The leading matrix, $\mathbf{M}_0$ is non-singular and therefore, the determinant of $\overline{\mathbf{M}}(z)$ is a polynomial of degree 24. The fact that the curves have 12 intersections implies that 12 of the 24 roots of Determinant$(\overline{\mathbf{M}}(z)) = 0$ correspond to zero. Moreover the condition number of $\mathbf{M}_0$ is 6.5552325. As a result, we can reduce it to an eigenvalue problem of a 24×24 matrix:

$$\mathbf{M} = \begin{pmatrix} \mathbf{0} & \mathbf{I} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{I} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{I} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{I} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{I} \\ -\mathbf{M}_0^{-1}\mathbf{M}_6 & -\mathbf{M}_0^{-1}\mathbf{M}_5 & -\mathbf{M}_0^{-1}\mathbf{M}_4 & -\mathbf{M}_0^{-1}\mathbf{M}_3 & -\mathbf{M}_0^{-1}\mathbf{M}_2 & -\mathbf{M}_0^{-1}\mathbf{M}_1 \end{pmatrix}.$$

The 12 known eigenvalues of $\mathbf{M}$ can be used in choosing the shifts in the QR algorithm, as highlighted in (13). After choosing a sufficient number of shifts corresponding to $s = 0$ (at most 12), the problem reduces to computing the eigendecomposition of a 12×12 Hessenberg matrix.

$\mathbf{M}$ has only two non-zero real eigenvalues $z = 2.794796$ and $z = -1.787942$. As a result, the two real intersections correspond to $x = 0.357807$ and $x = -0.559302$. The corresponding y values are obtained from the eigenvectors. The points of intersection are $(0.357807, -0.816289)$ and $(-0.559302, -0.9587738)$.

4.2 Improving the Accuracy

The accuracy of the intersection points can be further improved by a few iterations of Newton's method. The solutions computed using the eigendecomposition algorithms are used as the starting points for the Newton's iteration. We highlight the method for the intersection of parametric curves. It is also applicable to algebraic curve intersection.

Given $\mathbf{P}(t) = (x(t), y(t), w(t))$ and $\mathbf{Q}(u) = \overline{x}(u), \overline{y}(u), \overline{w}(u))$ we formulate the equations:

$$F(t,u) : \frac{x(t)}{w(t)} - \frac{\overline{x}(u)}{\overline{w}(u)} = 0$$

$$G(t,u) : \frac{y(t)}{w(t)} - \frac{\overline{y}(u)}{\overline{w}(u)} = 0$$

At each step of the Newton's iteration we improve the current solution $(t_0, u_0)^T$ to $(t_1, u_1)^T$ by

$$\begin{pmatrix} t_1 \\ u_1 \end{pmatrix} = \begin{pmatrix} t_0 \\ u_0 \end{pmatrix} - \begin{pmatrix} F_t(t_0,u_0) & G_t(t_0,u_0) \\ F_u(t_0,u_0) & G_u(t_0,u_0) \end{pmatrix} \begin{pmatrix} F(t_0,u_0) \\ G(t_0,u_0) \end{pmatrix},$$

where $F_t(t_0, u_0)$ and $G_u(t_0, u_0)$ correspond to the partial derivatives. This is repeated until we are able to achieve the desired accuracy (by looking at the norm of the vector $(F(t_0, u_0)\ G(t_0, u_0))^T$. In almost cases we have been able to compute the intersections accurately up to 11 or 12 digits using one or two steps of Newton's iteration after the eigendecomposition.

5 Higher Order Intersections

In the previous section, we presented an algorithm to compute the simple intersections of parametric curves. In this section we extend the analysis to higher order intersections.

According to Bezout's theorem two rational or algebraic curves of degree m and n intersect in mn points (counted properly) [Wal50]. 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 (the coefficients of the control polygon for rational parametric curves) results in* 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 $\mathbf{p}$ is a singular point on one of the curves. Some examples are highlighted in Fig. 5. 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.

In the previous section we showed that the intersection points correspond to the mn eigenvalues of $\mathbf{C}$ in (18) or the generalized eigenvalue problem, $\mathbf{C}_1 s + \mathbf{C}_2$ in (20). 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

$$\text{Determinant}(\mathbf{C} - s\mathbf{I}) = 0 \quad \text{or} \quad \text{Determinant}(\mathbf{C}_1 s - \mathbf{C}_2) = 0$$

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

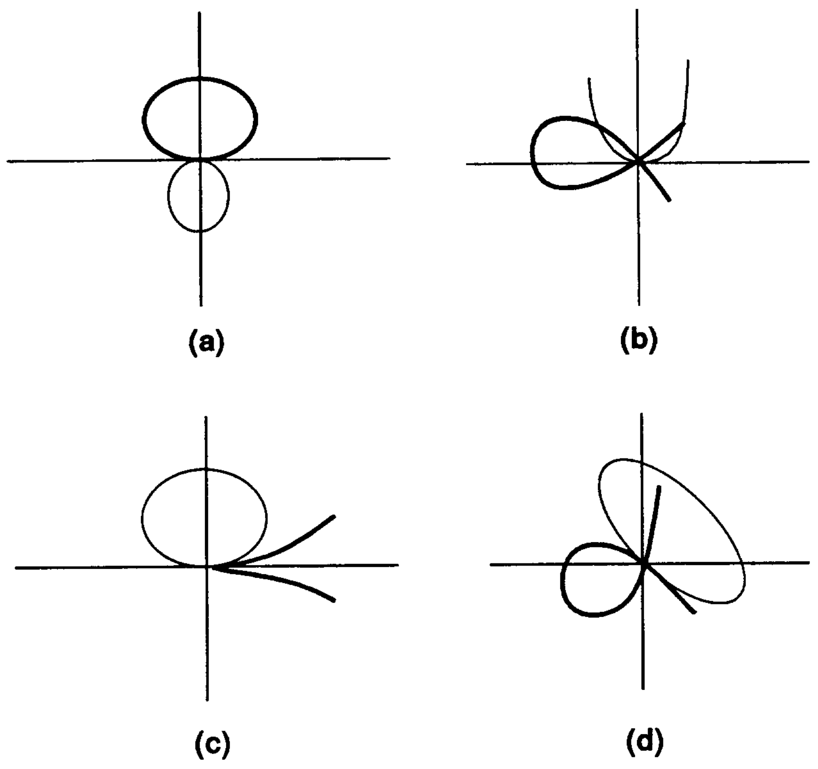

Figure 5: Higher order intersections

kernel of $\mathbf{C} - s_0\mathbf{I}$ or $\mathbf{C}_1 s_0 - \mathbf{C}_2$. 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 (9). 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 (10). 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.

5.1 Multiple 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 [BDM89]. Let $\mathbf{A}_\epsilon$ be a 11×11 matrix of the form:

$$\mathbf{A}_\epsilon = \begin{pmatrix} 0 & 1 & 0 & \dots & 0 & 0 \\ 0 & 0 & 1 & \dots & 0 & 0 \\ \vdots & \vdots & \dots & \dots & \vdots & \vdots \\ 0 & 0 & \dots & 0 & 1 & 0 \\ \epsilon & 0 & 0 & \dots & 0 & 0 \\ 0 & 0 & \dots & 0 & 0 & 0.5 \end{pmatrix}.$$

$\mathbf{A}_\epsilon$ is a block diagonal matrix with a 10×10 block at the upper left and a 1×1 block at the lower right. If $\epsilon = 0$, the upper left block is a 10×10 Jordan block [Wil65]. As a result, it has a single eigenvalue, $s = 0$, of algebraic multiplicity 10 and geometric multiplicity 1. The corresponding eigenvector is $\mathbf{v} = [1\ 0\ 0\ \ldots\ 0]^T$. For small values of ϵ the eigenvalues become distinct numbers all with absolute value $\epsilon^{\cdot 1}$ and the eigenvectors which have rotated away from $\mathbf{v}$ by about $\epsilon^{\cdot 1}$ radians. For $\epsilon = 1e - 10$, $\epsilon^{\cdot 1} = 0.1$, which is a much larger change. As a result the eigenvalue of $\mathbf{A}_\epsilon$ at $\epsilon = 0$ and the corresponding eigenvector is an ill-posed problem[6]. 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. In general, the arithmetic mean of this cluster of eigenvalues is much less sensitive to small perturbations than the individual eigenvalues. In Section 3.9 we summarized the bounds for condition numbers of clusters of eigenvalues. In particular, it has been shown that the mean of the cluster of eigenvalues is well-conditioned and it is bounded by $\| \epsilon \|$ as shown in (14). More details on the analysis are given in [Kat80].

We apply this technique to identify clusters of eigenvalues using their condition numbers. Given a cluster of eigenvalues, we compute the arithmetic mean of the eigenvalues of the cluster and the condition number of the mean. The latter information tells us about the accuracy of the multiple eigenvalue. In the rest of this section we demonstrate this method on the intersection of parametric curves.

Given two curves that intersect with multiplicity k at $\mathbf{p}$, our algorithm computes the implicit representation and reduces the problem to an eigenvalue problem. Let s be the eigenvalue corresponding to $\mathbf{p}$. The reduction to eigenvalue problem involves floating point operations on the coefficients of parametric curves, linear transformations, matrix inversion and matrix multiplication. As a result, the matrix $\mathbf{C}$ or $\mathbf{C}_1$ and $\mathbf{C}_2$ correspond to a slightly perturbed intersection problem[7]. Furthermore, k of its eigenvalues, say $s_1, s_2, \ldots, s_k$ are close to s. We proceed by identifying the cluster $s_1, s_2, \ldots, s_k$, computing the accurate number of digits and thereby expressing the eigenvalues with right number of digits as $s_1', s_2', \ldots, s_k'$. Finally we estimate the mean as

$$s' = \frac{s_1' + s_2' + \ldots + s_k'}{k}.$$

The condition number of the mean tells us about the number of accurate digits in s'. Depending upon the number of accurate digits we can also decide whether the

[6]This is due to the fact that the sensitivity of the eigenvalue is not proportional to the norm of the perturbation ϵ, but a root of ϵ.

[7]The perturbed problem is the intersection of parametric curves whose coefficients have been slightly perturbed.

problem is ill-posed.

To identify a cluster we use the condition numbers of the eigenvalues. In particular let the eigenvalues of $\mathbf{C}$ or $\mathbf{C}_1 s + \mathbf{C}_2$ be $s_1, s_2, \ldots, s_n$ and their condition numbers be $C_1, C_2, \ldots, C_n$. Based on condition numbers we estimate the error in the eigenvalue as

$$E_i = \epsilon \parallel \mathbf{C} \parallel \mathrm{cond}_i,$$

where ϵ is the machine precision. The exponent of E_i tells us about the number of accurate digits in s_i. Let us denote the number of accurate digits in s_i as d_i. Each s_i consists of a mantissa of l digits[8] and an exponent. However, we only use d_i of the l digits. As a result the mantissa of s_i consists of d_i digits at most. A cluster consists of all those eigenvalues whose exponents and the mantissa (consisting of up to d_i digits) are equal. In other words a cluster consists of k eigenvalues, $s_1' = s_{i_1}, s_2' = s_{i_2}, \ldots, s_k' = s_{i_k}$ such that they all have the same exponent and their mantissas agree to the first d_j digits, where

$$d_j = \mathrm{minimum}(d_{i_1}, d_{i_2}, \ldots, d_{i_k}),$$

and $s_{i_1}, s_{i_2}, \ldots, s_{i_k}$ is the largest subset of n eigenvalues satisfying this property. The rest of the algorithm proceeds by taking the mean of this cluster. To speed up the cluster identification process we sort the eigenvalues (containing up to d_i digits of mantissa) based on their magnitudes. The eigenvalues corresponding to the same cluster would appear next to each other in the sorted list. We highlight this algorithm on some examples.

Example 5.1 *Consider the intersection of cubic curve,* $\mathbf{P}(t) = (x(t), y(t)) = (t^2 - 1, t^3 - t)$ *with the parabola* $\mathbf{Q}(u) = (\overline{x}(u), \overline{y}(u)) = (u^2 + u, u^2 - u)$ *(as shown in Fig. 6). 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 6×6

[8]In case the computations are being performed in IEEE 64 bit floating point arithmetic, $l = 16$.

matrix

$$\mathbf{C} = \begin{pmatrix} 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ -1 & 0 & 1 & -1 & 0 & 0 \\ -1 & 0 & 1 & -2 & -1 & 0 \\ -1 & 0 & 1 & -2 & -2 & -1 \end{pmatrix} \qquad (23)$$

The eigenvalues of this matrix and their error estimates are:

Intersection Number (i)	*Eigenvalue* s_i	*Cond. Number* C_i	*Accurate Digits* d_i	s_i'
1.	*-2.58740105196*	*1.6402*	*15*	*-2.58740105196*
2.	*-0.20629947401 + j 1.374729636*	*1.38682*	*15*	*-0.20629947401 + j1.374729636*
3.	*-0.20629947401 - - j 1.374729636*	*1.386829*	*15*	*-0.20629947401 - - j 1.374729636*
4.	*-8.8380412e-17 + j 2.72753258842e-9*	*1.79612e08*	*8*	*0.0*
5.	*-8.8380412e-17 - j 2.72753258842e-9*	*1.79612e08*	*8*	*0.0*
6.	*0.0*	*56.35*	*14*	*56.35*

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' *and* s_6' *belong to the same cluster. Upon taking their average we find that their arithmetic mean is* 0.0 *and its condition number is* 26.82. *As a result, the mean is correct up to* 14 *digits and therefore,* $s' = 0.0$ *is an eigenvalue of multiplicity three.*

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

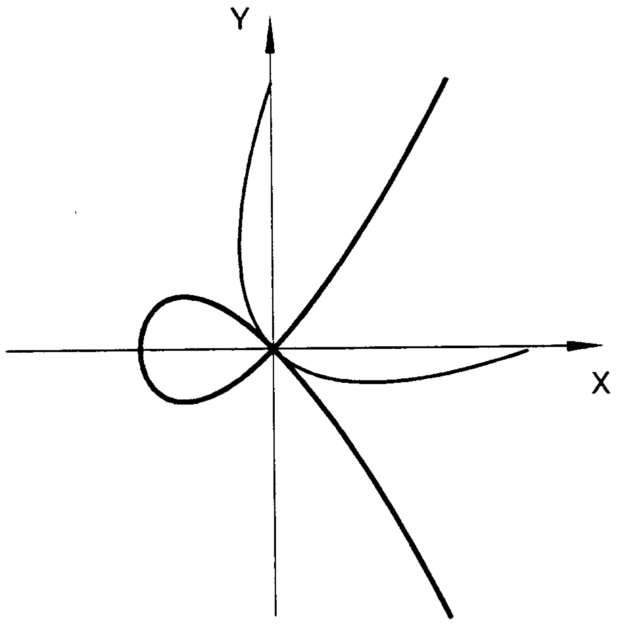

Figure 6: Higher order intersection of a cubic curve and a parabola

5.2 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 $f(t) = f_n t^n + f_{n-1}t^{n-1} + \ldots + f_1 t + f_0$. Without loss of generality we assume that $f_n = 1$. It is a well known fact in linear algebra that the roots of $f(t)$ correspond exactly to the eigenvalues of the companion matrix [Wil65]:

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

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 verify whether it is a root of $g(t)$ by computing the rank of $\mathbf{G} - t_1\mathbf{I}$ using singular value decompositions. To verify whether t_1 is a root of multiplicity greater than one, we 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 $\mathbf{G}$.

5.3 Parametric Curve Intersection

Given the higher multiplicity eigenvalue s and the corresponding intersection point, $\mathbf{p} = \mathbf{Q}(s) = (\overline{x}(s), \overline{y}(s), \overline{w}(s))$, 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 $\mathbf{C}_1 s - \mathbf{C}_2$. This can be accurately computed using singular value decompositions (SVD) of $\mathbf{C} - s\mathbf{I}$ or $\mathbf{C}_1 s - \mathbf{C}_2$. 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 $k((1-t_0)^{m-1}\ t_0(1-t_0)^{m-2}\ \ldots\ t_0^{m-1})^T$ 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:

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

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

where $(X_1, Y_1) = (\frac{\overline{x}(s)}{\overline{w}(s)}, \frac{\overline{y}(s)}{\overline{w}(s)})$. The number of common roots of these two equations is computed by the algorithm highlighted 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) [Der85].

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 $k((1-t_0)^{m-1}\ t_0(1-$

$t_0)^{m-2} \ \ldots \ t_0^{m-1})^T$. 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 5.2 *Let us again consider the intersection of a parabola and a cubic curve with a loop shown in Fig. 6. The curves are*

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

In Example 5.1 we computed the entries of the matrix $\mathbf{C}$, *as shown in (23) 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 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.*

5.4 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 $k(1 \;\; y_0 \;\; y_0^2 \;\; \ldots \;\; y_0^{m-1})^T$. 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.

Example 5.3 *Given*

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

and

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

These curves have been highlighted in Fig. 7. $P(x,y)$ is a nodal quartic having a self intersection at the origin, whereas $Q(x,y)$ has a cusp at the origin. The algorithm of intersection proceeds by computing by eliminating x from these two equations using the 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

$$\mathbf{M}(x) = \mathbf{M}_0 + \mathbf{M}_1 x + \mathbf{M}_2 x^2 + \mathbf{M}_3 x^3 + \mathbf{M}_4 x^4,$$

where the leading matrix is

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

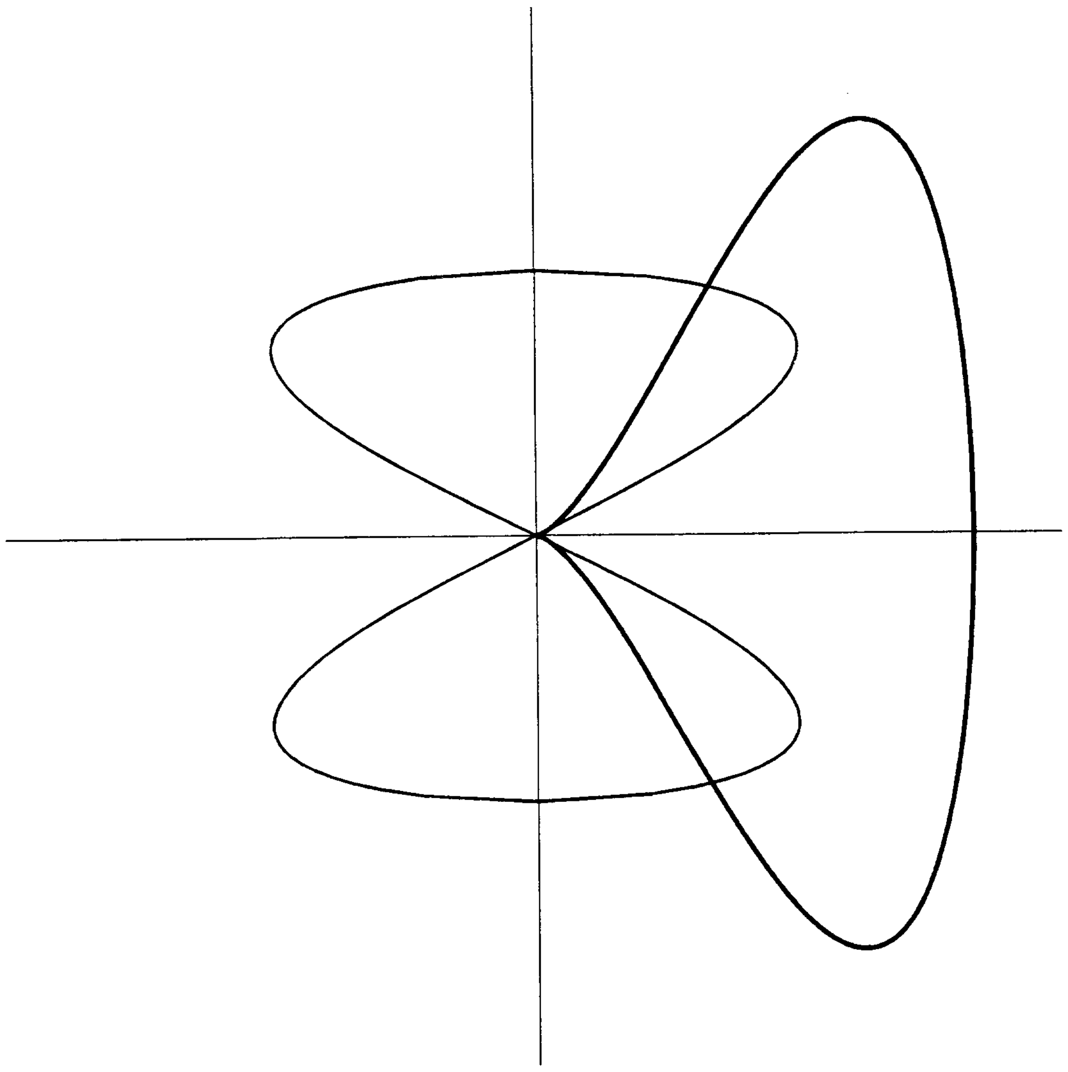

Figure 7: Intersection of quartic algebraic curves

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* 16×16 *matrix is*

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

$\mathbf{C}$ *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.

6 Conclusion

In this chapter we have highlighted a new technique for computing the intersection of parametric and algebraic curves. 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. The algorithm is very robust and can accurately compute the intersection points. 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.

References

[ABB$^+$92] 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.

[Abh88] S. S. Abhyankar. What is the difference between a parabola and a hyperbola. *The Mathematical Intelligencer*, 10(4), 1988.

[BBB87] 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.

[BDM89] 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).

[Chi90] E.W. Chionh. *Base Points, Resultants and the Implicit Representation of Rational Surfaces.* PhD thesis, University of Waterloo, Department of Computer Science, 1990.

[CK92] 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.

[Dem89] 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.

[Der85] Tony D. Derose. *Geometric Continuity: A Parametrization independent measure of continuity for computer aided geometric design.* PhD thesis, University of California at Berkeley, 1985.

[EC90] G. Elber and E. Cohen. Hidden curve removal for free form surfaces. *Computer Graphics*, 24(4):95–104, 1990.

[FR87] R.T. Farouki and V.T. Rajan. On the numerical condition of polynomials in bernstein form. *Computer Aided Geometric Design*, 4:191–216, 1987.

[GBDM77] 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.

[GL89] G.H. Golub and C.F. Van Loan. *Matrix Computations.* John Hopkins Press, Baltimore, 1989.

[GLR82] I. Gohberg, P. Lancaster, and L. Rodman. *Matrix Polynomials.* Academic Press, New York, 1982.

[Hob91] J.D. Hobby. Numerically stable implicitization of cubic curves. *ACM Transactions on Graphics*, 10(3):255–296, 1991.

[Hof89] C.M. Hoffmann. *Geometric and Solid Modeling.* Morgan Kaufmann, San Mateo, California, 1989.

[Kat80] T. Kato. *Perturbation Theory for Linear Operators.* Springer Verlag, Berlin, 2 edition, 1980.

[KM83] 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.

[LR80] 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.

[Mac02] F.S. Macaulay. On some formula in elimination. *Proceedings of London Mathematical Society*, pages 3–27, May 1902.

[Man90] D. Manocha. Regular curves and proper parametrizations. In *Proceedings of International Symposium on Symbolic and Algebraic Computations*, pages 271–276, 1990.

[Man92] 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.

[MC91] D. Manocha and J.F. Canny. Rational curves with polynomial parametrization. *Computer-Aided Design*, 23(9):645–652, 1991.

[MC92] D. Manocha and J.F. Canny. Detecting cusps and inflection points in curves. *Computer Aided Geometric Design*, 9:1–24, 1992.

[MM89] R.P. Markot and R. L. Magedson. Solutions of tangential surface and curve intersections. *Computer-Aided Design*, 21(7):421–427, 1989.

[Sal85] G. Salmon. *Lessons Introductory to the Modern Higher Algebra.* G.E. Stechert & Co., New York, 1885.

[Sed83] T.W. Sederberg. *Implicit and Parametric Curves and Surfaces.* PhD thesis, Purdue University, 1983.

[Sed86] T.W. Sederberg. Improperly parametrized rational curves. *Computer Aided Geometric Design*, 3:67–75, 1986.

[Sed89] T.W. Sederberg. Algorithms for algebraic curve intersection. *Computer-Aided Design*, 21(9):547–555, 1989.

[SK59] J.G. Semple and G.T. Kneebone. *Algebraic Curves.* Oxford University Press, London, 1959.

[SP86] T.W. Sederberg and S.R. Parry. Comparison of three curve intersection algorithms. *Computer-Aided Design*, 18(1):58–63, 1986.

[Stu91] B. Sturmfels. Sparse elimination theory. Technical Report 91-32, Mathematical Sciences Institute, Cornell University, 1991. Presented at *Computational Algebraic Geometry and Commutative Algebra* in Cortona, Italy in June 1991.

[SWZ89] T.W. Sederberg, S. White, and A. Zundel. Fat arcs: A bounding region with cubic convergence. *Computer Aided Geometric Design*, 6:205–218, 1989.

[Wal50] R.J. Walker. *Algebraic Curves.* Princeton University Press, New Jersey, 1950.

[Wil59] 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.

[Wil65] J.H. Wilkinson. *The algebraic eigenvalue problem.* Oxford University Press, Oxford, 1965.