\[
\frac{dy}{dx} = \frac{\frac{dy}{dt}}{\frac{dx}{dt}}
\]

At first sight, continuity of \(\frac{dy}{dx}\) would seem assured by the continuity of \(\frac{dy}{dt}\) and \(\frac{dx}{dt}\) (\(x(t), y(t)\) each being cubic splines). However, Epstein gives an example using the parametrisation (5.1) where

\[
\frac{dy}{dt} = \frac{dx}{dt} = \frac{d^2y}{dt^2} = \frac{d^2x}{dt^2} = 0
\]

at a knot, and so, by L'Hopital's rule

\[
\frac{dy}{dx} = \frac{\frac{d^3y}{dt^3}}{\frac{d^3x}{dt^3}}
\]

In this case, therefore, the tangent does have a slope discontinuity at that knot. For parametrisation (5.2), Epstein proves that \(\frac{d^2y}{dt^2}\) and \(\frac{d^2x}{dt^2}\) cannot vanish together, and so slope continuity of the tangent is guaranteed.

A popular curve drawing technique among designers is the method of Manning [1974]. He takes a rather different definition of a parametric cubic spline: \(x(t), y(t)\) are piecewise cubics, not necessarily splines themselves but simply subject to the restriction that the resulting curve \((x(t), y(t))\) has continuous tangent and curvature vectors. This is a less restrictive definition, and Manning makes good use of the extra freedom to ensure the smoothness of the curve.

As in the single-valued case, local methods are often used in preference to splines. An extension of the osculatory method has been proposed by McConalogue [1970, 1971], and his method is widely used in general purpose graphics packages. It works as follows.

Consider the curve between data points \((x_i, y_i)\) and \((x_{i+1}, y_{i+1})\). The slopes \(\frac{dx}{dt}\) and \(\frac{dy}{dt}\) at \((x_i, y_i)\) are estimated
by fitting parametric quadratics through \((t_{i-1}, x_{i-1}), (t_i, x_i)\), \((t_{i+1}, x_{i+1})\) and \((t_{i-1}, y_{i-1}), (t_i, y_i), (t_{i+1}, y_{i+1})\). The chord-length parametrisation (5.2) is used at this step. Having obtained the slopes at \((x_i, y_i)\) and \((x_{i+1}, y_{i+1})\) a parametric cubic is fitted between the points; this has the form:

\[
\begin{align*}
x(t) &= a_0 + a_1t + a_2t^2 + a_3t^3 \\
y(t) &= b_0 + b_1t + b_2t^2 + b_3t^3
\end{align*}
\]

The values and slopes at the end-points provide eight conditions, which determine \(a_0, a_1, a_2, a_3, b_0, b_1, b_2, b_3\) in terms of the remaining parameter \(T\). It would be possible to take \(T\) as the length of the chord joining \((x_i, y_i)\) and \((x_{i+1}, y_{i+1})\), but McConalogue aims to relate \(T\) more to the arc length. Note that the arc length \(s\) satisfies

\[
\frac{ds}{dt} = \sqrt{\left(\frac{dx}{dt}\right)^2 + \left(\frac{dy}{dt}\right)^2}. 
\]

If the slopes at the end-points are normalized, then

\[
\frac{ds}{dt} = 1 \text{ at } t = 0, T.
\]

To keep \(t\) as close to \(s\) as possible throughout, McConalogue asks that

\[
\frac{ds}{dt} = 1 \text{ at } t = T/2.
\]

This extra condition is enough to fix \(T\).

Just as the osculatory method has an interpretation as a blending method, so has McConalogue’s method. One can write the parametric cubic in each interval as a blend of the parametric quadratics used to determine the slopes at the end-points. This parabolic blending method is usually attributed to Overhauser [1968] in the CAD literature (see also Rogers and Adams, 1976, and Brewer and Anderson, 1977). McConalogue’s method is simply Overhauser’s method of parabolic blending in a different guise.

Other local methods for single-valued curves can likewise be extended to the parametric case. Again unwanted inflection points can be a problem but one solution has been put forward by Bolton [1975]. This is the method of biarc curves, where
the curve in any interval is composed of two circular arcs each having the correct slope at one end-point and joined smoothly in the middle. A point of inflection is only created if it can reasonably be inferred from the data. The method is used in the ship-building industry.

6. CURVE DESIGN

Methods for curve design have been thoroughly explored by CAD research workers. Only a brief mention of the topic is given here, so that the techniques involved can be introduced to those from outside the CAD field and related to the methods for curve drawing described in the earlier sections. Readers interested in pursuing the subject further are recommended to read the papers of Forrest [1972] and Gordon and Riesenfeld [1974a,b].

Fig. 16 shows a parametric cubic spline drawn through a set of data points. To change the shape of the curve, one could alter one of the data points but this would require recalculation of the spline and the result might be hard to predict. In particular the alteration of the data point might affect the shape of the curve at some distance from the point.

Recall that a parametric cubic spline can be expressed as

\[ s_x(t) = \sum_{i=1}^{k+4} a_i N_i(t) \]

\[ s_y(t) = \sum_{i=1}^{k+4} b_i N_i(t) \]

The B-spline coefficients for the spline of Fig. 16, taken as pairs \((a_i, b_i)\), are plotted in Fig. 17. Notice that the polygon formed by these points 'mirrors' the shape of the curve. Control over the shape of the curve can be obtained by varying the defining polygon of the spline. Because B-splines are zero except in a small interval, the alteration of one pair of coefficients only has a local effect on the curve. In Figs. 18, 19, two points of the defining polygon are altered in turn, in an effort to improve the S-shape of the curve. Notice that only a small area of the curve is altered at each stage.

This is the basic idea behind the B-spline curves of Gordon and Riesenfeld [1974b], although the original suggestion of using a polygon to define a curve is due to Bezier [1972] and Forrest [1972]. The designer typically starts with an
B-SPLINE POLYGON

Fig. 17

FIRST ALTERATION

Fig. 18
An interpolating parametric spline is constructed as described in section 5, and the associated polygon formed. A dialogue is then set up in which the designer modifies the vertices of the polygon until the associated curve is of the required shape.

Besides varying the vertices of the polygon, it is common also to vary the order of the spline. The first order (linear) spline associated with the polygon is just the polygon itself. An important property of a B-spline curve of any order is that it is variation diminishing: it has at most as many inflections as the defining polygon and may well have fewer.

There is further scope for design in varying the knot positions. A recent paper by Hartley and Judd [1978] advocates the choosing of knot positions to obtain a good parametrization of the curve, i.e. so that evenly spaced parameter values should correspond roughly to evenly spaced points along the curve.

Some designers have expressed dissatisfaction with the Bezier approach, finding it awkward to work with points off
the curve. The local parametric methods described in section 5 offer an alternative approach, allowing the designer to work with points on the actual curve and yet still retain local control of the shape. (Overhauser 1968, Brewer and Anderson, 1977).

7. PLOTTING A FUNCTION OF ONE VARIABLE

A common requirement, especially amongst mathematicians and physicists, is a routine to plot a function of a single variable, \( y = f(x) \) say. The simplest approach is to evaluate \( f(x) \) at a number of equally-spaced points, and join up these values with straight lines. The number of points required to produce a smooth curve would depend on the resolution of the device.

There are some difficulties with this crude approach. As mentioned in section 4, one generally wishes to pass as few line segments to the graphics package as possible, and so aim should be not to use line segments of equal length but to approximate the function with long segments where it is nearly straight, and with short segments where it bends sharply. One solution is to preprocess the evaluated points and to discard unnecessary points by merging into a single line segment any sequence of points which lie within a given tolerance of the line. One such preprocessing algorithm is described by McLain [1978]. Note, however, that in several of these algorithms the measurement of the points from the line is made in terms of Euclidean distance. This is only sensible if the axes are equally scaled, and can give disastrous results otherwise. One finds that points can be discarded which are perhaps close to a line segment in terms of the user's co-ordinate system, but some considerable distance in terms of the actual plotting surface. A good strategy when the axes are unequally scaled is to ask the user to supply two tolerances \( \delta_x, \delta_y \) say, and to test separately distances parallel to the two co-ordinate axes.

If the function to be plotted is expensive to evaluate, it is a good idea first to approximate it by a cubic spline, and then to plot the spline using the method described in section 4. A good automatic algorithm for cubic spline approximation is described by Curtis [1970] and implemented by Powell [1972]. The user is asked to supply a tolerance \( \varepsilon \), and a cubic spline \( s(x) \) is constructed such that

\[
|s(x) - f(x)| \leq \varepsilon \quad (7.1)
\]
for all $x$ in the interval $\{a,b\}$ over which the function is to be approximated.

Briefly, the method works as follows. Initially a small set of equally spaced knots $\{\lambda_1, \lambda_2, \ldots, \lambda_k\}$ is chosen in the interval $\{a,b\}$, the function is evaluated at the knots and a cubic spline interpolant is constructed. Now the error in cubic spline interpolation is related to the size of the third derivative discontinuities of the spline at the knots, or more precisely

$$\sup_{\lambda_i < x < \lambda_{i+1}} |s(x) - f(x)| \leq \frac{h^3}{384} (\max(\Delta_i, \Delta_{i+1})) + O(h^6) \quad (7.2)$$

where $\Delta_i$ is the size of the third derivative discontinuity at $\lambda_i$ (Curtis and Powell, 1967a,b). It is easy therefore to examine the error in each knot interval, and to place extra knots in the intervals where the error is greatest. A new spline is constructed, and the process continues until the inequality (7.1) is satisfied. Although the result (7.2) only applies to equally spaced knots, the error estimate can be extended to cover variable knot spacing.

This technique is strongly recommended when the function is expensive to evaluate, since consideration is given to where the evaluations should be made, rather than simply calculating the function at equal intervals.

8. CONCLUSIONS

The main point of the paper has been to distinguish clearly two situations in curve drawing, and to describe the different approaches needed to handle the two cases. First, there is the situation where a single-valued curve is needed - this occurs frequently in simple graph-plotting when there is some function underlying the data which is single-valued and it makes nonsense for the drawn curve to turn back on itself. Second, there is the situation where a rotation-invariant, possibly multi-valued, curve is required - this applies, for example, in the drawing of shapes.

In the single-valued case, it is usual to construct some function $y = f(x)$ which interpolates the data: there is a choice between splines, with good continuity properties, and local methods, which allow the curve to be drawn as the data is collected. In both cases, the 'looseness' of curves
can be a problem and special techniques to counteract this have been devised.

Parametric curves are used in the rotation-invariant, multivalued situation. Again the user has a choice between splines and local methods.

REFERENCES

Ackland, T.G. (1915) "On osculatory interpolation, where the given values of the function are at unequal intervals". J. Inst. Actuar., 49, 369-375.


McLain, D.H. (1979) "Interpolation methods for erroneous data". Chapter 4 of this volume.


Maude, A.D. (1973) "Interpolation - mainly for graph plotters". Comp. J., 16, 64-65.


DISCUSSION

Professor M.L.V. Pitteway (Brunel University):

A cubic curve can be generated directly on an incremental plotter with an algorithm involving six additions and one test for each increment drawn, i.e. it is not necessary to first represent the cubic through a piecewise linear approximation. There are no problems with error accumulation provided the calculation is done correctly - see Botting and Pitteway,
Computer Journal, Vol. 11, 1968, p. 120, and Pitteway, Computer Journal, Vol. 10, 1967, pp 282-289. In particular, if the cubic has integer coefficients (so that it is specified precisely), the algorithm will follow for example a closed loop indefinitely, without ever drifting more than half an increment from the intended curve.

M.A. Sabin (Kongsberg Ltd. now with CAD Centre, Cambridge.)

Some graph plotter packages do not permit the user direct plotter control, but insist on straight line increments. Moreover, many modern graphic display devices include vector generator hardware. Both obviously require the piecewise linear approximation.

Brodie:

I agree with Sabin that most users interface to graphics systems at the level of straight line pieces rather than plotter increments or raster positions. Thus I think the piecewise linear approximation of a cubic is important. However I would expect that hardware curve generators will soon be offered on many graphics devices and one hopes that manufacturers will take note of the work of Botting and Pitteway in the design of this hardware.

Sabin:

Two comments which may be of interest:

(i) Regarding cubic splines, the end-condition

\[
\frac{d^2 s(x_1)}{dx^2} = \frac{d^2 s(x_2)}{dx^2}
\]

\[
\frac{d^2 s(x_n)}{dx^2} = \frac{d^2 s(x_{n-1})}{dx^2}
\]

gives good results with reasonably dense data or a smooth curve.

(ii) It is possible and useful to determine parameter values from the problem. If, for example, we can constrain

\[ t = x \]

then the parametric fit reduces to the single-valued case.
Another example is the fitting of an aerofoil which is a double-valued function, with a smooth transition between the two leaves at the leading edge. The parametrization $t^2 = x$ is convenient, with $t > 0$ and $t < 0$ on the two leaves.

S. Heatherington (Durham County Council, Engineer's Department):

Much work has been done in Durham County Council in applying spline techniques in the field of Highway Design (References: Heatherington, IMA Bulletin, November 1974; Open University, M351, Units 9 and 12). Whilst there is perhaps nothing original in the mathematical theory behind this work it is probably worth mentioning that in road design there is often a need to have continuity of both first and second derivatives at the end points. This situation may arise, for example when the line of a road improvement ties into a stretch of existing road or crosses a bridge. In these circumstances more constraints are placed on the data set than can be accommodated.

The solution adopted is to introduce a further data point into the data set such that it may "float" until the required end conditions are met. In the single valued case this means that a "floating" point is introduced midway between the x ordinate of the end point and penultimate point, and because the y co-ordinate is free to move this presents a degree of freedom to overcome the constraint given by the curvature (or second derivative).

---

Fig. 20
In the parametric case the added knot is introduced into the solution without specifying either the x or y but constraining the parameter t.

![Diagram showing ARC, DATA POINTS, and FREE TO MOVE](image)

**Fig. 21**

Although the diagrams may imply the solution is found by applying an iterative technique, in fact the problem is resolved very simply and analytically to give a unique mathematical solution.

Mrs. J. Butland (Bradford University):

I have developed a local method which uses parametric piecewise cubics for all types of data, and is incapable of generating a multivalued curve from "single-valued data".

The rule for determining the gradients of the curve at data points is based on two assumptions:

(i) That all known features of the curve such as turning points and inflections are described by the data and the generated curve must not introduce any more.

(ii) That when successive data points approximate a straight
line, they must be joined by a curve which approximates a straight line.

An equal-interval parametrization is used, and the value of $\frac{dx}{dt}$ at $t = 1$ is given by the formula:

$$\frac{dx}{dt} = \frac{2m_1m_2}{m_1 + m_2}, \quad m_1m_2 > 0$$

0 otherwise

where $m_1 = x(i) - x(i-1)$

$m_2 = x(i+1) - x(i)$,

with a similar rule for $\frac{dy}{dt}$

Brodlie:

Mrs. Butland's method is very interesting. It can indeed be proved that it generates a single-valued curve from "single-valued data", i.e. data whose x-values are strictly increasing. If the data is not single-valued, then multivalued curves are drawn.

However, the very fact that single-valuedness is ensured means that the method cannot be invariant under a rotation of the co-ordinate system. To see this, consider the data points below

```
 x   x
 x   x
 x   x
```

and suppose the co-ordinate system is rotated until the third point lies almost on top of the second point.

```
 x   x
 x   x
```

It is easy to see that the only rotation-invariant curve which is single-valued when the data is single-valued is the 'curve' formed by joining the points with straight lines.

Hence I regard Mrs. Butland's method as a useful addition to the set of methods for drawing single-valued
curves, but I would prefer to use a rotation-invariant method (such as McConalogue's method) for multivalued curves. The method does work extremely well in the single-valued case, as can be seen from the two figures below which show how the curves drawn by the method through the data of Figs. 4 and 5. I am grateful to Mrs. Butland for supplying these plots.

The most interesting feature of Mrs. Butland's method is its monotonicity property. If \( x(i+1) > x(i) \) then the curve increases monotonically in \( x \) between the two points – hence the single valued property. Exactly the same holds for the \( y \)-values. An immediate consequence is that all maxima and minima of the curve occur at data points, and there are no extraneous points of inflection.

A method for single-valued curves which is more along the lines of the local methods described in section 3, but which uses the ideas of Mrs. Butland's parametric method to obtain a monotonicity property, can be defined as follows. The slope \( m \) at a data point \((x_i', y_i')\) is estimated as

\[
\frac{1}{m} = \frac{1}{2} \frac{1}{m_{i-1}} + \frac{1}{m_i} \quad m_{i-1} m_i > 0
\]

\[
m = 0 \quad \text{otherwise},
\]

where \( m_{i-1} \) is the slope of the line joining \((x_{i-1}', y_{i-1}')\), \((x_i', y_i')\) and \( m_i \) is the slope of the line joining \((x_i', y_i')\), \((x_{i+1}', y_{i+1}')\). As usual, a cubic polynomial is fitted in each interval to match the data values and the estimated slopes.

Note that \( m \) satisfies

\[
0 < m/m_{i-1} < 2 \quad \text{and} \quad 0 < m/m_i < 2
\]

and together these conditions are sufficient to guarantee that the piecewise cubic curve which is generated is monotonic between data points – i.e. if \( y_{i+1} > y_i \), the curve increases monotonically between \( x_i \) and \( x_{i+1} \). Again maxima and minima of the curve occur at data points, and there are no unwanted points of inflection. In fact, monotonicity is guaranteed under the weaker conditions:
\[ 0 \leq \frac{m}{m_{i-1}} < 3 \quad \text{and} \quad 0 \leq \frac{m}{m_i} < 3 \]


The method just described makes no allowance for variably-spaced data points. An extension which has been found to work well in practice is to define the slope at a data point by the formula

\[ \frac{1}{m} = \frac{\alpha}{m_{i-1}} + \frac{(1-\alpha)}{m_i} \quad m_{i-1}m_i > 0 \]

\[ m = 0 \quad \text{otherwise,} \]

where \( \alpha = \frac{1}{3} \left( 1 + \frac{h_i}{h_i + h_{i-1}} \right) \)

with \( h_{i-1} = x_i - x_{i-1} \), \( h_i = x_{i+1} - x_i \). Note that \( 1/3 < \alpha < 2/3 \), and the conditions

\[ 0 \leq \frac{m}{m_{i-1}} < 3, \quad 0 \leq \frac{m}{m_i} < 3 \]

are both satisfied.