Curve Fitting with Conic Splines

THEO PAVLIDIS
Bell Laboratories

Conic splines are formed by arcs of conics, each defined by its endpoints and the tangents at them plus an intermediate point. Instead of the common general equation that depends on five parameters, an equation with a single parameter is used, thus simplifying significantly the curve fitting problem. The resulting *guided* conics resemble Bezier polynomials and for parabolas are identical to them. Such splines can be used conveniently both for interactive design and for automatic curve fitting. They allow circular, elliptical, and hyperbolic arcs to be included in the spline family, while the common forms using a B-spline basis allow the inclusion of parabolic arcs only. Conic splines are described either in a rational parametric or in algebraic form $f(x, y) = 0$. A simple estimate for the distance of a point from such a curve is given and is used to test the quality of approximations. The data to be fitted are first approximated by a polygon, and then simple heuristics are used to decide which sequences of vertices should be approximated by conics. The conics found by the applications of the heuristics are usually close approximations of the data and need no further adjustments. When adjustments are needed, the interval is split and a conic is fitted on each part. It is shown theoretically that exact knot placement at the optimal locations is less important for higher order splines than for polygons. Examples of application of the method to the fitting of font and other contours are given. Comparisons with other methods suggest that conic splines require no more knots than cubic splines for similar quality of approximation.

Categories and Subject Descriptors: G.1.2 [**Numerical Analysis**]: Approximation; I.3.6 [**Computer Graphics**]: Methodology and Techniques; I.5.4 [**Pattern Recognition**]: Applications

General Terms: Algorithms

Additional Key Words and Phrases: Approximation by splines, Bezier polynomials, variable knot splines, font description, interactive graphics, optical character recognition

1. INTRODUCTION

The representation of curves by splines has found applications in graphics, computer-aided design and manufacturing, and numerous other fields ([1, 5, 7, 19], etc.). Originally, splines were given in the $y = f(x)$ form, but this has certain disadvantages when one deals with arbitrary curves on the plane that may have "infinite" slope [6]. Therefore the parametric form of representation is commonly used. The spline in the plane is defined through two scalar splines, $x(t)$ and $y(t)$, that have the same values of t at the knots. For example, a spline of degree n with

Author's address: Theo Pavlidis, Bell Laboratories (2C-456), Murray Hill, NJ 07974.

m knots may be given by the following equation:

$$\mathbf{P}(t) = \sum_{i=0}^{n} \mathbf{a}_{ji}(t - t_j)^i \qquad \text{for} \quad t_j \le t \le t_{j+1} \quad 0 \le j \le m - 1. \tag{1.1}$$

$\mathbf{P}(t)$ is a vector with elements $x(t)$ and $y(t)$, $\mathbf{a}_{ji}$ is the coefficient vector, t is the parameter, and the t_j are the values of the parameter at the knots. If we use B-splines for a basis, then the parametric equation takes the form

$$\mathbf{P}(t) = \sum_{i=0}^{m} \mathbf{P}_i N_{i,n}(t), \tag{1.2}$$

where $N_{i,n}(t)$ is the ith B-spline of degree n [19, pp. 252–267]. This form, originally proposed by Riesenfeld [26], has been particularly popular in graphics applications and has an appealing intuitive interpretation. The coefficient vectors $\mathbf{P}_i$ may be interpreted as vertices of a *guiding* (or *control*) *polygon* and may be used in interactive graphics to control the shape of a curve. Thus they exhibit the major advantage of *Bezier polynomials* and are also easier to manipulate.

In spite of its popularity, the above representation has certain disadvantages. In particular, some commonly used curves, such as conics, are excluded by the formalism of eqs. (1.1) or (1.2). Conics are widely used in CAD applications [3, 7, 8], and it seems that in many cases piecewise conic curves give results at least as good as those given by cubic splines. The popularity of cubic splines is due partly to historical reasons and partly to the following property: frequently, we want local approximations where the curve between two knots is specified by the location of the points and their tangents. Thus we need four degrees of freedom, and if we use the formalism $y = f(x)$, or that of eq. (1.1), we must go to cubics ($n = 3$). Conics, however, have five degrees of freedom and therefore are adequate for such local fitting. (Parabolas expressed in this way have four degrees of freedom.) There are three more reasons for preferring conics: (a) conics are used to generate certain kinds of data (sometimes through "French curves"), so that our curve fitting matches the generating process; (b) conics have been studied for over two thousand years,[1] and there is wealth of mathematical results about them; and (c) it is much easier to find the intersection of a line (especially one parallel to one of the axes) with a conic than with a cubic. The solution of this problem is required in many graphics applications, and its difficulty for cubics has led to the development of the recursive subdivision algorithms [4]. Cubics have a clear advantage over conics only in the following cases: (a) when continuity of curvature (or second derivative) is important, and (b) when interpolation without tangent specification is required. Then one has a system of equations like (1.2) that must be solved in order to find $\mathbf{P}_i$. This system is well behaved numerically for cubics ($n = 3$) [31] but not for quadratics ($n = 2$).

While conics cannot be described through polynomials of the form of eq. (1.2), they can be expressed through the following rational parametric form, often used in CAD applications [3], [7, pp. 138–144], [8, pp. 21–25]:

$$x(t) = \frac{x_0 t^2 + x_1 t + x_2}{w_0 t^2 + w_1 t + w_2}, \tag{1.3a}$$

[1] Although there were long interruptions because research funding was suspended.

$$y(t) = \frac{y_0 t^2 + y_1 t + y_2}{w_0 t^2 + w_1 t + w_2}. \tag{1.3b}$$

They may also be described by the algebraic form

$$f(x, y) = 0. \tag{1.4}$$

The main result of the paper is a method for finding a mixture of conic arcs and straight-line segments approximating a given set of data points. This is described in Sections 6–8. This method is independent of the form used to describe conics and could be used with either of the two forms above. However, all parametric forms suffer from the following limitation: when splines are used to approximate a set of data points, one must establish a correspondence between the data points and the parameter t in order to compute the error. This problem does, however, disappear when the algebraic form is used. We present in Section 5 expressions for estimating the distance of a point from a conic given in algebraic form. It would seem that it would be difficult to find points of a conic given by eq. (1.4), but this is not the case. We present a solution in Section 4.

We have not attempted to design an optimal approximation algorithm for two reasons. One is that it can be shown theoretically that approximations by splines with tangent continuity at the knots have the property that error norm at suboptimal solutions is close to the error norm at the optimal ones (see Appendix B). A more important reason is that it is very difficult (if not impossible) to devise mathematical criteria for approximation that agree with the human perception of high-quality approximation. For example, the preservation of various symmetries or "lining-up" relations is very important for aesthetic reasons but very difficult to incorporate in an optimization algorithm. Therefore it is not advisable to make a great effort to find the optimal approximation under, say, an integral square error norm when this is not exactly what we want. For applications where high-quality approximations are essential, it is necessary to include postediting of the results by a human observer. For applications where low-quality approximations suffice a suboptimal mathematical approximation could be acceptable.

2. DEFINITION OF CONIC SPLINES

We define conic splines by an extension of the definition of guided splines.

Definition 2.1. The following sets of points and parameters define a conic spline: (a) the vertices of a guiding polygon $\mathbf{V}_i$, $0 \le i \le k$; (b) coefficients p_i that are between zero and one and specify the location of the knot on the line segment $\mathbf{V}_i\mathbf{V}_{i+1}$, $1 \le i < k$; and (c) coefficients q_i, between zero and one, denoting the location of a point of the curve along the line joining $\mathbf{V}_{i+1}$ with the middle of the line joining the knots before and after it, $1 \le i \le k - 1$.

The knots $\mathbf{K}_i$ are then defined as

$$\mathbf{K}_i = (1 - p_i)\mathbf{V}_i + p_i\mathbf{V}_{i+1}, \tag{2.1}$$

and together with the sides of the guiding polygon, they specify the endpoints

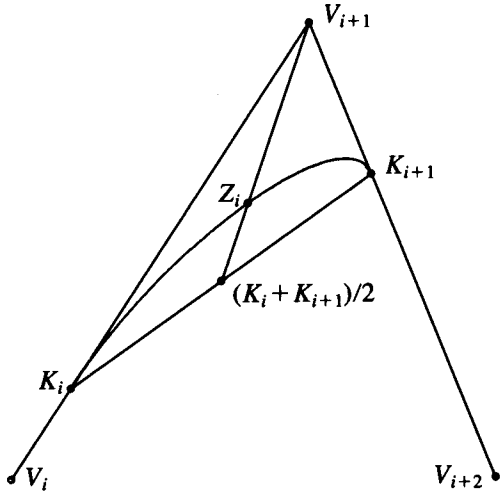

Fig. 1. Specification of a conic arc according to Definition 2.1.

and their tangents for each conic. The conic is specified completely by a fifth point:

$$\mathbf{Z}_i = q_i \mathbf{V}_{i+1} + (1 - q_i) \frac{\mathbf{K}_i + \mathbf{K}_{i+1}}{2}. \tag{2.2}$$

We have five conditions, therefore, the conic is well defined, unless some of the guiding points are collinear (in which case the conic is reduced to linear segments). Figure 1 shows an example of a well defined conic. It is well known that if q_i equals 0.5, then the conic is a parabola [8, pp. 30–33]. In that case it is also easy to prove that if all the coefficients p_i equal 0.5, then the conic spline is the same as a second-degree spline given by eq. (1.2) with uniform knot distribution. [19, chap. 11].

Next we examine various singular cases. If p_{i+1} is zero, then the ith arc of the spline is the straight-line segment $\mathbf{K}_i\mathbf{V}_{i+1}$. In order to have a straight-line segment with continuity of the tangents at both endpoints, we must select three guiding points along the same line. If $\mathbf{V}_i$, $\mathbf{V}_{i+1}$, and $\mathbf{V}_{i+2}$ are collinear, then $\mathbf{K}_i$ and $\mathbf{K}_{i+1}$ are also on the same line, as is $\mathbf{Z}_i$, regardless of the value of q_i. If p_i equals one and p_{i+1} equals zero, then both $\mathbf{K}_i$ and $\mathbf{K}_{i+1}$ equal $\mathbf{V}_{i+1}$. Also, $\mathbf{Z}_i$ equals $\mathbf{V}_{i+1}$, regardless of the value of q_i. The ith arc then reduces to a single point and the spline exhibits a *corner*. This is a case of tangent discontinuity produced by knot coalescence [5, 25]. There is no way to introduce a discontinuity in the curve by further knot coalescence unless we allow values of p_i other than zero and one. If $\mathbf{V}_i$ coincides with $\mathbf{V}_{i+2}$, but $\mathbf{V}_{i+1}$ is different, then the spline exhibits a *cusp*. If q_i is zero or one, then the conic arc reduces to a polygonal arc regardless of the value of p_i or p_{i+1}.

The following example illustrates one of the advantages of the new formalism.

Example 2.1. We wish to draw a curve of a desired shape in an interactive graphics system. To this end we specify first the guiding polygon (Figure 2). If we were to use the form of eq. (1.2), the points where the curve intersects the sides

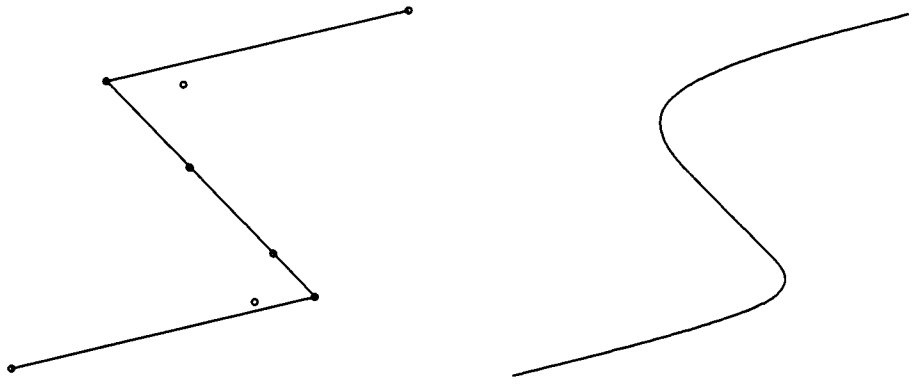

Fig. 2. Specification of a guiding polygon, points through which the curve must pass, and the resulting curve.

of the polygon would not be under our direct control because they are specified by the values of the parameter t at the knots. (Also if we were to use cubics, we would not have direct control over the location of the curve or its tangents.) On the other hand, we may specify these points interactively and then use Definition 2.1 to draw the spline. We have the option of specifying the fifth point if we wish to control how close the curve passes to a vertex. In an interactive system there is no need to specify parameters p_i and q_i explicitly. They can be replaced by a specification of the knots and any fifth point.

3. EQUATIONS THAT DESCRIBE CONICS

The customary general equation for a conic is

$$f(x, y) = ax^2 + 2hxy + by^2 + 2ex + 2gy + c = 0. \tag{3.1}$$

Although there are six coefficients in eq. (3.1), one of them can be chosen arbitrarily, so that a conic has only five degrees of freedom. We assume without loss of generality that *the sign of the first nonzero coefficient in eq.* (3.1) *is positive.* It is well known that the conic is an *ellipse* if the quantity $ab - h^2$ is positive, a *hyperbola* if $ab - h^2$ is negative, and a *parabola* if $ab - h^2$ is zero. The conic is a *circle* if $a = b$ and $h = 0$. The conic degenerates into a pair of straight lines if $f(x, y)$ can be expressed as the product of two first-degree polynomials. In spite of its generality this form is not convenient for many applications or for proving properties about conics. We next describe some alternative representations.

3.1 Matrix Equation for Conics

We use the term *coordinate vector* to refer to the column vector $\mathbf{x}$ with components x and y. If we define $\mathbf{Q}$ to be the matrix

$$\mathbf{Q} = \begin{bmatrix} a & h \\ h & b \end{bmatrix}, \tag{3.2}$$

and $\mathbf{g}$ the column vector with components $2e$ and $2g$, then eq. (3.1) becomes

$$f(\mathbf{x}) = \mathbf{x}'\mathbf{Q}\mathbf{x} + \mathbf{g}'\mathbf{x} + c = 0. \tag{3.3}$$

The primed symbol of a vector or a matrix is used to denote the transpose. Note that $ab - h^2$ is the determinant of $\mathbf{Q}$.

The *center* of a conic is defined as the point with coordinate vector

$$\mathbf{x}_c = -\frac{1}{2}\mathbf{Q}^{-1}\mathbf{g}, \tag{3.4}$$

or in scalar form

$$x_c = \frac{-eb + gh}{ab - h^2}, \qquad y_c = \frac{-ag + eh}{ab - h^2}. \tag{3.5}$$

Note that the center is not defined for parabolas. For ellipses and hyperbolas we may move the origin of the coordinates to the center and obtain a simpler form of eq. (3.3)

$$f(\mathbf{x}) = \mathbf{x}'\mathbf{Q}\mathbf{x} + c_1 = 0, \tag{3.6}$$

where the constant c_1 is given by

$$c_1 = c - \frac{1}{4}\mathbf{g}'\mathbf{Q}^{-1}\mathbf{g}. \tag{3.7}$$

The *axes* of a conic are parallel to the two eigenvectors of $\mathbf{Q}$, and if $\lambda_{\max}$ denotes the larger eigenvalue of $\mathbf{Q}$ and $\lambda_{\min}$ the smaller, we find from eq. (3.6) that the norms of the two axes are, respectively,

$$\|\mathbf{a}_{\max}\| = \sqrt{-\frac{c_1}{\lambda_{\min}}} \quad \text{and} \quad \|\mathbf{a}_{\min}\| = \sqrt{-\frac{c_1}{\lambda_{\max}}}. \tag{3.8}$$

Note that for a circle $\mathbf{Q}$ is proportional to the identity matrix; thus any vector through the center is an axis. A straightforward calculation shows that the eigenvalues are given by

$$\lambda_{\max,\min} = \tfrac{1}{2}[a + b \pm \sqrt{(a - b)^2 + 4h^2}].$$

If h is zero, then the two eigenvalues equal a and b respectively.

3.2 Guided Form for Conics

A form that is useful for conic splines is the *guided form* whereby the conic is expressed through the equations of certain lines. We recall that the general form for the equation of a straight line is

$$a_x x + a_y y + a_0 = 0. \tag{3.9a}$$

If $\mathbf{a}$ is the two-dimensional column vector of the coefficients, then the above equation can be written in the more concise form

$$\mathbf{a}'\mathbf{x} + a_0 = 0, \tag{3.9b}$$

where $\mathbf{a}'$ denotes the transpose of the vector $\mathbf{a}$. It can be shown that replacing the coordinates of any point into eq. (3.9) yields the distance of the point from the line times a constant. If a straight line is specified by the coordinates of its two

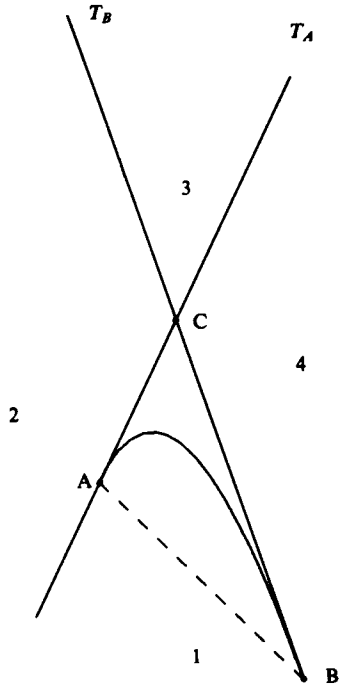

Fig. 3. Illustration of the definition of a set of conics by two points and the tangents at them. A single conic is specified by selecting a fifth point E that can be in any of the sectors 1, 2, 3, and 4.

endpoints x_1, y_1 and x_2, y_2, then the above equation becomes

$$x(y_1 - y_2) + y(x_2 - x_1) + y_2x_1 - y_1x_2 = 0. \tag{3.10}$$

Furthermore, in that case the distance of a point (u, v) from that line equals $|d/D|$ where

$$d = u(y_1 - y_2) + v(x_2 - x_1) + y_2x_1 - y_1x_2, \tag{3.11}$$

and

$$D = \sqrt{(y_1 - y_2)^2 + (x_2 - x_1)^2}. \tag{3.12}$$

The sign of d indicates the side of the line on which the point lies.

Using these expressions, we can write the equation for a guided conic. Suppose that we are given two points A and B and the tangents of the conic at points T_A and T_B. Since conics have five degrees of freedom, these four conditions describe a one-parameter family of conics. Let $\mathbf{a}'\mathbf{x} + a_0 = 0$ be the equation of line T_A, $\mathbf{b}'\mathbf{x} + b_0 = 0$ the equation of T_B, and $\mathbf{u}'\mathbf{x} + u_0 = 0$ the equation of chord AB. Let K be the parameter of the family of conics defined in this way. Then their equation is

$$(\mathbf{a}'\mathbf{x} + a_0) \cdot (\mathbf{b}'\mathbf{x} + b_0) = K \cdot (\mathbf{u}'\mathbf{x} + u_0)^2. \tag{3.13}$$

See [28, pp. 234–235] and [29, p. 72] for a proof. If we denote by C the intersection of the two tangents (Figure 3), we have the following geometrical interpretation of eq. (3.13).

PROPOSITION 3.1. *If A, B, and C are three points, then for each number k there exists a conic passing through A and B and tangent to AC and CB with the property that for each one of its points P, the distance of P from AB times k is the geometric mean of the distances of P from lines AC and CB.*

We have used k rather than K in the statement of Proposition 3.1 because the constant includes scale factors from the equations of the lines.

For a given set of tangents and endpoints, the selection of a fifth point E, through which the curve must pass, determines K. It is possible to establish a relation between the form of the conic and the part of the plane where the point is chosen. First we observe that since two points and their tangents uniquely define a parabola, then we should be able to specify a value of K so that eq. (3.13) represents a parabola without having to select another point. A rather lengthy but straightforward calculation yields the following value of K

$$K = -\frac{1}{4}\frac{(a_x b_y - b_x a_y)^2}{u_x u_y(a_x b_y + b_x a_y) - u_x^2 a_y b_y - u_y^2 a_x b_x}. \tag{3.14}$$

We can define the three lines of eq. (3.13) in terms of eq. (3.10) by using the following ordered pairs of points: A and C, B and C, and A and B. Then the big fraction of eq. (3.14) reduces to one and the value of K corresponding to the parabola is $-1/4$. With this specification for the lines we derive the following results.

PROPOSITION 3.2. *If a conic is defined by a triangle ABC and a fourth point E, and the equations of the lines in eq.* (3.13) *are like those in* (3.10), *then the following statements are true. K is negative whenever point E is selected in the plane sector containing chord AB. This sector is marked by a* 1 *in Figure* 3. *In this case all three points E, A, and B belong to the same conic arc. The parabola divides that sector into two parts. The conic will be an ellipse only when E is in sector* 1 *below the parabola. In that case K will be less then* $-1/4$. *In all other cases the conic will either be a hyperbola or will degenerate into pairs of lines. If E is in sector* 3, *then K is negative, but E belongs to a different branch of the hyperbola than A and B. K is positive if E is located in either sector* 2 *or* 4. *Then A and B lie on different hyperbolic branches. K can be zero only if we select E on one of the tangents and the conic degenerates into a pair of lines. Also if E is selected on AB, then K is infinite and the conic degenerates to chord AB.*

It is simple to prove that if the conic is a parabola, then eq. (3.13) is the same as the Bezier equation [19, pp. 221-223]. There is another convenient form for parabolas that is related to the guided form. Let $\mathbf{c}'\mathbf{x} + c_0 = 0$ be the equation of a line through A and parallel to the line joining C with the midpoint M of chord AB (see Figure 4). Let $\mathbf{x}_2$ be the coordinate vector of B. Then the equation of the parabola is

$$(\mathbf{c}'\mathbf{x} + c_0)^2 + (\mathbf{a}'\mathbf{x} + a_0)\cdot(\mathbf{a}'\mathbf{x}_2 + 1) = 0 \tag{3.15}$$

[28, p. 195]. We can show that the parabola passes through the midpoint F of segment CM and that at F it has a tangent parallel to chord AB. If D and E are

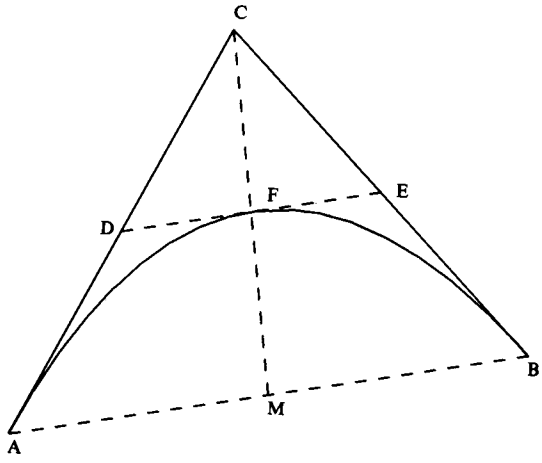

Fig. 4. Illustration of the properties of a parabola defined by two points, A, and B, and the tangents of the curve at them. C is the intersection of the two tangents, M the midpoint of chord AB, and F the midpoint of segment CM.

the points where this tangent intersects AC and BC, respectively, then the parabola is subdivided into two arcs, AF and FB, and each arc is specified again by its endpoints and tangents. This is the same as the recursive Bezier subdivision [19, pp. 227–230].

3.2 Guided Rational Parametric Form for Conics

It can be shown [7, pp. 138–144] that the rational parametrtic form of eq. (1.3) can be used to describe a guided conic by the following equation:

$$\mathbf{x}(t) = \frac{w_0(1-t)^2\mathbf{x}_A + 2w_1t(1-t)\mathbf{x}_C + w_2t^2\mathbf{x}_B}{w_0(1-t)^2 + 2w_1t(1-t) + w_2t^2}, \tag{3.16}$$

where $\mathbf{x}_A$, $\mathbf{x}_B$, and $\mathbf{x}_C$ are the coordinate vectors of points A, B, and C of Figure 3, and w_0, w_1, and w_2 are the parameters whose selection specifies each particular conic belonging to the family. As a matter of fact it is the value of

$$\frac{w_0w_2}{w_1^2} \tag{3.17}$$

that is important—the conic does not change if we vary the w_i's while keeping the above ratio fixed. It can be shown [8, pp. 25–27] that this ratio equals q_i as defined in eq. (2.2). It is straightforward to verify that $\mathbf{x}(0) = \mathbf{x}_A$, $\mathbf{x}(1) = \mathbf{x}_B$, $\dot{\mathbf{x}}(0) = 2w_1/w_0(\mathbf{x}_C - \mathbf{x}_A)$, and $\dot{\mathbf{x}}(1) = 2w_1/w_2(\mathbf{x}_B - \mathbf{x}_C)$. If we denote by t_E the value of t at point E with coordinate vector $\mathbf{x}_E$, then eq. (3.16) yields

$$(1-t_E)^2w_0(\mathbf{x}_A - \mathbf{x}_E) + 2w_1t_E(1-t_E)(\mathbf{x}_C - \mathbf{x}_E) + t_E^2w_2(\mathbf{x}_A - \mathbf{x}_E) = 0. \tag{3.18}$$

In other words the w_i's and t_E can be found from the projections of vector EC on vectors AC and AB. (Note that two of the four unknowns can be chosen arbitrarily.) While these computations may involve slightly more work than finding K from eq. (3.13), we should point out that additional work is required to find the coefficients of the conic from K, while only the solution of eq. (3.18) is required for the rational parametric form.

4. PLOTTING A CONIC

If we use the parametric form (1.1), then it is simple to find a sequence of points on the curve for plotting or for any other purpose. This is also the case with the parametric form of eq. (1.3), except that we require two additional divisions for each point to be plotted. Things are somewhat more complicated for nonparametric forms. The following result provides a mechanism for finding such points under the algebraic representation, provided that we already know one point on the curve.

4.1 Incremental Plotting

PROPOSITION 4.1. *If an ellipse or hyperbola is given by eq.* (3.6), *then there exist matrices* $\mathbf{B}$ *such that if* $\mathbf{x}$ *is a point on the conic, then* $\mathbf{Bx}$ *is also a point on the same conic. All such matrices are of the form*

$$\mathbf{B} = \begin{bmatrix} \rho + h\tau & b\tau \\ -a\tau & \rho - h\tau \end{bmatrix}, \tag{4.1}$$

with ρ *and* τ *satisfying the equation*

$$\rho^2 + (ab - h^2)\tau^2 = 1. \tag{4.2}$$

PROOF. Straightforward by showing by direct substitution that any matrtix $\mathbf{B}$ given by eqs. (4.1) and (4.2) satisfies also the following equation

$$\mathbf{B}'\mathbf{Q}\mathbf{B} = \mathbf{Q}. \tag{4.3}$$

This implies that $\mathbf{Bx}$ will also be on the conic because of eq. (3.6). It is also possible to show that this construction is equivalent to the well known geometric construction of an ellipse by means of the *concentric circle* method [27, pp. 133–134]. □

A matrix $\mathbf{B}$ may be used to plot ellipses and hyperbolas in a numerically stable way. For this purpose τ is taken to be small and ρ near one. For ellipses one could define an angle ϕ such that

$$\rho = \cos\phi \quad \tau = \frac{\sin\phi}{\sqrt{ab - h^2}}, \tag{4.4a}$$

while for hyperbolas we can select a variable χ such that

$$\rho = \cosh\chi \quad \tau = \frac{\sinh\chi}{\sqrt{h^2 - ab}}. \tag{4.4b}$$

Normalization of the coefficients of eq (3.1) so that

$$ab - h^2 = 1 \tag{4.5}$$

simplifies subsequent calculations. The only difficulty occurs when the original value of $ab - h^2$ is near zero. From a practical viewpoint it is best in such cases to treat the conic as a parabola. For the applications discussed in this paper this is quite acceptable since we are approximating rather than interpolating curves.

At first look the use of a matrix multiplication per point seems an unduly complicated method for plotting conics. However, it is competitive with the rational parametric method. The operation **Bx** requires 4 scalar multiplications and 2 additions plus 2 more additions if the center is not the origin of coordinates. Equation (1.3) requires two divisions plus 6 additions, if the divided differences method is used for calculating the quadratic polynomials.[2] In contrast, some books on computer graphics contain algorithms for conic plotting that require as many as 20 operations per point (10 multiplications and 10 additions). If $h = 0$, (i.e., if the conic axes are parallel to the coordinate axes), and if the known point of the conic is at the tip of an axis, then **Bx** is reduced to the simple expression

$$X \cos \phi \qquad Y \sin \phi$$

for an ellipse with axes X and Y.

A matrix **B** may not be used to plot parabolas (eq. (3.6) does not hold for them), but parabolas can be plotted easily as polynomial splines. For the sake of generality we give in Appendix A a result that can be used for the incremental plotting of parabolas.

4.2 Recursive Plotting

All the plotting techniques described above assume that the coefficients of the equation describing the conic have been found. It is possible to plot a conic without having that information, at the expense of more computation per point. Such constructions are based on the following important theorem for conics.

PASCAL'S THEOREM. *If a hexagon is inscribed in a conic, then the three pairs of opposite sides have collinear intersections.*

See [28, pp. 245–246] or [29, pp. 83–85] for a proof of this theorem. Thus, given five points, we may select a line on which the three intersections must lie, and then construct the sixth point. See [28, pp. 247–248] for details. If we let two pairs of points on the hexagon coincide, then two of the sides become tangents and we have the situation shown in Figure 5. There, A and B are double points and the six lines of the hexagon are paired: AC with EB, AD with BC, and DE chord AB. This is the configuration of practical interest for guided conics.

If only D is given, then we can construct E by drawing an arbitrary line through P_2, finding points P_1 and P_3 as intersections on that line with AC and AB, respectively, and then finding the intersection of lines AP_2 and DP_3. Forrest [9] gives another algorithm based on the same principle. Since these techniques require finding the intersection of three pairs of lines for each point on the conic, they are clearly inferior to incremental plotting or parametric plotting. They become competitive only when one needs very few points and wants to avoid the overhead of finding the coefficients of the conic or of solving eq. (3.18).

---

[2] The two methods will be equally expensive if we assume a machine where a multiplication is twice and division three times as expensive as addition, or if we assume that the three operations have equal costs. These assumptions are reasonable for modern computers where the register-to-register arithmetic operation time is comparable to the time of the memory to register fetch. (E.g., pp. B-1 to B-28 of PDP11/04/34/45/55 Processor Handbook, Digital Equipment Corporation, Maynard, Mass., 1976.)

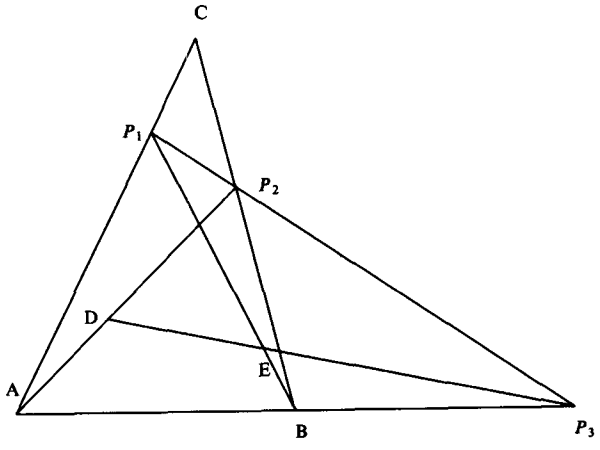

Fig. 5. Illustration of Pascal's Theorem. The endpoints of the conic arc are A and B, and C is the intersection of the tangents. $P_1P_2P_3$ is the Pascal line, while D and E are two interior vertices. (Note that A and B are double points and that the six lines of the hexagon are paired as AC with EB, AD with BC, and DE with chord AB.)

If we wish to find the tangent at a given point, then we repeat the above construction by having D and E coincide: line DP_3 will be the tangent. (In that case we would not have chosen an arbitrary line through P_2, but would have found P_1 as the intersection of AC and BD.) These constructions provide the elements for a recursive algorithm. Given triangle ABC and a point on the conic D, we find the tangent at D and let A_1 be the point where it intersects AC, and B_1 the point where it intersects CB. In addition, we find two more points, D_1 and D_2. Now we have two arcs, one defined by triangle AA_1D and point D_1, and another defined by triangle DB_1B and point D_2. Since the same set of operations may be applied to each of these two arcs, we have a recursive method. It is easy to show that at each step the angle between the tangents increases; therefore the difference between the parabola defined by the triangle and all other possible conics keeps decreasing. Thus at some point we can stop the subdivision and instead complete the plotting by *parabolic arcs*. Furthermore we do not need to find the points D_1 and D_2 for the last recursion. If the number of recursions is small compared to the number of points to be plotted, then the recursive method becomes competitive since a parabolic arc can be drawn incrementally with only four operations per point, and part of the cost of the recursion can be written off against the overhead of finding the conic coefficients and initialization.

5. DISTANCE OF A POINT FROM A CONIC

While for a straight line with an equation of the form $f(x, y) = 0$, the quantity $f(x, y)$ is proportional to the distance of a point from the line, the same is not true for conics. We present here some properties of conics that are useful for estimating such a distance. Some of these results are "new" in the sense that we could not find a reference describing them, but given the vast literature on the subject of conics it is doubtful that they have not appeared in print before. Anyway, all the proofs are straightforward applications of analytic geometry and linear algebra. Let $\mathbf{x}_0$ be a point on the conic, $\mathbf{x}$ a point not on the conic, and $\mathbf{r}$ a vector joining the two such that

$$\mathbf{x} = \mathbf{x}_0 + \mathbf{r}. \tag{5.1}$$

Substituting eq. (5.1) into eq. (3.3), we obtain

$$f(\mathbf{x}) = (\mathbf{x}_0 + \mathbf{r})'\mathbf{Q}(\mathbf{x}_0 + \mathbf{r}) + \mathbf{g}'(\mathbf{x}_0 + \mathbf{r}) + c. \tag{5.2}$$

Carrying out the multiplications, and considering that, by hypothesis, $f(\mathbf{x}_0) = 0$, we find

$$f(\mathbf{x}) = 2\mathbf{x}_0'\mathbf{Q}\mathbf{r} + \mathbf{r}'\mathbf{Q}\mathbf{r} + \mathbf{g}'\mathbf{r}. \tag{5.3}$$

If we select $\mathbf{x}_0$ at the foot of the normal from $\mathbf{x}$ to the conic, then the length of $\mathbf{r}$ will be the distance of this point from the curve. Clearly, $f(\mathbf{x})$ is a rather complicated function of that distance and one can construct examples where, depending on the location of $\mathbf{x}$, we have different values of $f(\mathbf{x})$ for the same length of $\mathbf{r}$. Equation (5.3) shows that the use of $f(\mathbf{x})$ as a measure of the distance of a point from a conic is not justified. Nevertheless, it can be used as a starting point for deriving simple distance estimates.

5.1 Distance of a Point from a Parabola

It is necessary to handle parabolas differently from ellipses and hyperbolas. The latter two types of conics have a center while parabolas do not. However parabolas have the property that the matrix $\mathbf{Q}$ has a zero eigenvalue. If we select $\mathbf{r}$ to be an eigenvector of $\mathbf{Q}$ corresponding to that eigenvalue, then $\mathbf{Q}\mathbf{r}$ is zero and we have

$$f(\mathbf{x}) = \mathbf{g}'\mathbf{r}. \tag{5.4}$$

Because the direction of $\mathbf{r}$ is fixed, its scalar product with $\mathbf{g}$ is proportional to the norm of $\mathbf{r}$. It can be shown easily that the eigenvector in question is parallel to line CM in Figure 4. If $\mathbf{x}_1$ and $\mathbf{x}_2$ are the endpoints (A and B in Figure 4), and $\mathbf{x}_b$ the point where the tangents intersect (C in Figure 4), then the equation of the parabola can be written as

$$\begin{aligned} f(\mathbf{x}) = {} & [dy_{12}(x - x_1) - dx_{12}(y - y_1)]^2 \\ & + [(y_b - y_1)(x - x_1) - (x_b - x_1)(y - y_1)] \\ & \cdot [(x_b - x_1)(y_2 - y_1) - (y_b - y_1)(x_2 - x_1)] \end{aligned} \tag{5.5}$$

where

$$dx_{12} = x_b - \frac{x_1 + x_2}{2} \qquad \text{and} \qquad dy_{12} = y_b - \frac{y_1 + y_2}{2}.$$

This is a special case of eq. (3.15), which produces the following result.

PROPOSITION 5.1. *For a parabola given in the form of eq.* (3.15), *the quantity* $f(\mathbf{x})$ *is proportional to the distance of the point from the parabola computed along a segment parallel to the line joining the intersection of the tangents with the midpoint of the chord* (*CM in Figure* 4). *If the special form of eq.* (5.5) *is used, then the distance is given by the following equation*:

$$D(\mathbf{x}) = \frac{2f(\mathbf{x})}{[(y_2 - y_1)(x_b - x_1) - (x_2 - x_1)(y_b - y_1)]^2} \sqrt{dx_{12}^2 + dy_{12}^2}. \tag{5.6}$$

Clearly, $D(\mathbf{x})$ is a good estimate of the distance along the normal to the curve only when the foot of the normal is near point F in Figure 4.

5.2 Distance of a Point from an Ellipse or a Hyperbola

For ellipses and hyperbolas we can find a useful relation for the distance of a point from the curve computed along the line joining the point and the center of the conic. Let $\mathbf{x}_c$ be the coordinates of the center. Equation (5.1) can then be written as

$$\mathbf{x} - \mathbf{x}_c = \mathbf{x}_0 - \mathbf{x}_c + \mathbf{r}. \tag{5.7}$$

If the vectors $\mathbf{x} - \mathbf{x}_c$ and $\mathbf{x}_0 - \mathbf{x}_c$ are collinear, then there exists a nonnegative scalar μ such that

$$\mathbf{r} = \mu(\mathbf{x}_0 - \mathbf{x}_c). \tag{5.8}$$

μ exceeds one if $\mathbf{x}$ is "outside" the conic, and it is less than one if $\mathbf{x}$ is "inside" the conic. Substituting eq. (5.8) into eq. (5.3), and using eq. (3.4) for $\mathbf{x}_c$, we find

$$f(\mathbf{x}) = (2\mu + \mu^2)(\mathbf{x}_0'\mathbf{Q}\mathbf{x}_0 + \mathbf{g}'\mathbf{x}_0 + \frac{1}{4}\,\mathbf{g}'\mathbf{Q}^{-1}\mathbf{g}). \tag{5.9}$$

We shall use the symbol $q(\mathbf{x}_0)$ to denote the expression $\mathbf{x}_0'\mathbf{Q}\mathbf{x}_0 + \mathbf{g}'\mathbf{x}_0$. Since $\mathbf{x}_0$ was assumed to be on the conic, using eq. (3.3) we have $q(\mathbf{x}_0) = -c$. Substituting this value and using eq. (3.7), we find

$$f(\mathbf{x}) = -c_1(2\mu + \mu^2). \tag{5.10}$$

Elimination of μ between Eqs. (5.8) and (5.10) yields

$$\|\mathbf{r}\| = \|\mathbf{x}_0 - \mathbf{x}_c\| \left| \sqrt{1 - \frac{1}{c_1} f(\mathbf{x})} - 1 \right|, \tag{5.11}$$

or, in a simpler form

$$\|\mathbf{r}\| = \|\mathbf{x}_0 - \mathbf{x}_c\| \left| \sqrt{-\frac{1}{c_1} q(\mathbf{x})} - 1 \right|, \tag{5.12}$$

where $q(\mathbf{x}) = f(\mathbf{x}) - c$. Clearly, the norm of the vector $\mathbf{x}_0 - \mathbf{x}_c$ is bounded from above and below by the size of the two axes.[3] Also, the distance of $\mathbf{x}$ from the curve is no greater than $\mathbf{r}$, and it equals $\mathbf{r}$ along the axes. Therefore we can find upper and lower bounds for it. Using eq. (3.8), we find

$$\left| \sqrt{-\frac{1}{c_1} q(\mathbf{x})} - 1 \right| \|\mathbf{a}_{\min}\| \le D(\mathbf{x}) \le \left| \sqrt{-\frac{1}{c_1} q(\mathbf{x})} - 1 \right| \|\mathbf{a}_{\max}\|.^4 \tag{5.13}$$

While eq. (5.13) suggests that the distance $D(\mathbf{x})$ is a monotonic function of $f(\mathbf{x})$, this is strictly true only for the bounds. Thus $f(\mathbf{x})$ is only a gross indicator of the

---

[3] One can show that the angle between the normal to a conic at a point $\mathbf{x}$ and the line joining $\mathbf{x}$ with the center is at most $\pi/2 - 2\tan^{-1}\sqrt{a/b}$. Therefore $\mathbf{r}$ and $\mathbf{x}_0 - \mathbf{x}_c$ do form a small angle if the axes have comparable lengths.

[4] The reader can confirm that when an ellipse is given in the canonical form:

$$x^2/A^2 + y^2/B^2 - 1 = 0,$$

then eq. (5.13) becomes

$$|\sqrt{x^2/A^2 + y^2/B^2} - 1|B \le D(x) \le |\sqrt{x^2/A^2 + y^2/B^2} - 1|A.$$

distance, and its use to specify a conic (e.g., by minimizing sums of squares of $f(\mathbf{x})$ for various points), is not recommended. On the other hand, eq. (5.13) could be used to check whether a *given* approximation by a conic is acceptable. If the coefficients of the conic have been normalized so that $ab - h^2 = 1$, then it is easily shown that the geometric mean of the two axes equals $(|c_1|)^{1/2}$.

PROPOSITION 5.2. *If the equation of the conic has been normalized so that* $ab - h^2 = 1$, *and the two axes do not differ much in length, then the distance of a point* $\mathbf{x}$ *from the conic is given approximately by*

$$D(\mathbf{x}) \approx |\sqrt{f(\mathbf{x}) - c_1} - \sqrt{|c_1|}| = |\sqrt{q(\mathbf{x}) + c - c_1} - \sqrt{|c_1|}|. \tag{5.14}$$

Eq. (5.14) can be proved by substituting the geometric mean of the axes into eq. (5.13). It is also easily confirmed that it holds exactly when the conic is a circle. A further simplification of eq. (5.14) can be made by replacing c_1 with c. To demonstrate this we can use eq. (3.7) and denote $\frac{1}{4}\,\mathbf{g}'\mathbf{Q}^{-1}\mathbf{g}$ by v. Then the last expression of eq. (5.14) becomes

$$D_0(\mathbf{x}) = \sqrt{q(\mathbf{x}) + v} - \sqrt{c + v}. \tag{5.15}$$

For points near the conic, $q(\mathbf{x})$ is close to $-c$. Then

$$\sqrt{q(\mathbf{x}) + v} - \sqrt{c + v} = \sqrt{1 + \frac{v}{c}}\,[\sqrt{q(\mathbf{x})} - \sqrt{c}]. \tag{5.16}$$

The estimate

$$D_1(\mathbf{x}) = \sqrt{q(\mathbf{x})} - \sqrt{c} \tag{5.17}$$

is simpler to compute than $D_0(\mathbf{x})$, and eq. (5.16) shows that the ratio between D_0 and D_1 is approximately constant. Special cases of these expressions have been used in [2].

5.3 On the Selection of a Fifth Point for Guided Conics

The following is proved by straightforward substitutions.

PROPOSITION 5.3. *Let* K_i *be the coefficient* K *in eq.* (3.13) *determined by requiring that the conic pass through the point* $\mathbf{x}_i$. *Then define*

$$f_i(\mathbf{x}) = (\mathbf{a}'\mathbf{x} + a_0) \cdot (\mathbf{b}'\mathbf{x} + b_0) - K_i \cdot (\mathbf{u}'\mathbf{x} + u_0)^2. \tag{5.18}$$

Let $\mathbf{x}_j$ *be another point that may be used to specify a conic. Then if* $\mathbf{x}_i$ *and* $\mathbf{x}_j$ *are equidistant from chord AB,*

$$f_i(\mathbf{x}_j) = -f_j(\mathbf{x}_i). \tag{5.19}$$

In other words, when we have to choose between two points (equidistant from the chord) for specifying a guided conic, it makes no difference which one of the two points we pick because both result in the same value of $|f(\mathbf{x})|$ for the other point and therefore yield the same value for eq. (5.14).

6. APPROXIMATION BY CONIC SPLINES

Specifying a curve through a guiding (control) polygon is a method popular among experienced users—naive users, however, may prefer to draw the curve

itself in a rough way and expect the automatic production of a "good looking" curve. In other applications, data must be processed without the help of a human operator. When curve fitting is to be done noninteractively, it is necessary to select the location of the knots and the other parameters by an algorithmic procedure. References [14], [23], and [24] are three recent papers on this topic with an emphasis on graphics applications. (Reference [24] contains many references to the earlier literature.) All use the parametric representation of eq. (1.1). The selection of a_{ij} and t_j is made while minimizing certain error criteria that measure how far the spline is from the given data points $\mathbf{X}_k$, $(k = 1, \ldots, N)$.

Earlier writers who have studied approximations by conics have used five parameters to define a conic and then solved a least squares minimization problem with constraints for finding the conic that fits a set of data points. (See, for example, [2] or [13]. Reference [2] contains many citations to earlier papers with similar methods.) Because there is no simple expression for the distance of a point from the conic, some have used the value of $f(x, y)$. In this paper we rely on the guided form of conics introduced in Section 3, which is a convenient form to use for conic splines because the continuity constraints are satisfied automatically. However, this simplification still leaves us with a nasty problem because both the location of the knots and the direction of the curve tangents at them are not known. We overcome this problem by first finding a polygonal approximation of the data, and then selecting the knots and their tangents on the basis of the properties of the polygon found. The major advantage of this strategy is that the expensive knot search procedure is performed for the polygons and that therefore the fitting of conics is a one-, or at most, two-pass operation.

Roughly speaking, we use the triangular inequality among functions. We find a polygon that is near our data points and then a set of conic arcs that approximates the polygon. The error norm between the last approximation and the data points is less than the sum of the error norms of the previous two approximations. Clearly, the result is not going to be as good as the result of a direct approximation. However, it is important to stop and think what "good" means when we fit curves. If we decide that good means minimizing a well-defined mathematical cost function, then the approach of this paper leaves much to be desired. However there is little agreement as to what the proper cost function is. Therefore it is counterproductive to invest significant effort in mathematical optimization if we doubt that the mathematically optimal solution is what we actually want. In design applications where shape is very important, the results will be postedited anyway, while in pattern recognition applications we are interested only in the general shape of a curve, and details of the representation are usually ignored. (E.g., one may be interested only in knowing whether an arc is convex or concave.)

We chose to start with polygonal approximation because it is much easier than approximation by higher order curves. This is true for many reasons: first, one may use interpolants rather than approximants with a significant savings in computation costs. It is well known that the maximum error with respect to the interpolant is at most twice that of the optimal approximation [5, pp. 40–41]. That condition is achieved when there are no changes in the sign of the error; usually the maximum error from the interpolant is below that upper limit. Second, eq. (3.10) provides a very simple formula for computing the distance of a point

from a line. If the data to be fitted indeed have polygonal parts, then performing the cheaper approximation first makes sense: we will do the higher order approximations only when we need them. One of the intended applications of this work is the fitting of curves on contours of alphanumeric characters of various fonts (see Section 9). Many characters, for instance Helvetica A, have no curvilinear parts. There are also some deeper reasons for using polygons for solving the knot location problem. These are described in Appendix B.

The use of a polygonal approximation to determine sequences of vertices which can be replaced by a conic has been suggested before, particularly in [13] and [21]. Pavlidis and Ali, however, stopped short of actually finding the conics, while Liao attempted to fit conics in all parts of the polygon without any prescreening. Lozover and Preiss [14] also used a rough polygonal approximation as a first pass and then determined a cubic spline whose knots are the vertices of the polygon. In Section 7 we present criteria that use the size of the angles of the vertices and the ratio of the lengths of subsequent sides to classify a vertex as *hard*, *soft*, or *break*. Hard vertices are left as polygonal vertices, while groups of adjacent soft vertices are considered for approximation by conics. In addition, we label sides as breaks if they separate two conics. Section 8 includes methods for selecting the knots and tangents at them. The following is an outline of the algorithm.

6.1 Outline of Algorithm

(1) Find an approximating polygon.
(2) Classify vertices as hard, soft, or break according to the criteria of Section 7.
(3) Identify breaks from among sides joining soft vertices according to the criteria of Section 7.
(4) Fit conics on all sequences of soft vertices using the method presented in Section 8. If a conic does not fit well, subdivide the interval and try again.

Since the rest of the paper does not depend on the way the polygon is found, we dispense with a discussion of polygonal approximations. We assume only that we are given N data points $\mathbf{X}_k$, $k = 1, \ldots, N$, a maximum error tolerance e_m, and that we want to determine a polygonal approximation of the data points such that for each data point there is a line within e_m from it. There are numerous algorithms for doing this. (See [18] for a survey of earlier work.) The specific algorithm used in this paper is described in [19, pp. 281–292] and is analyzed in [20]. Its results have the following properties.

(a) All data points are within distance e_m of a side of the polygon.
(b) All vertices of the polygon are also data points.
(c) It is not possible to replace two sides by one without having the maximum error exceed e_m.

We emphasize that the algorithm does not depend at all on the way the conic is represented and is equally adaptable to the algebraic or rational parametric forms.

7. CLASSIFICATION OF POLYGONAL VERTICES

The central idea of this section is that if a polygon with more than five vertices is inscribed in a conic, then there are certain relations that must be satisfied by

the size of its angles and sides. If we are given an arbitrary polygon and we want to find whether parts of it can be inscribed in conic arcs, then it is reasonable to search for sequences of vertices that satisfy the relations that vertices of an inscribed polygon do. In what follows, the term *hard* is used to describe a polygon vertex that is unlikely to be part of a conic arc, while *breaks* are points where two different conic arcs may meet.

We shall use properties of conics to classify polygonal vertices as hard, soft, or breaks. A soft vertex is one that is not hard. It may or may not be a break. While all our criteria are stated as heuristics, they are motivated by Pascal's theorem, stated and discussed in Section 4. If we are given two endpoints and tangents at them, the theorem implies certain bounds on the vertex size and the difference in the size of adjacent vertices. Referring to Figure 5, we observe that for any triangle ABC and two points D and E inside it we have

PROPOSITION 7.1. *Angles ADE and DEB have a sum that is greater than* $\pi + ACB$.

A simple geometric calculation using the collinearity of P_1, P_2, P_3, shows that

$$ADE = \pi - P_2AB - DP_3A \qquad \text{and} \qquad DEB = \pi - P_1BA + DP_3A.$$

Subtracting the first equation from the second we find

PROPOSITION 7.2. *The difference between two adjacent angles of a polygon inscribed in a conic satisfies the following equation*:

$$DEB - ADE = P_2AB - P_2BA + 2DP_3A.$$

The difference becomes maximum in the limiting case when P_3 tends to B and both P_1 and P_2 tend to C. Then DP_3A approximately equals P_2BA, and the difference is found to be less than $\pi - ACB$. However, if side DE forms a small angle with chord AB, then the right-hand side of the equation of Proposition 7.2 is near zero.

These observations lead us to some simple criteria: first, that of the size of the angle at a vertex. In principle, no matter how sharp an angle is, we can always fit a conic that passes through that vertex and the two adjacent vertices and also stays within distance e_m of the sides. (We have exactly five conditions.) However, such a conic is not likely to be extensible to include other vertices. If we use the guided form of conics of eq. (3.13), then we can show that the distance of the vertex from the unique parabola is proportional to the sine of the angle between the tangents. The sine function is small not only when the angle is small, but also when it is near 180°. Therefore soft vertices should have angles that are near 180°, not a surprising conclusion. Thus we have

HEURISTIC 7.1. *A vertex is classified as hard if it is less than* $180° - a_1$, *or greater than* $180° + a_1$, *for a given angle* a_1.

Appendix B describes some experiments demonstrating that, at least for some applications, the distribution of the size of the polygonal angles is bimodal and that there is, therefore, a certain latitude in the selection of the size of the threshold for classifying a vertex as hard or soft. If a vertex is classified as hard, then it is preserved in the final approximation and separates conics from each other. However, conics may be separated at points where there is no derivative

discontinuity, and therefore the knot will be near a vertex that is not hard. We classify such vertices as breaks and find them by means of the following criteria. The first criterion for breaks is based on the ratio of side lengths. The following result is easily proven using eq. (3.6).

PROPOSITION 7.3. *Let O be the center of a conic and A and B two points on it. Let M be the middle of chord AB, and P the point where line OM intersects the conic. Find a point C such that if N is the middle of chord BC, and R the point where ON intersects the conic, then*

$$MO/PO = NO/RO. \tag{7.1}$$

Then Chords AB and BC are parallel to rays (lines from the center) of a concentric conic similar to the one given.

Note that the point C may not always exist. But given A, B, C, P, and R, one can always find a conic passing through them.

COROLLARY. *The ratio of the lengths of chords AB and AC is bounded by the ratio of the lengths of the axes of the given conic.*

This is a particularly useful result for our purpose. Let P be a given polygon. We want to test the hypothesis that two of its sides form chords of a conic under the conditions of the theorem. Furthermore, we assume that the axes of the conic have a ratio within given limits. (This is reasonable for many applications, such as curve fitting for font description.) If the side ratio is outside these limits, then the sides cannot be chords. Since we expect the theorem to hold approximately when its hypotheses do not hold exactly, we can develop the following heuristic for distinguishing between hard and soft vertices.

HEURISTIC 7.2. *If the ratio of two adjacent sides exceeds a given value R_1, then the vertex is classified as a break.*

We can find a criterion for labeling sides as breaks by observing that since conics have no inflection points, we should always separate groups of angles that are greater than 180° from those that are less than 180°. A side adjacent to such a pair of angles should be classified as a break. In addition, we should define as breaks sides where the two adjacent angles are significantly different. This is dual to Heuristic 7.2.

HEURISTIC 7.3. *A side is classified as a break if the angles adjacent to it are on different sides of* 180° *or they differ by more than a given threshold A_1.*

Many of these heuristics have been motivated by Proposition 7.2, which states that the difference between the angles is small if angle ACB of Figure 3 is large. In order to ensure that this is the case, we introduce a fourth heuristic.

HEURISTIC 7.4. *A vertex is classified as a break if its addition to the conic would cause the angle between the two endpoint tangents to be less than a given angle A_2. (Otherwise the conic arc would exceed* 180° $- A_2$.)

This heuristic was found to be very useful in practice. During tests we observed that conics that covered wider arcs almost always also violated the error criteria.

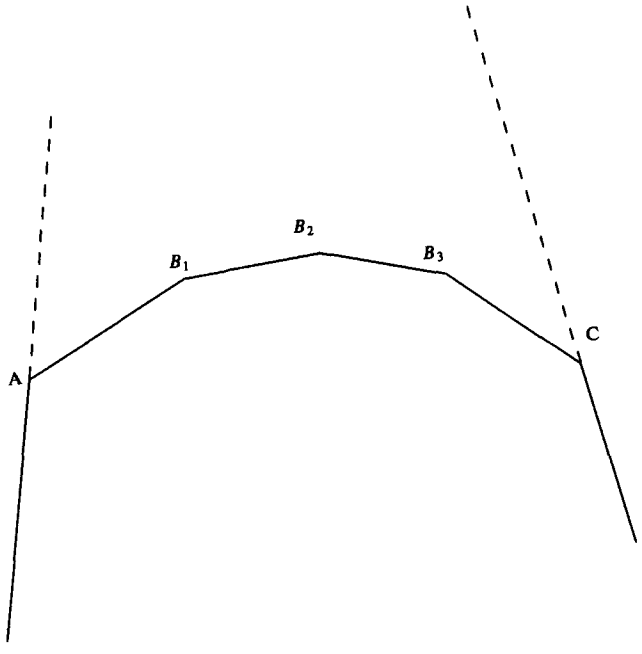

Fig. 6. Arrangement of polygonal vertices that are considered for approximation by a conic. A and C are hard vertices and B_i is soft according to the criteria of Section 7. The broken lines are the tangents at the endpoints.

8. FITTING A CONIC

The labeling of the vertices and sides of a polygon according to the criteria of Section 7 is a very simple operation. (The most expensive computation is the calculation of the angles of the vertices.) Suppose that A is a hard vertex, and B_1, . . . , B_n a sequence of soft vertices, as shown in Figure 6. Then line AB_1 is taken as the tangent of the conic at the knot near A. We must select a point on line segment AB_1 at which to start the conic arc. The following heuristic has been found to give good results.

HEURISTIC 8.1. *If the ratio of side AB_1 to side B_1B_2 is less than R_1, then select the starting point of the conic at A. Otherwise select the starting point of the conic arc along segment AB_1 at a distance from B_1 equal to half the size of segment B_1B_2.*

The motivation for using half the length is that this would be the optimal selection of points if we were fitting a parabola. (Recall Figure 4.) If a vertex is a break, then we consider the vertices on either side. If A_0 is the previous vertex, we select line A_0A as the tangent and use the midpoint of the line as the starting point, unless B_1B_2 is less than half of A_0A_1. Then we apply Heuristic 8.1. If a side has been classified as a break, then it is taken as a tangent and its midpoint is used as starting point. Similar criteria are used for the end of a sequence of soft vertices.

1. **For** each vertex **do**:
   2. **If** the vertex has an angle not near 180°, **or** if the ratio of the adjacent sides is too big, **or** if it differs from the next or previous vertex too much, **then** mark it.
3. **For** each vertex **do**:
   4. **If** the $flag_{old}$ is one **do**:
      5. Use as P_1 and P_2, P_3 and P_4 respectively.
   6. **Else if** the vertex is not marked **do**:
      7. Skip all unmarked vertices until either a marked vertex is reached or the angle between the first and the current polygon side is too big. In the latter case set the $flag_{new}$ to one.
      8. Set P_4 to be the current vertex and P_3 the previous.
   9. **If** P_2 and P_3 coincide, then select the parabola from P_1 to P_4 with tangents the lines P_1P_2 and P_2P_4.
   10. **Else do**:
      12. Select as a fifth point the vertex as close as possible to the middle between P_2 and P_3, or if there is no such vertex select the midpoint of the side P_2P_3.
      13. **If** the $flag_{old}$ is set, select P_0 as the midpoint of the segment P_1P_2.
      14. **Else if** the side P_1P_2 is less than a specified multiple of the next side use P_1 for P_0. **Else** select P_0 on P_1P_2 at a distance from P_2 equal half the length of the next side.
      15. Select P_n in a similar way as P_0, but replacing P_1 by P_3 and P_2 by P_4.
      16. Compute the equation of the conic according to eq. (5.18) or eq. (3.19).
   17. **If** $flag_{new}$ equals one, **then** set $flag_{old}$ to one. **Else** set $flag_{old}$ to zero.
   18. Set $flag_{new}$ to zero.

Fig. 7. Conic Fit. *Notation*: $flag_{\text{new}}$ is set to one when we stop a sequence of vertices because of Heuristic 8.1. $flag_{\text{old}}$ is set to one to indicate that the previous sequence of vertices was stopped because of Heuristic 8.1. P_1 and P_2 define the first tangent and P_3 and P_4 the second.

In addition to endpoints we must select a fifth point to specify the conic completely. It is tempting to attempt the selection of these points by means of an optimization procedure. If the soft vertices have the configuration of Figure 6, then their distances from the chord would not vary much, and according to Proposition 6.3 it should not matter which one we select for a fifth point. For this reason we select the vertex closest to the middle as the fifth point. Figure 7 lists the conic fitting procedure in detail.

After a conic is fitted, we compute the distance of the vertices from it using eq. (5.14). If the axes have significantly different lengths, then we may use eq. (5.13). We have found that in most instances the conics fitted initially give good results, so usually there is no need to modify the approximation. If high-quality fits are desired, then the following procedure improves the fit. The set of vertices defining the conic is split into two parts separated by the vertex where the error is *zero*. If the conic ends in a hard vertex or a break, new conics are fitted on both parts. If the conic has been terminated because of the turn criterion (Heuristic 8.1), then the first part only is fitted and the second part is continued until a stop condition is encountered.

Intuitively, one might have expected that the interval should be split at the point of maximum error. This is the point used for polygonal approximations [20]. However, things are different in the case of conics. According to basic properties of polynomial approximations, one expects two locations of maximum error [10]. If we do not wish to go from one to three intervals, we have the problem of selecting one of the two maxima. Often such maxima occur not far from knots and we obtain a very unbalanced split (see the examples of Section 9). Another

reason has to do with the asymptotic distribution of spline knots [5, pp. 180–190] and [15]. It has been shown in the $y = f(x)$ case, that when knots are selected to minimize the integral square error (ISE), their distribution follows the $k + 1$th derivative of the $f(x)$, if the splines are of the kth order. For a linear spline the knot distribution follows the second derivative. Points inside an interval where the error is maximum usually have high curvature and therefore high values for the second derivative. For quadratic splines the knot distribution follows the third derivative. Maxima of the third derivative correspond to zeros of the fourth. For many functions zeros of derivatives differing by two occur near each other [10]. (They coincide in the case of the sine or cosine functions.) Thus for quadratics the knots should be near inflection points, and these are more likely to occur where the error is zero.

If a curve contains no hard vertices or breaks, then the algorithm can either start at an arbitrary point and apply Heuristic 8.1 to fit conic arcs, or (at the user's option) it may fit an ellipse to the whole set of points. There is little difference between open and closed curves for the algorithm. For open curves the first and last point are simply labeled as hard.

9. INTENDED APPLICATIONS, EXAMPLES AND COMPARISONS

This paper has been motivated by four types of applications. One is curve fitting on data that are entered manually into an interactive graphics system. A second is approximation of digitized drawings for data compaction and, possibly, for eventual pattern recognition. The other two applications involve alphanumeric characters and other symbols found in documents. The first of these is *font scaling*. Given a set of fonts for some output device, we want to produce another set for another output device whose resolution is substantially different from the first. For example, we may want to scale fonts from a phototypesetter with a resolution of 1000 lines per inch to a bit map display that has only 100-line-per-inch resolution. It is well known that direct scaling of binary images produces unsatisfactory results. If the output device has gray scale output, then we may use gray levels to produce satisfactory results [11, 30]. If it does not, then we must solve the scaling problem carefully. Knuth [12] has used mathematical descriptions of fonts that can be transformed with precision to produce actual fonts. He has called such descriptions METAFONT. These descriptions can be used effectively for scaling as follows: the description is scaled while one ensures that symmetry and other important properties of the shape of the character are preserved. Then the binary matrix is generated from the scaled description. While the use of abstract descriptions has clear advantages, their derivation is rather laborious and requires that the designer be familiar with curve fitting. What we propose to do is to start with a given font from a high-resolution device (or an artist's drawing of it) and then to "decompile" it and produce a METAFONT type description for it. The selection of conic arcs for the description is not unreasonable since they have been used by font designers (see [12], Figures 4–6). The only problem with font "decompiling" is that, according to Section 6, the error of approximation is not very sensitive to the knot location for knots where the tangent is continuous. Therefore decompiling is unlikely to provide descriptions

that can be used immediately as a METAFONT, and some further human intervention will be needed.

The fourth application is *recognition of alphanumeric characters* using structural techniques [18]. In order to recognize a symbol we obtain either a tracing of its contour or a tracing of its "skeleton" that results from thinning. The curves that approximate the skeleton or contour are then used for deciding the identity of the character. Such methods have been used in the past with some success [21], and we hope that better algorithms for encoding will improve the performance of the recognizers [22].

In principle, the same curve fitting algorithm could be used for both font decompiling and character recognition. There are some differences though. Font descriptions require far higher quality representations than the descriptions for character recognition, while the computing resources for font decompiling are usually much greater.[5] It is possible to take advantage of this trade-off between quality and speed by using coarser tolerance in curve fitting for recognition, approximate formulas for the computation of conics, and even approximations of simpler functions such as square root. Both the quality and speed requirements for smoothing of interactively entered data are between the extremes of font scaling and character recognition.

The algorithm has been implemented in C under the UNIX[6] operating system and run on both a PDP11/70 and a VAX11/780 machine. Figures 8–10 show examples of its application on character contours obtained from the original font descriptions. All the examples show both the conics fitted in the first attempt *without corrections* in (a), and *after corrections* in (b) and (c). The examples show clearly that splitting the interval at points where the error is zero is preferable to splitting at points where the error estimate is maximum. The approximations are represented by solid lines, overlaid on the originals, represented by dotted lines. In addition, Figure 8d shows the guiding polygon. The data of these examples are similar to those used in [23], and a comparison of Figures 9 and 10 with Figures 11a and 10c of that work shows that conic splines require about the same number of knots as parametric cubic splines. The new method seems to be faster, though. The whole process (polygon and conic fitting) required about one CPU second per character on the PDP11/70, and half that time on the VAX11/780. The method of [23] requies about 60 seconds on a DORADO (M. Plass, personal communication). However, this comparison should be interpreted with caution, given that the tests were made on different machines, with programs written by different people and in different languages.

Figure 11 shows the approximation of an epicycloid, produced by the same equations as the epicycloid used in [24]. It requires 15 knots after splitting some interval to bring the maximum error within 1.5 grid units. Reeves and Sermer [24] have reported between 14 and 31 knots for cubic splines, depending on the accuracy. For 496 data points the total processing time was 1.5 CPU seconds on

[5] A character recognition machine may be required to process 100 characters per second under the constraint that the total cost of the machine be under $50,000. On the other hand it is quite reasonable to expect that a mainframe machine would devote 30 seconds per character so that a complete font of 120 characters could be "decompiled" in one CPU hour.

[6] UNIX is a trademark of Bell Laboratories.

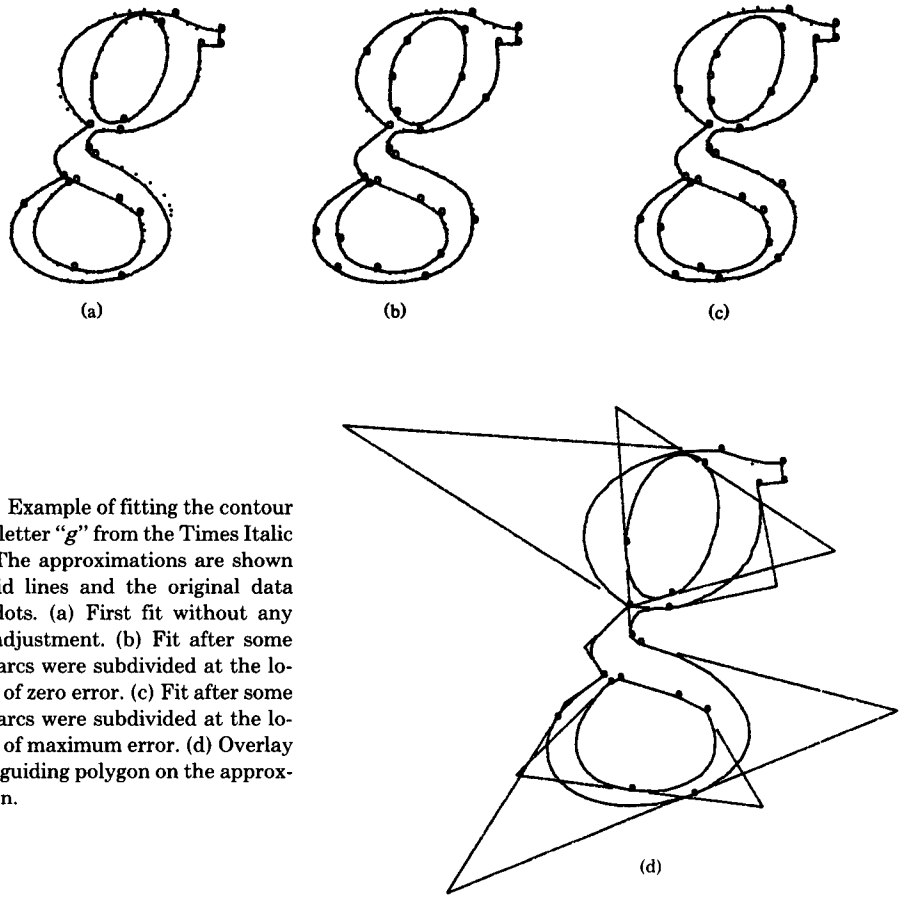

Fig. 8. Example of fitting the contour of the letter "*g*" from the Times Italic font. The approximations are shown in solid lines and the original data with dots. (a) First fit without any knot adjustment. (b) Fit after some conic arcs were subdivided at the location of zero error. (c) Fit after some conic arcs were subdivided at the location of maximum error. (d) Overlay of the guiding polygon on the approximation.

the VAX11/780. The times reported by [24] are between 2 and 3 seconds. Figure 12 shows the application on a free-drawn contour on an interactive graphics device. There were 401 data points, and they required about the same time as the epicycloid.

Clearly, there is room for improvement, and the approximations obtained do not always have the minimum number of knots. One possibility is to label fewer vertices as hard by making the requirement more stringent. We could then attempt to fit a conic in the "arm" of the penguin in Figure 12. However the increase in the number of attempts would result in an increase in computational costs. We could also have considered variations in the location of the knots and possibly of the merge intervals. Such techniques are well known (and computationally costly), and we did not want to obscure the quality of the results obtained by a simple algorithm using the conic splines.

The speed of the method is quite fast for font "decompiling," and also fast enough for interactive graphics applications. However one second per character is too slow by a factor of 100 for optical character recognition (OCR) applications. It is expected that the time can be reduced because of a smaller number of

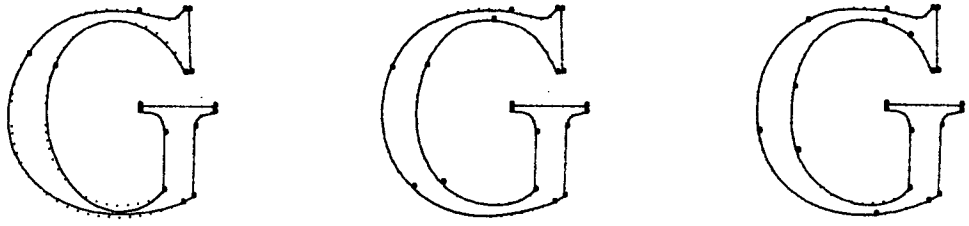

Fig. 9. Example of fitting the contour of the letter "G" from the Times Roman font. (a), (b), and (c) as in Figure 8.

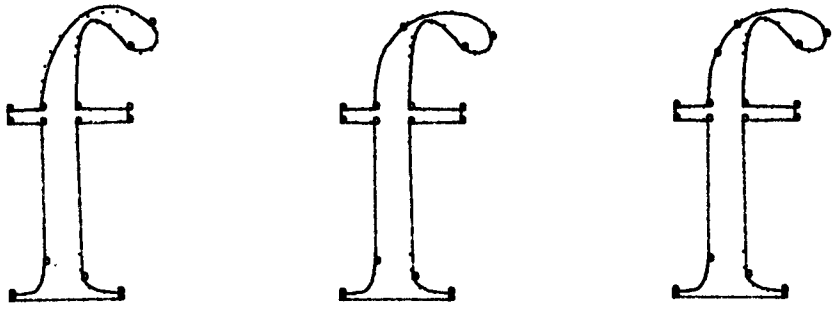

Fig. 10. Example of fitting the contour of the letter "f" from the Times Roman font. (a), (b), and (c) as in Figure 8.

samples used per contour. The font descriptions were for the maximum point size (36) and 1000 samples per inch. Ten point characters sampled at 240 samples per inch (a commonly used resolution) will have about one-tenth the number of contour points. This introduces a speedup factor of about 10. Also, for OCR applications we need not compute the actual conics, as long as we know some of their features. The ability of the algorithm to find good conics with the first attempt is critical in this respect. Finally, some special-purpose hardware may be used for fitting lines.

APPENDIX A. INCREMENTAL PLOTTING OF PARABOLAS

The simplest way to plot parabolas is by using divided differences on the parametric form. Then only four additions per point are required. We present an incremental technique only for the sake of generality.

PROPOSITION A1. *For each parabola there exist matrices* $\mathbf{B}$ *and vectors* $\mathbf{m}$, *such that if* $\mathbf{x}$ *is a point on the parabola,* $\mathbf{Bx} + \mathbf{m}$ *is also a point on the same parabola.*

PROOF: It is well known that the parametric representation of a parabola has the form

$$\mathbf{x}(t) = \mathbf{P}t^2 + \mathbf{R}t + \mathbf{S} \tag{A1}$$

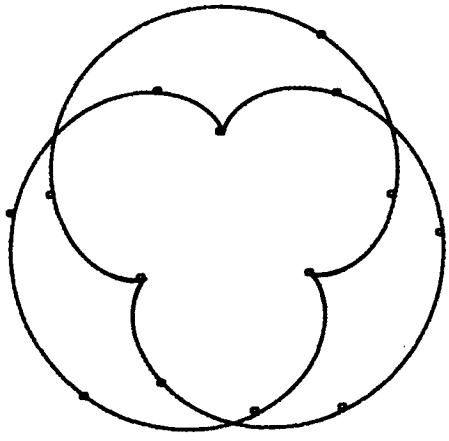

Fig. 11. Example of fitting an epicycloid with conic arcs.

for appropriately chosen vectors **P**, **R**, and **S**. Let **n** be a vector orthogonal to **P**. Taking the scalar product of both sides of eq. (A1) with **n** and solving with respect to t, we find

$$t = \frac{\mathbf{n}'\mathbf{x} - \mathbf{n}'\mathbf{S}}{\mathbf{n}'\mathbf{R}}. \tag{A2}$$

Note that $\mathbf{n}'\mathbf{R}$ will be nonzero unless **R** is zero or collinear to **P**. In both cases the parabola degenerates into a straight line. If we compute now the value of **x** for another value of the parameter equal to $t + s$, and substitute t from eq. (A2), we find

$$\mathbf{x}(t+s) = \mathbf{x}(t) + s\mathbf{R} + s^2\mathbf{P} + 2s\left[\frac{\mathbf{n}'\mathbf{x} - \mathbf{n}'\mathbf{S}}{\mathbf{n}'\mathbf{R}}\right]\mathbf{P}. \tag{A3}$$

Let us now define a matrix **B** as

$$\mathbf{B} = \mathbf{I} + 2s\frac{\mathbf{P}\mathbf{n}'}{\mathbf{n}'\mathbf{R}}, \tag{A4}$$

and a vector

$$\mathbf{m} = s\mathbf{R} + s^2\mathbf{P} - 2s\frac{\mathbf{n}'\mathbf{S}}{\mathbf{n}'\mathbf{R}}\mathbf{P}. \tag{A5}$$

I stands for the identity matrix. Eq. (A5) can then be written as

$$\mathbf{x}(t+s) = \mathbf{B}\mathbf{x}(t) + \mathbf{m}. \tag{A6}$$

Both **B** and **m** depend only on s and not on t, and therefore when applied on any point of the parabola they yield another point, specifically one which is s units of the parameter later. □

This result would not have been very useful had there been no way of finding a matrix **B** and a vector **m** without resorting to the parametric representation.

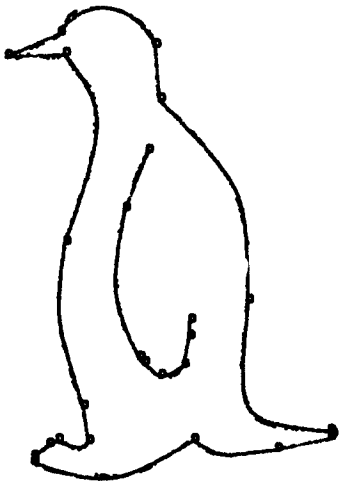

Fig. 12. Example of fitting a sketch entered through a digitizing tablet in an interactive graphics system.

However, it can easily be shown that given an initial point $\mathbf{K}_i$, a final point $\mathbf{K}_{i+1}$, and a guiding point $\mathbf{V}_{i+1}$, then vectors $\mathbf{P}$, $\mathbf{R}$, and $\mathbf{S}$ are given by

$$\mathbf{P} = \mathbf{K}_{i+1} + \mathbf{K}_i - 2\mathbf{V}_{i+1}, \tag{A7a}$$

$$\mathbf{R} = 2(\mathbf{V}_{i+1} - \mathbf{K}_i), \tag{A7b}$$

$$\mathbf{S} = \mathbf{K}_i. \tag{A7c}$$

We can then use eqs. (A4) and (A5) to find $\mathbf{B}$ and $\mathbf{m}$.

APPENDIX B. A COMPARISON OF OPTIMAL AND SUBOPTIMAL SOLUTIONS

We list here some results that, while not directly related to the approximations used in this paper, provide some insight into why more elaborate techniques may not yield better quality approximations. If we use the integral square error (ISE) as a measure for the quality of fit, then we can express the knot location problem in terms of conventional optimization theory. Furthermore it is possible to compute explicitly the partial derivatives of the ISE with respect to the knot location in the parametric and $y = f(x)$ cases. This was done by the author in a sequence of papers [16, 17]. When the $y = f(x)$ form is used, the explicit computation of the first partial derivatives of the ISE and the application of Lagrange multipliers produce the following result.

THEOREM B1: *A necessary condition for the optimal location of the knots for a continuous piecewise linear approximation is that the unconstrained optimal approximation on each subinterval results in a continuous approximation over the whole interval* [17].

This theorem does not imply that continuity constraints may be ignored during the optimization, but does suggests other ways to simplify the computation [17]. There are no similar simplifications for higher order approximations.

Another observation is based on the computation of the matrix of the second derivatives. It is well known that if that matrix is positive definite, then a point where all the first partial derivatives are zero is a minimum. An important question in many optimization problems is the sensitivity of the cost function to deviations from the optimal location. When the optimum is found as a point where the first partials are zero, then the sensitivity is less than when the optimum is found on the boundary of a region of constraints (for example, in linear programming problems) because the larger the partial derivatives are the more the cost function varies. When we compare two optima, both found through zeros of the first derivatives, then the second derivatives provide information about the sensitivity. The larger they are, the greater the variation of the first derivatives. For the $y = f(x)$ case the following result holds.

THEOREM B2. [16]. *The matrix M of the second derivatives is tridiagonal. If we neglect terms that are proportional to the square of pointwise errors divided by the length of subintervals, then M reduces to a diagonal form with elements given by*

$$M_{ii} \approx 2e_l(x_i)e_l'(x) - 2e_r(x)e_r'(x), \tag{B1}$$

where e_l stands for the error computed with the approximation to the left of the breakpoint x_i and e_r for the approximation computed to the right. The primed terms denote derivatives with respect to x.

If the approximation is continuous, then the errors from both the left and the right are equal. Thus for a polygonal approximation the terms are proportional to the product of the pointwise error times the difference in the slopes. This leads to the not surprising conclusion that the ISE is more sensitive to the location of the knots where the slope changes by a greater amount. If we insist on slope continuity, then the right-hand side of eq. (B1) becomes zero. Then M contains only terms proportional to the square of pointwise errors divided by interval lengths. For any reasonable approximation the ratio of the pointwise error over the length of a subinterval is expected to be small compared to the difference in slopes at a vertex of a polygon. Thus the value of ISE will be less sensitive to the location of the knots. The practical conclusion of all this is that *the error of approximation is less sensitive to the knot location for higher order approximations than it is for polygons.* Therefore suboptimal solutions are expected to be not much worse than optimal ones and we are justified in ignoring the exact knot location problem for the conics, as long as we can obtain estimates of their approximate location.

APPENDIX C. ON THE DISTRIBUTION OF THE SIZE OF POLYGON VERTICES

For font "decompiling" the selection of a_1 is facilitated by an experimental observation about the size of the angles actually found in polygonal approximations of the outlines of characters from various fonts. Table I summarizes these results. This table was obtained by finding the polygonal approximations for each of the 120 characters of each font with the method mentioned in Section 5. About two-thirds of the vertices in all fonts fall in the range 150–210°. Direct display of the results showed that indeed these were vertices placed in the curved parts of

Table I. Distribution of Polygonal Angles for Various Fonts

	Percent of vertices in each angle range					
Font	±30°	30–90°	90–150°	150–210°	210–270°	270–330°
Times Roman	0.06	5.03	18.71	65.10	10.22	0.88
Times Italic	0.26	5.40	20.20	63.79	9.61	0.74
Helvetica	0.32	7.47	12.03	72.28	6.91	0.99
Spartan	0.00	7.34	13.32	69.88	8.13	1.32

Table II. Distributions of Polygonal Angles in Subranges[a]

	Distribution in subranges				
Font	90–110°	110–120°	120–130°	130–140°	140–150°
Times Roman	304	38	37	82	178
Times Italic	261	50	37	95	238
Helvetica	243	14	12	8	26
Spartan	182	26	13	28	54
Times Bold	310	51	29	63	169
Greek	306	36	46	100	240

	Distribution in subranges				
Font	210–220°	220–230°	230–240°	240–250°	250–270°
Times Roman	149	40	16	16	128
Times Italic	131	58	17	36	82
Helvetica	19	5	12	8	130
Spartan	26	15	16	17	111
Times Bold	152	38	17	24	144
Greek	176	56	19	26	152

[a]Absolute Counts.

Table III. Sparsest 10° Range (Percent of total in parentheses)

Font	Between 90 and 180°	Between 180 and 270°
Times Roman	120–130 (1.08)	230–240 and 240–250 (0.47)
Times Italic	120–130 (1.10)	230–240 (0.50)
Helvetica	130–140 (0.32)	220–230 (0.20)
Spartan	120–130 (0.57)	220–230 (0.66)

the character outlines. A more detailed histogram in the 90°–150° and 210°–270° reveals a bimodality, as shown in Tables II and III.

If the distribution of angles were uniform, each 10° interval would contain about 3 percent of the angles. Because of the sparsity of the distribution around 130° and 230° it is reasonable to assume that the vertices found by the polygonal approximation have two origins. Some correspond to true vertices of the contour (around 90° or 270°) and others to approximations of the curved parts. Furthermore the low density of the distribution in the in-between region suggests that *we need not by very accurate in the selection* of the threshold angle a_1. In the examples given in Section 9, we have selected a_1 equal to 70° so that angles less than 110° or greater than 250° are classified as hard.

ACKNOWLEDGMENTS

I want to thank Lorinda Cherry, Ed Gilbert, Doug McIlroy, and Chris van Wyk for many helpful comments on earlier drafts of this paper, and also Robin Forrest for many comments and for bringing to my attention many of the references on the rational parametric form.

REFERENCES

1. BARNHILL, R. E., AND RIESENFELD, R. F., Eds. *Computer Aided Geometric Design.* Academic Press, New York, 1974.
2. BOOKSTEIN, F. L. Fitting conic sections to scattered data. *Comput. Gr. Image Process. 9* (1979), 56–71.
3. CHASEN, S. H. *Geometric Principles and Procedures for Computer Graphic Applications.* Prentice-Hall, Englewood Cliffs, N. J., 1978.
4. COHEN, E., LYCHE, T., AND RIESENFELD, R. Discrete B-splines and subdivision techniques in computer-aided geometric design and computer graphics. *Comput. Gr. Image Process. 14* (1980), 87–111.
5. DE BOOR, C. *A Practical Guide to Splines.* Springer-Verlag, New York, 1978.
6. DIMSDALE, B. Convex cubic splines. *IBM J. Res. Dev. 22* (1978), 168–178.
7. FAUX, I. D., AND PRATT, M. J. *Computational Geometry for Design and Manufacture.* Horwood, Chichester, England, 1979.
8. FORREST, A. R. Curves and surfaces for computer-aided design. Ph.D. dissertation, Cambridge Univ., Cambridge, England, July 1968.
9. FORREST, A. R. Conic Sections. Draft of a book chapter.
10. ISAACSON, E., AND KELLER, H. B. *Analysis of Numerical Methods.* Wiley, New York, 1966.
11. KAJIYA, J., AND ULLNER, M. Filtering high quality text for display on raster scan devices. In *Proc. SIGGRAPH '81* (Dallas, Texas, August 3–7), ACM, New York, 1981.
12. KNUTH, D. E. TEX and METAFONT, *New Directions in Typesetting.* Digital Press and American Mathematical Society, Bedford, Mass., 1979.
13. LIAO, Y. Z. A two-stage method of fitting conic arcs and straight-line segments to digitized contours. In *Proc. IEEE Pattern Recognition and Image Processing Conference* (Dallas, Texas, 1981), pp. 224–229.
14. LOZOVER, O., AND PREISS, K. Automatic generation of a cubic B-spline representation for a general digitized curve. In *Eurographics 81.* J. L. Encarnacao, Ed., Elsevier North-Holland, New York, 1981, pp. 119–126.
15. MCCLURE, D. E. Nonlinear segmented function approximation and analysis of line patterns. *Q. Appl. Math. 33* (1975), 1–37.
16. PAVLIDIS, T. Optimal piecewise polynomial L_2 approximation of functions of one and two variables. *IEEE Trans. Comput. C-24* (1975), 98–102.
17. PAVLIDIS, T. Polygonal approximations by Newton's method. *IEEE Trans. Comput. C-26* (1977), 800–807.
18. PAVLIDIS, T. *Structural Pattern Recognition.* Springer-Verlag, New York, 1977.
19. PAVLIDIS, T. *Algorithms for Graphics and Image Processing.* Computer Science Press, Rockville, Md., 1982.
20. PAVLIDIS, T. Curve fitting as a pattern recognition problem. *Proc. 6th Int. Conf. Pattern Recognition* (Munich, Oct. 1982), IEEE Computer Society Press, Silver Spring, Md., pp. 853–859.
21. PAVLIDIS, T., AND ALI, F. A hierarchical syntactic shape analyzer. *IEEE Trans. Pattern Anal. Mach. Intell. 1* (1979), 2–9.
22. PAVLIDIS, T., AND CHERRY, L. L. Vector and arc encoding of grapics and text. In *Proc. 6th Int. Conf. Pattern Recognition* (Munich, Oct. 1982), IEEE Computer Society Press, Silver Spring, Md., pp. 610–613.
23. PLASS, M., AND STONE, M. Curve-fitting with piecewise parametric cubics. *Xerox PARC Tech. Rep.*, Xerox Palo Alto Research Center, Palo Alto, Calif. In Proc. *SIGGRAPH '83* (Detroit, July 25–29), ACM, New York.
24. REEVES, W. T., AND SERMER P. Efficient representation of curves in computer graphics. *Tech. Rep. 153/81,* Dept. of Computer Science, Univ. of Toronto, Toronto, Canada, June 1981.

25. RICE, J. R. *The Approximation of Functions*, vol. 2. Addison-Wesley, Reading, Mass., 1969.
26. RIESENFELD, R. Applications of B-spline approximation to geometric problems of computer-aided design. *Tech. Rep. UTEC-CSc-73-126*, Computer Science Dept., Univ. of Utah, Salt Lake City, Utah, March 1973.
27. RULE, J. T., AND COONS, S. A. *Grapics*. McGraw-Hill, New York, 1961.
28. SALMON, G. *Conic Sections*, 6th ed., Chelsea, New York, 1954.
29. SCOTT, C. A. *Projective Methods in Plane Analytical Geometry*, 3d ed., Chelsea, New York, 1961.
30. WARNOCK, J. E. The display of characters using gray level sample arrays. In *Proc. SIGGRAPH'80*, (Seattle, Wash., July 14–18, 1980), ACM, New York, 1980.
31. YAMAGUCHI, F. A new curve fitting method using a CRT computer display. *Comput. Gr. Image Process.* 7 (1978) 425–437.

Received August 1982; revised February 1983; accepted March 1983