Interactive interpolation and approximation by Bézier polynomials

A. R. Forrest

Computer-Aided Design Group, Computer Laboratory, University of Cambridge, Cambridge CB2 3QG

One of the main problems in computer-aided design is how to input shape information to the computer. The paper describes a method developed for the interactive interpolation and approximation of curves which has been found in practice to provide a natural interface between the mathematically unsophisticated user and the computer.

(Received May 1971)

1. Introduction

In *computational geometry* (the computer representation, analysis and synthesis of *shape* information*) interpolation and approximation techniques are often used for both curves and surfaces. However, the properties of 'shape' are different from the properties of functions and the well-known techniques of functional interpolation and approximation are not necessarily suitable. Shape is, for example, an axis independent phenomenon and vector valued parametric curves and surfaces are often adopted for this reason alone. Indeed, parametric notation will be used throughout this paper, both for convenience and to emphasise that the paper is concerned with the approximation of shapes rather than the approximation of functions. Sometimes, but by no means always, the functional form of a particular shape is known—a curve may be a circular arc. In such cases conventional fitting techniques may be used, but in general what is required is an acceptably close fit (to within a given tolerance) which maintains the *character* of the curve or surface and is *smooth* or *fair*.

This paper considers the *interactive design* of curves, *ab initio*, and the *interactive approximation* of curves. Just as mathematical techniques are modified or developed to cope with shape, so they must be amended when interaction between a computer and a human operator is involved. A particularly elegant technique has been developed by Bézier (1968a, 1968b, 1970) of Régie Renault. This paper develops the mathematical properties of Bézier's methods for interactive approximation.

In the car industry the problem is to find a mathematical representation for a stylist's clay model or sketch. The data has two basic kinds of error: measurement error and error due to the stylist and the inherent properties of his working medium. The former are to some extent predictable but the latter errors are only really apparent to the stylist himself. There is no definable 'best' fit; rather the goodness of fit depends on human judgement. It is thus logical to use an interactive technique because no fully automatic technique can be expected to distinguish between the intent of the stylist and his errors and will at the best employ *ad hoc* procedures. **Fig. 1** shows the kind of curve data which might be encountered in the car industry and acceptable and unacceptable (in the author's opinion) solutions. In order to be useful the interactive procedure must be easy to apply and a user should not need to know the mathematical principles involved. At Renault, data from a small clay model, or a hand sketched curve, is plotted, full size, on a drafting machine. The stylist then estimates graphically the parameters of an approximating curve which is then drawn by the machine. Three-dimensional curves are approximated in two plane projections. An acceptable approximation is usually achieved in a few iterations by adjusting the curve parameters. In some cases, no doubt, the stylist will do some redesigning as well as fitting. As smoothness is of paramount importance the system in effect gives the designer a 'perfect' medium in which to work rather than an imperfect one such as clay because the shapes which are created are basically smooth and irregularities must deliberately be designed. It is much easier to create a 'bump' in a curve than to remove an unintended 'bump' caused by bad data.

2. The basic principle

A curve segment, in Bézier's method, is defined by a polygon, two of whose vertices are the end points of the arc, **Fig. 2.** For a curve which is an nth degree polynomial or which is expressed as a linear combination of $(n + 1)$ linearly independent functions of the parameter the polygon has $(n + 1)$ vertices. Let the vertices be denoted by the vectors $\vec{P}(i)$, $0 \le i \le n$. Then $\vec{P}(0)$ is the start point of the arc and $\vec{P}(n)$ is the final point. (We shall employ the notation $\vec{P}(t)$ to denote the N component vector in the real Cartesian space R_N. Usually, $N = 2$ or 3.)

The Bézier curve defined by the polygon will *not* in general pass through the vertices other than $\vec{P}(0)$ and $\vec{P}(1)$ but will have its derivatives at $\vec{P}(0)$ and $\vec{P}(n)$ defined by the vertices. Thus at $\vec{P}(0)$ the curve is tangent to the first side of the polygon and at

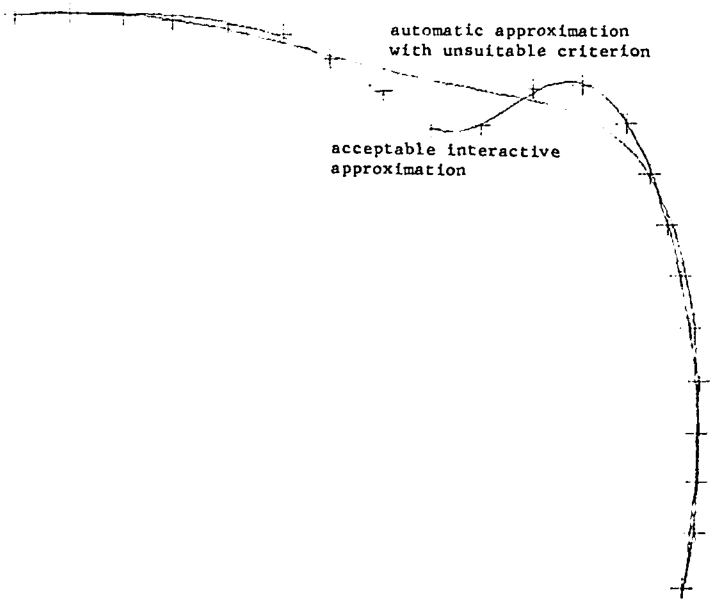


Fig. 1

*The term computational geometry has been used by Minsky and Papert (1969) as a pseudonym for pattern recognition. We employ here a less restricted definition.

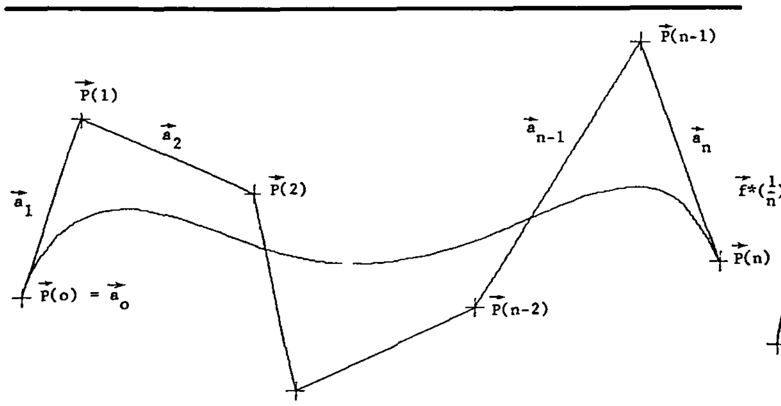
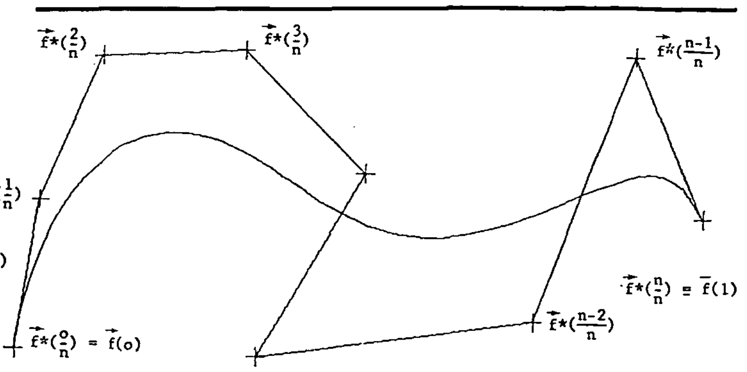


Fig. 2. **Bézier's notation**



$\vec{f}*(\frac{2}{n})$
$\vec{f}*(\frac{3}{n})$
$\vec{f}*(\frac{n-1}{n})$
$\vec{f}*(\frac{1}{n})$
$\vec{f}*(\frac{n}{n}) = \vec{f}(1)$
$\vec{f}*(\frac{n-2}{n})$
$\vec{f}*(\frac{o}{n}) = \vec{f}(o)$

Fig. 3. **Generalised notation**

$\vec{P}(n)$ to the last side of the polygon; in general the mth derivative at $\vec{P}(0)$ is dependent on the vertices $\vec{P}(0)$ to $\vec{P}(m)$ and the mth derivative at $\vec{P}(n)$ is dependent on the vertices $\vec{P}(n - m)$ to $\vec{P}(n)$. There are two basic ways of defining the curve—in terms of the polygon vertices, and in terms of the polygon sides. We shall first consider the latter as it is the form used by Bézier (1968a, 1968b, 1970).

Let the sides of the defining polygon be denoted by the vectors $\vec{a}(i)$ where

$$\vec{a}(i) = \vec{P}(i) - \vec{P}(i-1) \quad 1 \leq i \leq n \tag{2.1}$$

and let

$$\vec{a}(0) = \vec{P}(0) .$$

Then the Bézier curve is given by:

$$\vec{Q}(t) = \sum_{i=0}^{n} \vec{a}(i) . \underset{n}{f_i}(t) \tag{2.2}$$

where t is the parameter and the functions $\underset{n}{f_i}(t)$ have properties consistent with the end conditions previously mentioned. As we shall be adopting a slightly different formulation we shall not go into details of these properties (see Bézier 1968a, 1968b, 1970). In all cases $\underset{n}{f_0}(t) \equiv 1$.

In the case where the functions $\underset{n}{f_i}(t)$ are polynomials of degree n, and for $t \in [0, 1]$, we have:

$$\underset{n}{f_i}(t) = \frac{-(-t)^i}{(i-1)!} \cdot \frac{d^{i-1}\,\Phi_n(t)}{dt^{i-1}} \quad 1 \leq i \leq n \tag{2.3}$$

where

$$\Phi_n(t) = \frac{1 - (1-t)^n}{t} \tag{2.4}$$

Hence

$$\vec{Q}(0) \equiv \vec{P}(0) \quad \text{and} \quad \vec{Q}(1) \equiv \vec{P}(n)$$

3. The general Bézier polynomial

There are two reasons for developing a formulation of the Bézier curve in terms of polygon vertices rather than polygon sides. Firstly the formulation becomes more elegant, and secondly as a general principle it is better to program in terms of absolute vectors rather than a chain of relative vectors, irrespective of the particular user interface, when transformations such as rotation are to be applied to the vectors because rounding errors do not have the cumulative effect which sometimes give rise to poor drawings. This can be particularly noticeable when transformations are performed, for reasons of speed, in a small satellite graphics computer.

Let $\vec{f}(t)$ denote the curve to be approximated and $\vec{g}(t) =$ $\vec{Bé}_n(\vec{f}; t)$ denote an nth degree Bézier polynomial approximation of $\vec{f}(t)$, for $t \in [0, 1]$. Let the vertices of the polygon be denoted by $\vec{f}*\left(\frac{i}{n}\right)$, $0 \leq i \leq n$, **Fig. 3.** The asterisk is used to indicate that the vertices do not lie on the curve being approximated but are related, in some fashion to be specified, to that curve. Then:

$$\vec{Bé}_n(\vec{f}; t) = \sum_{i=0}^{n} \vec{f}*\left(\frac{i}{n}\right) . J_{n,i}(t) \tag{3.1}$$

$$= \sum_{i=0}^{n} \vec{f}*\left(\frac{i}{n}\right) \cdot \binom{n}{i} t^i (1-t)^{n-i} \tag{3.2}$$

$J_{n,i}(t)$ denotes the interpolation function associated with the ith vertex of an n-sided polygon. (The author uses $I_{r,i}(t)$ to denote the interpolation function dual to the rth derivative at the point $t = t_i$ *on* the curve (Forrest, 1970a); $J_{n,i}(t)$ is used to emphasise that these interpolation functions are dual to vertices which do *not* in general lie on the curve.)

Note that

$$\vec{f}*\left(\frac{0}{n}\right) \equiv \vec{f}(0)$$

and

$$\vec{f}*\left(\frac{n}{n}\right) \equiv \vec{f}(1)$$

The interpolation functions $J_{n,i}(t)$ have the following properties:

$$J_{n,0}(0) = 1, J^p_{n,0}(1) = 0 \text{ for } 0 \leq p < n \tag{3.3}$$

$$J_{n,n}(1) = 1, J^q_{n,n}(0) = 0 \text{ for } 0 \leq q < n \tag{3.4}$$

where the indices denote derivatives with respect to t, and for $1 \leq i \leq n - 1$:

$$J^p_{n,i}(0) = 0, \quad 0 \leq p < i \tag{3.5}$$

$$J^q_{n,i}(1) = 0, \quad 0 \leq q < (n-i) \tag{3.6}$$

$$J^i_{n,i}(0) = (-1)^i\, J^i_{n,0}(0) = \frac{n!}{(n-i)!} \tag{3.7}$$

$$J^{n-i}_{n,i}(1) = (-1)^{n-i}\, J^{n-i}_{n,n}(1) = (-1)^{n-i} \frac{n!}{(n-i)!} \tag{3.8}$$

In addition

$$J^1_{n,i}\left(\frac{i}{n}\right) = 0, \quad 1 \leq i \leq (n-1) \tag{3.9}$$

i.e. the interpolation functions have maximum values for $t \in [0, 1]$ at $t = i/n$. The maximum value is:

$$J_{n,i}\left(\frac{i}{n}\right) = \binom{n}{i} \frac{i^i (n-i)^{n-i}}{n^n} \tag{3.10}$$

Thus the vertex $\vec{f}^*(i/n)$ has its maximum influence on the shape of the curve $B\acute{e}_n(\vec{f}; t)$ at the point $t = i/n$.

The Bézier polynomials, in parametric form, are axis independent, because the polynomial interpolation functions obey the Cauchy relationship:

$$\sum_{i=0}^{n} J_{n,i}(t) = 1 \tag{3.11}$$

(since the successive polynomial interpolation functions $J_{n,i}(t)$ are terms in the binomial expansion of $[(1 - t) + t]^n$).

Because of the binomial form of the interpolation functions, there are symmetry relationships between the $J_{n,i}(t)$ of the form:

$$J_{n,i}(t) \equiv J_{n,n-i}(1 - t) \tag{3.12}$$

Writing $B\acute{e}_n(\vec{f}; t) = \vec{g}(t)$ for the sake of brevity, the end derivatives of the Bézier polynomial in terms of the polygon vertices are:

$$\vec{g}^p(0) = \frac{n!}{(n-p)!} \sum_{i=0}^{p} (-1)^{p-i} \binom{p}{i} \vec{f}^* \left(\frac{i}{n}\right) \tag{3.13}$$

$$\vec{g}^q(1) = \frac{n!}{(n-q)!} \sum_{i=0}^{q} (-1)^{i} \binom{q}{i} \vec{f}^* \left(\frac{n-i}{n}\right) \tag{3.14}$$

Conversely, in terms of end derivatives, we have:

$$\vec{f}^* \left(\frac{i}{n}\right) = \sum_{k=0}^{i} \binom{i}{k} \frac{(n-k)!}{n!} \vec{g}^k(0) \tag{3.15}$$

$$= \sum_{k=0}^{n-i} (-1)^k \binom{n-i}{k} \frac{(n-k)!}{n!} \vec{g}^k(1) \tag{3.16}$$

These relationships are useful when Bézier polynomials are to be manipulated when joined piecewise; their derivation is obvious.

For the general range $[a, b]$ the Bézier polynomial of degree n may be written:

$$B\acute{e}_n(\vec{f}; t) = \sum_{i=0}^{n} \vec{f}^* \left(\frac{(n-i)a + ib}{n}\right) \binom{n}{i} \frac{(t-a)^i (b-t)^{n-i}}{(b-a)^n} \tag{3.17}$$

For computational reasons it is often convenient to use the power series expansion of the Bézier polynomial, particularly as the coefficients of the power series can be obtained from the polygon vertices by a difference technique

$$B\acute{e}_n(\vec{f}; t) = \sum_{i=0}^{n} \vec{C}_i \,.\, t^i \tag{3.18}$$

where

$$\vec{C}_i = \binom{n}{i} \sum_{k=0}^{i} (-1)^{i+k} \binom{i}{k} \vec{f}^* \left(\frac{k}{n}\right) \tag{3.19}$$

i.e.

$$\vec{C}_i = \binom{n}{i} \nabla_i^i \left\{ \vec{f}^* \left(\frac{i}{n}\right) \right\} \tag{3.20}$$

where ∇_i is the backwards difference operator

$$\nabla_i \left\{ \vec{f}^* \left(\frac{i}{n}\right) \right\} = \vec{f}^* \left(\frac{i}{n}\right) - \vec{f}^* \left(\frac{i-1}{n}\right) \tag{3.21}$$

We assume $\vec{f}^* \left(\frac{i}{n}\right) \equiv 0$ for $i < 0$.

4. Comparisons with the Bernstein polynomial

The Bézier polynomial approximation bears a striking resemblance to the corresponding nth degree Bernstein polynomial (Davis, 1963):

$$Bn(\vec{f}; t) = \sum_{i=0}^{n} \vec{f} \left(\frac{i}{n}\right) . \binom{n}{i} t^i (1-t)^{n-i} \tag{4.1}$$

where the $(n + 1)$ points $\vec{f}(i/n)$ are equally distributed, with respect to the independent variable t, along the arc of the curve being approximated. Bernstein polynomials are normally mentioned in interpolation theory in connection with the proof of the Weierstrass Approximation Theorem (Davis, 1963); $Bn(\vec{f}; t)$ will provide a uniform approximation to $\vec{f}(t)$, for a sufficiently large n, provided $\vec{f}(t)$ is C^0 continuous (continuous only in position), and will converge uniformly to $\vec{f}(t)$ and its derivatives as $n \to \infty$. In particular if $\vec{f}(t)$ is a polynomial of degree m, $Bn(\vec{f}; t)$ will *not* coincide with $\vec{f}(t)$ for $n = m$, but only for $n = \infty$.

It will be seen that the Bernstein polynomial is in fact defined by a polygon whose vertices lie *on* the curve being approximated. In contrast to the Bernstein polynomial we must assume C^n continuity (continuous up to and including the nth derivative) for the curve $\vec{f}(t)$ if we are to derive the polygon vertices of an nth degree Bézier polynomial from derivatives of order up to n of $\vec{f}(t)$. This does, however, ensure that the convergence process is fast for such curves, and in particular that $B\acute{e}_n(\vec{f}; t) \equiv \vec{f}(t)$ if $\vec{f}(t)$ is an mth degree polynomial and $n > m$. However, it would be wrong to think solely of defining the polygon vertices in terms of the end derivatives. Where the Bézier polynomials are used interactively to approximate non-analytic curves such as car body lines (as at Renault) polygon vertices may be chosen by eye, and convergence becomes a matter of opinion and taste; a similar approach can be taken to the interactive approximation of analytic functions, but in this case a measure of error is available.

If, on the other hand, we use Bézier methods to design a curve *ab initio* then in a sense we are approximating the polygon or a curve (e.g. the polygon itself) through the vertices (Gordon, 1971). In such circumstances the Bézier method is simply a vector valued extension of the Bernstein approximation. Thus when discussing the convergence or variation diminishing properties (Gordon, 1971) of the Bézier method, we must take care in distinguishing what we are approximating (the polygon, or some other curve) and the method by which the vertices are selected (in terms of end derivatives or by interactive trial and error).

5. The general Bézier curve

As Bézier (1968a, 1968b, 1970) has pointed out, his interpolation method is not confined to polynomials. In general:

$$B\acute{e}_n(\vec{f}; t) = \sum_{i=0}^{n} \vec{f}^* \left(\frac{i}{n}\right) . J_{n,i}(t) \tag{5.1}$$

where the $J_{n,i}(t)$ are the $(n + 1)$ interpolation functions which may be formed by linear combinations of $(n + 1)$ linearly independent functions of the variable t. The interpolation functions $J_{n,0}(t)$ and $J_{n,n}(t)$ may be generated directly from the conditions:

$$J_{n,0}(0) = 1, \quad J_{n,0}^p(1) = 0, \quad 0 \leq p < n \tag{5.2}$$

$$J_{n,n}(1) = 1, \quad J^q_{n,n}(0) = 0, \quad 0 \le q < n \tag{5.3}$$

The remaining interpolation functions may then be derived from:

$$J^p_{n,i}(0) = 0, \quad 0 \le p < i \tag{5.4}$$

$$J^q_{n,i}(1) = 0, \quad 0 \le q < (n - i) \tag{5.5}$$

and *either*

$$J^i_{n,i}(0) = (-1)^i . J^i_{n,0}(0) \tag{5.6}$$

or

$$J^{n-i}_{n,i}(1) = (-1)^{n-i} . J^{n-i}_{n,n}(1) \tag{5.7}$$

For axis independence

$$\sum_{i=0}^{n} J_{n,i}(t) \equiv 1 \tag{5.8}$$

E.g. for the basis functions:

$$\left[\cos^2 \frac{\pi}{2} t \sin^2 \frac{\pi}{2} t \cos \frac{\pi}{2} t \sin \frac{\pi}{2} t\right]$$

$$J_{3,0}(t) = \cos^2 \frac{\pi}{2} t + 2 \sin^2 \frac{\pi}{2} t - 2 \sin \frac{\pi}{2} t$$

from (5.2) and

$$J_{3,3}(t) = 2 \cos^2 \frac{\pi}{2} t + \sin^2 \frac{\pi}{2} t - 2 \cos \frac{\pi}{2} t$$

from (5.3)

Therefore:

$$J^1_{3,0}(0) = -\pi \quad J^2_{3,0}(0) = \frac{\pi^2}{2}$$

$$J^1_{3,3}(1) = \pi \quad J^2_{3,3}(1) = \frac{\pi^2}{2}$$

whence

$$J_{3,1}(t) = -2 \sin^2 \frac{\pi}{2} t + 2 \sin \frac{\pi}{2} t$$

$$J_{3,2}(t) = -2 \cos^2 \frac{\pi}{2} t + 2 \cos \frac{\pi}{2} t$$

with maxima at $t = \frac{1}{3}$ for $J_{3,1}(t)$ and at $t = \frac{2}{3}$ for $J_{3,2}(t)$.

Note

$$\sum_{i=0}^{3} J_{3,i}(t) = 1$$

Although the use of bases other than polynomials has not been investigated to any extent, a different extension of Bézier's work has proved fruitful. The use of rational cubic curves for computer-aided design has been suggested by the author and others (Coons, 1967; Forrest, 1968; Lee, 1969) and an elegant form of these curves can be derived when they are defined in terms of a three sided polygon (Forrest, 1970b). A particularly welcome feature of the form is that the factors controlling the shape of the curve are not affected by the parametrisation (i.e. rate of traverse of the arc with respect to the parameter t), a feature not shared by some other formulations of the curve. **Fig. 4** shows various Bézier curves with similar polygons but different bases.

6. Curve splitting and changing of order of polygons

When working interactively it is often found that a particular curve segment is not sufficiently powerful or flexible (i.e. does not have sufficient degrees of freedom) to adopt a desired shape. There are two possible ways to resolve this difficulty: the segment may be split into two or more segments, retaining

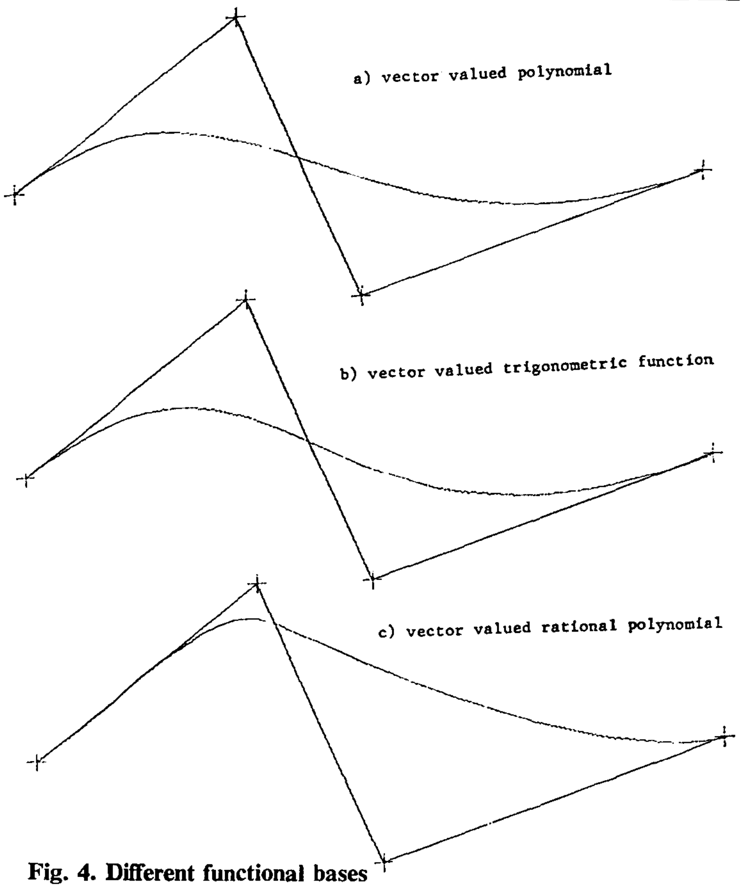

a) vector valued polynomial
b) vector valued trigonometric function
c) vector valued rational polynomial

Fig. 4. Different functional bases

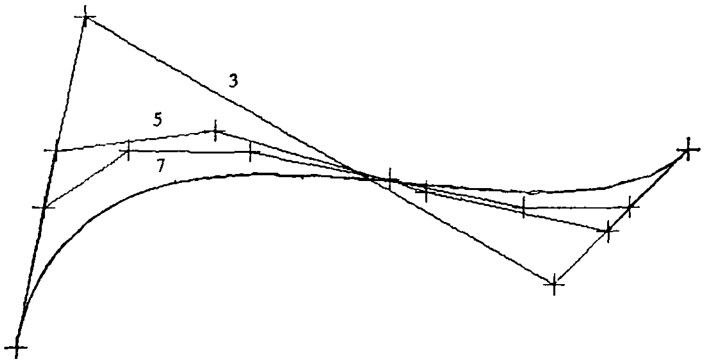


Fig. 5. Increasing polygon order

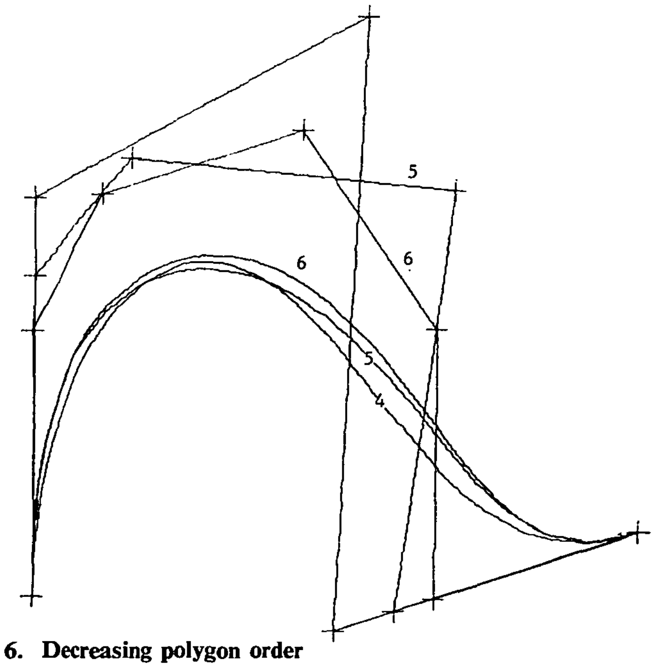


Fig. 6. Decreasing polygon order

initially the same shape, or a higher order curve segment, again of the same shape, may be substituted. Curve splitting is simple, mathematically, and may be advantageous where it is desired to use only curves of up to a certain order. Increasing the order of a curve whilst retaining the same shape is slightly more involved. It is probably simplest to use the following easily proved procedure (Talbot, 1971) to increase the order from n to $n + 1$:

$$\vec{f^*}\left(\frac{i}{n+1}\right) = \frac{1}{n+1}\left[i\vec{f^*}\left(\frac{i-1}{n}\right) + (n+1-i)\vec{f^*}\left(\frac{i}{n}\right)\right]$$
$$0 \le i \le n+1 \qquad (6.1)$$

Fig. 5 shows an example of increasing the order of Bézier polynomials. Note how the higher order polygons converge towards the curve.

It might sometimes be desired to attempt to decrease the order of the curve from n to $(n - 1)$ in order to reduce complexity. This is a rather more tricky procedure as some information has to be discarded. If n is odd, then in effect two vertices are merged, but if n is even then the middle vertex is eliminated. In general, the reduction of degree of a curve will cause a change in shape, but the procedure outlined here is such that, in the case of polynomials at least, if the polynomial described by an n-sided polygon is in fact a polynomial of degree less than n then the reduction of the number of sides from n to $(n - 1)$ will not change the shape of the curve.

The two cases n even and n odd differ slightly. In both cases we use the formulae:

$$\vec{f^*}\left(\frac{i}{n-1}\right) = \frac{1}{(n-i)}\left[n\vec{f^*}\left(\frac{i}{n}\right) - i\vec{f^*}\left(\frac{i-1}{n-1}\right)\right] \qquad (6.2)$$

and

$$\vec{f^*}\left(\frac{n-i-1}{n-1}\right) = \frac{1}{(n-i)}\left[n\vec{f^*}\left(\frac{n-i}{n}\right) - i\vec{f^*}\left(\frac{n-i}{n-1}\right)\right] \qquad (6.3)$$

For n even, compute $\vec{f^*}\left(\frac{i}{n-1}\right)$ from (6.2) for $0 \le i \le \frac{n-2}{2}$

and $\vec{f^*}\left(\frac{n-i-1}{n-1}\right)$ from (6.3) for $0 \le i \le \frac{n-2}{2}$.

If the curve described by the n-sided polygon was in fact of degree $<n$, then:

$$\vec{f^*}\left(\frac{n/2}{n}\right) = \frac{1}{2}\left[\vec{f} * \left(\frac{(n-2)/2}{n-1}\right) + \vec{f^*}\left(\frac{n/2}{n-1}\right)\right] \qquad (6.4)$$

For n odd, compute $\vec{f^*}\left(\frac{i}{n-1}\right)$ from (6.2) for $0 \le i \le \frac{n-1}{2}$

and $\vec{f^*}\left(\frac{n-i-1}{n-1}\right)$ from (6.3) for $0 \le i \le \frac{n-3}{2}$. Set

$\vec{f^*}\left(\frac{\frac{n-1}{2}}{n-1}\right)$ equal to the mean of the two values obtained from

(6.2) and (6.3). If these values were identical then the original curve was of degree $<n$. **Fig. 6** shows the effects of decreasing the order of Bézier polynomials using this algorithm.

In computational geometry piecewise techniques are often appropriate. The conditions for derivative continuity between two Bézier polynomials may be obtained from the expressions for end derivatives. First and second derivative continuity can readily be assured by simple graphical constructions, but higher order continuity necessitates more complex constructions.

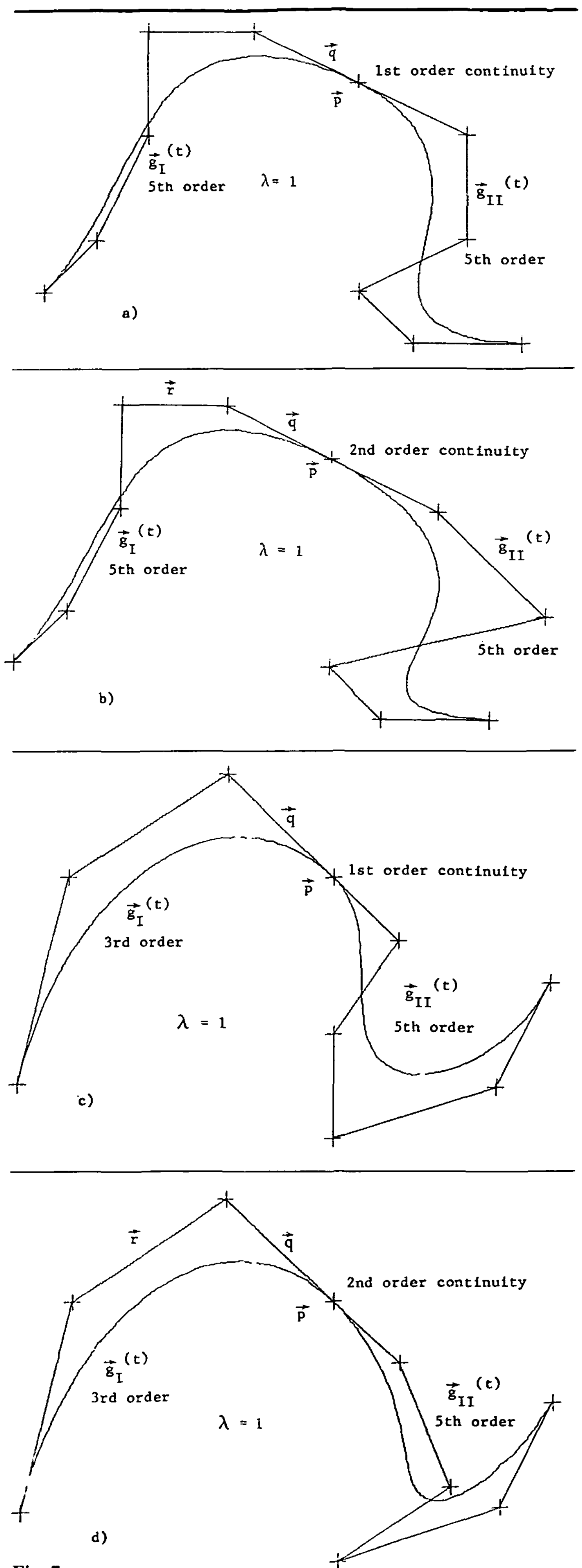


Fig. 7

Suppose

$$\vec{g}_I(t) = \sum_{i=0}^{m} \vec{f}_I^*\left(\frac{i}{m}\right)\cdot\binom{m}{i} t^i(1-t)^{m-}$$

and

$$\vec{g}_{II}(t) = \sum_{j=0}^{n} \vec{f}_{II}^*\left(\frac{j}{n}\right)\cdot\binom{n}{j} t^j(1-t)^{n-j}$$

are to be joined so that

$$\vec{g}_I(1) = \vec{f}_I^*(1) = \vec{g}_{II}(0) = \vec{f}_{II}^*(0) = \vec{p}$$

Let

$$\vec{g}_I(1) - \vec{f}_I^*\left(\frac{m-1}{m}\right) = \vec{q} \qquad (6.5)$$

and

$$\vec{f}_I^*\left(\frac{m-1}{m}\right) - \vec{f}_I^*\left(\frac{m-2}{m}\right) = \vec{r} \qquad (6.6)$$

Then for first order continuity, in the Cartesian sense,

$$\vec{g}_I'(1) \equiv \lambda \vec{g}_{II}'(0) \qquad (6.7)$$

where λ is a $+ve$ scalar constant.

For second order continuity, in the Cartesian sense,

$$\vec{g}_I''(1) \equiv \mu \vec{g}_{II}''(0) \qquad (6.8)$$

and for curvature continuity, $\mu = \lambda^2$.

Evaluating the end derivatives for $\vec{g}_I$ and $\vec{g}_{II}$ we can show: for first order continuity,

$$\vec{f}_{II}^*\left(\frac{1}{n}\right) = \vec{p} + \frac{\lambda m}{n}\vec{q} \qquad (6.9)$$

and in addition for curvature continuity,

$$\vec{f}_{II}^*\left(\frac{2}{n}\right) = \vec{p} + 2\frac{\lambda m}{n}\vec{q} + \lambda^2\frac{m(m-1)}{n(n-1)}(\vec{q} - \vec{r}) \qquad (6.10)$$

In the case $m = n$, $\lambda = 1$

$$\vec{f}_{II}^*\left(\frac{1}{n}\right) = \vec{p} + \vec{q} \qquad \text{(Fig. 7(a))}$$

and

$$\vec{f}_{II}^*\left(\frac{2}{n}\right) = \vec{p} + 3\vec{q} - \vec{r} \qquad \text{(Fig. 7(b))}$$

These relationships lead to simple graphical constructions for 1st and 2nd order continuity. Higher order relationships can be derived but are obviously more complex. In **Fig. 7(c)**, **Fig. 7(d)** $\lambda = 1$, $m = 3$, $n = 5$.

7. Interactive design and approximation by Bézier methods

It might be thought that because all the vertices of the polygon can be related to the end derivatives of the curve, there is little point in thus disguising what is essentially a Taylor or Hermite type approximation or interpolation. This is not so, for two reasons which are connected with computational geometry. In the first place, as remarked earlier, when the Bézier method is used for approximation the polygon vertices need not be related to the end derivatives of the curve to be approximated but may be selected manually (in some interactive system); they will, of course, control the end derivatives of the approximant. Secondly, for reasons of axis independence, convenient bounding of segments, ease of affine transformation, etc., parametric curves are often used in computational geometry. This introduces problems. In a Lagrange type of interpolation, parameter values have to be assigned to each of the interpolation

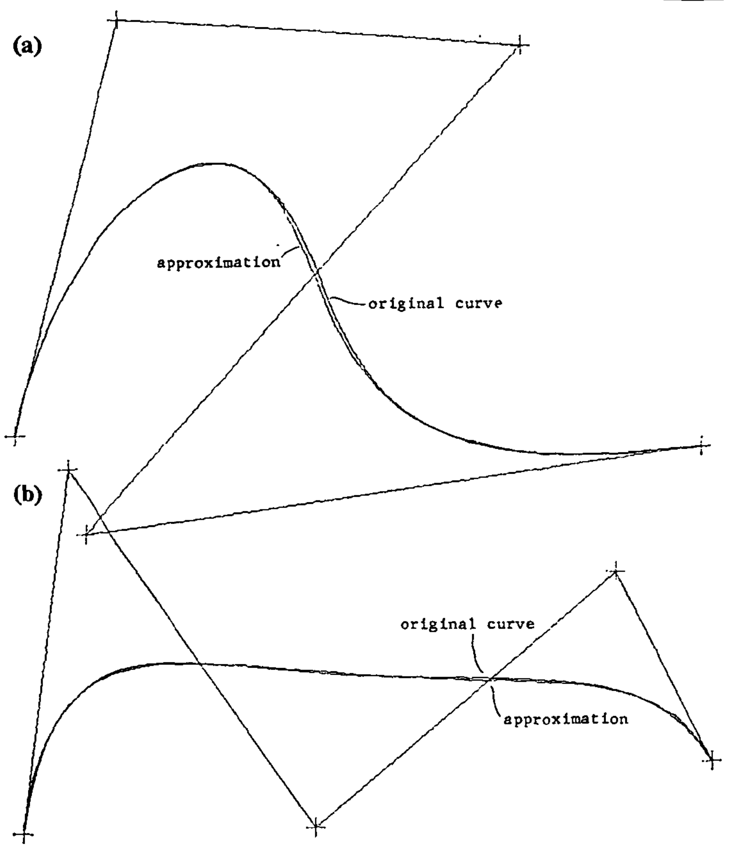


Fig. 8. (a) 6th order polynomial approximated by 5th order polynomial (b) 4th order polynomial approximated by 4th order polynomial

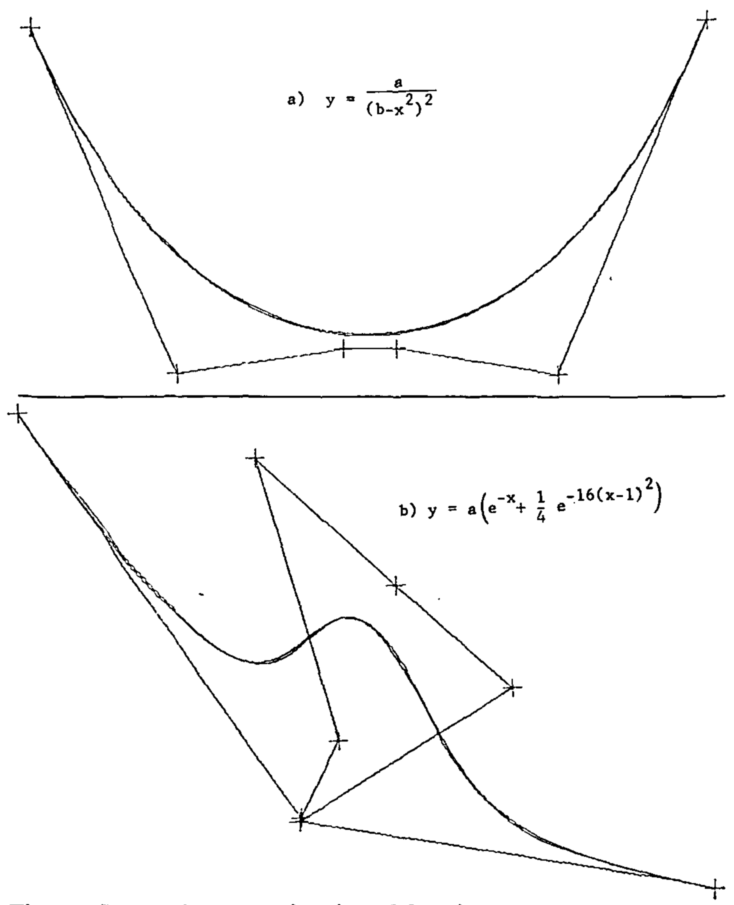

a) $y = \frac{a}{(b-x^2)^2}$
b) $y = a\left(e^{-x} + \frac{1}{4}e^{-16(x-1)^2}\right)$

Fig. 9. Interactive approximation of functions

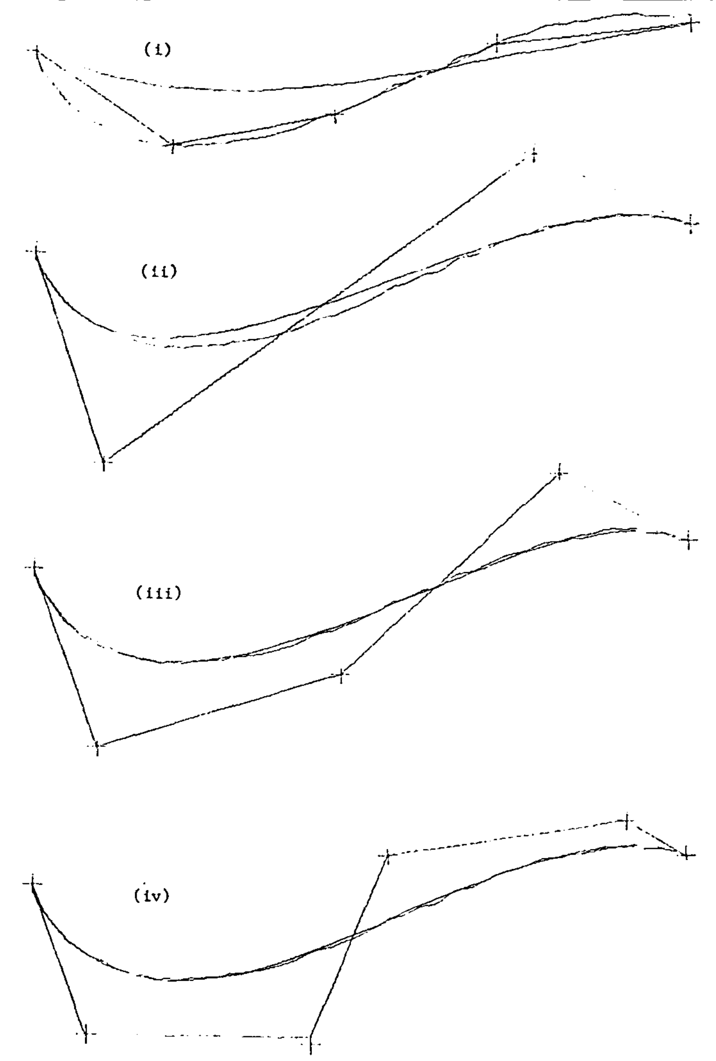


Fig. 10(a). Successive approximations to a hand-drawn curve

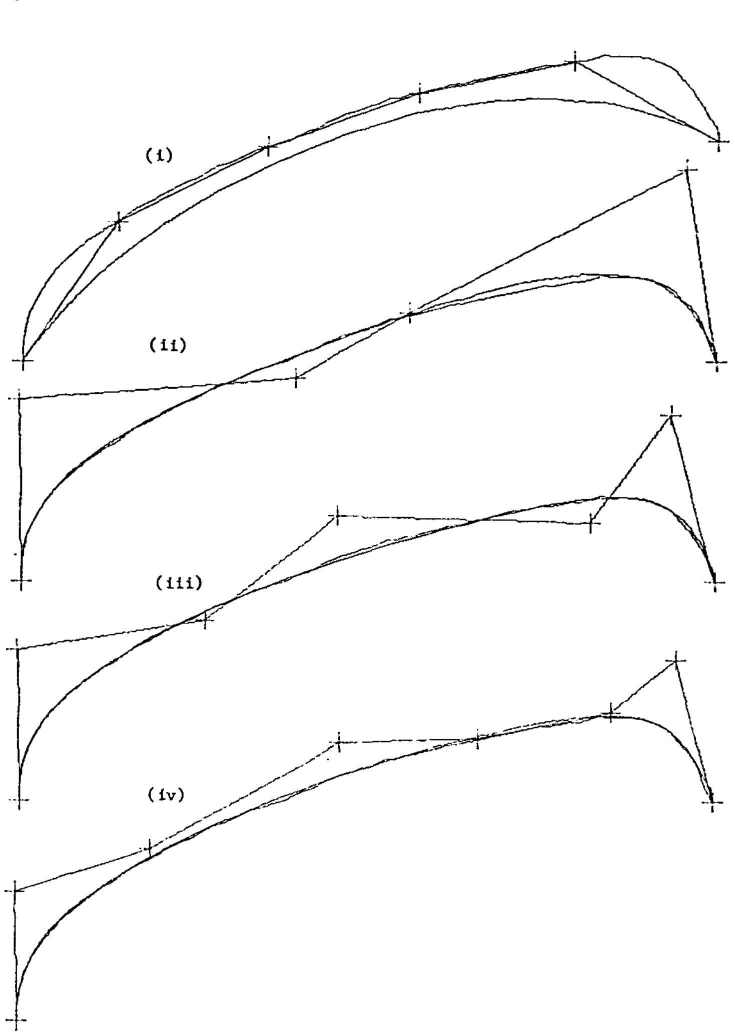


Fig. 10(b). Successive approximations to a hand-drawn curve

points and even to a mathematically sophisticated user this presents difficulties. The shape of the interpolated curve depends somewhat critically on the selected parametric values; automatic methods are not generally satisfactory. In Hermite or Taylor type interpolation, parametric derivative vectors must be defined. The difference between the tangent and the tangent vector is sometimes difficult for users to grasp, as is the exact role of the tangent vector's magnitude. The difficulties are compounded as the order of derivative increases. With Bézier's method both these problems are avoided. The user need not consider parametric derivatives or parameter values (although the polygon vertices are associated with the parameter values $t = i/n$, and control the end parametric derivatives) because the polygon vertices directly control the curve shape in a manner which may readily be appreciated. With very little experience a user can predict the shape of curve which can be generated by a particular polygon. **Figs. 8, 9 and 10** show examples of the use of Bézier polynomials for approximation. Fig. 8 illustrates the approximation of vector valued polynomials, Fig. 9 the approximation of other analytic functions (explicit, not vector valued, in this case) and Fig. 10, the successive approximation of hand drawn curves, starting from the vector valued Bernstein approximation.

Bézier has shown (in the English edition of his book, 1970) that the hodographs of polynomial Bézier curves can readily be computed using the sides of the curve polygon.

Given the curve

$$\vec{g}(t) = \sum_{i=0}^{n} \vec{f}^{*}\left(\frac{i}{n}\right)\binom{n}{i} t^{i}(1-t)^{n-i}$$

then the $(n-1)$th degree Bézier polynomial $h(t)$ whose successive polygon vertices are

$$\left[\vec{f}^{*}\left(\frac{k+1}{n}\right) - \vec{f}^{*}\left(\frac{k}{n}\right)\right], 0 \leq k \leq n-1 ,$$

is the hodograph of $\vec{g}(t)$, i.e.:

$$\vec{h}(t) = \sum_{k=0}^{n-1}\left[\vec{f}^{*}\left(\frac{k+1}{n}\right) - \vec{f}^{*}\left(\frac{k}{n}\right)\right]\binom{n-1}{k} t^{k}(1-t)^{n-1-k}$$

$$= \frac{1}{n}\vec{g}'(t) \qquad (7.1)$$

(The proof is simple and is omitted.)

The hodograph provides a convenient *graphical* method for detecting points with zero curvature and points of inflexion. If a vector can be drawn from the origin tangent to the hodograph $\vec{h}(t)$ then there is a point of zero curvature (**Fig. 11(a)**) or a point of inflexion (**Fig. 11(b)**) at the corresponding point on $\vec{g}(t)$. If the hodograph passes through the origin, there is a cusp at the corresponding point on $\vec{g}(t)$ (**Fig. 11(c)**). In practice the hodograph also provides a good indication of whether a curve under design is tending towards an undesirable shape (an unwanted flat or point of inflexion).

8. Practical experience in the use of Bézier methods

It is clearly evident from the decision of Régie Renault to increase the number of its computer-controlled drafting machines and numerically controlled machine tools that Bézier's method is well-suited to the types of problems encountered at Renault.

Experience at Cambridge University has been gained from four Bézier curve programs. Pankhurst's program (1970) is for the design of single curve segments of any order; polygon vertices are changed positionally by typed commands. The order of the curve may be increased or decreased, as outlined in Section 6, and a vertex may be constrained to move along one of the polygon sides which intersect at that vertex. Bézier's curves have also been implemented by Armit (1970) in his Multiobject system. Again the curve is manipulated by the typed commands of an interactive language and more powerful facilities are provided. In addition, curve splitting is implemented, and multiple curve segments may be constructed. Although intended primarily for design, the system can also be used for interactive approximation to point data. Talbot (1971) starts from an assembly of cubic spans with C^0 continuity, each span being obtained by a local least squares fit. Successive improvements to the curve shape with automatic creation and maintenance of C^1 continuity between spans are effected by moving polygon vertices.

All Figures in this paper were produced by a small experimental program written by the author for a PDP-7 computer. Input to the program is either a data tape describing the curve to be approximated (this method was used for Figs. 4(*b*), (*c*), 8 and 9) or curves hand drawn using light pen (as in Fig. 10). The program computes Bézier polynomials of 2nd to 9th degree. At present only single curve segments may be handled. The program is intended primarily for interactive approximation, and unlike the Pankhurst, Armit and Talbot programs, a visual comparison between the required curve, whether hand-drawn or analytic, and the approximation is always available. As an initial approximation the vertices of the polygon are equally distributed along the arc of the curve to be approximated; the initial approximation is thus a vector valued Bernstein polynomial (top curve in Figs. 10(*a*) and 10(*b*)). Thereafter interior vertices of the polygon may be repositioned by light pen with the curve distorting accordingly in real time.

There are several shortcomings of the author's program which must be borne in mind. Approximations are made visually and a discrepancy of less than 1 part in 500 cannot be detected. For example, most of the errors in Figs. 8 and 9 could not be detected on the c.r.t. display but were revealed when the display files were plotted. However, Bézier's system using a high resolution plotter overcomes the disadvantages of a low resolution device. It proves difficult to draw a really smooth curve with the light pen, and it would be reasonable to assume that much better approximations to hand-drawn smooth curves than those shown in Fig. 10 could be obtained. Surprisingly, no difficulty has been found in manipulating high order curves. A much better initial approximation would be the vector valued Lagrange polynomial through points spaced equidistantly on the curve, but this has not been implemented because of the limited arithmetic capabilities of the PDP-7. More experience is needed in using Bézier curves in a piecewise manner with automatic maintenance of continuity. The present program has already demonstrated to the author's satisfaction that successful approximations can readily be constructed by the mathematically uninitiated.

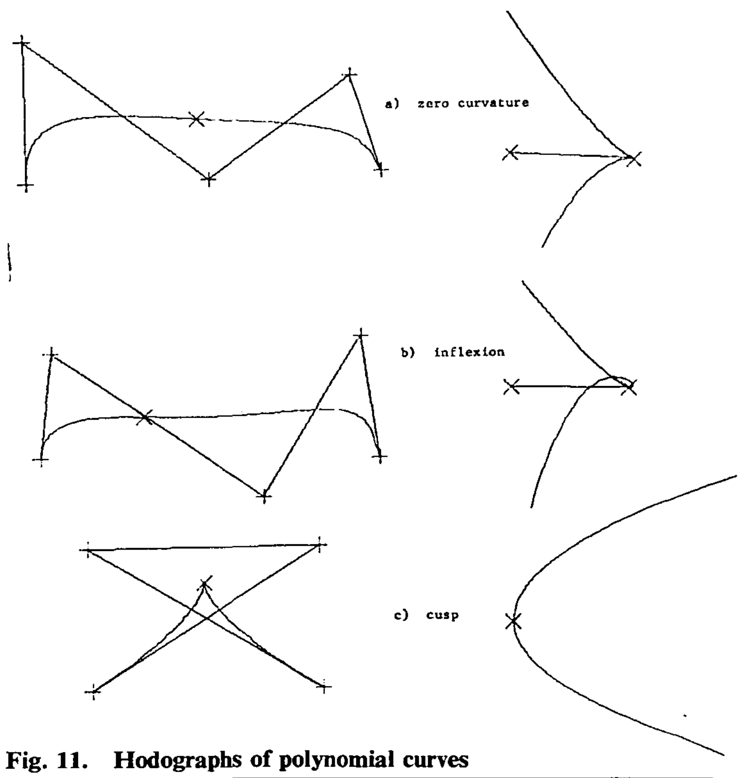


Fig. 11. Hodographs of polynomial curves

9. Extensions to surfaces

Bézier's method may obviously be extended to the interpolation and approximation of surfaces. There are three basic ways in which this may be achieved. The method actually employed

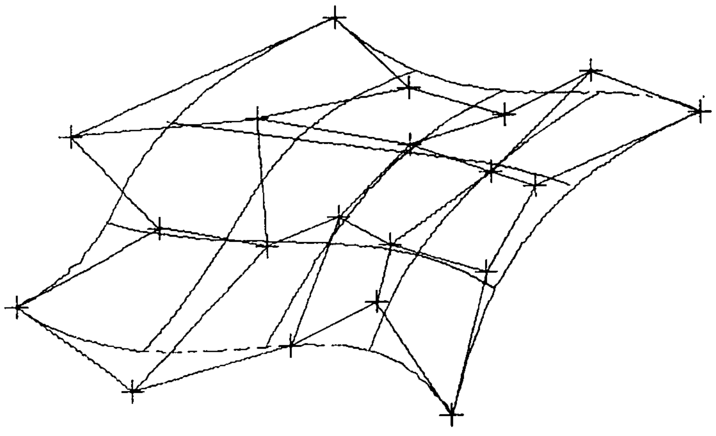


Fig. 12. Typical Bézier surface and net

by Bézier and others (Bézier 1968a, 1970, Armit, 1970, Sabin, 1969) is the product method in which the surface is defined by a grid of points, only four of which (the surface corner points) lie on the surface, **Fig. 12.** In the polynomial case,

$$\vec{g}(t, u) = \sum_{i=0}^{m} \sum_{j=0}^{n} \vec{f}^*\left(\frac{i}{m}, \frac{j}{n}\right)\binom{m}{i} t^i(1 - t)^{m-i} \binom{n}{j} u^j(1 - u)^{n-j} \qquad (9.1)$$

A surface may also be defined by Bézier interpolation to a single family of curves

$$\left[\vec{f}^*\left(\frac{i}{m}, u\right) \text{ or } \vec{f}^*\left(t, \frac{j}{n}\right)\right]$$

in a manner akin to lofting:

$$\vec{g}(t, u) = \sum_{i=0}^{m} \vec{f}^*\left(\frac{i}{m}, u\right)\binom{m}{i} t^i(1 - t)^{m-i} \qquad (9.2)$$

(in the polynomial case)

Thirdly, the surface may be defined in a manner analogous to that of Coons (Coons, 1967, Forrest, 1968) by two families of curves:

$$\vec{g}(t, u) = \sum_{i=0}^{m} \vec{f}^*\left(\frac{i}{m}, u\right)\binom{m}{i} t^i(1 - t)^{m-i}$$
$$+ \sum_{j=0}^{n} \vec{f}^*\left(t, \frac{j}{n}\right)\binom{n}{j} u^j(1 - u)^{n-j}$$
$$- \sum_{i=0}^{m} \sum_{j=0}^{n} \vec{f}^*\left(\frac{i}{m}, \frac{j}{n}\right)\binom{m}{i} t^i(1 - t)^{m-i}$$
$$\binom{n}{j} u^j(1 - u)^{n-j} \qquad (9.3)$$

Of these methods the first is probably the most suitable for interactive design.

10. Conclusions

The Bézier method is one of the most convenient methods devised for the interactive approximation of curves. Moreover, the method extends naturally to surface description and provides a simple way of controlling surface parameters. Surface twist vectors which have been stumbling blocks in interactive surface design (they are essential for doubly curved surfaces) are easily defined, in a disguised manner.

It is hoped that this paper will stimulate further work both in the field of shape description and in the field of interactive approximation.

11. Acknowledgements

The author is indebted to Professor Bézier, Professor W. J. Gordon of Syracuse University, and members of the Cambridge CAD Group for criticisms of a preliminary version of this paper.

References

Armit, A. P. (1970). Systems for Interactive Design of Three-dimensional Shapes, Cambridge University CAD Group Ph.D. Thesis, November, 1970.

Bézier, P. (1970). *Emploi des Machines à Commande Numérique*, Masson et Cie., Paris, 1970 (to be published, revised, in English translation by John Wiley, London and New York).

Bézier, P. (1968a). Procédé de Définition Numérique des Courbes et Surfaces Non Mathématiques; Système UNISURF, *Automatisme* 13, May 1968.

Bézier, P. (1968b). How Renault uses Numerical Control for Car Body Design and Tooling, *Society of Automative Engineers*, Paper SAE 680010.

Coons, S. A. (1967). Surfaces for Computer-Aided Design of Space Forms, *MIT MAC-TR*-41.

Davis, P. J. (1963). *Interpolation and Approximation*, Ginn-Blaisdell, New York.

Forrest, A. R. (1971). Computational Geometry, *Proc. Roy. Soc. Lond. A*. 321, pp. 187-195.

Forrest, A. R. (1970a). Coons' Surfaces and Multivariable Functional Interpolation. Cambridge University CAD Group, Document No. 38, July 1970.

Forrest, A. R. (1968). Curves and Surfaces for Computer-Aided Design. Cambridge University CAD Group Ph.D. Thesis, July 1968.

Forrest, A. R. (1970b). The Twisted Cubic Curve. Cambridge University CAD Group Document, No. 50, November 1970.

Gordon, W. J. (1971). Private communication, May 1971.

Lee, T. N. P. (1969). Three-Dimensional Curves and Surfaces for Rapid Computer Display, Ph.D. Thesis, Harvard University, April 1969.

Minsky, M., and Papert, S. (1969). *Perceptrons: An Introduction to Computational Geometry*, MIT Press, Cambridge, Mass.

Pankhurst, R. J. (1970). Notes on Experiments with Command Languages for Graphic Interaction. Cambridge University CAD Group Document No. 37, February 1970.

Sabin, M. A. (1969). A 16-Point Bicubic Formulation Suitable for Multipatch Surfaces. British Aircraft Corporation, Weybridge. VTO/MS/155, March 1969.

Talbot, J. E. (1971). Experiments towards interactive graphical design of motor bodies. Cambridge University CAD Group Document No. 56, April 1971.

Book review

Picture Language Machines, by S. Kaneff (editor), 1970; 425 pages. (*Academic Press Ltd.*, £4·50)

This book is concerned with pictures and with attempts to describe them in a precise language having meaning to a machine (the term 'machine' implying an appropriately programmed computer). These descriptions are formulated in such a way that the machine receives essentially the same information about a picture as would an observer viewing the picture.

Generally speaking, it is the structure of a picture which carries what is regarded as the important information. Thus the relative positions of black areas in a picture is more informative than a count of the areas. In the same way, the ordering of words in a sentence, together with a notion of how such words interrelate, enables the meaning of a sentence to be conveyed. It is suggested that pictures can be regarded as two-dimensional sentences and that studies of the structured description of natural languages, as has been proposed by Chomsky, can be adapted to the analysis of pictures.

A conference held in Canberra in 1966 drew together workers from various disciplines including Picture Languages, Graphical Communication, Interaction Systems, Pattern Recognition, Linguistics and Psychology in an attempt to explore aspects of these areas of study which would throw light on the development of picture language machines. As is stated in the Foreword by Dr. Max Clowes, the first four papers were intended to be primarily tutorial in character to provide an introduction to the notion of a picture language machine whereas the remaining 11 papers reported current research interests of the authors.

The volume taken as a whole provides a valuable review of this relatively new and fast developing subject and is particularly useful in a field where publications tend to be distributed over a wide range of journals. The important discussions during the conference are reported verbatim; these discussions add useful comments to the formal papers and are effective in putting over some of the atmosphere of the Canberra conference. The photographed typescript is beautifully printed and the illustrations are both clear and plentiful. Academic Press Ltd. have produced an attractive 425-page volume at the reasonable price of £4·50.

M. J. B. Duff (London)

Errata

In Algorithm 69, Trigonometric curve fitting to equally or unequally spaced data (this *Journal*, Volume 14, Number 2, pp. 213-214) there were a number of typing mistakes. These were all at the top of the second column of page 214.

Line 2 should read

'*we will have a full set of coefficients A;*'.

In line 11 '*A*[*mmax*, *mmax*]' should read '*a*[*mmax*, *mmax*]'.
In line 16 '*A*[*k*, *i*]' should read '*a*[*k*, *i*]'.
In line 17 '*A*[*k*, *k*];' should read '*a*[*k*, *k*]'.