Algorithms for Intersecting Parametric and Algebraic Curves I: Simple Intersections

DINESH MANOCHA
University of North Carolina at Chapel Hill

JAMES DEMMEL
University of California at Berkeley

The problem of computing the intersection of parametric and algebraic curves arises in many applications of computer graphics and 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 article 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 problem of intersection is reduced to that of computing the eigenvalues and eigenvectors of a numeric matrix. 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.

Categories and Subject Descriptors: F.2.1 [**Analysis of Algorithms and Problem Complexity**]: Numerical Algorithms and Problems—*computations on matrices*; I.3.5 [**Computer Graphics**]: Computational Geometry and Object Modeling—*curve, surface, solid and object representations*

General Terms: Algorithms, Performance, Theory

Additional Key Words and Phrases: Algebraic, curves, eigenvalues, intersection, parametric, resultants, robustness

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 [Hoffmann

Dinesh Monocha was supported by an IBM Fellowship, a David and Lucille Packard Fellowship, National Science Foundation grant ASC-9005933 and DARPA ISTO contract DAEA 18-90-C-0044. James Demmel was supported in part by National Science Foundation grants DCR-8552474, ASC-8715728, and ASC-9005933.

Authors' addresses: D. Monocha, Department of Computer Science, University of North Carolina at Chapel Hill, Chapel Hill, NC 27599; J. Demmel, Computer Science Division and Mathematics Department, University of California at Berkeley, Berkeley, CA 94720.

1989; Markot and Magedson 1989; Sederberg 1989; Sederberg and Parry 1986]. 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. [Collins and Krandick 1992; Elber and Cohen 1990]. 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 [Sederberg 1983], Bézier subdivision [Lane and Riesenfeld 1980], and interval arithmetic [Koparkar and Mudur 1983] 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 [Sederberg 1983]. 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 that of computing the roots of the resulting polynomial.

The Bézier subdivision relies on the convex-hull property of Bézier curves and de Casteljau's 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 curve segments are approximated by straight lines up to certain tolerance, their intersection point is accepted as the intersection of two curves. It has been improved by Sederberg et al. [1989] 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, we obtain 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 Sederberg [1986]. 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 [Sederberg et al. 1989]. 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 [Hobby 1991]. 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 [Sederberg and Parry 1986]. 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 [Wilkinson 1959]. As a result the intersection algorithm involves (1) representing matrix entries as linear combinations of Bernstein polynomials, (2) multiplying Bernstein polynomials for expanding the determinant, (3) 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 [Sederberg 1989]. However, resultant-based algorithms are considered to be the fastest for lower-degree curves.

In this article 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*,[1] and fast implementations are available as part of packages like EISPACK and LAPACK [Anderson et al. 1992; Golub and Van Loan 1989]. The algorithm for intersecting algebraic curves is similar to that for parametric curves.

The rest of the article 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 problem of intersecting parametric and algebraic curves can be reduced to that of computing roots of polynomial expressed as a matrix determinant. We also highlight a number of properties of the matrix determinants corresponding to

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

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.

2. PARAMETRIC AND ALGEBRAIC CURVES

A rational Bézier curve is of the form [Bartels et al. 1987]:

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

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

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

For polynomial curves the denominator term is a constant. Other rational formulations like B-splines can be converted into a series of rational Bézier curves by knot insertion algorithms [Bartels et al. 1987]. Thus, the problem of intersecting rational curves can be reduced to intersecting rational 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 Figure 1.

Algebraic plane curves are generally expressed in standard power basis:

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

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.

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* [Walker 1950]. Algorithms to compute the proper parametrizations of curves have been described in Manocha [1990]; Manocha and Canny [1992]; and Sederberg [1986].

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 [Walker

Num.	s_0	E_i	$u_0 = \frac{s_0}{1+s_0}$	$\alpha = \frac{v_1}{v_3}$	$\beta = \frac{v_2}{v_3}$	$t_0 = \frac{\beta}{\alpha+\beta}$	(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

Figure 1

1950]. 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.*

2.1 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 [Macauley 1902; Salmon 1885]. 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 nontrivial solution. As far as geometric and solid modeling are concerned, the use of resultants was resurrected by Sederberg for implicitizing parametric curves and surfaces [Sederberg, 1983]. In this paper we will deal with the resultants of two polynomials in one unknown. Surveys on various formulations of resultants are given in Sederberg [1983] and Sturmfels [1991], and effective techniques for computing and applying them are presented in Manocha [1992].

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 Bezout or Cayley and Sylvester [Salmon 1885]. Each of them expresses the resultant as the determinant of a matrix. The order of the matrix is different for different methods. We use Cayley's formulation of Bezout resultant since it results in a matrix of lower order as compared to Sylvester's formulation.

Let the two polynomials be $F(x)$ of degree m and $G(x)$ of degree n. Without loss of generality we assume that $m \geq n$. Lets 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

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

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

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

Let us denote the $m \times m$ matrix by **M**. The determinant of **M** is the resultant of $F(x)$ and $G(x)$ [Salmon 1885]. 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 **M** is singular, and $\mathbf{v} = [1 \;\; x_0 \;\; x_0^2 \;\; \cdots \;\; x_0^{m-1}]^T$ is a vector in the *kernel* of **M**. In other words, if we substitute $x = x_0$ in (2), the product of **M** and the right-hand-side vector **v** is a null vector:

$$\begin{pmatrix} P_{0,0} & P_{0,1} & \cdots & P_{0,m-1} \\ P_{1,0} & P_{1,1} & \cdots & P_{1,m-1} \\ \vdots & \vdots & \vdots & \vdots \\ P_{m-1,0} & P_{m-1,1} & \cdots & P_{m-1,m-1} \end{pmatrix} \begin{pmatrix} 1 \\ x_0 \\ x_0^2 \\ \vdots \\ x_0^{m-1} \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ \vdots \\ 0 \end{pmatrix}.$$

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. It has also been used to implicitize Bézier curves in Goldman et al. [1984].

2.2 Implicitizing Parametric Curves

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

$$\begin{aligned} \mathbf{p}(t) &= (x(t), y(t), w(t)) \\ &= \left(\sum_{i=0}^{m} w_i X_i B_{i,m}(t), \right. \\ &\qquad \left. \sum_{i=0}^{m} w_i Y_i B_{i,m}(t), \sum_{i=0}^{m} w_i B_{i,m}(t) \right). \end{aligned}$$

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

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

is a constant.

To implicitize the curve we consider the following system of equations

$$F_1(t)\colon Xw(t) - x(t) = 0$$
$$F_2(t)\colon Yw(t) - y(t) = 0. \tag{3}$$

Consider them as polynomials in t, and X, Y are treated as symbolic coefficients. The implicit representation corresponds to the resultant of (3).

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 $(f(t)g(\alpha) - f(\alpha)g(t))/(t - \alpha)$ corresponds to a polynomial and can be expressed as product of matrices and vectors, as shown in (2). In other words,

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

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 the determinant of the matrix $\mathbf{M}$, where $\mathbf{M}$ is defined as:

$$\mathbf{M} = X\mathbf{M}_1 + Y\mathbf{M}_2 + \mathbf{M}_3. \tag{4}$$

Given $f(t), g(t)$, polynomials of degree m, we compute the $m \times m$ coefficient matrix $\mathbf{P}$ corresponding to $(f(s)g(t) - f(t)g(s))/(s - t)$ in the following manner. Let $F[0] \cdots F[m]$ and $G[0] \cdots G[m]$ correspond to the coefficients of the polynomials $f(s)$ and $g(s)$. Moreover, let a monomial in two variables s and t correspond to $a_{i,j}s^i t^j$, where $a_{i,j}$ is the coefficient. Such a monomial is represented as a typed-structure *Monomial*, where

$$\text{Monomial.coeff} = a_{i,j},$$
$$\text{Monomial.s} = i$$

and

$$\text{Monomial.t} = j.$$

Furthermore, let $Poly[0] \cdots Poly[max]$ denote a collection of *Monomial* structures. 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. $Monomial.coeff = F[i] * G[j]$.
      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. $Monomial.coeff = -F[i] * G[j]$.
      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) $\mathbf{P}[j][k] = \mathbf{P}[j][k] + Monomial.coeff$.

One advantage of the algorithm lies in the fact that the computation is purely numeric. As a result, we can use error analysis techniques and come up 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 the power basis. However, converting from Bézier to power basis can introduce numerical errors [Farouki and Rajan 1987]. To circumvent this problem we perform a reparametrization.

Given

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

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

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

Let $s = (t)/(1 - t)$, and the resulting parametrization is

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

The rest of the algorithm proceeds by computing the implicit representation of $\bar{\mathbf{p}}(s)$ and computing a matrix formulation by Cayley's method, as shown in (2). This corresponds to linearizing the system of equations (3) into

$$\mathbf{M} \begin{pmatrix} 1 \\ s \\ s^2 \\ \vdots \\ s^{m-1} \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ \vdots \\ 0 \end{pmatrix}.$$

Let us substitute $s = (t)/(1 - t)$ and multiply the right-hand side vector by $(1 - t)^{m-1}$. The resulting linear formulation is

$$\mathbf{M} \begin{pmatrix} (1-t)^{m-1} \\ t(1-t)^{m-2} \\ t^2(1-t)^{m-3} \\ \vdots \\ t^{m-1} \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ \vdots \\ 0 \end{pmatrix}. \tag{5}$$

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

2.3 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 (4).

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

$$(1 \;\; t_1 \;\; t_1^2 \;\; \cdots \;\; t_1^{m-1})^T.$$

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

Singular- or higher-multiplicity points on the parametric curves are analyzed in Manocha and Demmel [1992], and many properties of the matrix corresponding to the implicit representation are highlighted as well.

Example 2.1. Consider the parametric curve:

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

Using the algorithm we obtain its implicit representation as

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

Given $(X, Y) = (4, 1)$, we see that (1) the matrix obtained after substituting these values is singular and (2) a vector in its kernel is $[1\ 0\ 0]^T$. As a result, the preimage of $(4, 1)$ is $t = 0$.

2.4 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 an $m \times m$ matrix $\mathbf{M}$, whose entries are linear combinations of symbolic coefficients X, Y. The second parametrization $\mathbf{Q}(u) = (\bar{x}(u), \bar{y}(u), \bar{w}(u))$ is substituted into the matrix formulation, (4), as

$$X = \frac{\bar{x}(u)}{\bar{w}(u)},$$

$$Y = \frac{\bar{y}(u)}{\bar{w}(u)}.$$

The entries of the resulting matrix are rational functions in terms of u, and we multiply them by $\bar{w}(u)$ to obtain a matrix with polynomial entries. The latter matrix is represented as $\mathbf{M}(u)$. It follows from the properties of the implicit representation and resultants that the intersection points correspond to the roots of

$$\text{Determinant}(\mathbf{M}(u)) = 0. \tag{6}$$

In the next section, we reduce the problem of computing roots of the above equation into an eigenvalue problem.

2.5 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 the power basis or the 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 Figure 2. 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.

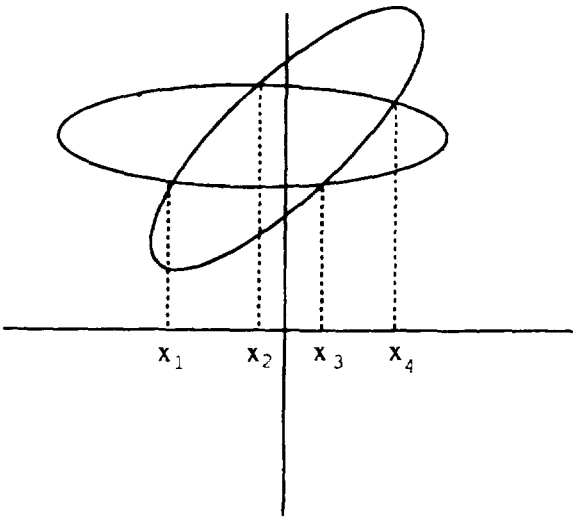

Fig. 2. Intersection and projection of two ellipses.

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

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

and

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

where both $F_i(x)$ 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 the 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

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

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

$$\text{Determinant}(\mathbf{M}(x)) = 0. \tag{7}$$

Example 2.2. 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

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

In case the two algebraic curves are not of the same degree, Cayley's formulation of the resultant may result in an extraneous factor. That is, given algebraic curves, $F(x, y)$ of degree m and $G(x, y)$ of degree n, Cayley's resultant corresponding to Determinant $(\mathbf{M}(x))$ can have degree greater than mn. In such cases, the algorithm may result in some extraneous solutions, which can be eliminated later by substituting them back into the equations $F(x, y) = 0$ and $G(x, y) = 0$.

The other possibility is to use Sylvester's formulation of the resultant of two polynomial equations. After eliminating y from the two equations, Sylvester's resultant produces a matrix of order $m + n$, $\mathbf{M}(x)$. The construction of Sylvester's resultant is explained in Hoffmann [1989].

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.

3.1 Eigenvalues and Eigenvectors

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

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

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 are the elements on its diagonal. Efficient algorithms for computing eigenvalues and eigenvectors are well known [Golub and Von Loan 1989], and their implementations are available as part of packages EISPACK [Garbow et al. 1977], and LAPACK [Anderson et al. 1992; Demmel 1989]. Most algorithms make use of the similarity transformations of the form $\mathbf{A}' = \mathbf{QAQ}^{-1}$, where $\mathbf{Q}$ is any nonsingular $n \times n$ 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 [Golub and Van Loan 1989]. $\mathbf{Q}$ is an orthogonal matrix if $\mathbf{QQ}^T = \mathbf{I}$. Given $\mathbf{A}$, the eigendecomposition algorithm converts it into a Hessenberg matrix, $\mathbf{H}$, using a sequence of similarity transformations by orthogonal matrices.

A Hessenberg matrix is of the form

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

$\mathbf{H}$ and $\mathbf{A}$ are related as

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

where $\mathbf{Q}$ is an orthogonal matrix. Given $\mathbf{H}$, the eigendecomposition algorithm proceeds by similarity transformations by orthogonal matrices. Each of these similarity transformation corresponds to a QR iteration of the form

$$\mathbf{H} - q\mathbf{I} = \mathbf{UR}, \tag{8}$$

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

$$\mathbf{H} = \mathbf{RU} + q\mathbf{I}.$$

The shifts are chosen appropriately such that the matrix converges to its *real Schur decomposition* of the form [Golub and Van Loan 1989; Wilkinson

1965]:

$$\mathbf{QAQ}^{-1} = \begin{pmatrix} R_{11} & R_{12} & \dots & R_{1m} \\ 0 & R_{22} & \dots & R_{2m} \\ \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \dots & R_{mm} \end{pmatrix}, \tag{9}$$

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. Often a matrix has complex eigenvalues, and the above algorithm is modified to double shift consisting of a complex number and its conjugate. More details are given in Golub and Van Loan [1989]. 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 $\mathbf{Q}$, computing eigenvectors corresponds to solving quasi-triangular systems [Golub and Van Loan 1989; Wilkinson 1965]. The running time of these algorithms is $\mathcal{O}(n^3)$. 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, s is an eigenvalue of $\mathbf{A}$, and let the QR factorization of $\mathbf{H} - s\mathbf{I}$ be

$$\mathbf{H} - s\mathbf{I} = \mathbf{U}'\mathbf{R}'. \tag{10}$$

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 of a Hessenberg matrix of lower order.

3.2 Generalized Eigenvalue Problem

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

$$\mathbf{Ax} = s\mathbf{Bx}.$$

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 nonsingular 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:

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

However, $\mathbf{B}$ may have a high condition number, and such a reduction may be numerically unstable. A better algorithm, called the QZ algorithm [Golub and Van Loan 1989], applies orthogonal transformations to $\mathbf{A}$ and $\mathbf{B}$ to reduce A to Hessenberg form, to reduce B to upper triangular form, and then *implic-*

itly perform the QR algorithm on $B^{-1}A$ without ever forming it. This algorithm is in EISPACK [Garbow et al. 1977] and in the most recent release of LAPACK. 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 the *QR* algorithm for computing eigenvalues and eigenvectors of a matrix.

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 roundoff errors. The rank of a matrix can also be computed by Gauss elimination. However, there arise many situations where near rank deficiency prevails. Rounding errors and fuzzy data make rank determination a nontrivial exercise. In these situations, the numerical rank is easily characterized in terms of the SVD.

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

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

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

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

Moreover, $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_n \geq 0$. 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 [Golub and Van Loan 1989]. The relationship between the elements of $\mathbf{A}$, singular values, and singular vectors can be expressed as

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

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 [Golub and Van Loan 1989].

3.4 Condition Numbers

The condition number of a problem measures the sensitivity of a solution to small changes in the input. A problem is *ill-conditioned* if its condition number is large and *ill-posed* if its condition number is infinite. These condition numbers are used to bound errors in computed solutions of numerical problems. More details on condition numbers are given in Golub and Van Loan [1989] and Wilkinson [1965]. The implementations of these condition number computations are available as part of LAPACK [Bai et al. 1993].

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

3.5 Condition Number of a Square Matrix

The *condition number of a square matrix* corresponds to $\sigma_1(\mathbf{A})/\sigma_n(\mathbf{A})$, 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.

3.6 Condition Number of Simple Eigenvalues

Let s be a *simple eigenvalue*[2] of the $n \times n$ matrix $\mathbf{A}$ with unit right eigenvector $\mathbf{x}$ and unit left eigenvector $\mathbf{v}$. 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 (the Euclidean length) of a vector. Let $\mathbf{P} = (\mathbf{x} \cdot \mathbf{y}^T)/(\mathbf{y}^T \cdot \mathbf{x})$. Therefore, $\|\mathbf{P}\|_2 = 1/|\mathbf{y}^T\mathbf{x}|$. 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 $\|\mathbf{P}\|_2$. Higher multiplicity eigenvalues are being considered in Manocha and Demmel [1992].

3.7 Accuracy of Right Eigenvectors

Detailed error bounds for eigenvectors are given in Anderson et al. [1992] and Wilkinson [1965]. We will not use these detailed bounds beyond the following implication. We need to take ratios of certain eigenvector components, therefore we choose those components with the highest relative accuracy. Since the bounds are on the 2-norm of the error, this tells us that the largest components are likely to have the highest relative accuracy.

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 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

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

determinant as shown in (6). 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 (7). Let us represent it as a *matrix polynomial*

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

where $\mathbf{M}_i$ is an $m \times m$ matrix with numeric entries. More details on matrix polynomials and their properties are given in Gohberg et al. [1982]. On dividing the $\mathbf{M}(u)$ by $(1-u)^n$ we obtain a matrix polynomial of the form

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

Substitute $s = u/(1-u)$, and the new polynomial is of the form

$$\mathbf{L}(s) = \mathbf{M}_n s^n + \mathbf{M}_{n-1} s^{n-1} + \cdots + \mathbf{M}_0. \tag{11}$$

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 $P(s)$ = Determinant($\mathbf{L}(s)$) = 0 in the domain $[0, \infty]$. This can result in very large solutions which may be quite inaccurate or even overflow 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 reducing it to a generalized eigenvalue problem.

Let us consider the case when $\mathbf{M}_n$ is a nonsingular and well-conditioned matrix. Computation of the inverse of $\mathbf{M}_n$ does not, therefore, introduce significant numerical errors. Let

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

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

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

As a result $\bar{\mathbf{L}}(s_0)$ is a singular matrix, and there is at least one nontrivial vector in its kernel. Let us denote such an $m \times 1$ vector as $\mathbf{v}$, that is

$$\bar{\mathbf{L}}(s_0)\mathbf{v} = \mathbf{0}, \tag{12}$$

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*

$$\mathbf{C} = \begin{bmatrix} \mathbf{0} & \mathbf{I}_m & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{I}_m & \cdots & \mathbf{0} \\ \vdots & \vdots & \cdots & \vdots & \vdots \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \cdots & \mathbf{I}_m \\ -\overline{\mathbf{M}}_0 & -\overline{\mathbf{M}}_1 & -\overline{\mathbf{M}}_2 & \cdots & -\overline{\mathbf{M}}_{n-1} \end{bmatrix}, \tag{13}$$

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* $s = s_0$ *is of the form*:

$$\begin{bmatrix} \mathbf{v} & s_0\mathbf{v} & s_0^2\mathbf{v} & \cdots & s_0^{n-1}\mathbf{v} \end{bmatrix}^T,$$

where $\mathbf{v}$ *is the vector in the kernel of* $\overline{\mathbf{L}}(s_0)$ *as defined in* (12).

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 mn. Let $s = s_0$ be an eigenvalue of $\mathbf{C}$. As a result there is a nontrivial vector $\mathbf{V}$ in the kernel of $\mathbf{C} - s_0\mathbf{I}$. Furthermore, we represent $\mathbf{V}$ as

$$\mathbf{V} = \begin{bmatrix} \mathbf{v}_1^T & \mathbf{v}_2^T & \cdots & \mathbf{v}_n^T \end{bmatrix},$$

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

$$\begin{bmatrix} \mathbf{0} & \mathbf{I}_m & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{I}_m & \cdots & \mathbf{0} \\ \vdots & \vdots & \cdots & \vdots & \vdots \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \cdots & \mathbf{I}_m \\ -\overline{\mathbf{M}}_0 & -\overline{\mathbf{M}}_1 & -\overline{\mathbf{M}}_2 & \cdots & -\overline{\mathbf{M}}_{n-1} \end{bmatrix} \begin{bmatrix} \mathbf{v}_1 \\ \mathbf{v}_2 \\ \vdots \\ \mathbf{v}_{n-1} \\ \mathbf{v}_n \end{bmatrix} = s_0 \begin{bmatrix} \mathbf{v}_1 \\ \mathbf{v}_2 \\ \vdots \\ \mathbf{v}_{n-1} \\ \mathbf{v}_n \end{bmatrix}. \tag{14}$$

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

$$\begin{aligned} \mathbf{v}_2 &= s_0\mathbf{v}_1, \\ \mathbf{v}_3 &= s_0\mathbf{v}_2, \\ &\vdots \\ \mathbf{v}_n &= s_0\mathbf{v}_{n-1} \end{aligned}$$

and

$$-\overline{\mathbf{M}}_0\mathbf{v}_1 - \overline{\mathbf{M}}_1\mathbf{v}_2 - \overline{\mathbf{M}}_2\mathbf{v}_3 - \cdots - \overline{\mathbf{M}}_{n-1}\mathbf{v}_n = s_0\mathbf{v}_n.$$

These relations imply

$$\mathbf{v}_i = s_0^{i-1}\mathbf{v}_1, \quad \text{for } 1 \le i \le n$$

and

$$-\left(\overline{\mathbf{M}}_0 + s_0\overline{\mathbf{M}}_1 + s_0^2\overline{\mathbf{M}}_2 + \cdots + s_0^{n-1}\overline{\mathbf{M}}_{n-1} + s_0^n\mathbf{I}\right)\mathbf{v}_1 = \mathbf{0}.$$

Equating the above relation with (12) results in the fact that s_0 is a solution of Determinant($\overline{\mathbf{L}}(s)$) = 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 nonsingular, $P(s)$ 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 $\mathbf{C}$. □

The matrix polynomials have been used to solve general systems of nonlinear polynomial equations. More details are highlighted in Manocha [1992]. The relationship between the eigenvalues of $\mathbf{C}$ and the roots of $P(s)$ has been proved using similarity transformations in Gohberg et al. [1982]. At times the leading matrix $\mathbf{M}_n$ is singular or close to being singular (due to high condition number). Therefore, reducing it to an eigenvalue problem using Theorem 4.1 may not be numerically stable. 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*

$$\mathbf{C}_1 = \begin{bmatrix} \mathbf{I}_m & \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & \mathbf{I}_m & \mathbf{0} & \cdots & \mathbf{0} \\ \vdots & \vdots & \cdots & \vdots & \vdots \\ \mathbf{0} & \mathbf{0} & \cdots & \mathbf{I}_m & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} & \mathbf{M}_n \end{bmatrix}$$

$$\mathbf{C}_2 = \begin{bmatrix} \mathbf{0} & \mathbf{I}_m & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{I}_m & \cdots & \mathbf{0} \\ \vdots & \vdots & \cdots & \vdots & \vdots \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \cdots & \mathbf{I}_m \\ -\mathbf{M}_0 & -\mathbf{M}_1 & -\mathbf{M}_2 & \cdots & -\mathbf{M}_{n-1} \end{bmatrix}, \tag{15}$$

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 4.1 and can also be proved using similarity transformations, as highlighted in Gohberg et al. [1982].

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 $u_0 = s_0/(1 + s_0)$. 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 $t \in [0, 1]$. Therefore, it is important 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 (5) and Theorem 4.1.

Let us assume that (x_0, y_0, w_0) is a simple point on $\mathbf{P}(t)$. Algorithms to deal with higher multiplicity points are described in Manocha and Demmel [1992]. Substitute for $(X, Y, W) = (x_0, y_0, w_0)$ in the matrix $\mathbf{M}$ as shown in (5), 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, Manocha and Demmel [1992]. The vector in the kernel corresponds to $\mathbf{v}$ shown in (12). 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, $t_0 = v_2/(v_1 + v_2)$. The relationship between the eigenvalue s_0 of $\mathbf{C}$, 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) is expressed as

$$(x_0, y_0, w_0) = \mathbf{Q}\left(\frac{s_0}{1+s_0}\right) = \mathbf{P}\left(\frac{v_2}{v_1+v_2}\right). \tag{16}$$

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 simple eigenvalue system (almost by 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 (11) is transformed into

$$\begin{aligned} \mathbf{P}(\bar{s}) &= (c\bar{s}+d)^n \mathbf{M}\left(\frac{a\bar{s}+b}{c\bar{s}+d}\right) \\ &= \mathbf{M}_n(a\bar{s}+b)^n + \mathbf{M}_{n-1}(a\bar{s}+b)^{n-1}(c\bar{s}+d) + \cdots \\ &\quad + \mathbf{M}_1(a\bar{s}+b)(c\bar{s}+d)^{n-1} + \mathbf{M}_0(c\bar{s}+d)^n \\ &\Rightarrow \mathbf{P}(\bar{s}) = \mathbf{P}_n\bar{s}^n + \mathbf{P}_{n-1}\bar{s}^{n-1} + \cdots + \mathbf{P}_1\bar{s} + \mathbf{P}_0, \end{aligned}$$

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; other-

wise 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 a singular or 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 on the signs of a, b, c, and d, where $s_1 = -b/a$ and $s_2 = -d/c$.

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 seemingly 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 as right eigenvectors of the matrix.

We used the LAPACK implementation of the *QR* algorithm for eigendecomposition [Anderson et al. 1992]. The *QR* algorithm computes all the eigenvalues of the given matrix. However, we are interested in the positive real eigenvalues only. To account for numerical errors, the implementation selects all eigenvalues of the form $\alpha + j\beta$, where $\alpha > -\epsilon$, $|\beta| < \epsilon$, and $j = \sqrt{-1}$, ϵ is a small positive constant. 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

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

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 $\mathbf{v} = [v_1\ v_2\ \cdots\ v_m]^T$ as:

If $s_0 \geq 1$,

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

otherwise

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

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 QR algorithm computes all the eigenvalues of the given matrix. It is difficult to restrict them to computing eigenvalues in the domain of interest without any heuristics, although some progress on large matrices on parallel machines has recently been made [Bai and Demmel 1993]. The order of the matrix, say p, corresponds to the product of the degree of the two curves, and the number of eigenvalues is equal to the order of the matrix. 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 able to accurately compute the intersections of curves of degree as high as ten. In practice it is possible to obtain accurate eigendecompositions for matrices of order greater than 100. This is in contrast with computing roots of high-degree univariate polynomials (which may be an ill-conditioned problem) 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 seemingly 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 Sederberg [1983]. The control points of two Bézier curves along with the weights (as shown in Figure 3) 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,

$$\begin{aligned}\mathbf{P}(t) = (x(t), y(t), w(t)) = (&4(1-t)^3 + 30(1-t)^2 t + 30(1-t)t^2 + 6t^3,\\ &(1-t)^3 + 36(1-t)^2 t + 4t^3, (1-t)^3\\ &+6(1-t)^2 t + 6(1-t)t^2 + t^3).\end{aligned}$$

The implicit representation has a matrix determinant formulation given as

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

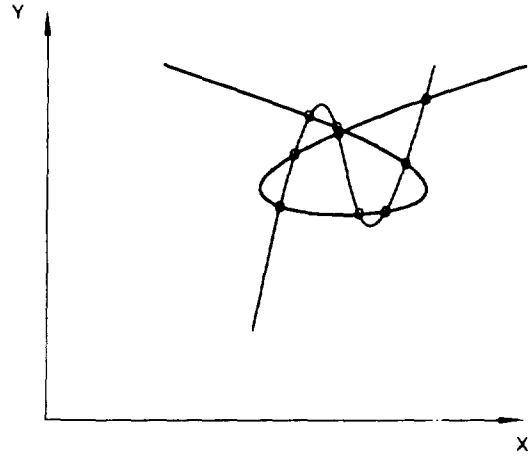

Fig. 3. Intersection of rational cubic Bézier curves.

We substitute the homogeneous representation of the second parameterization,

$$\mathbf{Q}(u) = (\bar{x}(u), \bar{y}(u), \bar{w}(u)) = (7(1-u)^3 + 6(1-u)^2u + 54(1-u)u^2 + 3u^3, 4(1-u)^3 + 12(1-u)^2u + 12(1-u)u^2 + 4u^3, (1-u)^3 + 6(1-u)^2u + 6(1-u)u^2 + u^3)$$

into the implicit representation $\mathbf{M}$, divide the resulting matrix $\mathbf{M}(u)$ by $(1-u)^3$, and reparameterize by $s = u/(1-u)$. The resulting matrix polynomial has the form

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

The exact condition number of the leading matrix is 9.525. LINPACK's estimator returns the approximate value as 7.0621.[3]

[3] Using double-precision arithmetic we have reduced matrix polynomials with leading matrices of condition numbers up to 100,000 to eigenvalue problems.

Multiplying $\mathbf{M}(s)$ with the inverse of the leading matrix and reducing it to an eigenvalue problem (using Theorem I) results in the matrix

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

The eigendecomposition of $\mathbf{C}$ results in 9 points of intersection. The intersections points are computed using the relationship highlighted in (16). They are:

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,
—t_0 is the parameter in the domain of the 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 on 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

$$E_i = \epsilon \|\mathbf{C}\| \text{cond}_i, \tag{17}$$

where $\epsilon = 2.2204 * 10^{-16}$ 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 compute 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 accounts 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. Let us again consider the intersection of algebraic plane curves highlighted in Example 2.2. 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:

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

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

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

The entries of the matrix are polynomials of degree 6. However, the determinant of $\mathbf{M}(x)$ is a polynomial of degree 12 since the two curves intersect in 12 points, according to Bezout's 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 since we are only interested in 12 eigenvalues.

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

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

The leading matrix $\mathbf{M}_0$ is nonsingular, 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 $\text{Determinant}(\overline{\mathbf{M}}(z)) = 0$ correspond to $z = 0$. Moreover the condition number of $\mathbf{M}_0$ is 6.5552325. As a result, we 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 $\mathbf{M}$ are used in choosing the shifts in the QR algorithm, as highlighted in (10). After choosing a sufficient number of shifts corresponding to $s = 0$ (at most 12), the problem reduces to that of computing the eigendecomposition of a 12×12 Hessenberg matrix.

$\mathbf{M}$ has only two nonzero 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 is 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 $\mathbf{P}(t) = (x(t), w(t))$ and $\mathbf{Q}(u) = (\bar{x}(u), \bar{y}(u), \bar{w}(u))$ we formulate the equations

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

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

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 most 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. CONCLUSION

In this article 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 reduced to an eigenvalue problem. The algorithm is very robust and can accurately compute the intersection points. The numerical accuracy of the operations is well understood, and as a result we are able to come up with tight bounds on the errors in the computation. Good implementations of matrix computations are available as part of standard library packages like EISPACK and LAPACK, and we used them in our implementation. A similar approach is directly applicable to curve–surface intersections and ray-tracing parametric and algebraic surfaces.

ACKNOWLEDGMENT

We are grateful to the reviewers for their comments.

REFERENCES

ANDERSON, E., BAI, Z., BISCHOF, C., DEMMEL, J., DONGARRA, J., DU CROZ, J., GREENBAUM, A., HAMMARLING, S., AND SORENSEN, D. 1992. *LAPACK User's Guide, Release 1.0.* SIAM, Philadelphia, Pa.

BARTELS, R. H., BEATTY, J. C., AND BARSKY, B. A. 1987. *An Introduction to Splines for Use in Computer Graphics and Geometric Modeling.* Morgan Kaufmann, San Mateo, Calif.

BAI, Z. AND DEMMEL, J. 1993. Design of a parallel nonsymmetric eigenroutine toolbox, Part I. In *Proceedings of the 6th SIAM Conference on Parallel Processing for Scientific Computing.* SIAM, Philadelphia, Pa. Long version available as UC Berkeley Computer Science report UCB/CSD/92-718, 1992.

BAI, Z., DEMMEL, J., AND MCKENNEY, A. 1993. On computing condition numbers for the nonsymmetric eigenproblem. *ACM Trans. Math. Softw. 19*, 3, 202–223.

COLLINS, G. E. AND KRANDICK, W. 1992. An efficient algorithm for infallible polynomial complex root isolation. In *Proceedings of the International Symposium on Symbolic and Algebraic Computation.* ACM, New York, 189–194.

DEMMEL, J. 1989. LAPACK: A portable linear algebra library for supercomputers. In *Proceedings of the 1989 IEEE Control Systems Society Workshop on Computer-Aided Control System Design.* IEEE, New York.

ELBER, G. AND COHEN, E. 1990. Hidden curve removal for free form surfaces. *Comput. Graph. 24*, 4, 95–104.

FAROUKI, R. T. AND RAJAN, V. T. 1987. On the numerical condition of polynomials in bernstein form. *Comput. Aided Geo. Des., 4*, 3, 191–216.

GARBOW, B. S., BOYLE, J. M., DONGARRA, J., AND MOLER, C. B. 1977. *Matrix Eigensystem Routines – EISPACK Guide Extension.* Lecture Notes in Computer Science, vol. 51. Springer-Verlag, Berlin.

GOLUB, G. H. AND VAN LOAN, C. F. 1989. *Matrix Computations.* John Hopkins Press, Baltimore, Md.

GOHBERG, I., LANCASTER, P. AND RODMAN, L. 1982. *Matrix Polynomials.* Academic Press, New York.

GOLDMAN, R., SEDERBERG, T. W., AND ANDERSON, D. C. 1984. Vector elimination: A technique for the implicitization, inversion and intersection of planar parametric rational polynomial curves. *Comput. Aided Geo. Des. 1*, 4, 327–356.

HOBBY, J. D. 1991. Numerically stable implicitization of cubic curves. *ACM Trans. Graph. 10*, 3, 255–296.

HOFFMANN, C. M. 1989. *Geometric and Solid Modeling.* Morgan Kaufmann, San Mateo, Calif.

KOPARKAR, P. A. AND MUDUR, S. P. 1983. A new class of algorithms for the processing of parametric curves. *Comput. Aided Des. 15*, 1, 41–45.

LANE, J. M. AND RIESENFELD, R. F. 1980. A theoretical development for the computer generation and display of piecewise polynomial surfaces. *IEEE Trans. Patt. Anal. Mach. Intell. 2*, 1, 150–159.

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

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

MANOCHA, D. 1992. Algebraic and numeric techniques for modeling and robotics. Ph.D. thesis, Computer Science Div., Dept. of Electrical Engineering and Computer Science, Univ. of California, Berkeley, Calif.

MANOCHA, D. AND CANNY, J. F. 1992. Detecting cusps and inflection points in curves. *Computer Aided Geometric Design*, 9:1–24.

MANOCHA, D. AND DEMMEL, J. 1992. Algorithms for intersecting parametric and algebraic curves. Tech. Rep. UCB/CSD 92/698, Computer Science Div., Univ. of California, Berkeley, Calif.

MARKOT, R. P. AND MAGEDSON, R. L. 1989. Solutions of tangential surface and curve intersections. *Comput. Aided Des. 21*, 7, 421–427.

SALMON, G. 1885. *Lessons Introductory to the Modern Higher Algebra.* G. E. Stechert and Co., New York.

SEDERBERG, T. W. 1989. Algorithms for algebraic curve intersection. *Comput. Aided Des. 21*, 9, 547–555.

SEDERBERG, T. W. 1986. Improperly parametrized rational curves. *Comput. Aided Geo. Des. 3*, 1, 67–75.

SEDERBERG, T. W. 1983. Implicit and parametric curves and surfaces. Ph.D. thesis, Purdue Univ., West Lafayette, Ind.

SEDERBERG, T. W. AND PARRY, S. R. 1986. Comparison of three curve intersection algorithms. *Comput. Aided Des. 18*, 1, 58–63.

SEDERBERG, T. W., WHITE, S., AND ZUNDEL, A. 1989. Fat arcs: A bounding region with cubic convergence. *Comput. Aided Geo. Des. 6*, 3, 205–218.

STURMFELS, B. 1991. Sparse elimination theory. In D. Eisenbud and L. Robbiano, editors, *Computational Algebraic Geometry and Commutative Algebra*. Cambridge University Press.

WALKER, R. J. 1950. *Algebraic Curves*. Princeton University Press, New Jersey.

WILKINSON, J. H. 1965. *The Algebraic Eigenvalue Problem*. Oxford University Press, Oxford, U.K.

WILKINSON, J. H. 1959. The evaluation of the zeros of ill-conditioned polynomials. Parts i and ii. *Num. Math. 1*, 150–166 and 167–180.

Received August 1992; revised August 1993; accepted September 1993