Continuity conditions for spline curves

J. R. Manning

Shoe and Allied Trades Research Association, Satra House, Rockingham Road, Kettering, Northamptonshire

The conditions for continuity of direction and curvature at a knot in a vector-valued spline are derived. A method of normalising tangent vector magnitudes at knots is suggested. Many examples are displayed of closed spline curves constructed to pass through a series of knots, continuous in slope and curvature, with the segments normalised. Some of the figures represent shoe components; all the figures generated are acceptable, whether or not the defining knots are evenly spaced.

(Received February 1973)

1. Background to the problem

This paper is concerned with the problem of constructing a smooth aesthetically pleasing curve to pass through an ordered set of point vectors. A general solution is obtained for vectors in N-space, although the aesthetic criterion is only meaningful in the cases $N = 2$ and $N = 3$.

The technique has applications in the field of computer-aided design where it may be necessary to create a curve (for example the profile of a high heel) which is pleasing to the eye. The designer is not attempting to approximate to a curve already specified and so there is no question of minimising some 'error' function.

Another case where it is useful to be able to construct a smooth curve through a sequence of points is exemplified in **Fig. 1,** which represents the outline of an insole. The eye is capable of imagining a smooth curve through these 22 points, and a draughtsman might draw one with the aid of a set of French curves. We seek to generate such a curve by means of an algorithm which can be thought of as an N-dimensional French curve. With such an algorithm available, it is not necessary to specify the whole of the insole outline; it suffices to specify the 22 point vectors of Fig. 1.

Two cases in particular are considered:

(*a*) A smooth *closed* curve passing through the vectors

$$P_1, P_2, P_3, \ldots, P_n, P_1$$

(*b*) A smooth *open* curve passing through the vectors

$$P_0, P_1, P_2, \ldots, P_n,$$

with given directions for the tangent vectors at

$$P_0, P_n.$$

The defining vectors $P_0, P_1, \ldots$ are known as 'knots'. In general the knots are not coplanar, and the curve is 3-dimensional. In the special case where all the knots are coplanar, the smooth curve will lie completely in the same plane, and the problem reduces to a 2-dimensional one.

The method of solution adopted is to derive a vector-valued spline curve made up of n segments, each segment linking two consecutive knots. A general account of the theory of splines is given by Ahlberg *et al.* (1967). The requirement that the spline should be smooth is interpreted to mean that:

(*a*) The direction of the tangent vector should be continuous at the knots.

(*b*) The curvature vector should be continuous at the knots (in magnitude as well as direction).

These two conditions are generally sufficient to ensure that the curve appears smooth at a knot, in the sense that the eye cannot detect the position of the knot on the resulting composite curve. Whereas a discontinuity in tangent direction is obvious as a corner, and a discontinuity in the curvature vector can be detected by a practised eye, it seems that a discontinuity in the torsion or the rate of change of curvature of a curve is not visible.

The simplest form of spline segment which permits these requirements to be satisfied is the parametric cubic, and in practice a parametric cubic turns out to be adequate for most applications.

Ferguson (1964) has described one way in which these continuity conditions can be realised—but the Ferguson conditions generate curves which are aesthetically unacceptable unless the knot vectors are approximately equally spaced. The Ferguson method is to take the parametric intervals between knots as uniform, and to demand that the tangent vectors be continuous (in magnitude as well as in direction) at the knots. When the knot vectors are unequally spaced, the result is that the curve tends to cut corners where the knot vectors are widely spaced, and to develop unwanted loops and overshoots where the knot vectors are closely spaced. These defects can be overcome to some extent by taking the parametric interval between knots as proportional to chord length, but the results are still unsatisfactory if any one segment of the curve turns through a large angle.

The present paper derives a method for generating an acceptable solution, whether or not the knot sections are equally spaced. The method depends on intrinsic properties of the curve (unit tangent vector, curvature vector) and not on the particular parametrisation adopted.

2. Tangent and curvature vectors of a parametric curve

Let P be a vector-valued function such that $P(u)$ is a point vector tracing out a space curve as u is varied, and let $s(u)$ represent the cumulative arc length measured along the curve. dP/ds is the *unit* tangent vector

$$dP/ds = \frac{dP}{du}\bigg/\frac{ds}{du} = P'/s'$$

where primes denote differentiation w.r.t. u. Therefore

$$s' = |P'|$$
$$dP/ds = P'/|P'|$$

The curvature vector is

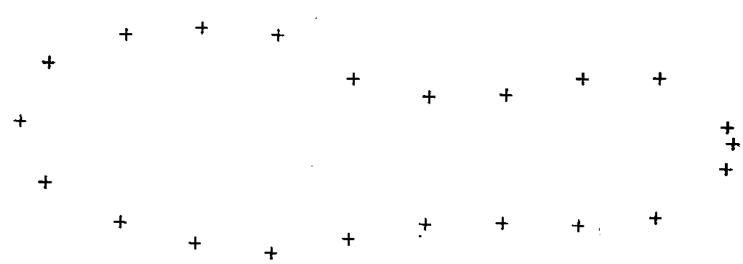

Fig. 1 **Insole shape defined by 22 points**

$$d^2P/ds^2 = \kappa n ,$$

where κ is the curvature and n the unit normal.

We wish to express this vector as a function of P, not involving s.

$$d^2P/ds^2 = \frac{d}{ds}\left(\frac{P'}{|P'|}\right) = \frac{d}{du}\left(\frac{P'}{|P'|}\right)\Big/ s' = \left(\frac{P''}{|P'|} - \frac{P'}{|P'|^2}\cdot\frac{d|P'|}{du}\right)\Big/|P'| \quad (1)$$

If A is any vector

$$A.A = |A|^2$$

and by differentiation

$$A.A' = |A|\, d|A|/du$$

therefore writing P' for A, the expression (1) becomes

$$\frac{d^2P}{ds^2} = \frac{P''}{P'.P'} - \frac{P'.P''}{(P'.P')^2}P' \quad (2)$$

This can be written more neatly as

$$\kappa n = \frac{d^2P}{ds^2} = P' \times (P'' \times P')/[(P'.P')]^2$$

(2) is the expression we need for the curvature vector involving only the derivatives of P with respect to u.

3. Continuity conditions

Suppose a spline curve has a knot at $u = u_0$ and is represented by

$$P = P_1(u) \quad (u \leqslant u_0)$$
$$P = P_2(u) \quad (u \geqslant u_0)$$

subject to the continuity condition $P_1(u_0) = P_2(u_0)$.

For continuity of *slope*, the direction of the tangent vector must be continuous, but not necessarily its magnitude.

$$P_2'(u_0) = hP_1'(u_0) \text{ where } h \text{ is a } \textit{positive} \text{ scalar} \quad (3)$$

For continuity of curvature, the curvature vector κn must be continuous and, therefore, from equation (2) at the point $u = u_0$:

$$\frac{P_1''}{P_1'.P_1'} - \frac{(P_1'.P_1'')P_1'}{(P_1'.P_1')^2} = \frac{P_2''}{P_2'.P_2'} - \frac{(P_2'.P_2'')P_2'}{(P_2'.P_2')^2}$$

Substitute $P_2' = hP_1'$ and multiply by $(P_1'.P_1')$

$$P_1'' - \frac{(P_1'.P_1'')P_1'}{P_1'.P_1'} = \frac{P_2''}{h^2} - \frac{h^2(P_1'.P_2'')P_1'}{h^4(P_1'.P_1')}$$

$$h^2P_1'' - P_2'' = \frac{[h^2P_1'.P_1'' - P_1'.P_2'']}{P_1'.P_1'}P_1' \quad (4)$$

The right-hand side of (4) is a scalar multiple of P_1', say kP_1'. Then

$$h^2P_1'' - P_2'' = kP_1' \quad (5)$$

Equation (4) becomes an identity when $h^2P_1'' - kP_1'$ is substituted for P_2''. Thus there are no restrictions on the value of k in equation (5), which is an arbitrary scalar.

3.1. *Summary of continuity conditions*

Condition for continuity of tangent direction at $u = u_0$:

$$P'(u_0^+) = hP'(u_0^-) \quad h > 0 \quad (3)$$

Condition for continuity of curvature vector at $u = u_0$:

$$P''(u_0^+). = h^2P''(u_0^-) - kP'(u_0^-) \quad k \text{ arbitrary} \quad (5)$$

Ferguson (1964) specifies $h = 1$ and $k = 0$ as conditions for continuity of slope and curvature; we see that they are sufficient but not necessary and are, in fact, unduly restrictive.

Sabin (1969) recognises that k is arbitrary, but does not consider the case $h \neq 1$ in the context of curvature continuity.

4. Normalisation of a cubic spline segment

We now turn from considering continuity at the knots of a spline to the condition for 'fairness' between the knots, and we confine our attention to cubic splines, in which $P(u)$ is a cubic polynomial in u.

Generally,

$$P(u) = R_0 + uR_1 + u^2R_2 + u^3R_3 \quad (6)$$

where R_0, R_1, R_2 and R_3 are vector coefficients.

If the segment runs from the point A (position vector P_A) to the point B (position vector P_B) while u runs from 0 to 1, then

$$P_A = R_0$$
$$P_B = R_0 + R_1 + R_2 + R_3$$

If further we write P_A' for the value of dP/du at the point A, and P_B' for dP/du at the point B,

$$P_A' = R_1$$
$$P_B' = R_1 + 2R_2 + 3R_3$$

Equation (6) can be written as

$$P(u) = P_A + uP_A' + u^2(3(P_B - P_A) - 2P_A' - P_B') + u^3(P_A' + P_B' - 2(P_B - P_A)) \quad (7)$$

Thus the cubic spline segment is uniquely determined by the point vectors P_A and P_B and the tangent vectors P_A' and P_B' at the two ends (see **Fig. 2**).

4.1. *The magnitudes of tangent vectors*

The directions of the vectors P_A' and P_B' are the directions of the tangents at A and B respectively of the curve $P(u)$. It is not intuitively obvious what geometrical significance is to be attached to the *magnitudes* of the tangent vectors at A and B. Forrest (1968) has studied this question. If we fix P_A and P_B and the *directions* of the tangent vectors P_A' and P_B', then equation (7) represents a two-parameter family of curves. **Fig. 3** illustrates some members of this family for the 2-dimensional case when P_A' and P_B' are coplanar, each making an angle of 45° with $P_B - P_A$. Write W for $|P_B - P_A|$, the distance between A and B.

For curve:

(i) $|P_A'|$ and $|P_B'|$ are small compared with W
(ii) $|P_A'|$ and $|P_B'|$ are comparable with W
(iii) $|P_A'|$ and $|P_B'|$ are large compared with W
(iv) $|P_A'|$ is large and $|P_B'|$ is small compared with W.

We note that the larger $|P_A'|$ is, the longer the arc persists in its initial direction before curving away towards B. This is to be expected, because $|P'| = ds/du$, and a large value for $|P'|$ implies that the arc length is large for a small increment in u. Intuitively, curve (ii) looks the smoothest or 'fairest' curve, where the spline segment AB is close to a circular arc. (A circular arc cannot be represented exactly by a parametric cubic; and a parametric quadratic is always a parabola).

In **Fig. 4**, P_A' and P_B' are shown each making an angle θ with the line AB, and AQB is a circular arc also making an angle θ

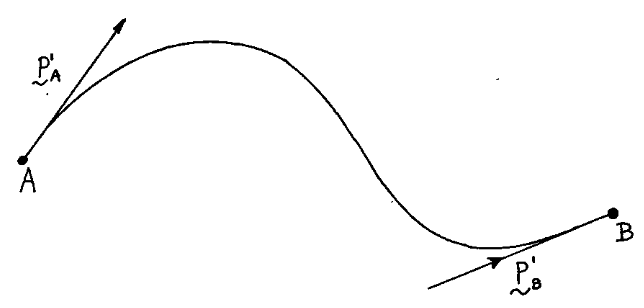

Fig. 2 Spline segment linking *A* and *B* with given tangent vectors

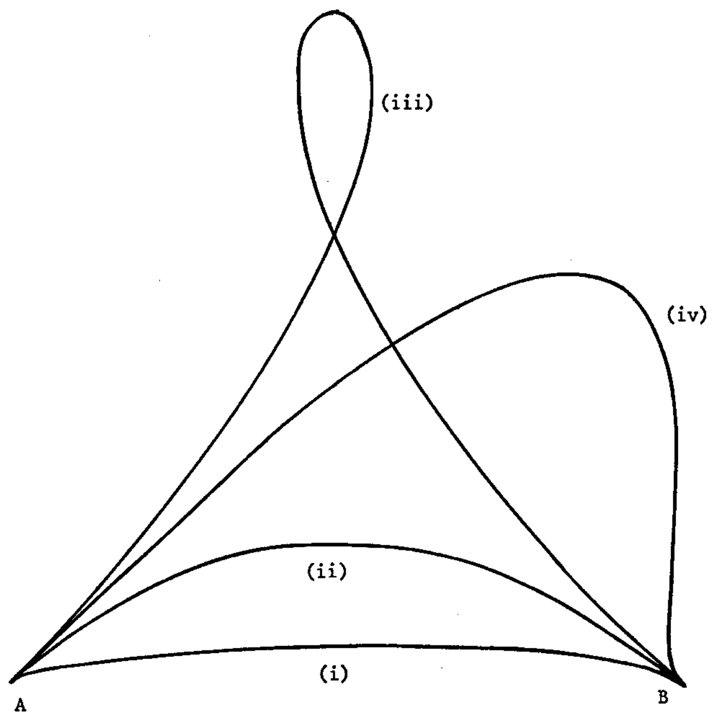

Fig. 3 Effect of varying tangent magnitudes at A and B

with the chord AB. If a symmetrical cubic spline segment (equation (7) with $|P_A'| = |P_B'|$) is to pass through Q, it is easy to show that

$$|P_A'| = |P_B'| = 2W/(1 + \cos\theta) \quad (8)$$

This is curve (ii) of Fig. 3; it is a reasonable approximation to the circular arc AQB, and lies outside it, except where it touches at A, Q and B.

Fig. 5 is a generalisation of Fig. 4 in which P_A' makes an angle θ_A and P_B' a smaller angle θ_B with AB.

The 'fair' curve is one which curves away rapidly from A and therefore $|P_A'|$ ought to be less than $|P_B'|$.

A simple generalisation of equation (8) is:

$$|P_A'| = 2W/[1 + \alpha\cos\theta_B + (1-\alpha)\cos\theta_A]$$
$$|P_B'| = 2W/[1 + \alpha\cos\theta_A + (1-\alpha)\cos\theta_B] \quad (9)$$

where α is some constant between 0·5 and 1.

The formulae reduce to (8) in the symmetrical case $\theta_A = \theta_B$; furthermore the quantities W, θ_A, and θ_B are all well-defined also in the 3-dimensional case when P_A' and P_B' are not coplanar.

In a previous paper (Manning, 1972) the value $\alpha = 1$ was suggested, but this has the disadvantage that a tangent vector becomes infinite when either θ_A or θ_B approaches 180°.

To decide on the most appropriate value for α, series of plane curves were plotted for various combinations of θ_A, θ_B and α. The most acceptable curves (admittedly a subjective judgement) were generated for values of α close to 2/3. Accordingly the formulae (9) have been adopted for defining the magnitudes of P_A' and P_B', with $\alpha = 2/3$.

The tangent vectors (and the spline segments) are now said to be *normalised.*

5. Curve fitting—closed curve

Suppose we are given a series of n knots

$$P_1, P_2, \ldots, P_{n-1}, P_n$$

and we wish to construct a smooth closed curve linking them, the composite curve being made from n cubic spline segments. For convenience we define

$$P_0 = P_n,\ P_{n+1} = P_1$$

(i.e. indices are to be computed modulo n).

Let T_i be the *unit* tangent vector at the point P_i, and let l_i and r_i be the *magnitudes* of the tangent vectors to the left and right of P_i respectively (**Fig. 6**).

By differentiating equation (7) twice and putting $u = 1$ we obtain the following expression for P'' at the point P_i^-:

$$-6(P_i - P_{i-1}) + 2r_{i-1}T_{i-1} + 4l_iT_i$$

Similarly P'' at the point P_i^+ is:

$$6(P_{i+1} - P_i) - 4r_iT_i - 2l_{i+1}T_{i+1}$$

Substituting these expressions for P_1'' and P_2'' respectively in equation (5), and noting that h must be replaced by r_i/l_i, the ratio of the magnitudes of the tangent vectors either side of the knot P_i we obtain the condition for curvature continuity at P_i as

$$K_iT_i = 3\{r_i^2(P_i - P_{i-1}) + l_i^2(P_{i+1} - P_i)\} - r_{i-1}r_i^2T_{i-1} - l_i^2l_{i+1}T_{i+1} \quad (i = 1, \ldots, n) \quad (10)$$

where K_i is another arbitrary scalar.

The normalising conditions are (*cf.* equation (9))

$$l_{i+1} = \frac{2|(P_{i+1} - P_i)|}{\left\{1 + (\tfrac{2}{3}T_i + \tfrac{1}{3}T_{i+1}) \cdot \dfrac{(P_{i+1} - P_i)}{|(P_{i+1} - P_i)|}\right\}}$$

$$r_i = \frac{2|(P_{i+1} - P_i)|}{\left\{1 + (\tfrac{2}{3}T_{i+1} + \tfrac{1}{3}T_i) \cdot \dfrac{(P_{i+1} - P_i)}{|(P_{i+1} - P_i)|}\right\}} \quad (i = 1, \ldots, n) \quad (11)$$

If all vectors are 3-dimensional, these are $5n$ equations in $5n$ unknowns

$$l_1, \ldots, l_n, r_0, \ldots, r_{n-1}, K_1, \ldots, K_n, T_1, \ldots, T_n.$$

(Note that T_i is a *unit* tangent and has only two independent components.)

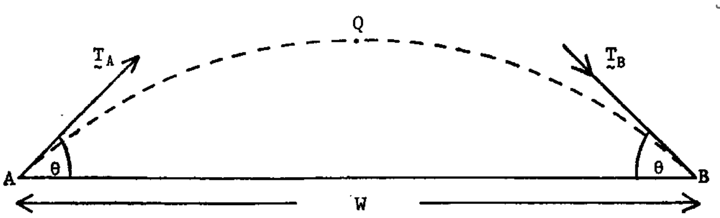

Fig. 4 Circular arc

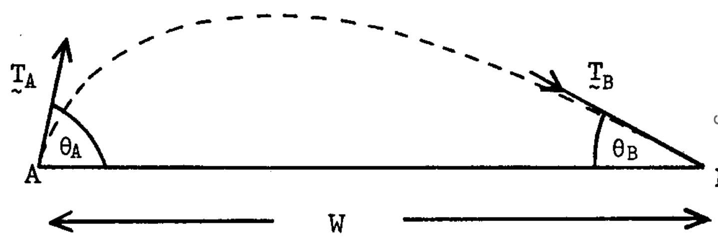

Fig. 5 Asymmetric spline

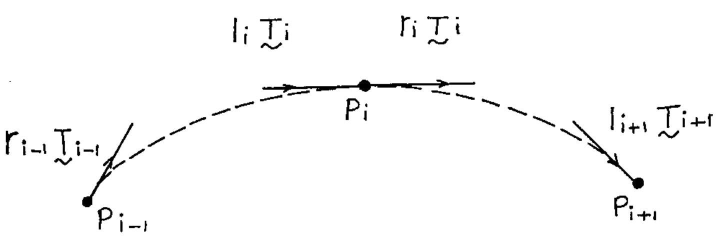

Fig. 6 Section of spline showing continuity of tangent vector direction, but not magnitude, at knot P_i

Thus the restrictions are just sufficient in number to define the composite curve.

Formally, equations (10) and (11) apply to vectors in N-dimensional space. Since, however, a cubic spline segment lies in a 3-space, cubic splines are not suitable for constructing curves in N-space, when $N > 3$.

6. Computational algorithm

A solution of equations (10) and (11) can be arrived at iteratively by the following algorithm:

1. Choose reasonable initial values for the unit tangents T_i (e.g. set T_i parallel to $P_{i+1} - P_{i-1}$).
2. Calculate the tangent magnitudes l_i and r_i using equation (11).
3. Insert these values of l_i, r_i, T_i in the right-hand side of equation (10) and so calculate new values for the unit tangents $\tilde{T}_i$.
4. Replace T_i by $\tilde{T}_i$.
5. Repeat stages 2 and 3 and 4 until the process converges (see Section 8).

There is one difficulty which concerns the constants K_i. We are not interested in the magnitude of K_i but we do need to know its sign to derive the 'correct' sense for the unit tangent $\tilde{T}_i$. If the vectors $T_{i-1}, T_i, T_{i+1}, P_{i+1} - P_i, P_i - P_{i-1}$ all make small angles with one another (as in Fig. 6), then $K_i > 0$ as can be seen from equation (10) in conjunction with the normalising equation (11). If we are trying to fit a fair curve without unnecessary loops, curlicues and other violent changes of direction, it is possible to *define* K_i as a positive scalar. Alternatively, we can define K_i such as to make $T_i . \tilde{T}_i \geqslant 0$, thus ensuring that the direction of T_i does not change by more than 90° with each iteration.

Usually, both methods lead to the same result, but there are exceptions, as will be seen, where they lead to different results.

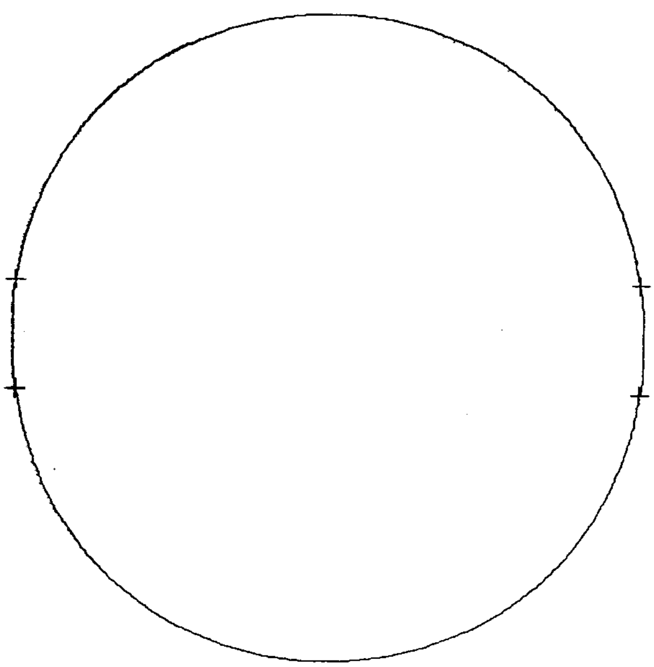

Fig. 7 Smooth curve through four points at the corners of a rectangle

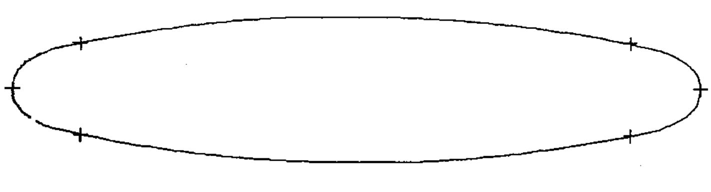

Fig. 8 Quasi-ellipse defined by six knots

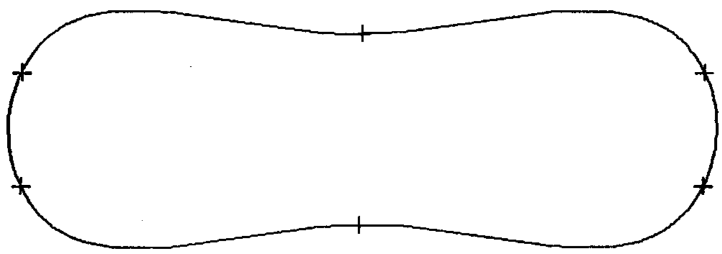

Fig. 9 Oblate circle, not resembling an ellipse

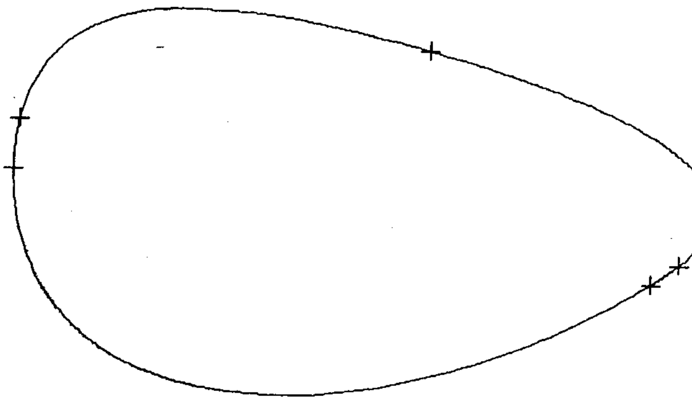

Fig. 10 Oval through 5 knots

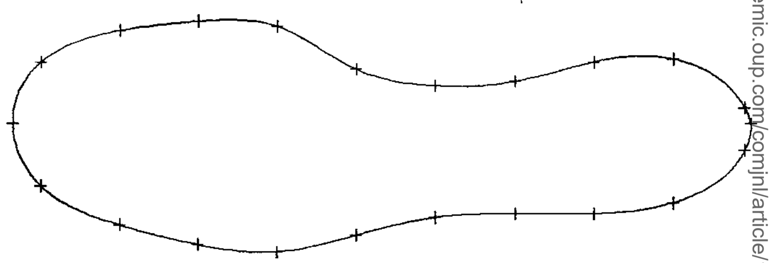

Fig. 11 Insole through 22 knots

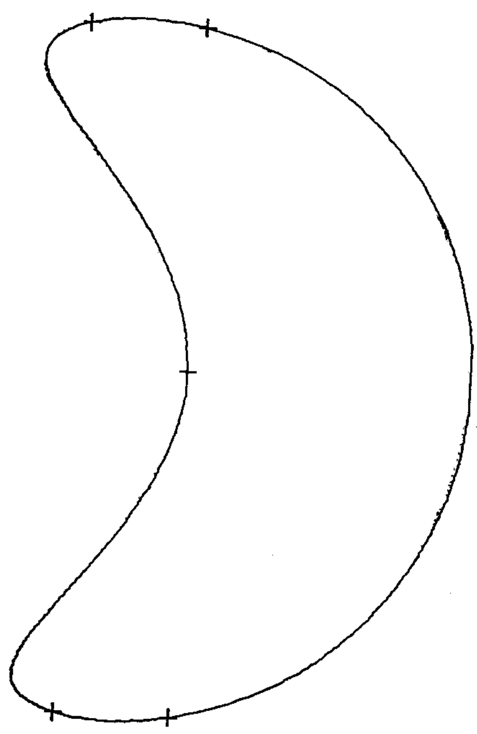

Fig. 12 Curve defined by 5 knots

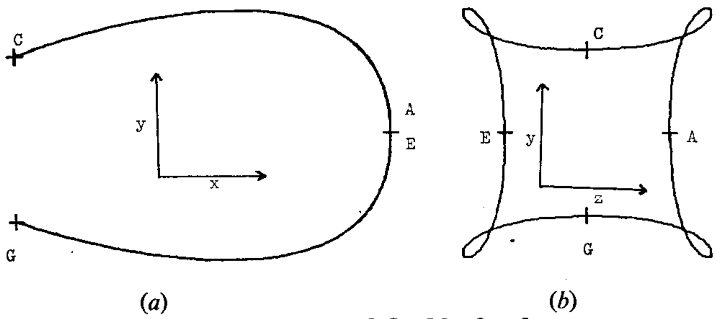

Fig. 13 **Projections of space curve defined by four knots**

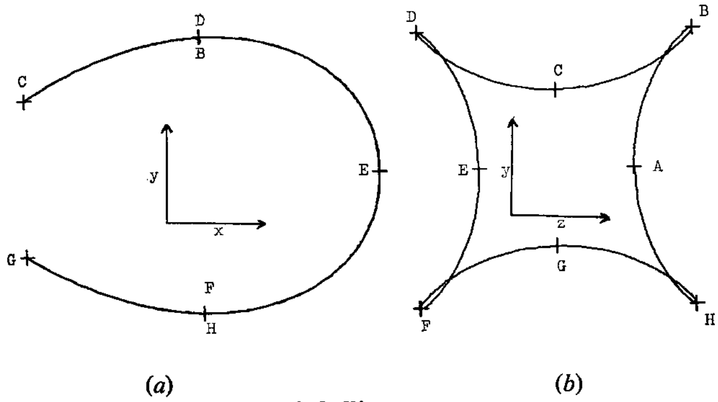

Fig. 14 **Projections of 'tennis-ball' seam**

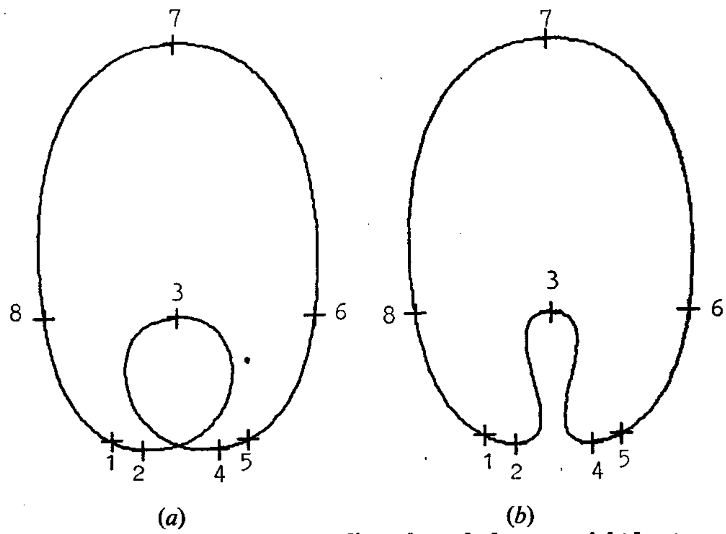

Fig. 15 **Ambiguous case—Two splines through the same eight knots, both with continuity of slope and curvature**

7. Applications

The equations (10) and (11) generally define a smooth curve through a set of points in two or three dimensions. Continuity of curvature is assured, and therefore the method does not require the points to be even approximately evenly spaced; for example **Fig. 7** shows the curve defined by four knots at the corners of a rectangle; the result approximates to a circle. If we require an ellipse passing through the same four knots, two extra knots must be added at the ends of the major axis, as in **Fig. 8.** If instead two knots are added at the ends of the minor axis (**Fig. 9**) the resulting curve bears no resemblance to an ellipse. Evidently, if the general shape of the curve is known, then the defining points must be more closely spaced in regions of high curvature. The figures tend to be characterised by bold, sweeping curves, as in the oval (**Fig. 10**) defined by five knots. **Fig. 11** is the result of applying the algorithm to the insole shape (Fig. 1).

Fig. 12 is an example of a lune defined by five knots.

At least four knots are needed to define a twisted 3-dimensional curve. **Fig. 13**(*a*) and (*b*) show two projections of a curve through the points

$$\left.\begin{array}{l} A\ (a, 0, b) \\ C\ (-a, b, 0) \\ E\ (a, 0, -b) \\ G\ (-a, -b, 0) \end{array}\right\} \text{ with } b/a = 55/24$$

The figure has the same general character as the shape of a seam on a tennis ball (and the ratio $b/a = 55/24$ was chosen after measuring an actual tennis ball), but it does not lie on the spherical surface $x^2 + y^2 + z^2 = a^2 + b^2 = r^2$. If we wish to construct a curve which truly approximates to a tennis ball seam, we need to add four more points lying on the equator:

$$\begin{array}{l} B\ (0, r/\surd 2, r/\surd 2) \\ D\ (0, r/\surd 2, -r/\surd 2) \\ F\ (0, -r/\surd 2, -r/\surd 2) \\ H\ (0, -r/\surd 2, r/\surd 2). \end{array}$$

Fig. 14(*a*) and (*b*) show the *xy* and *yz* projections of this figure, whose maximum distance from the spherical surface is $0{\cdot}003r$.

8. Existence and uniqueness of solutions

The examples in Section 7 demonstrate that the algorithm of Section 6 frequently converges to a solution.

There may be cases, however, in which the algorithm does not converge, but none has yet been found. Furthermore, the equations (10) and (11) need not have a unique solution—a trivial example is furnished when there are only two knots P_1 and P_2 in which case the equations are satisfied by a figure

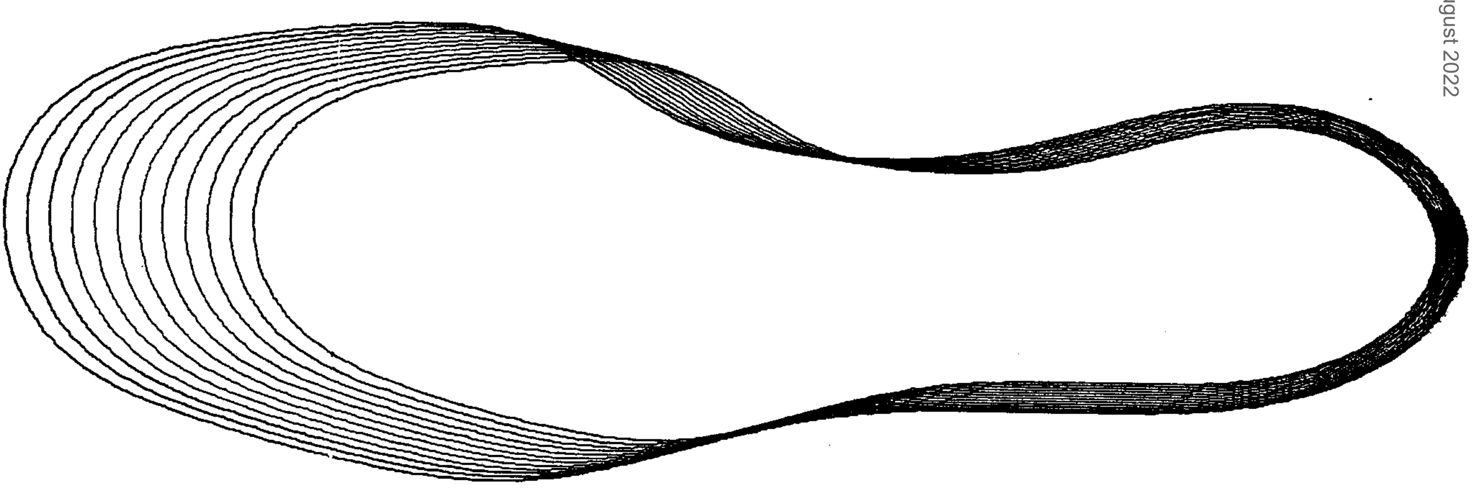

Fig. 16 **Graded set of insoles**

consisting of two quasi-semicircles, but the figure could lie in any plane containing P_1 and P_2.

A more interesting example of non-uniqueness is shown in **Fig. 15**(*a*) and (*b*). Both figures pass through the same eight (coplanar) knots, and both satisfy equations (10) and (11). Whereas Fig. 15(*a*) includes a closed loop, Fig. 15(*b*) contains an inlet. This type of ambiguity arises from the ambiguity over the sign of the constant K_i, briefly mentioned at the end of Section 6. Fig. 15(*a*) was derived by constraining the constants K_i to be positive, whereas 15(*b*) was derived by the second technique mentioned, of choosing the sign of K_i so as to minimise the change of slope at each iteration. The value of K_3 is negative for Fig. 15(*b*). The ambiguity is not a fault in the method, it is a consequence of the inadequacy of the data: the eight knots are insufficient to define where the curve is supposed to go.

It is perhaps worth pointing out that if the curve is thought of as an elastica, then Fig. 15(*b*), which has two points of inflexion, has more strain energy than Fig. 15(*a*), which has no points of inflexion.

A typical example of the practical use of the method is displayed in **Fig. 16**, which represents a graded set of insole patterns. The grading formula is not a proportional enlargement; Fig. 16 was obtained by applying a grading formula to the 22 defining knots and refitting a spline curve for each size.

9. Open curves

The modifications for fitting an open curve to the points $P_0, \ldots, P_n$ with prescribed tangent directions at P_0 and P_n, are minor:

equations (10) and (11) apply with $i = 1, \ldots, (n - 1)$;
T_0 and T_n are given as data.

References

AHLBERG, NILSON and WALSH (1967). *The theory of splines and their application*, Academic Press.

FERGUSON, J. (1964). Multivariable curve interpolation, *JACM*, Vol. 11, No. 2, pp. 221-228.

FORREST, A. R. (1968). *Curves and surfaces for computer-aided design* (Ph.D. Thesis) Joint computer-aided design group, Computer Laboratory, Cambridge University.

MANNING, J. R. (1972). *The representation of a last in numerical form*, RR 240, Shoe and Allied Trades Research Association, Kettering.

SABIN, M. A. (1969). *Spline curves*, Unpublished report VTO/MS/154, British Aircraft Corporation, Weybridge.

Book review

Structured programming, by O.-J. Dahl, E. W. Dijkstra and C. A. R. Hoare, 1972; 220 pages. (*Academic Press Inc.* (*London*) *Ltd.*, £4·20)

The three monographs in this book explore how a structural viewpoint may help in mastering the intellectual processes which mediate between conceiving and producing a program, thereby simplifying the tasks of designing, validating and modifying programs. The main topics of program structuring, data structuring and their relationship are discussed at a level demanding no more than familiarity with an ALGOL-like language. Although principally directed at programmers and of especial value to programming pedagogues, no-one connected with computing should fail to benefit from reading it.

Dijkstra's contribution makes his famous 'Notes on Structured Programming' generally available for the first time. The major concern is reflection in a program's structure of its operational meaning and the imposition on it of a hierarchical structure of 'virtual' machines to facilitate due consideration of alternative programs and proofs of correctness. Lack of space precludes justice to this monograph, ranging as it does from elements of proving program correctness to substitution of the conceptual peculiarity of recursion for the irrelevant efficiency considerations of those suffering from archaic architecture (which is also mentioned). The examples, three of them developed in detail, add considerable weight to the arguments but emphasis on large programs irrespective of individual abilities is perhaps unfortunate. One might assume that structured programming is only of especial use when dealing with such programs. If we define a large program to be one whose size causes the particular programmer involved conceptual difficulty, then the general significance of this monograph will be better appreciated.

Hoare deals with the application of similar considerations to data structuring, starting with the viewpoint that an axiomatisation of program requirements can serve as an aid to design, comprehension and correctness proofs. Regrettably, the static version of type is then presented as though it were the only possibility. Since there are typeless and dynamic-type programmers who claim advantage in the opportunity to refine their type intentions during coding, an examination of the conceptual pros and cons of different notions of type would have been more appropriate. The monograph proceeds to discuss unstructured data and five general kinds of structured data as well as recursive and sparse data structures. In each case a lucid description of the data type and its associated operations is followed by a discussion of representations. The axiomatic viewpoint only returns to prominence in the final section which is a very readable axiomatised summary of the preceding material. Although the prime intention is to offer abstract data structures as an aid to program design, this monograph would usefully serve as a basis for a course on data structures. It is particularly pleasing that representations are discussed without the need to MIX in confusing machine-level programming.

The final section of the book is Hoare's refinement ('restructuring'?) of lectures by Dahl relating SIMULA structuring mechanisms to some concepts which they can model. Whereas the first two monographs are essentially language independent this one is more in the nature of an introduction to a SIMULA view of program and data structures. A comparison in terms of conceptual power and clarity between label values and SIMULA 'processes', to take an example at random, would have been more directly pertinent to the apparent aims of this book.

Finally I exhort anyone about to read this book not to dismiss as 'obvious' principles which are eminently sensible—only a conscious application of those principles will yield tangible results! The book serves a most useful purpose in the enunciation and explication of such principles for programming. It must surely become a classic of the computing literature and should be read, and preferably taken to heart, by any programmer claiming a status above that of 'coder'. Moreover, the need to concentrate on structure extends over every area of computer science.

CLIFF LLOYD (Swansea)