A PRIMER ON POLYNOMIAL INTERPOLATION AND SPLINES

Bryant W. York

COINS TECHNICAL REPORT 79-5

This research was supported by the National Science Foundation under Grant MCS75-16098 A01. During the 1977-78 academic year the author received support from The National Fellowship Fund, Atlanta, Georgia. The author acknowledges the support of the Computing Center at the University of Massachusetts.

SPLINE PRIMER

TABLE OF CONTENTS

APPENDICES

SPLINE PRIMER

I. INTRODUCTION

For many years the hulls of ships and the bodies of automobiles were designed manually with the aid of the "draftsman's spline". The draftsman's spline is a thin piece of wood or metal which is elastically bent around supports known as "weights" or "ducks". Supports are placed at selected locations to obtain a smooth curve of a desired shape. The actual shape of the curve assumed by the physical spline is the shape which minimizes internal strain energy.

In the mid 1940's mathematicians [Sch46] began to develop the mathematics to describe these curves and a mathematical theory of splines was born. At first development proceeded slowly until the early 1960's when advances in electronic computers, graphical display hardware, and numerically controlled machines spurred research into the practical applications of splines to engineering and design problems. As a result splines are becoming more widely used in systems for computer-aided design and numerical control.

Splines are used mainly in the areas of functional approximation and curve fitting. Good curve fitting techniques have many application from stock market forecasting to computer vision. Splines have also been used to solve systems of differential equations and certain boundary value problems. Our

purpose in this report is to present the reader, who may have very little background in numerical analysis, with a brief introduction to the basic notions of polynomial and spline interpolation. The first four sections require only knowledge of elementary calculus; the fifth section makes use of the theory of linear spaces and basis functions.

II. POLYNOMIAL INTERPOLATION

DEFINITION: Linear interpolation is the process of estimating values of a function between two known values.

In high school, most of us were required to interpolate values in a table (usually a table of logarithms). Generally we used "linear interpolation". For example, given the two values from a table of common logarithms,

x	log10(x)
100	2.00000
101	2.00432

we might have been asked to find the value of log10(x) for x=100.7 , an intermediate value between 100 and 101. Linear interpolation assumes that the distances between the desired functional value and the known functional values are proportional to the distances between the desired argument value and the known argument values. This relationship is expressed in the following

equation,

$$\frac{(100.7 - 100)}{(101 - 100.7)} = \frac{\log10(100.7) - \log10(100)}{\log10(101) - \log10(100.7)} \qquad \text{(II.1)}$$

The linear interpolation assumption states that the intermediate functional values lie on a straight line connecting the two known functional values (see figure 1). An alternative statement of the "linear interpolation assumption" is that the function "behaves as a polynomial of degree 1" (straight line) on the interval between the two known values of the function. A polynomial of degree 1 is of the form:

$$y = f(x) = c_0 + c_1 x$$

and in this case we have f(x)=1.56800 + .00432x This solution was easily determined by first finding the slope of the line

$$\frac{\Delta y}{\Delta x} = \frac{2.00432 - 2.00000}{101 - 100} = .00432$$

and then finding the y-intercept by substituting the slope and one of the known argument-value pairs into the equation for the line

$$2.00000 = c_0 + .00432 \cdot 100$$

$$1.5680 = c_0$$

thus

$$f(x) = 1.5680 + .00432x \qquad \text{(II.2)}$$

Now, we can evaluate f(x) at x = 100.7 to obtain the intermediate value

$$f(100.7) = 2.003024$$

When we check a (5-place) table of common logarithms, we see that

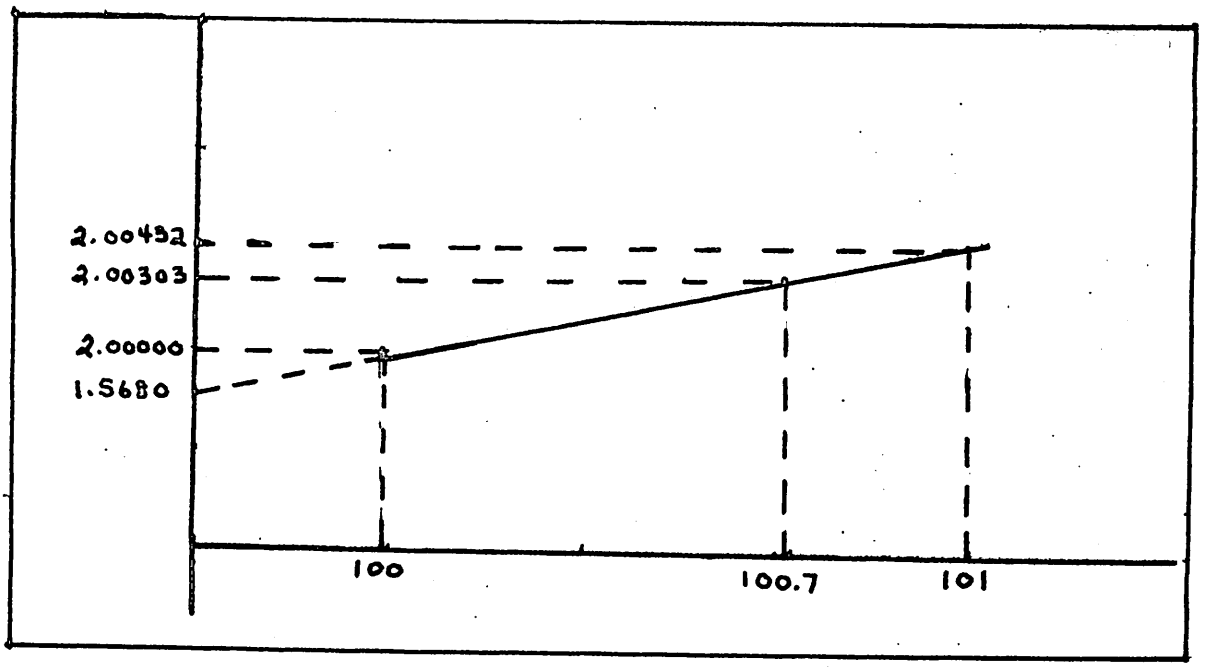


FIGURE 1 LINEAR INTERPOLATION

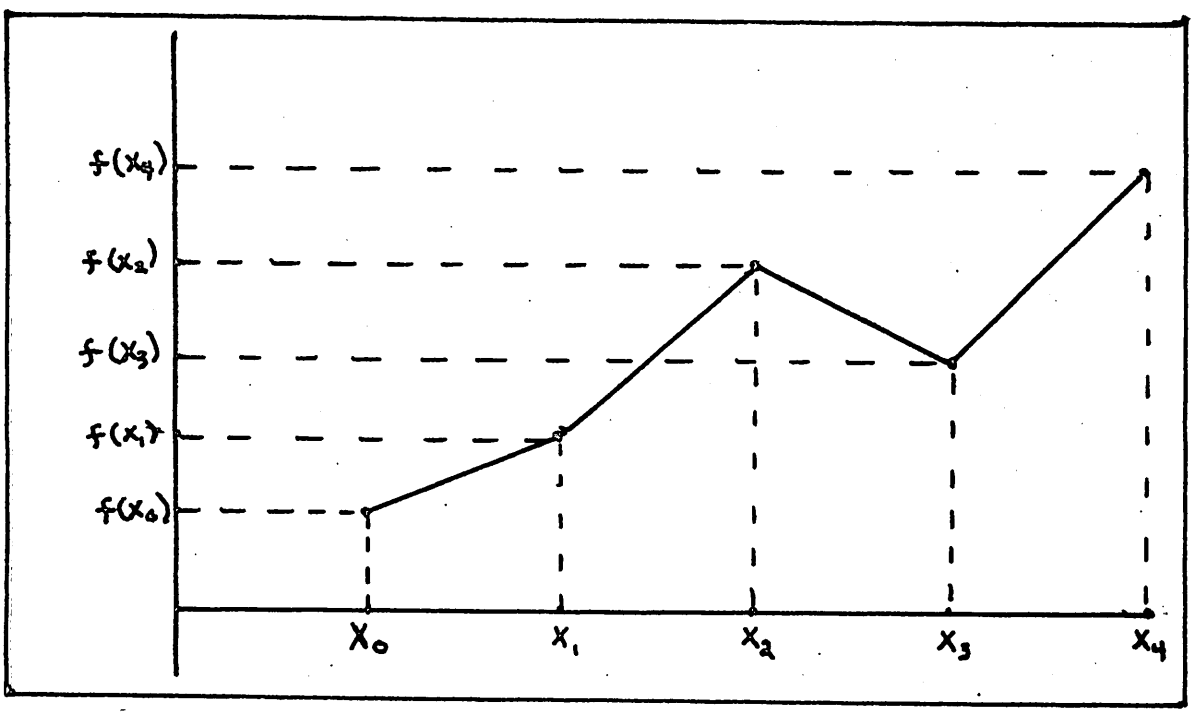


FIGURE 2 PIECEWISE LINEAR INTERPOLATION

the tabular value is 2.00303. Thus the linear interpolation scheme allows us to estimate an intermediate value of a function, which is known at two discrete points, to a reasonably high degree of accuracy in this case. The degree of accuracy of the estimation depends in large part on the true nature of the function and there are large classes of functions for which linear interpolation is unsuitable.

PIECEWISE LINEAR INTERPOLATION

Instead of being required to find a single value of the tabulated function, we were generally required to determine several functional values, so we extended the notion of simple linear interpolation to piecewise linear interpolation. Under the piecewise linear interpolation scheme, it is assumed that the underlying function behaves as a straight line on each subinterval (between successive argument pairs), but that successive straight line segments may differ in slope. All that is required is that they meet at the common end points (see figure 2).

The next step in sophistication is to assume that the function behaves as a higher degree polynomial over the interval in question. Before describing interpolation for polynomials of degree greater than 1, we will examine some important properties

of polynomials and introduce the notion of finite differences. The familiar form of a general polynomial of degree n is:

$$f(x) = c_0 + c_1x + c_2x^2 + \ldots + c_nx^n$$

Differentiation of this polynomial produces a polynomial of degree n-1

$$f'(x) = c_1 + 2c_2x + 3c_3x^2 + \ldots + nc_nx^{n-1}$$

If, in turn, f'(x) is differentiated a polynomial of degree n-2 is obtained, and so on until after n differentiations a polynomial of degree zero, a constant function, is eventually obtained.

$$f^{(n)}(x) = n!c_n$$

One more differentiation gives $f^{(n+1)}(x) = 0$. Thus for any polynomial function of degree n:

(1) The nth derivative is a constant.

(2) The (n+1)st and succeeding derivatives are zero.

These facts will be useful during the discussion of polynomial interpolation to follow.

ORDINARY FINITE DIFFERENCES

Suppose one is given a table of argument values and corresponding function values. From this table, it is possible to compute the "Finite Difference Table" which is the source of coefficients in most polynomial interpolation schemes. Given n

argument-function value pairs a difference table with n-1 difference levels is obtained (see figure 3). In the particular example shown in figure 3, the argument values are equally spaced and the distance between successive values is 1. This is the simplest case, often referred to as "ordinary finite differences". The difference table is formed by subtracting successive values at the previous level, for example

and
$$\Delta f(x_i) = f(x_{i+1}) - f(x_i)$$
$$\Delta^2 f(x_i) = \Delta f(x_{i+1}) - \Delta f(x_i) \qquad \text{for } i=0,1,2,3$$

Δf is read "the first ordinary difference of f"
$\Delta^2 f$ is read "the second ordinary difference of f"
and so on

note $x_i=i$

NEWTON'S FORWARD DIFFERENCE POLYNOMIAL INTERPOLATION FORMULA

Given the problem of finding a polynomial which agrees with the function f(x) for the arguments shown in figure 3, how would one make use of the difference table? First, since differences are discrete approximations to derivatives, let the differences in the table correspond to the derivatives of the unknown polynomial. Since fifth and higher differences are zero, the polynomial will be of degree 4 (fifth and higher derivatives are zero). Thus the polynomial and its derivatives are:

$$p(x) = c_0 + c_1x + c_2x^2 + c_3x^3 + c_4x^4 \qquad \text{(II.3)}$$

$$p^{(1)}(x) = c_1 + 2c_2x + 3c_3x^2 + 4c_4x^3 \qquad \text{(II.4)}$$

$$p^{(2)}(x) = 2c_2 + 6c_3x + 12c_4x^2 \qquad \text{(II.5)}$$

$$p^{(3)}(x) = 6c_3 + 24c_4x \qquad \text{(II.6)}$$

$$p^{(4)}(x) = 24c_4 \qquad \text{(II.7)}$$

x	$f(x)$	$\Delta f(x)$	$\Delta^2 f(x)$	$\Delta^3 f(x)$	$\Delta^4 f(x)$
0	10	10	20	-70	150
1	20	30	-50	80	
2	50	-20	30		
3	30	10			
4	40				

FIGURE 3 ORDINARY FINITE DIFFERENCE TABLE

Now, we must solve for the unknown constants, c_0, c_1, c_2, c_3, c_4. From these equations and the requirement that p(x) agree with f(x) at x=0 we get the following

$$c_4 = \frac{p^{(4)}(0)}{4!} \simeq \frac{\Delta^4 f(0)}{4!}$$

$$c_3 = \frac{p^{(3)}(0)}{3!} \simeq \frac{\Delta^3 f(0)}{3!}$$

$$c_2 = \frac{p^{(2)}(0)}{2!} \simeq \frac{\Delta^2 f(0)}{2!}$$

$$c_1 = \frac{p^{(1)}(0)}{1!} \simeq \frac{\Delta f(0)}{1!}$$

$$c_0 = p(0) = f(0)$$

Substitution into equation (II.3) yields

$$p(x) \simeq f(0) + x\Delta f(0) + x^2\frac{\Delta^2 f(0)}{2!} + x^3\frac{\Delta^3 f(0)}{3!} + x^4\frac{\Delta^4 f(0)}{4!} \qquad \text{(II.8)}$$

Now, we must force p(x) to agree with f(x) at x=1, 2, 3, 4. This is accomplished by replacing x^2 with $x(x-1)$, x^3 with $x(x-1)(x-2)$ and x^4 with $x(x-1)(x-2)(x-3)$ resulting in equation (II.9). The effect is that when x=0, all but the first term in equation (II.9) vanish; when x=1, all but the first and second term vanish, etc.

$$p(x) = f(0) + x\Delta f(0) + \frac{x(x-1)}{2!}\Delta^2 f(0) + \frac{x(x-1)(x-2)\Delta^3 f(0)}{3!} + \frac{x(x-1)(x-2)(x-3)}{4!}\Delta^4 f(0) \qquad \text{(II.9)}$$

It is clear that equations II.8 and II.9 are not algebraically equivalent; however, the intent is not to give the reader a

precise derivation, but to provide a feeling for what the formula represents. When f is expanded in a Taylor series about 0, the approximations for

$f(0)$, $f^{(1)}(0)$, $f^{(2)}(0)$, $f^{(3)}(0)$, $f^{(4)}(0)$ involve terms which actually introduce the "factorial polynomials" into the formula. Equation II.9 is known as the Newton Forward Difference Interpolation formula. It is the formula for a polynomial of degree n (where n+1 is the number of function values given) which agrees with the function at the given points. This polynomial may be used to interpolate values anywhere in the interval [0, n] . The complete derivation utilizing Taylor series expansion appears in Appendix F.

The reader should note several important points:

(i) The table of argument-function value pairs need not begin at x=0. The table may begin anywhere; however, a simple transformation must be performed on the argument if the Newton Forward Difference interpolation scheme is to be used. Consider the example illustrated in the difference table of Figure 4. In this case translation of the origin to x=100 is achieved by the transformation u=x-100. This results in a forward difference formula as a function of u,

$$p(u) = 15 + 10u - \frac{25u(u-1)}{2!} + \frac{70u(u-1)(u-2)}{3!}$$

When the interpolation formula is used for intermediate values of

x, the x values must be converted to the corresponding u values before substituting into the formula.

(ii) The arguments may be equally spaced at intervals greater than 1. The Newton Forward Difference formula may still be used by transforming the scale of x. Consider the example shown in Figure 5. The transformation is u = x/10; again the interpolation formula may be written as a function of u.

(iii) The transformations in (i) and (ii) above may be combined if necessary

(iv) There are many different polynomial interpolation formulas based on different ways of looking at the ordinary difference table. The most notable are:

a) Newton's Backward Difference Formula

b) Stirling's formula

c) Bessel's formula

Newton's Backward Difference formula is based upon the backward difference table. Stirling's and Bessel's formulas are based upon the central difference table. See [FRE39] and [HOR75] for details. In general these interpolation schemes are applied over an interval where the arguments are equally spaced. Often we have the situation in which the arguments are not equally spaced.

DIVIDED DIFFERENCES

When the argument values are not equally spaced, we may form

u	x	f(x)	$\Delta f(x)$	$\Delta^2 f(x)$	$\Delta^3 f(x)$
0	100	15	10	-25	70
1	101	25	15	45	
2	102	10	30		
3	103	40			

FIGURE 4 TRANSLATION OF FINITE DIFFERENCE TABLE ARGUMENTS TO ORIGIN

u	x	f(x)	$\Delta f(x)$	$\Delta^2 f(x)$	$\Delta^3 f(x)$
0	0	15	10	-25	70
1	10	25	-15	45	
2	20	10	30		
3	30	40			

FIGURE 5 SCALING OF FINITE DIFFERENCE TABLE ARGUMENTS TO UNIT INTERVAL

i	x	f(x)	DDf(x)	$DD^2 f(x)$	$DD^3 f(x)$	$DD^4 f(x)$
0	0	36	1.1000	.3139	-.0107	.0002
1	10	47	6.7500	-.1996	.0042	
2	18	101	-2.8333	.0962		
3	48	16	4.0000			
4	71	108				

FIGURE 6 DIVIDED DIFFERENCE TABLE

the "divided-difference table" (see figure 6). Divided differences are the ordinary differences divided by the difference in the associated arguments. For example

$$DDf(x_0) = (f(x_1) - f(x_0))/(x_1 - x_0) = (47 - 36)/10 = 1.10$$

$$DDf(x_1) = (f(x_2) - f(x_1))/(x_2 - x_1) = (101 - 47)/8 = 6.75$$

and

$$DD^2f(x_0) = (DDf(x_1) - DDf(x_0))/(x_2 - x_0) = .3139$$

Since x_0 and x_1 are involved in computing $DDf(x_0)$ and x_1 and x_2 are involved in computing $DDf(x_1)$, we associate the interval x_0 to x_2 with $DD^2f(x_0)$. The reader should convince himself that the values in the table are correct.

NEWTON'S FORWARD DIVIDED DIFFERENCES FORMULA

The Newton Forward Divided Difference Polynomial Interpolation formula can be obtained from the ordinary difference formula by making the following substitutions:

(i) Substitute $DD^kf(x_0)$ for $\Delta^kf(0)$ $k=1,2,\ldots,n$

(ii) For every factor $(x-i)$ substitute $(x-x_i)$ $i=1,2,\ldots,n$

Note that x is equivalent to (x-0).

We get the following formula:

$$p(x) = f(0) + (x - x_0)DDf(x_0) + \frac{(x - x_0)(x - x_1)}{2!} DD^2f(x_0)$$

$$+ \frac{(x - x_0)(x - x_1)(x - x_2)}{3!} DD^3f(x_0)$$

$$+ \frac{(x - x_0)(x - x_1)(x - x_2)(x - x_3)}{4!} DD^4f(x_0)$$

The reader can see how the above formula is easily generalized to degree n.

LIMITATIONS OF POLYNOMIAL INTERPOLATION

We will conclude this section with a few remarks about polynomial interpolation which point up some of its weaknesses, before proceeding to the section on splines.

The polynomial interpolation formulas shown above are actually equations for collocation polynomials. A collocation polynomial, p(x), is one which agrees in value with the function f(x) at certain points (those given in the table). There is a more sophisticated type of polynomial interpolation which produces an osculation polynomial. An osculation polynomial, p(x), of degree n and order k, is one which agrees in value with f(x) as well as its first k derivatives at the given values of x:

$$p(x) = f(x)$$
$$p^{(1)}(x) = f^{(1)}(x)$$
$$\vdots$$
$$p^{(k)}(x) = f^{(k)}(x)$$

Formulas for osculatory interpolation exist and they may be found in most numerical analysis texts. Both collocatory and osculatory interpolation are global interpolation schemes, since each value in the table helps to determine at least one of the coefficients and the resulting polynomial is defined over the interval $[x_0,x_n]$. When n is large, interpolation between arguments which are spaced relatively far apart may be poor due to the "wiggly" nature of high degree polynomials. For this reason polynomial interpolation may be unsuitable for certain applications.

At this point the reader should have a basic understanding of polynomial interpolation. We will build upon this foundation in the next few sections which deal specifically with spline function and curve interpolation.

III. SPLINE FUNCTIONS

Spline interpolation enjoys all of the advantages of polynomial interpolation and none of the disadvantages. Splines are basically piecewise polynomial functions which meet certain continuity criteria at the locations where the pieces are joined together. They allow the user to approximate a table of discrete values with several low degree polynomials instead of a single

high degree polynomial, thus avoiding the unwanted oscillations of the high degree polynomials. Figure 7 shows an example of a cubic spline through a set of data values.

This section will begin with a formal definition of a spline function, proceed to a particular representation for cubic spline functions, and finally discuss the various end conditions which may apply.

DEFINITION: A spline function S(x) of degree M-1 (order M) defined on the knot vector:
$(x_0, y_0), (x_1, y_1), \ldots, (x_n, y_n)$ satisfying the constraint $x_0 < x_1 < \ldots < x_n$ is a piecewise function such that:

(i) S(x) is a polynomial of degree M-1 on each subinterval $[x_i, x_{i+1}]$ for $i=0, 1, \ldots n-1$

(ii) S(x) and its first M-2 derivatives are continuous on the interval $[x_0, x_n]$

(iii) S(x) goes through the points of the knot vector - i.e. $S(x_i) = y_i$ for $i=0,1,\ldots,n$.

For each of the interior knots of the knot vector, the constraints that $S(x_i) = y_i$ and that the first M-2 derivatives be continuous give rise to constraint equations which, when solved, define the polynomial coefficients on each subinterval. Since there are n-2 equations in n unknowns, it is necesary to supply the (M-2)nd derivative at the initial and terminal knots, in order to solve

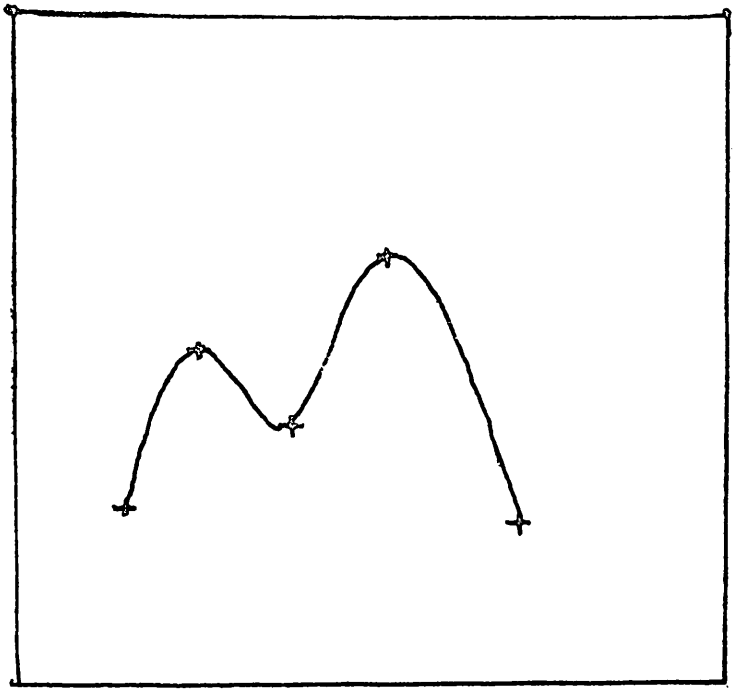


FIGURE 7 CUBIC SPLINE THROUGH 4 KNOTS

sub-interval	coefficients				polynomial
$[x_0,x_1]$	c_0^0	c_1^0	c_2^0	c_3^0	$P_0(x)=c_0^0+c_1^0x+c_2^0x^2+c_3^0x^3$
$[x_1,x_2]$	c_0^1	c_1^1	c_2^1	c_3^1	$P_1(x)=c_0^1+c_1^1x+c_2^1x^2+c_3^1x^3$
.					
.					
.					
$[x_{n-1},x_n]$	c_0^{n-1}	c_1^{n-1}	c_1^{n-1}	c_3^{n-1}	$P_{n-1}(x)=c_0^{n-1}+c_1^{n-1}x+c_2^{n-1}x^2+c_3^{n-1}x^3$

FIGURE 8 PIECEWISE POLYNOMIAL REPRESENTATION

the constraint equations. For a complete derivation of the constraint equations and their solution see Appendix E. Notice that the knot vector and the values of the (M-2)nd derivative at the terminal knots completely specify a degree M-1 spline through the knot vector. Different values for the (M-2)nd derivative at the terminal knots will produce different spline coefficients.

Now let us focus our discussion on <u>cubic</u> spline interpolation. Since a cubic spline is a cubic polynomial on each of the subintervals of the knot vector, we need four coefficients to describe each polynomial. Because there are n subintervals for the knot vector: $(x_0, y_0), (x_1, y_1), \ldots, (x_n, y_n)$, we need 4n coefficients in all as shown in Figure 8. The cubic spline function S(x) is defined to be:

$$S(x) = \begin{cases} p_0(x) & \text{if } x_0 \leq x \leq x_1 \\ p_1(x) & \text{if } x_1 \leq x \leq x_2 \\ \vdots & \\ p_{n-1}(x) & \text{if } x_{n-1} \leq x \leq x_n \end{cases} \qquad \text{(III.1)}$$

The procedure for determining the cubic spline which goes through the given knot locations and satisfies the given end conditions

must produce the polynomial coefficients in figure 8. In fact, the more efficient algorithms represent the polynomial functions $p_0(x), p_1(x), \ldots, p_{n-1}(x)$ in an alternative form which is non-redundant and lends itself to more efficient computation. Consider just one of the polynomials above

$$p_0(x) = c_0^0 + c_1^0 x + c_2^0 x^2 + c_3^0 x^3$$

defined on $[x_0, x_1]$ and satisfying the constraints $p_0(x_0) = y_0$ and $p_0(x_1) = y_1$; i.e. the polynomial $p_0(x)$ goes through the knots (x_0, y_0) and (x_1, y_1). If we define a variable $u = x - x_0$, then there is a polynomial function in u, say

$$g(u) = a_0 + a_1 u + a_2 u^2 + a_3 u^3$$

which describes the same polynomial as $p_0(x)$. In order to derive $g(u)$ from $p_0(x)$, first we must force $g(u)$ to go through (x_0, y_0). When $x=x_0$, $u=0$ thus we have $g(0) = a_0$, which implies that $a_0 = y_0$. Next we require $g(u) = p_0(x)$ for $x_0 < x \leq x_1$.

$$g(u) = a_0 + a_1 u + a_2 u^2 + a_3 u^3$$

$$g(u) = a_0 + a_1(x - x_0) + a_2(x - x_0)^2 + a_3(x - x_0)^3 \qquad \text{(III.2)}$$

Expanding (III.2)

$$g(u) = a_0 + a_1 x - a_1 x_0 + a_2 x^2 - 2a_2 x x_0 + a_2 x_0^2 + a_3 x^3 - 3a_3 x^2 x_0 + 3a_3 x x_0^2 - a_3 x_0^3 \qquad \text{(III.3)}$$

and collecting terms, yields

$$\begin{aligned} g(u) = {} & (a_0 - a_1x_0 + a_2x_0^2 - a_3x_0^3) + \\ & (a_1 - 2a_2x_0 + 3a_3x_0^2)x + \\ & (a_2 - 3a_3x_0)x^2 + \\ & a_3x^3 \end{aligned} \qquad \text{(III.4)}$$

Since we require that $g(u) = p_0(x)$ we get the following equations by matching coefficients in (III.4) with the coefficients of $p_0(x)$,

$$\begin{aligned} c_0^0 &= a_0 - a_1x_0 + a_2x_0^2 + a_3x_0^3 \\ c_1^0 &= a_1 - 2a_2x_0 + 3a_3x_0^2 \\ c_2^0 &= a_2 - 3a_3x_0 \\ c_3^0 &= a_3 \end{aligned} \qquad \text{(III.5)}$$

At this point, let us recall the relationships between a cubic polynomial function and its derivatives.

$$\begin{aligned} p(x) &= c_0 + c_1x + c_2x^2 + c_3x^3 \\ p^{(1)}(x) &= c_1 + 2c_2x + 3c_3x^2 \\ p^{(2)}(x) &= 2c_2 + 6c_3x \\ p^{(3)}(x) &= 6c_3 \end{aligned}$$

Therefore, we get

$$c_3^0 = \frac{p_0^{(3)}(x)}{6} = \frac{p_0^{(3)}(x)}{3!} = a_3 \qquad \text{(III.6)}$$

But, since the third derivative of a cubic polynomial function is constant, it does not matter at which point we evaluate the derivative. We can choose x_0 to be that point. Thus

$$c_3^0 = \frac{p_0^{(3)}(x_0)}{3!} = a_3$$

Next, we must determine a_2. By substitution above we get

$$p_0^{(2)}(x) = 2c_2^0 + 6c_3^0 x_0$$

$$p_0^{(3)}(x) = 6c_3^0$$

Thus

$$p_0^{(2)}(x) = 2c_2^0 + x_0 p_0^{(3)}(x_0)$$

or

$$c_2^0 = \frac{p_0^{(2)}(x_0) - x_0 p_0^{(3)}(x_0)}{2} \qquad \text{(III.7)}$$

but by equation (III.5)

$$c_2^0 = a_2 - 3a_3 x_0 = a_2 - \frac{p_0^{(3)}(x_0)}{2}$$

thus

$$a_2 = \frac{p_0^{(2)}(x_0)}{2!} \qquad \text{(III.8)}$$

By similar manipulation we get $a_1 = p_0^{(1)}(x_0)$. Thus we can write,

$$p_0(x) = y_0 + p_0^{(1)}(x_0)(x - x_0) + \frac{p_0^{(2)}(x_0)}{2!}(x - x_0)^2 + \frac{p_0^{(3)}(x_0)}{3!}(x - x_0)^3 \qquad \text{(III.9)}$$

Equation (III.9) is the alternative representation for a polynomial which is most often used in spline computations; it corresponds to the Taylor series expansion about x_0. Notice the similarity to the Newton Forward Difference Interpolation Formulas.

The cubic spline computation procedures compute the values of the second derivatives

$$p_1^{(2)}(x_1),\ p_2^{(2)}(x_2),\ p_3^{(2)}(x_3),\ldots,\ p_{n-1}^{(2)}(x_{n-1})$$

from the knot vector

$$(x_0, y_0),\ (x_1, y_1),\ldots,\ (x_n, y_n)$$

and the constraint equations. The values of the second derivatives $p^{(2)}(x_0)$ and $p^{(2)}(x_n)$, must be provided by the user in order for the algorithm to have enough constraints to solve for the values at the interior knots. Once the values of the second derivative have been found at the knot locations, the values of the first and third derivatives can easily be obtained from the following formulas.

$$p^{(1)}(x_i) = (y_{i+1} - y_i) - h_i(2p^{(2)}(x_{i+1}))/6 \qquad \text{(III.10)}$$

$$p^{(3)}(x_i) = \frac{p^{(2)}(x_{i+1}) - p^{(2)}(x_i)}{h_i} \qquad \text{(III.11)}$$

where $h_i = x_{i+1} - x_i$. Derivation of (III.10) appears in Appendix A.

Thus, in some sense, the knot vector and the second derivatives at the knot locations uniquely determine the cubic spline through those knot locations. This idea may be generalized to splines of higher degree. For a spline of order M (degree M-1), the values of the (M-2)nd derivative at the knot locations are required. Algorithms[1] for the computation of cubic splines appear in Appendices B and C.

END CONDITIONS

As noted earlier, different end conditions produce different spline functions for the same knot vector. There are three basic approaches to defining end conditions:

(i) Natural End Conditions

The two terminal (M-2)nd derivatives are set to zero:

$$p_0^{(2)}(x_0) = p_{n-1}^{(2)}(x_n) = 0$$

for cubic spline (M=4). This condition corresponds to zero curvature or equivalently, the slope of the spline is constant at the end points.

[1] The algorithms are reprinted with the permission of P. Baudelaire, Xerox PARC, Palo Alto, California.

(ii) Complete End Conditions

When the knot vector consists of points from some underlying function say $g(x)$ and the second derivatives $g^{(2)}(x_0)$ and $g^{(2)}(x_n)$, of g exist and are known at the terminal knots, then we set $p_0^{(2)}(x_0) = g^{(2)}(x_0)$ and $p_{n-1}^{(2)}(x_n) = g^{(2)}(x_n)$.

(iii) Polynomial Approximation of End Conditions

The terminal second derivatives are approximated by finding the cubic polynomial which collocates the first four knots using the Newton Forward Difference Formula. The second derivative of this polynomial is evaluated at the initial knot. After performing the same procedure on the last four knots, the second derivative of that polynomial is evaluated at the terminal knot. These two values are used as the initial and terminal second derivatives.

There is a fourth method which is sometimes used; it is called the "Not-a-Knot" End Condition. Under this method two equations are added to the spline constraint equations which, in effect, inactivate the first and last interior knots. For details see [deB79, p. 53].

It should be noted that there is no requirement that the same end condition method be used at each end of the interval.

IV. SPLINE CURVE INTERPOLATION

In the previous section we discussed spline functions; in this section we will discuss spline curves. For our purposes, a curve is merely a locus of points in the X - Y plane; however, in general there is no restriction on the number of dimensions. The curve may be open or closed (see figure 9). Closed curves are sometimes referred to as periodic and open curves are similarly referred to as nonperiodic. A curve may be expressed in analytic form such as the circle:
$x^2 + y^2 = 9$ or as an ordered list of coordinates (x_1, y_1), (x_2, y_2),..., (x_n, y_n). Suppose that we are given selected points (the knot vector) along a curve and we are told whether the curve is open or closed. How do we go about finding the order M spline which interpolates the knot vector?

PARAMETERIZATION

First we must decide how to parameterize the knot vector. In general a curve may be specified in parametric form if there exists an independent variable, say t, such that the locus of points can be expressed as an ordered pair of functions of the variable t, $(X(t), Y(t))$ and as t increases from some initial value to some terminal value the curve is traced out by evaluating the functions in the ordered pair $(X(t), Y(t))$. If such

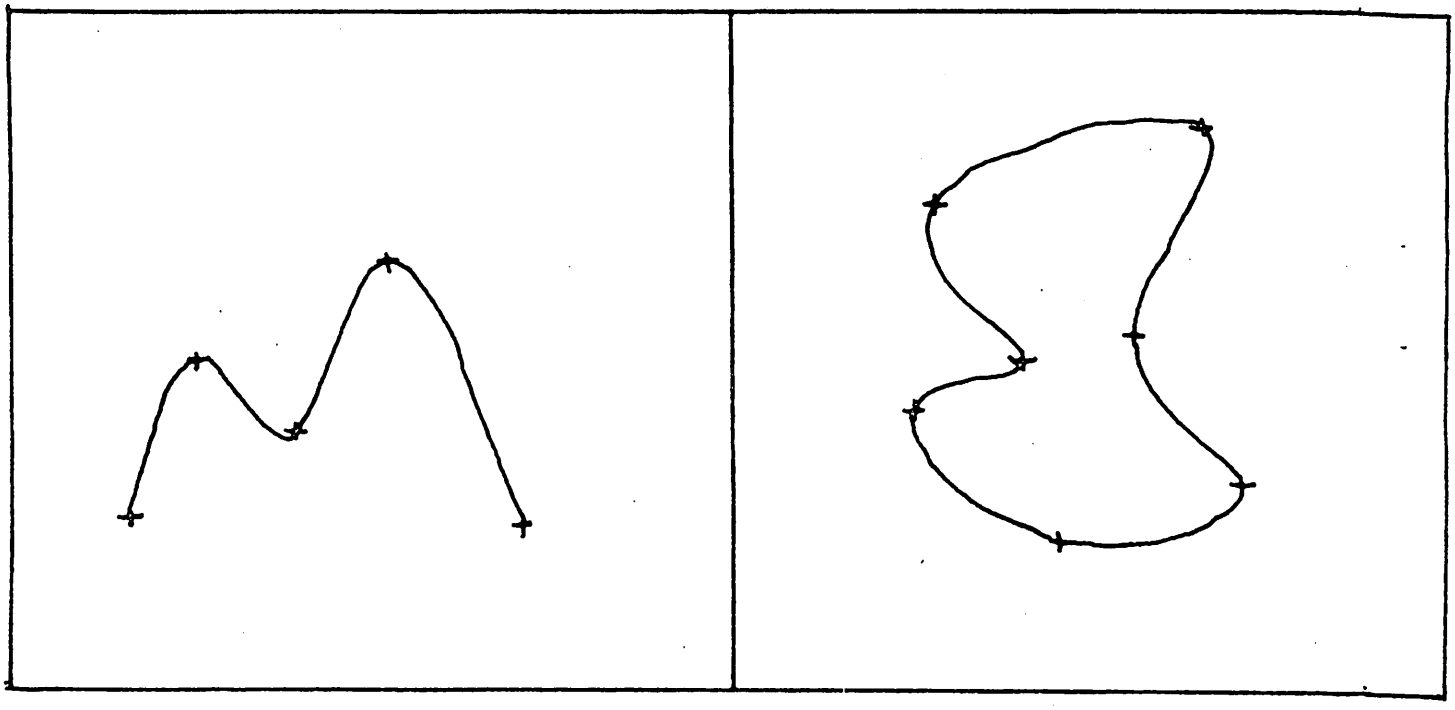
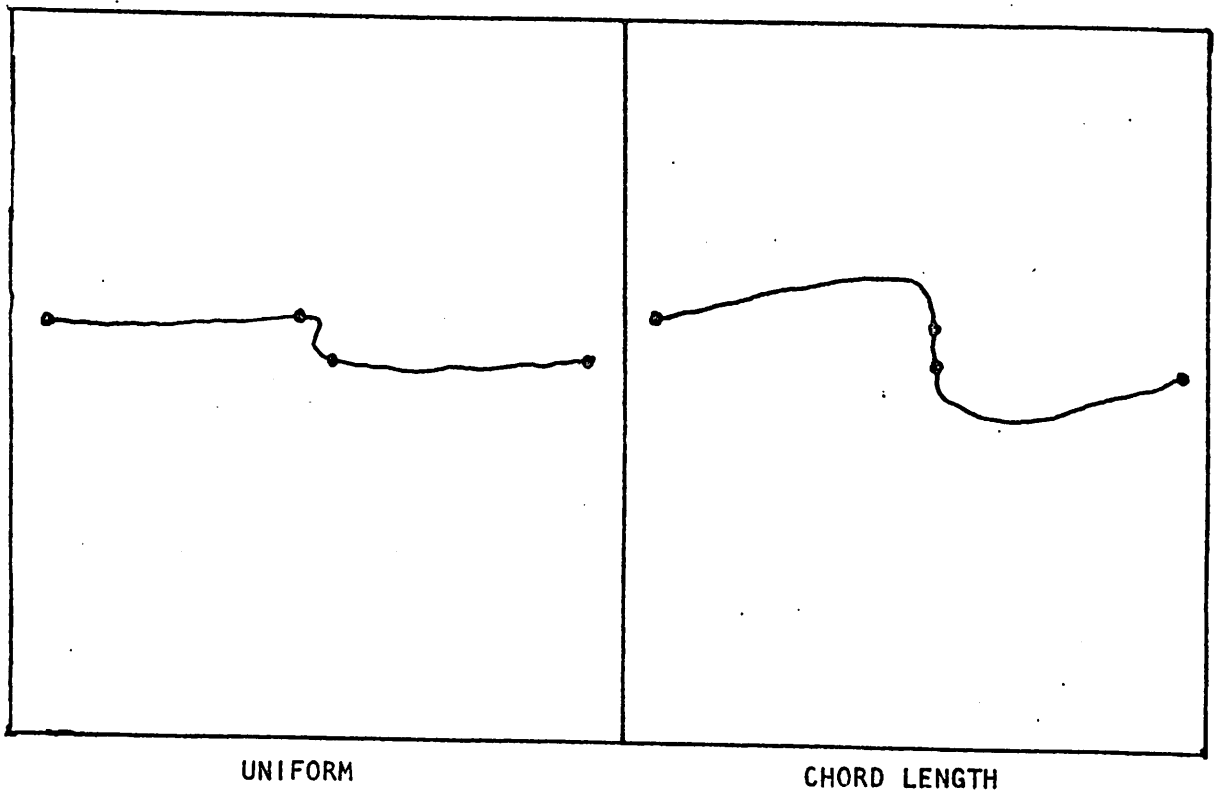


FIGURE 9 OPEN CUBIC SPLINE CURVE CLOSED CUBIC SPLINE CURVE



UNIFORM CHORD LENGTH

FIGURE 10 COMPARISON OF PARAMETERIZATIONS

an independent variable t is a parameter, then it is also true that any monotonically increasing function of t is also a parameter.

The spline interpolation formulas shown in Section III made use of the constraint $x_0 < x_1 < \ldots < x_n$ on the knot vector. It insures that division by zero does not occur in the computation of the divided differences which approximate the spline derivatives. Parameterization allows us to treat the x coordinates of the curve as a function of strictly increasing t and the y-coordinates of the curve as another function of strictly increasing t. In this way we can apply spline function interpolation to the two functions $X(t)$ and $Y(t)$ separately and together these two spline functions are a spline interpolation $(S_X(t), S_Y(t))$ to the curve $(X(t), Y(t))$.

Now let us turn to a procedure for computing the cubic spline curve interpolation to a set of X-Y data values.

Procedure for Natural Cubic Spline Curve Interpolation

Given the knot vector $(x_0, y_0), (x_1, y_1), \ldots, (x_n, y_n)$

1) form the two knot vectors

$(0, x_0), (1, x_1), \ldots, (n, x_n)$

$(0, y_0), (1, y_1), \ldots, (n, y_n)$

In this case the parameter t takes on the integer values 0, 1, . . . n. This is called the uniform parameterization because the t's are integers, and equidistant with a common interval of unity.

2) Use the spline function interpolation algorithms in Appendix B depending on whether the curve is open or closed to produce the spline second derivatives, $S_X^{(2)}$ and $S_Y^{(2)}$ of X(t) and Y(t) at the internal knot locations., given natural end conditions $S_X^{(2)}(0) = S_Y^{(2)}(0) = 0 \quad S_X^{(2)}(n) = S_Y^{(2)}(n) = 0$

$$S_X^{(2)}(0),\ S_X^{(2)}(1),\ldots,S_X^{(2)}(n)$$

and

$$S_Y^{(2)}(0),\ S_Y^{(2)}(1),\ldots,S_Y^{(2)}(n)$$

3) Use the second derivatives to compute first and third derivatives (see algorithms in Appendices B and C). Now evaluate the spline curve $(S_X(t),\ S_Y(t))$ by evaluating the spline functions $S_X(t)$ and $S_Y(t)$ as t ranges over [0, n]. Recall that $S_X(t)$ is evaluated according to the following formula:

$$S_X(t) = x_i + (t - t_i)S_X^{(1)}(t_i) + \frac{(t - t_i)^2}{2!} S_X^{(2)}(t_i)$$

$$+ \frac{(t - t_i)^3}{3!} S_X^{(3)}(t_i) \qquad \text{for } t_i \leq t \leq t_{i+1}$$

This procedure was used to generate the examples which appear in Appendix D. Two important topics which must be discussed are alternative parameterizations and multiple knots. They are presented in the following sections.

CUMULATIVE CHORD LENGTH PARAMETERIZATION

An often used alternative to the uniform parameterization is parameterization by chord length, the discrete analogue of parameterization by arc length. In this case the values of t which correspond to the knots are computed as the cumulative chord length along the knot vector. Thus

$$t_0 = 0$$

$$t_1 = \sqrt{(x_1 - x_0)^2 + (y_1 - y_0)^2}$$

$$t_2 = t_1 + \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

and in general $$t_i = \sum_{j=1}^{i-1} \overline{K_j K_{j+1}}$$

where $\overline{K_j K_{j+1}}$ denotes the Euclidean distance from the jth knot to the (j+1)st knot.

The uniform parameterization may be computed more quickly but it tends to produce curves which are somewhat "flattened". See figure 10. When the distance between successive argument values is larger than unity, the first divided difference is correspondingly smaller and the second divided difference is larger. The second difference is a discrete approximation to curvature and the larger the curvature, the less flattened the curve appears.

MULTIPLE KNOTS

Parameterization of the knot vector allows the same pair of (x,y) coordinates to appear next to each other in the knot vector. This situation is referred to as the "multiple knot" situation. For example the knot vector

(1,6), (2,7), (3,1), (3,1), (3,1), (4,9), (5,11)

may still be interpolated under a uniform parameterization. Under the uniform parameterization we get the two knot vectors

(t_i, x_i) $i=0,\ldots,6$

(0,1), (1,2), (2,3), (3,3), (4,3), (5,4), (6,5)

(t_i, y_i) $i=0,\ldots,6$

(0,6), (1,7), (2,1), (3,1), (4,1), (5,9), (6,11)

Spline functions may be interpolated to these knot vectors using the algorithm in Appendix B, since parameter t is strictly increasing.

Under the cumulative chord length parameterization we get the two knot vectors

(t_i, x_i) $i=0,\ldots,6$

(0,1), (1.414,2), (7.497,3), (7.497,3), (7.497,3), (15.559,4), (17.795,5)

(t_i, y_i) $i=0,\ldots,6$

(0,6), (1.414,7), (7.497,1), (7.497,1), (7.497,1), (15.559,9), (17.795,11)

where $1.414 \simeq \sqrt{2}$

$7.497 \simeq \sqrt{2} + \sqrt{37}$

$15.559 \simeq \sqrt{2} + \sqrt{37} + \sqrt{65}$

$17.795 \simeq \sqrt{2} + \sqrt{37} + \sqrt{65} + \sqrt{5}$

The t_i are not strictly increasing. Hence, if we try to apply the spline algorithm for non-uniform knot vectors, the algorithm will fail when it attempts to compute the necessary divided differences. Since $t_i = t_j$ for some $i \neq j$ in the knot vector, the algorithm must attempt to divide by zero at some point.

The effect of a multiple knot on the resulting spline is to reduce the degree of continuous differentiability of the spline

function at the knot location. From the definition at the beginning of section III, we recall that, for a spline of order M, the spline and its first M-2 derivatives are continuous over the whole interval. When we introduce a knot of multiplicity 2, the spline and its first M-1 derivatives are continuous everywhere in the interval; however, the M-2nd derivative is discontinuous at the location of the multiple knot. In a sense, we no longer have a spline under the strict definition. Each increase in the multiplicity of the knot causes a corresponding decrease in the continuous differentiability of the spline. See the examples in Appendix D. Splines with multiple knots are sometimes referred to as "deficient splines" [deB79] because of their lack of continuous differentiability.

To understand why multiple knots reduce the continuous differentiability of the spline, we must look carefully at the constraint equations. Only an intuitive explanation will be provided here.

When a knot of multiplicity 2 is introduced, it forces the first difference at the knot location to zero. When the multiplicity is iscreased by 1, the second difference is forced to zero, and so on (see figure 11). Each successive difference being forced to zero, in effect, disconnects the constraint equations which join the polynomials on the intervals adjacent to the multiple knot. Disconnecting the constraint equations forces

FIGURE 11(a)

t_i	y_i	Δ	Δ^2	Δ^3	Δ^4
1	1	4	-4	7	-5
2	5	0	3	2	
3	5	3	5		
4	8	8			
5	16				

FIGURE 11(b)

t_i	y_i	Δ	Δ^2	Δ^3	Δ^4
1	1	4	-4	4	7
2	5	0	0	11	
3	5	0	11		
4	5	11			
5	16				

FIGURE 11(c)

t_i	y_i	Δ	Δ^2	Δ^3	Δ^4
1	1	4	-4	-4	-4
2	5	0	0	0	
3	5	0	0		
4	5	0			
5	5				

FIGURE 11 SUCCESSIVE ZERO DIFFERENCES RESULTING FROM MULTIPLE KNOTS

the value of the left limit of the proper derivative to differ from the value of the right limit of the proper derivative. A discontinuity occurs when the left limit of a function at a given point differs from the right limit at the same point.

In the last two sections we have limited our discussion to to piecewise polynomial (PP) representations of splines. There are at least four different PP-representations [deB68], possessing varying degrees of redundancy. The PP scheme which represented a cubic spline with 4 coefficients c_0, c_1, c_2, c_3 on each of n subintervals is redundant in the sense that only n coefficients and the rules for integrating and differentiating polynomials are required to completely specify the spline. The particualr scheme employed in this section is the only non-redundant PP-representation; however, there is an alternative representation which is <u>not</u> a PP-representation. It is known as "B-spline representation" and it is the subject of the next section. Some additional mathematical concepts are necessary for a complete understanding of B-splines. The notions of linear space, linear combination, and basis function are required.

V. B-SPLINE INTERPOLATION

B-splines are merely an alternative form in which to represent the splines presented earlier in this paper. Given a specific knot vector:

$\{(x_0,y_0), (x_1,y_1),\ldots,(x_n,y_n)\}$ such that $x_0 < x_1 < \ldots < x_n$

we will denote by X the vector of x-values $X = \{x_0, x_1,\ldots,x_n\}$ and we will call X the mesh. It has been shown that the set of all spline functions of order M defined over a mesh X forms a linear space of functions which we will denote by S(M, X). This space has dimension n (the number of intervals in the mesh) and thus there are n basis functions of degree M-1 for this space. These n basis functions are referred to as the "B-spline basis" for S(M, X) and every spline function in S(M, X) may be written as a linear combination of these basis functions. Thus if S(x) belongs to S(M, X) then S(x) may be written

$$S(X) = \sum_{j=1}^{n} a_j B_j(x)$$

where each B_j is a B-spline basis function and each a_j is a real number. Before proceeding to B-spline interpolation, let's take a look at the B-spline basis functions in more detail.

B-SPLINE BASIS

First, two simple definitions are required:

DEFINITION 1: The restriction of the spline function S(x) to the subinterval (x_i, x_{i+1}) is called the ith span of S.

DEFINITION 2: The support of a function is the interval over which the function has non-zero values.

Every B-spline basis function, B_j, for the space S(M, X) is associated with some mesh value x_i and a finite support of M spans -- namely the interval $(x_i, x_{i+M \bmod n})$. B-spline basis functions may be defined in various ways; however, only two methods will be presented here. First we define the even order (M even) B-spline on the uniform mesh covering the interval [-M/2, M/2] as the function

$$B_{M-1}(x) = \frac{1}{r!} \sum_{j=-k}^{k} (-1)^{j+k} \binom{2k}{j+k} (j-x)_+^r \qquad (V.1)$$

where

$$M = 2k$$
$$r = 2k-1$$

where $(j-x)_+^r$ is the truncated power function.

The truncated power function is defined to be

$$(j-x)_+^r = \begin{cases} (j-x)^r & \text{if } j > x \\ 0 & \text{otherwise} \end{cases}$$

The function $B_{M-1}(x)$ has the following properties: It is symmetric about $x=0$, bell-shaped, and has support $[-M/2, M/2]$. The derivatives $B^{(p)}_{M-1}(x)$ for $p=1,2,\ldots,M-1$ are given by the following formula:

$$B^{(p)}_{M-1}(x) = \frac{1}{(r-p)!} \sum_{j=-k}^{k} (-1)^{j+k+p} \binom{2k}{j+k} (j-x)_+^{r-p} \qquad (V.2)$$

A graph of $B_3(t)$ appears in Figure 12. Because of the redundant calculations and the loss of arithmetic precision involved in the computation of the B-spline definition in equation (V.1), an alternative definition is often used which avoids these pitfalls. This alternative definition is recursive and uses the convention $0/0=0$. Throughout the remainder of this paper the notation $B_{i,M}(x)$ will be used instead of $B_{M-1}(x)$. $B_{i,M}(x)$ denotes the order M B-spline basis function associated with the mesh point x_i. For example, $B_{0,4}(x)$ denotes the cubic (M=4) B-spline basis function whose support consists of the interval $[x_0,x_4]$. The interval $[x_0,x_4]$ is the union of the 4 spans $[x_0,x_1]$, $[x_1,x_2]$, $[x_2,x_3]$, $[x_3,x_4]$.

RECURSIVE DEFINITION OF B-SPLINE

$$B_{i,1}(x) = \begin{cases} 1 & \text{if } x_i \le x \le x_{i+1} \\ 0 & \text{otherwise} \end{cases}$$

$$B_{i,M}(x) = \frac{x - x_i}{x_{i+M-1} - x_i} B_{i,M-1}(x) + \frac{x_{i+M} - x}{x_{i+M} - x_{i+1}} B_{i+1,M-1}(x)$$

for $M>1$

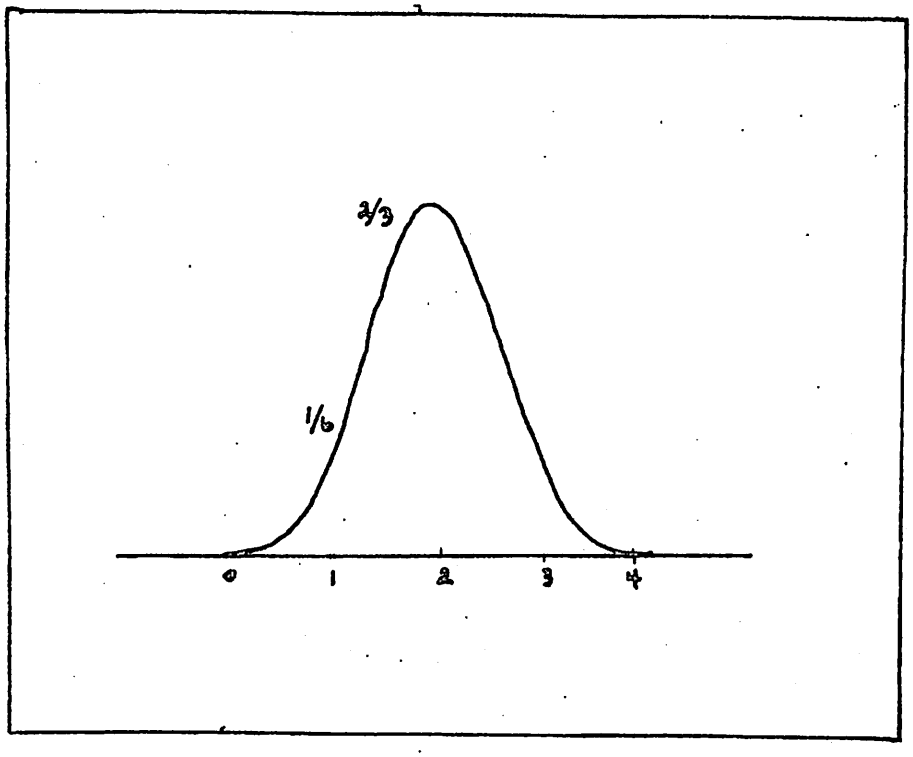


FIGURE 12 UNIFORM CUBIC B-SPLINE BASIS FUNCTION

Sample B-spline basis functions are shown in figure 13. The reader should note that for the periodic case, the B-spline basis functions on a uniform mesh are merely cyclic translates of a single canonical B-spline basis function. For example each basis function in figure 13(a) is a cyclic translate of $B_{0,2}$ and each basis function in figure 13(b) is a cyclic translate of $B_{0,3}$. This situation does <u>not</u> occur in the non-periodic case. The end conditions cause the basis functions at the ends of the interval to differ from the interior basis functions. The end conditions manifest themselves as multiple knots or multiple mesh values at the ends of the interval. Figure 13(c) is an example of non-periodic cubic B-spline basis functions on the mesh $X = \{0, 0, 0, 1, 2, 3, 4, 5, 6, 6, 6\}$.

Cox and deBoor have developed an algorithm for evaluation of arbitrary B-splines and their derivatives, which overcomes the inherent numerical instabilities of previous algorithms. The algorithm is recursive and it is based on the recursive definition for B-spline basis functions. The interested reader will find the algorithm described in Appendix G as well as several of the references [deB72, GOR74, deB79].

Throughout the remainder of the paper we will focus our attention on cubic B-splines. The reader should note an interesting fact which we will utilize in the following section. The cubic B-spline canonical basis function $B_{0,4}$ may be

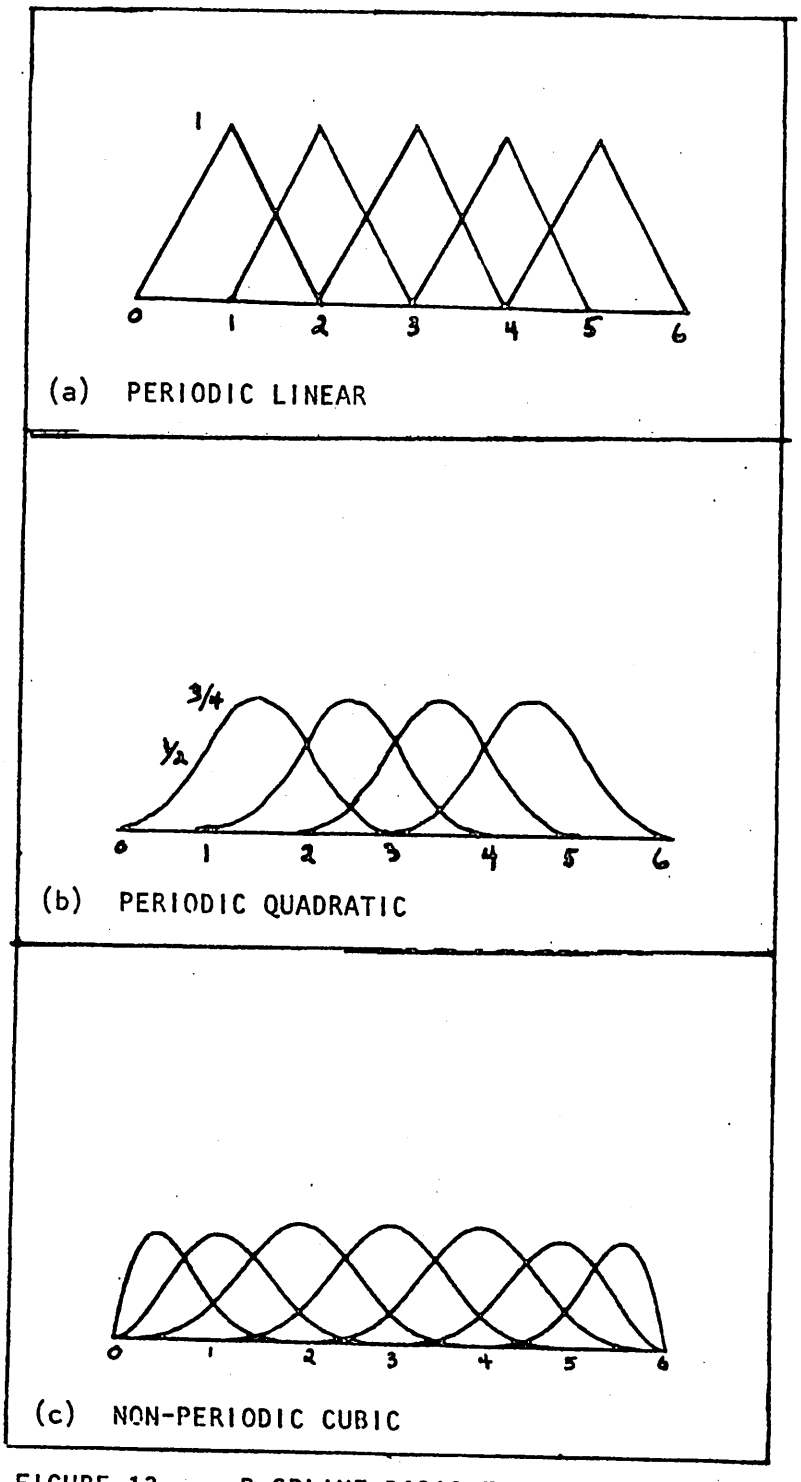


FIGURE 13 B-SPLINE BASIS FUNCTIONS

represented as a weighted sum of four polynomial functions

$$E_0(t) = -1/6t^3 + 1/2t^2 - 1/2t + 1/6$$

$$E_1(t) = 1/2t^3 - t^2 + 2/3 \qquad (V.3)$$

$$E_2(t) = -1/2t^3 + 1/2t^2 + 1/2t + 1/6$$

$$E_3(t) = 1/6t^3$$

The weights are the coordinate values at the vertices of the "defining polygon" for the cubic B-spline. Thus the value of an arbitrary B-spline function S(t) for some parametric value $0 \leq t \leq 1$ is

$$S(t) = E_0(t)V_i + E_1(t)V_{i+1} + E_2(t)V_{i+2} + E_3(t)V_{i+3} \qquad (V.4)$$

where V_i, V_{i+1}, V_{i+2}, V_{i+3} represent four consecutive vertices of the defining polygon for the B-spline S(t). We must now define precisely the idea of a defining polygon, explain how it is obtained, and describe its computational use.

Given a knot vector (x_0, y_0), $(x_1, y_1),\ldots,(x_n, y_n)$ for a cubic spline S(x), there exists a set of points (x_0', y_0'), $(x_1', y_1'),\ldots,(x_k', y_k')$ related to the knots such that each point on the curve S(x) may be represented as a <u>convex</u> combination of 4 consecutive points from

(x'_i,y'_i), $(x'_{i+1},y'_{i+1}),\ldots,(x'_{i+3},y'_{i+3})$. This ordered list of points is called the "defining polygon" and each point is a vertex of the defining polygon. Notice that the list of points may define either an open or a closed polygon and that the spline curve does <u>not</u> go through the points of the defining polygon. The defining polygon is obtained from the knot vector by a procedure called "inversion" and in general there are more points in the defining polygon than there are in the knot vector. An efficient inversion procedure due to Yamaguchi [YAM78] appears in Appendix H. Examples of cubic B-splines and their defining polygons appear in figure 14. Note also figure 15 which contains the four polynomials of the canonical cubic B-spline basis function. The inversion of the knot vector creates the B-spline representation from the PP-representation and allows us to more compactly represent the spline S(x). It should be obvious to the reader that we may parameterize the defining polygon by the same methods used for the knot vector. Thus B-spline curves may be interpolated as easily as B-spline functions. What remains to be shown is a method of evaluating a B-spline curve given its defining polygon. The formula is as follows

$$p(x) = [s^3\ s^2\ s\ 1]\ 1/6 \begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 0 & 3 & 0 \\ 1 & 4 & 1 & 0 \end{bmatrix} \begin{bmatrix} Y_i \\ Y_{i+1} \\ Y_{i+2} \\ Y_{i+3} \end{bmatrix} \qquad (V.5)$$

where i = integer part of n · x
s = fractional part of n · x
n = number of vertices - 3

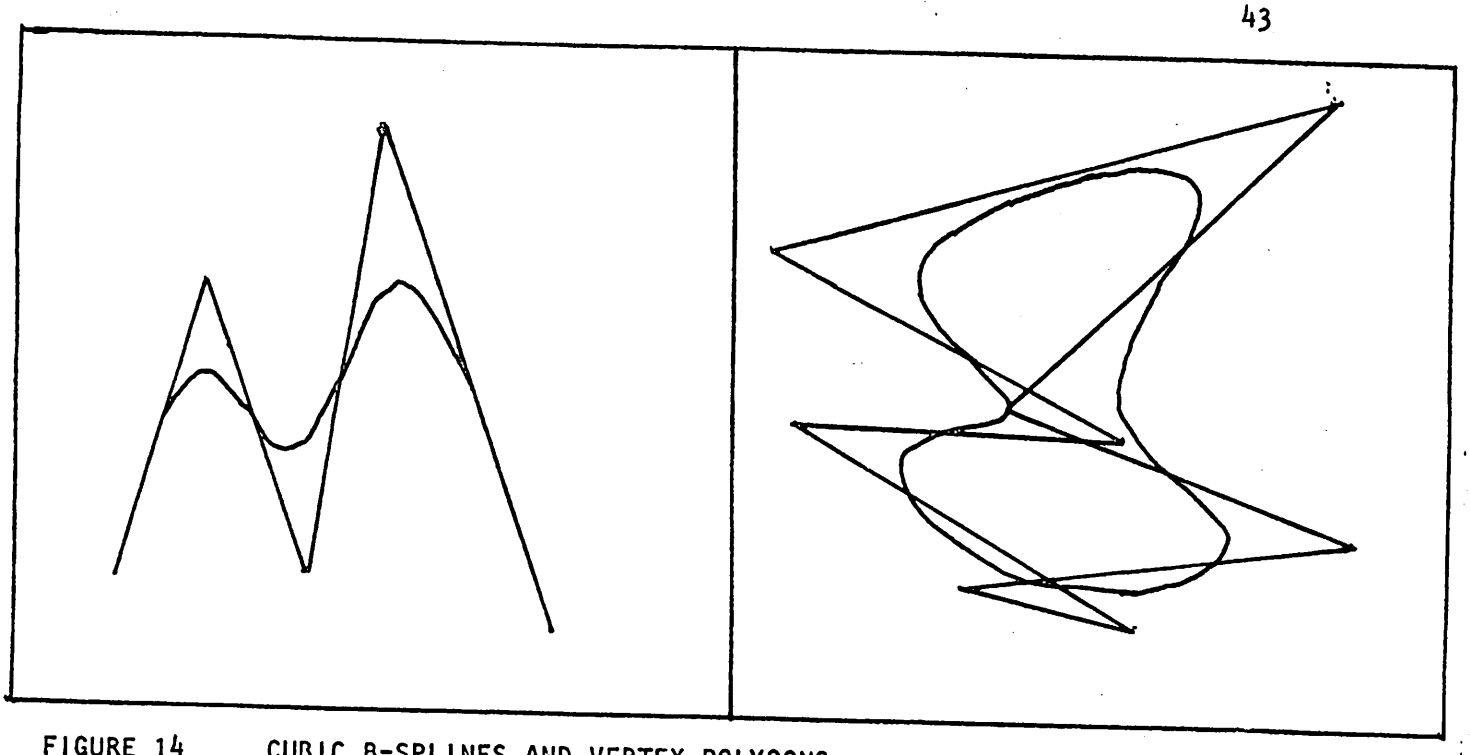


FIGURE 14 CUBIC B-SPLINES AND VERTEX POLYGONS

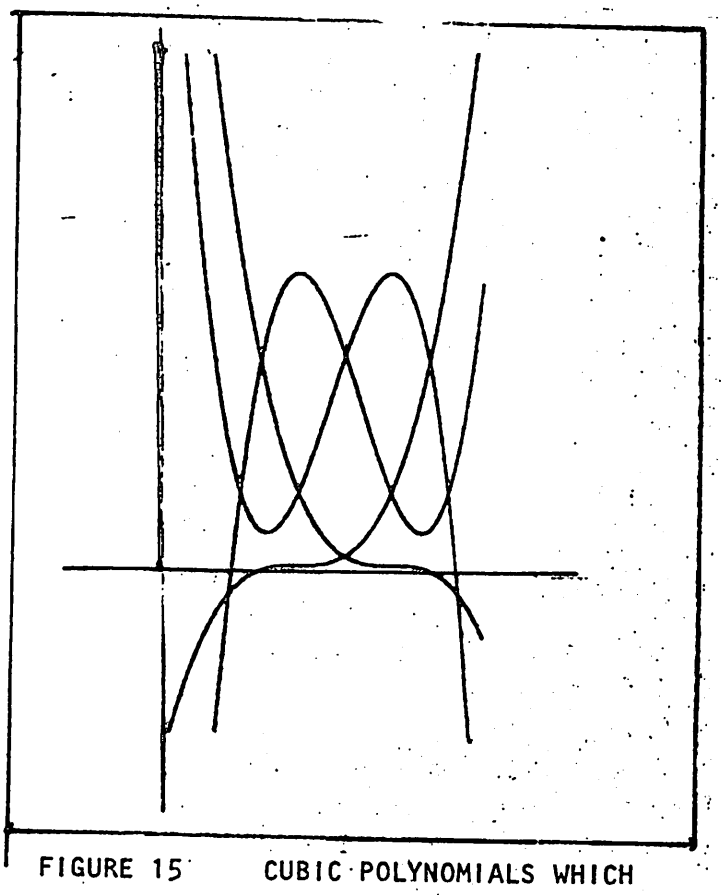


FIGURE 15 CUBIC POLYNOMIALS WHICH COMPRISE UNIFORM CUBIC B-SPLINE BASIS FUNCTION

The formula (V.2) may be generalized to any number of dimensions. For example, in three dimensions each vertex V_i has three coordinates (x_i, y_i, z_i) and P(x) is a point vector in 3-space. Instead of x we use the parameter t and let t range over the interval [0, k]. We evaluate P(t) for the x-coordinates of the successive vertices V_i, V_{i+1}, V_{i+2}, V_{i+3}, then for the y-coordinates, and finally for the z-coordinates. Notice that each of the column vectors in the matrix of formula (V.5) when multiplied by the row vector $[s^3\ s^2\ s\ 1]1/6$ produce the four polynomials in (V.3) which make up the canonical uniform cubic B-spline basis function. Before concluding this report, we will briefly mention some of the useful properties of B-splines.

PROPERTIES OF B-SPLINES

B-splines exhibit certain nice properties which we will summarize briefly:

1) Local approximation

Because of the local support of B-spline basis functions, B-spline approximation is a local approximation.

2) Variation diminishing

The approximation is always smoother than the underlying primitive function. Thus B-splines approximate straight lines exactly.

3) Convex hull

For a B-spline curve of degree M-1, any given point lies within the convex hull of the neighboring M vertices. (see Figure 16) These properties are clearly illustrated in [GOR74].

CONCLUDING REMARKS

This report has taken the reader from simple linear interpolation to B-spline representation and approximation of curves without imposing severe mathematical demands. The reader, who feels he has a possible application for splines, should carefully read the references, particularly [AHL67, deB79] to gain a more complete understanding before determining the suitability of splines to his particular problem. For the reader with a knowledge of the FORTRAN programming language and access to a computer system, a collection of spline programs may be obtained (on magnetic tape) from

IMSL
Sixth Floor
GNB Building
7500 Bellair Blvd.
Houston, Texas 77036

These programs were written by Carl deBoor and they are presented in his book, "A Practical Guide to Splines" [deB79]. Appendix D contains certain examples of spline curves which were generated using the algorithms in Appendices B and C coded in the APL programming language and plotted on a Tektronix 4013 display.

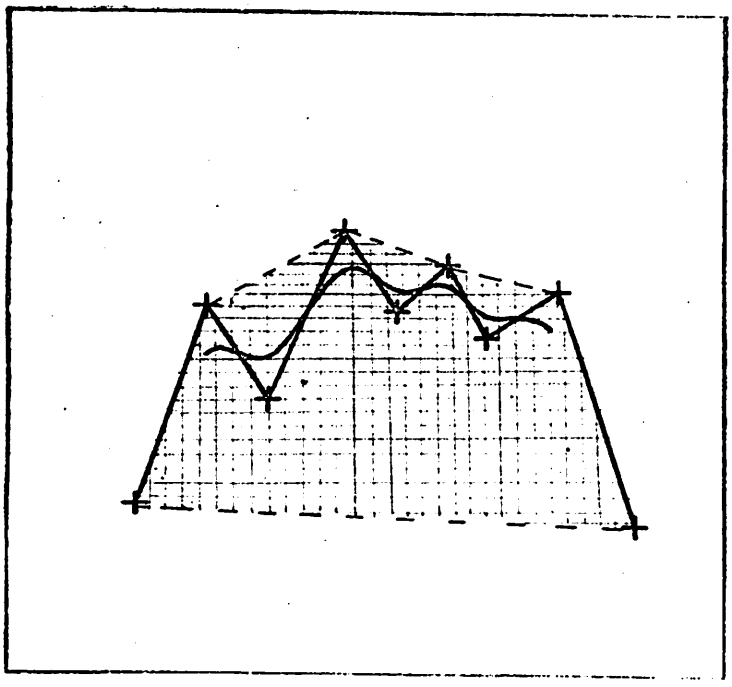


FIGURE 16(a) GLOBAL CONVEX HULL

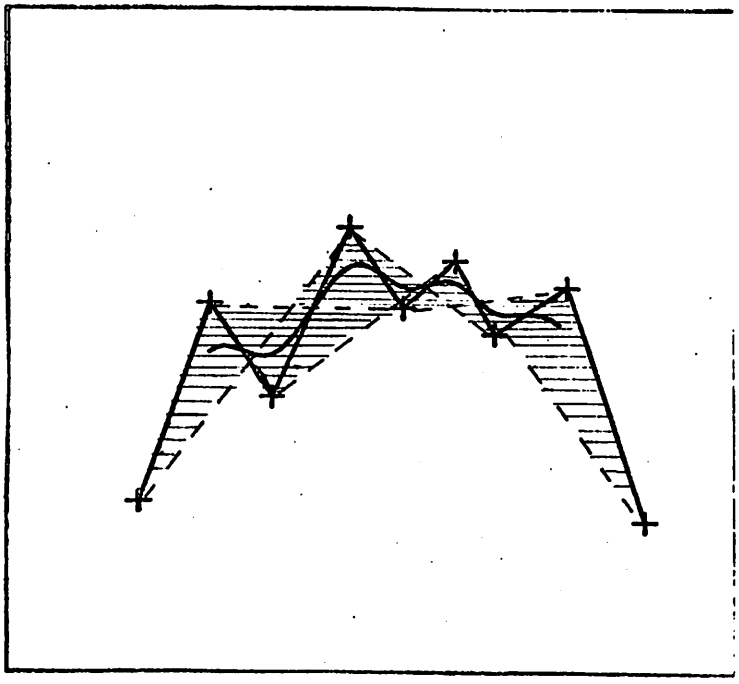


FIGURE 16(b) LOCAL CONVEX HULL OF B-SPLINE

FIGURE 16 STRONG CONVEX HULL PROPERTY OF B-SPLINES ILLUSTRATED IN (b) AS COMPARED TO USUAL CONVEX HULL IN (a)

REFERNCES

1. J. H. Ahlberg, E. N. Nilson, J. L. Walsh, The Theory of Splines and their Applications, Academic Press, New York, 1967

2. P. Baudelaire, R. M. Flegal, R. F. Sproull, "Spline Curve Techniques", Xerox PARC TR, Palo Alto, CA May 1977

3. C. deBoor, A Practical Guide to Splines, Springer-Verlag, New York, 1979

4. C. deBoor and J. R. Rice, "Least Squares Cubic Spline Approximation I - Fixed Knots", CSD TR 20 Purdue University, Lafayette, Indiana, April 1968

5. Freeman, Harry, Mathematics for Actuarial Students Cambridge University Press, Cambridge, England 1939

6. W. Gordon, R. Riesenfeld, "B-Spline Curves and Surfaces", in Computer Aided Geometric Design eds. Barnhill and Riesenfeld, Academic Press, New York, 1974

7. R. Hornbeck, Numerical Methods, Quantum Publishers, inc. New York, 1975

9. F. Yamaguchi, "A New Curve Fitting Method Using a CRT Computer Display", Computer Graphics and Image Processing 7, 425-437. (1978)

ADDITIONAL REFERENCES

1. Bezier, P. Numerical Control-Mathematics and Applications. (translated by A. R. Forrest). London: John Wiley and Sons (1972).
2. deBoor, C. "On Calculating with B-splines". J. Approx. Theory, Vol. 6 (1972), pp. 50-62.
3. Cox, M. G. "The Numerical Evaluation of B-Splines". National Physical Laboratory (Teddington, England), DNAC 4 (August 1971).
4. Forrest, A. R. "Interactive Interpolations and Approximation by Bezier Polynomials". Computer J., Vol. 15 (1972) pp. 71-79.
5. Greville, T. N. E. "Introduction to Spline Functions". in Theory and Applications of Spline Functions. (Greville, ed.), Academic Press 1969.
6. Riesenfeld, R. F. "Applications of B-spline Approximation to Geometric Problems of Computer Aided Design". Ph. D. Thesis, Syracuse U. (1973). Available at U. of Utah, UTEC-CSc-73-126.
7. Schoenberg, I. J. "On Spline Functions". with Supplement by T. N. E. Greville, Inequalities (O. Shisha, editor), Academic press (1967)

pp. 255-291.

8. Schoenberg, I. J. "On Variation Diminishing Approximation Methods". On Numerical Approximation. (R. E. Langer, Ed.) University of Wisconsin Press (1959), pp. 249-274.

9. Schoenberg, I. J. "Contributions to the Problem of Approximation of Equidistant Data by Analytic Functions". Quart. Appl. Math, Vol 4 (1946), pp. 45-99.

APPENDIX A

DERIVATION OF FIRST DERIVATIVES FOR CUBIC SPLINES

In this appendix we will show that $S'(x)$ can be represented in terms of the knot values (x_0,y_0), $(x_1,y_1),\ldots,(x_n,y_n)$ and the second derivatives of the interpolated spline function, namely $S''(x)$, by a series of algebraic manipulations. The purpose of this demonstration is to show that the knot vector and the cubic spline function second derivatives at the knots completely specify the spline function, since the first and third derivatives may easily be computed from the second derivatives.

Let's restate the basic relationships between a cubic polynomial and its derivatives:

$$P(x) = C_0 + C_1x + C_2x^2 + C_3x^3$$
$$P'(x) = C_1 + 2C_2x + 3C_3x^2$$
$$P''(x) = 2C_2 + 6C_3x$$
$$P'''(x) = 6C_3$$

Now it must be shown that $S'(x)$ can be derived from the knot vector (x_0,y_0), $(x_1,y_1),\ldots,(x_n,y_n)$ and the $S''(x)$ values at the knots - namely $S''(x_0)$, $S''(x_1),\ldots,S''(x_n)$. Consider the simple case where the x_i are uniformly spaced, $x_{i+1} = x_i + 1$. Since the spline $S(x)$ must go through the knots, we have:

$$S(x_0) = y_0$$

$$S(x_1) = y_1$$

$$\vdots$$

$$S(x_n) = y_n$$

Let $P(x)$ be the polynomial which represents $S(x)$ on the interval $[x_i, x_{i+1}]$ -i.e. $S(x)$ restricted to $[x_i, x_{i+1}]$

Let $\Delta y_i = y_{i+1} - y_i$

But $y_{i+1} - y_i = P(x_{i+1}) - P(x_i)$

$$= (C_0 + C_1 x_{i+1} + C_2 x_{i+1}^2 + C_3 x_{i+1}^3) -$$

$$(C_0 + C_1 x_i + C_2 x_i^2 + C_3 x_i^3)$$

but since $x_{i+1} = x_i + 1$ and $x_{i+1}^2 = x_i^2 + 2x_i + 1$ and

$x_{i+1}^3 = x_i^3 + 3x_i^2 + 3x_i + 1$ and

$x_{i+1} - x_i = 1$

we get

$$\Delta y_i = y_{i+1} - y_i = C_1(x_{i+1} - x_i) + C_2 x_{i+1}^2 - C_2 x_i^2 +$$

$$C_3 x_{i+1}^3 - C_3 x_i^3$$

$$\Delta y_i = C_1 + 2C_2 x_i + C_2 + 3C_3 x_i^2 + 3C_3 x_i + C_3$$

rearrange terms

$$\Delta y_i = C_1 + 2C_2 x_i + 3C_3 x_i^2 + C_2 + 3C_3 x_i + C_3$$

The first three terms of Δy_i are the formula for $P'(x_i)$ and we have only manipulated expressions for values of the knot vector so far. We must find an expression in terms of $P''(x_i)$ and $P''(x_{i+1})$ for the last three terms of the equation for Δy_i . Consider

$$2P''(x_i) + P''(x_{i+1}) = 2(2C_2 + 6C_3x_i) + (2C_2 + 6C_3x_{i+1})$$

$$= 6C_2 + 12C_3x_i + 6C_3x_{i+1}$$

but $x_{i+1} = x_i + 1$

$$\therefore 2P''(x_i) + P''(x_{i+1}) = 6C_2 + 12C_3x_i + 6C_3x_i + 6C_3$$

$$= 6C_2 + 18C_3x_i + 6C_3$$

Now

$$(2P''(x_i) + P''(x_{i+1}))/6 = C_2 + 3C_3x_i + C_3$$

Notice that the right hand side of this equation is precisely the portion we wish to remove from the equation for Δy_i . Therefore, by substitution we get

$$\Delta y_i = P'(x_i) + (2P''(x_i) + P''(x_{i+1}))/6$$

and

$$P'(x_i) = \Delta y_i - (2P''(x_i) + P''(x_{i+1}))/6$$

Thus it is clear that the first derivatives at the knots can be expressed in terms of the knots and the second derivatives at the knots. The above derivation can be generalized to the case where

the knots are <u>not</u> uniformly spaced,

$$x_{i+1} - x_i = h_i$$

The resulting equation is:

$$P'(x_i) = \Delta y_i - h_i(2P''(x_i) + P''(x_{i+1}))/6$$

or

$$P'(x_i) = (P(x_{i+1}) - P(x_i)) - h_i(2P''(x_i) + P''(x_{i+1}))/6$$

APPENDIX B

Cubic spline on unit mesh -- Algorithmic summary

natural end conditions	periodicity (implies $y_1 = y_N$)
Given N values $y_1, y_2, \ldots, y_N$ 1) for $i = 1$ to N-2, compute $\Delta^2 y_i = y_{i+2} - 2y_{i+1} + y_i$	
	1b) also compute $\Delta y_0 = y_2 - 2y_1 + y_{N-1}$
2) for $i = 1$ to N-3, compute $a_i = 4 - 1/a_{i-1}$ $b_i = \Delta^2 y_{i+1} - b_{i-1}/a_{i-1}$ with $a_0 = 4$ and $b_0 = \Delta^2 y_1$	2a) for $i = 2$ to N-2, compute $a_i = 4 - 1/a_{i-1}$ $b_i = \Delta^2 y_{i-1} - b_{i-1}/a_{i-1}$ $c_i = -c_{i-1}/a_{i-1}$ with $a_1 = 4$, $b_1 = \Delta^2 y_0$, and $c_1 = 1$ 2b) for $i = N-2$ to 1, compute $r_i = -(r_{i+1} + c_i)/a_i$ $s_i = (6b_i - s_{i+1})a_i$ with $r_{N-1} = 1$ and $s_{N-1} = 0$
3) for $i = N-1$ to 2, compute $Y''_i = (6b_{i-2} - Y''_{i+1})/a_{i-2}$ with $Y''_N = Y''_1 = 0$	3) for $i = 1$ to N-2, compute $Y''_i = r_i Y''_{N-1} + s_i$ with $Y''_{N-1} = \dfrac{6\Delta^2 y_{N-2} - s_1 - s_{N-2}}{r_1 + r_{N-2} + 4}$ and $Y''_N = Y''_1$
4) for $i = 1$ to N-1, compute $Y'_i = y_{i+1} - y_i - (2Y''_i + Y''_{i+1})/6$ $Y'''_i = Y''_{i+1} - Y''_i$	

APPENDIX C

Cubic spline -- Algorithmic summary

natural end conditions

Given N knots $(y_1, u_1), (y_2, u_2), \ldots, (y_N, u_N)$

1) for $i = 1$ to $N-1$, compute

$$h_i = u_{i+1} - u_i \quad \text{and} \quad \Delta y_i = (y_{i+1} - y_i)/h_i$$

2) for $i = 1$ to $N-2$, compute

$$d_i = \Delta y_{i+1} - \Delta y_i$$

3) for $i = 1$ to $N-3$, compute

$$a_i = 2(h_{i+1}+h_{i+2}) - h_{i+1}^2/a_{i-1}$$

$$b_i = d_{i+1} - h_{i+1}b_{i-1}/a_{i-1}$$

with $a_0 = 2(h_1 + h_2)$

and $b_0 = d_1$

4) for $i = N-1$ to 2, compute

$$Y''_i = (6b_{i-2} - h_i Y''_{i+1})/a_{i-2}$$

with $Y''_N = Y''_1 = 0$

periodicity (implies $y_1 = y_N$)

2b) also compute

$$h_0 = h_{N-1} = u_N - u_{N-1}$$

$$\Delta y_0 = \Delta y_{N-1} = (y_N - y_{N-1})/h_{N-1}$$

$$d_0 = \Delta y_1 - \Delta y_0$$

3a) for $i = 2$ to $N-2$, compute

$$a_i = 2(h_{i-1} + h_i) - h_{i-1}^2/a_{i-1}$$

$$b_i = d_{i-1} - h_{i-1}b_{i-1}/a_{i-1}$$

$$c_i = -h_{i-1}c_{i-1}/a_{i-1}$$

with $a_1 = 2(h_0 + h_1)$, $b_1 = d_0$, $c_1 = h_0$

3b) for $i = N-2$ to 1, compute

$$r_i = -(h_i r_{i+1} + c_i)/a_i$$

$$s_i = (6b_i - h_i a_{i+1})/a_i$$

with $r_{N-1} = 1$ and $s_{N-1} = 0$

4) for $i = 1$ to $N-2$, compute

$$Y''_i = rY''_{N-1} + s_i \text{ with}$$

$$Y''_{N-1} = \frac{6d_{N-2} - h_0 s_1 - h_{N-2} s_{N-2}}{h_0 r_1 + h_{N-2} r_{N-2} + 2(h_{N-2}+h_0)}$$

and $Y''_N = Y''_1$

5) for $i = 1$ to $N-1$, compute

$$Y'_i = \Delta y_i - h_i(2Y''_i + Y''_{i+1})/6$$

$$Y'''_i = (Y''_{i+1} - Y''_i)/h_i$$

APPENDIX D

EXAMPLES OF CUBIC SPLINES

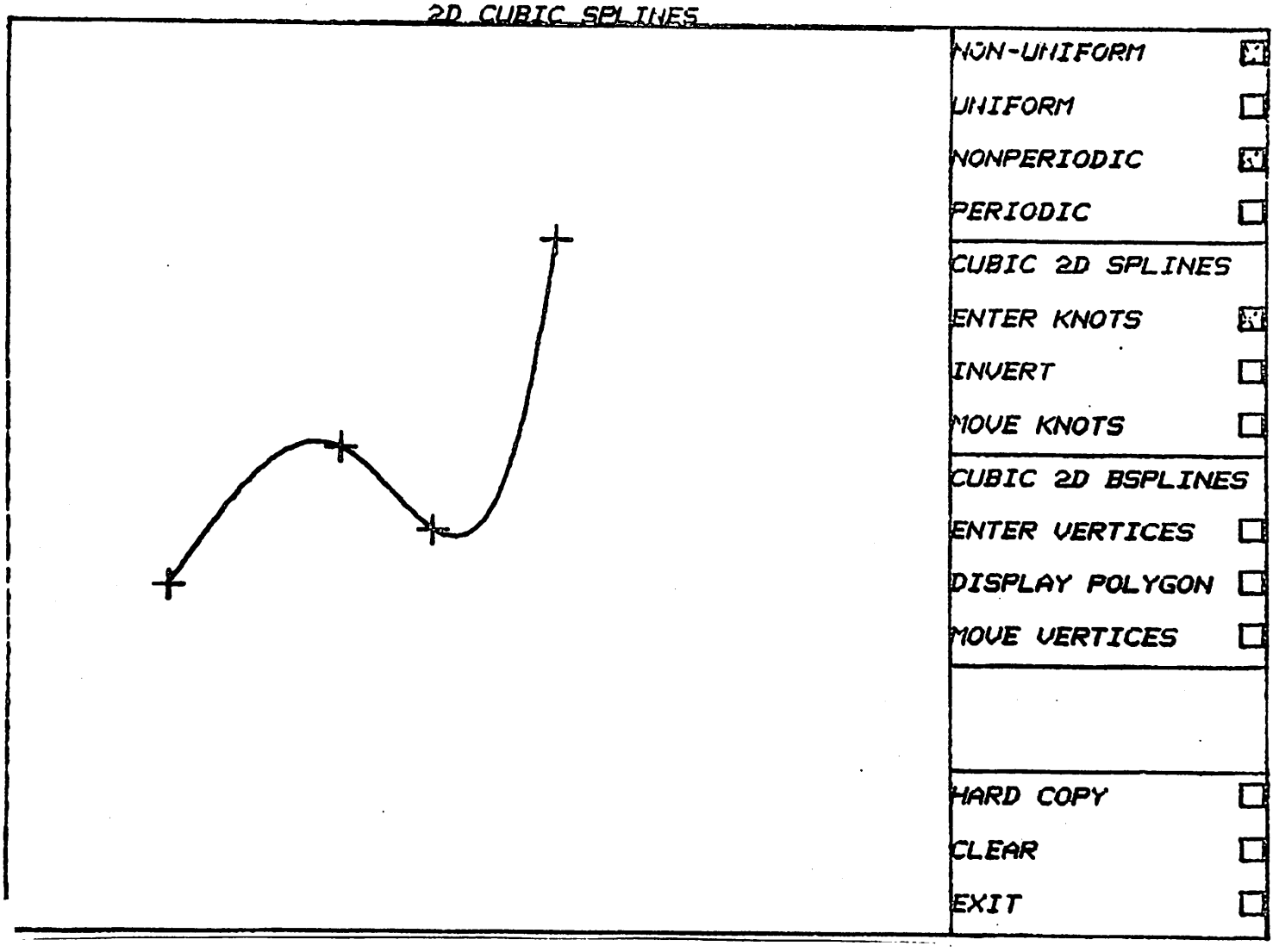


FIGURE D.1 OPEN (NON-PERIODIC) CUBIC SPLINE THROUGH FOUR KNOTS - CHORD LENGTH PARAMETERIZATION

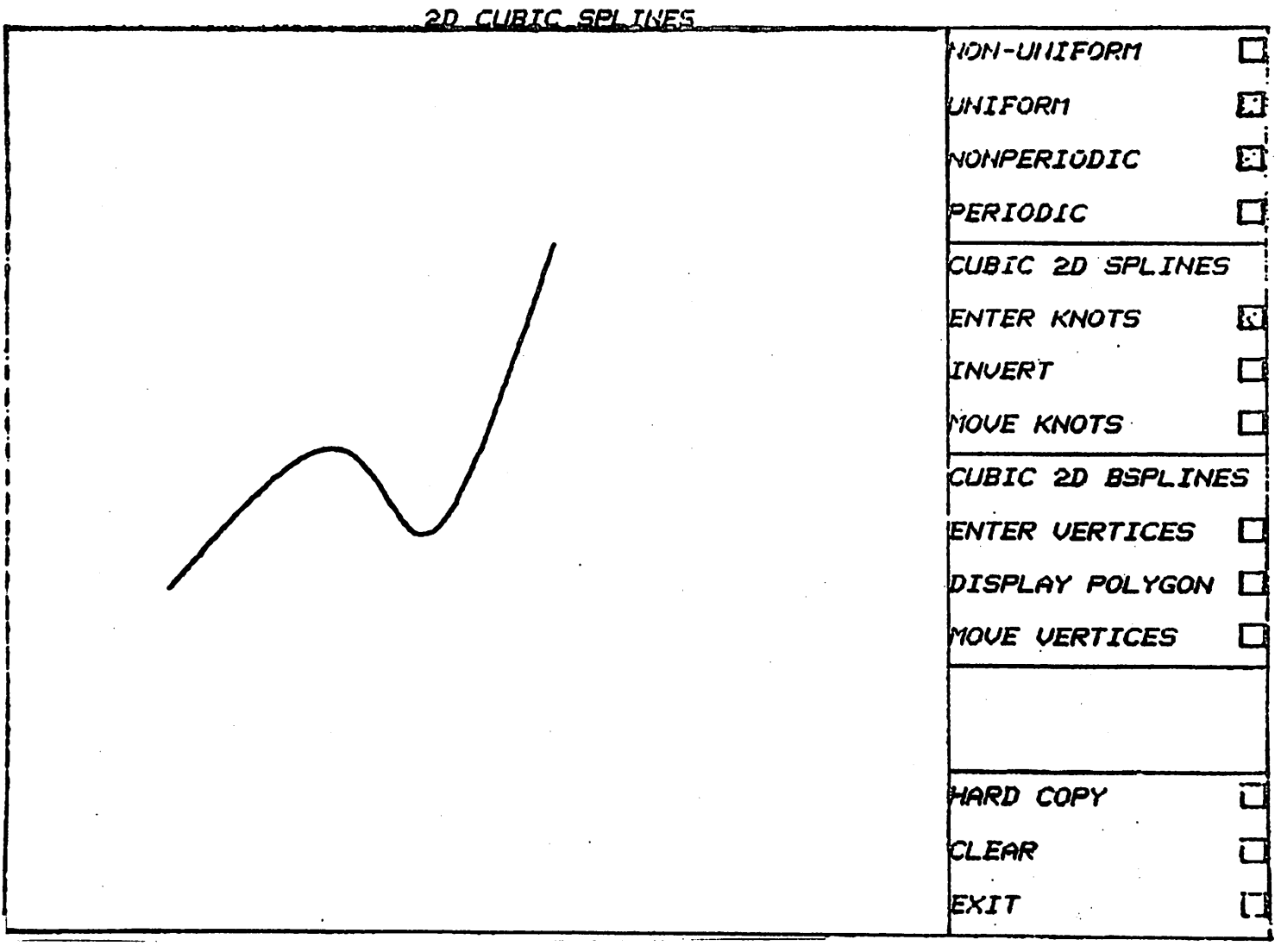


FIGURE D.2 OPEN CUBIC SPLINE THROUGH SAME FOUR KNOTS AS FIGURE D.1 - UNIFORM PARAMETERIZATION

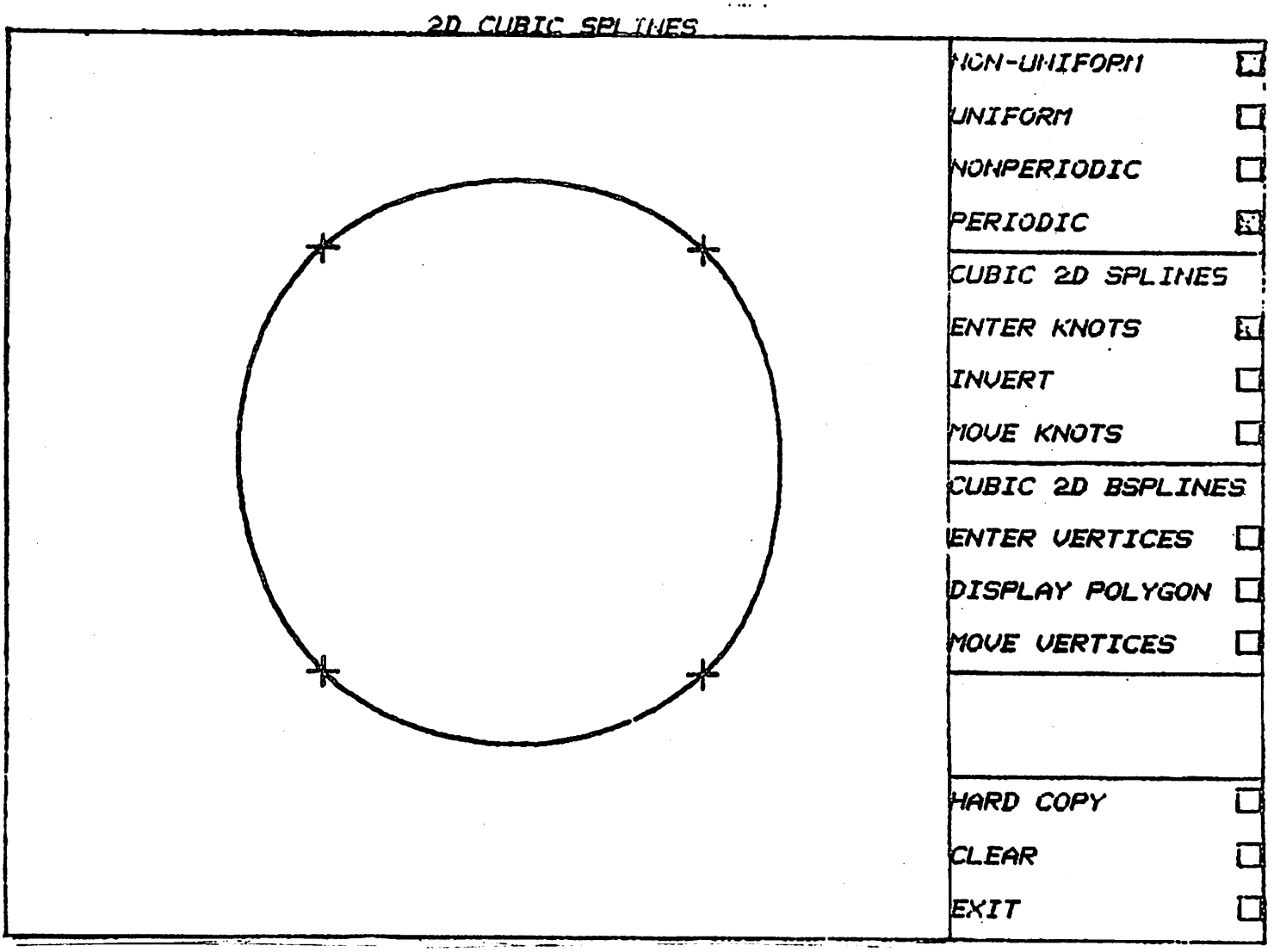


FIGURE D.3 CLOSED (PERIODIC) CUBIC SPLINE THROUGH FOUR KNOTS - CHORD LENGTH PARAMETERIZATION

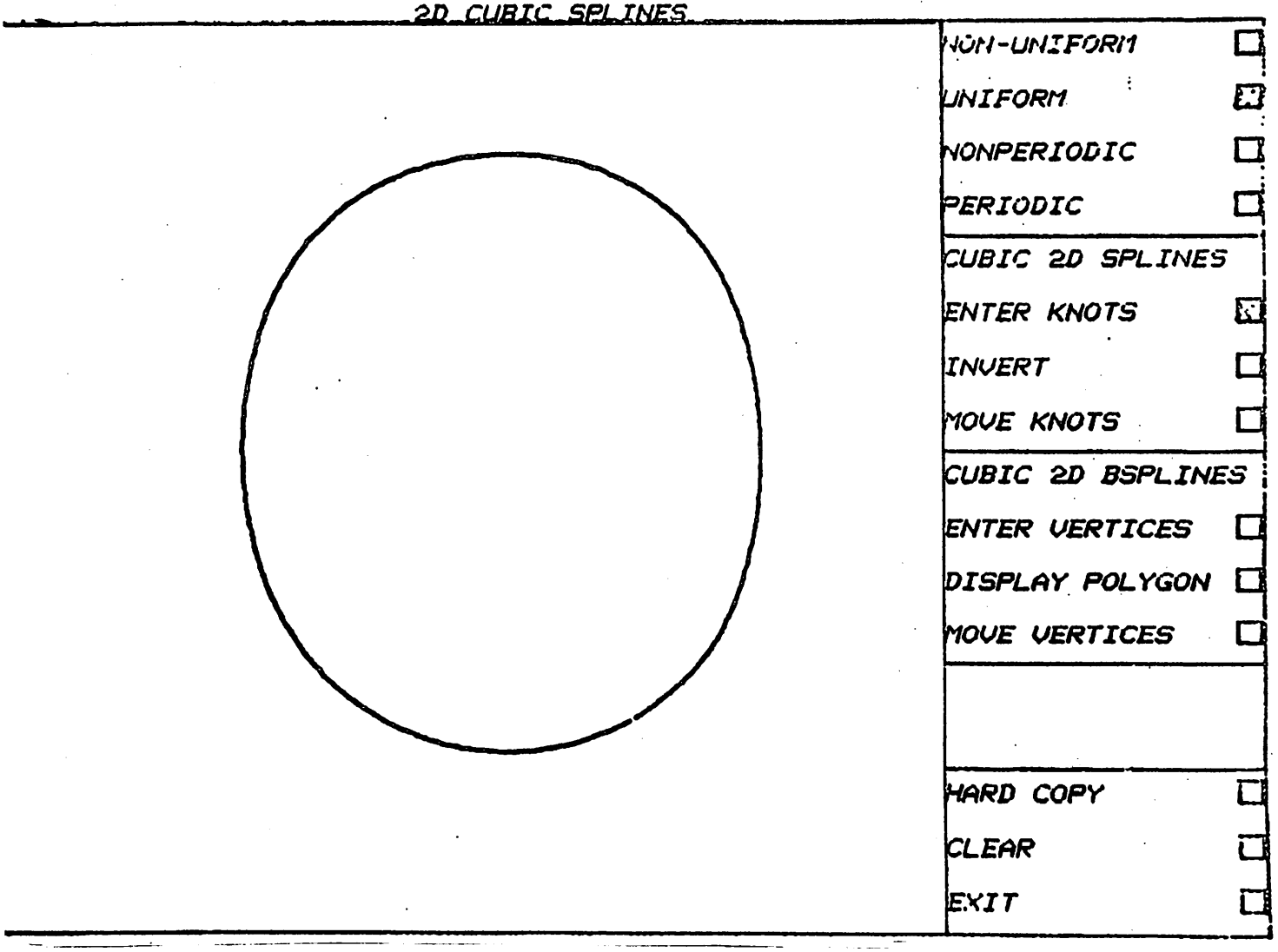


FIGURE D.4 CLOSED CUBIC SPLINE THROUGH SAME FOUR KNOTS AS FIGURE D.3 - UNIFORM PARAMETERIZATION

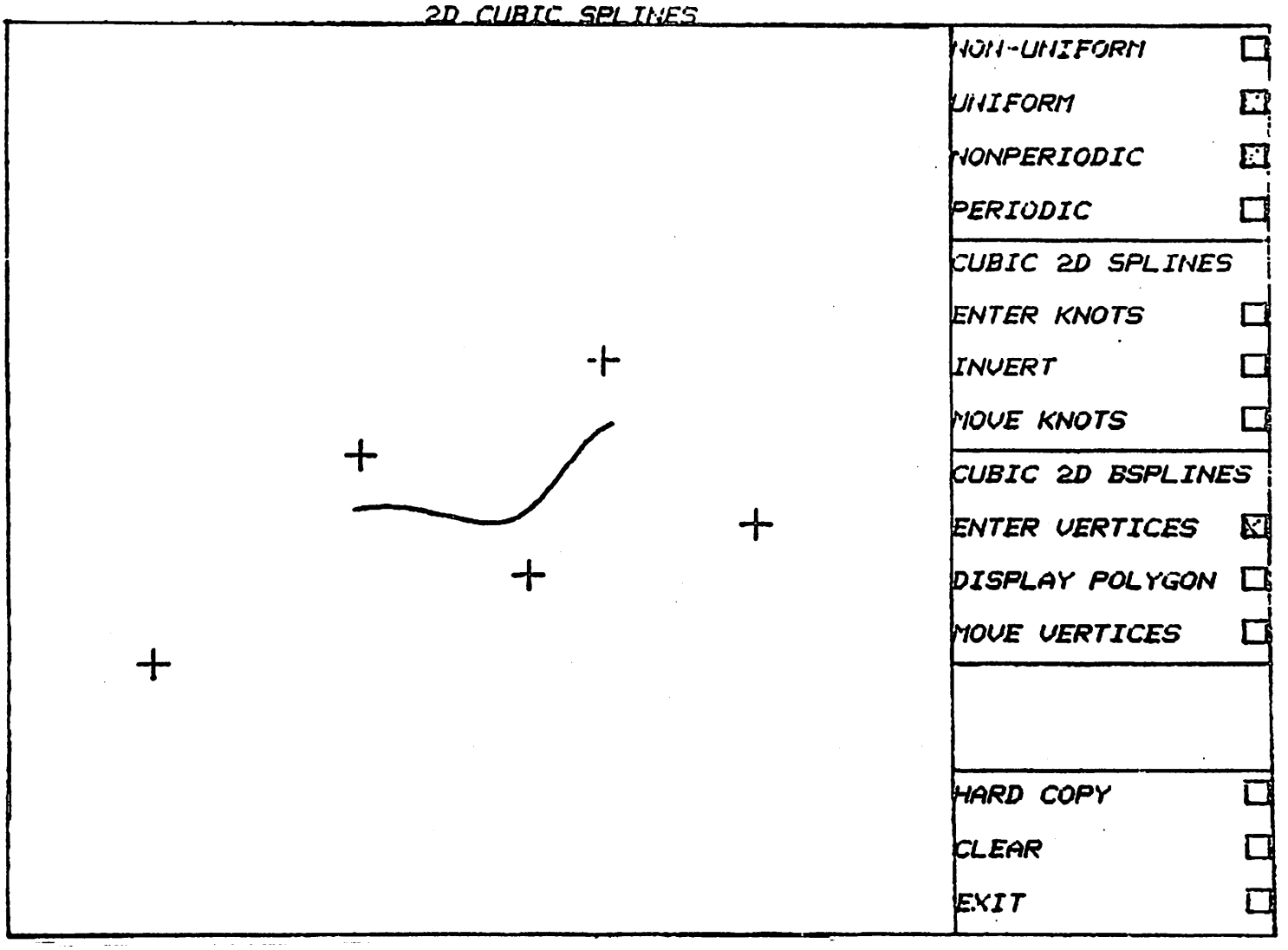


FIGURE D.5 OPEN CUBIC B-SPLINE AND VERTICES OF ITS DEFINING POLYGON

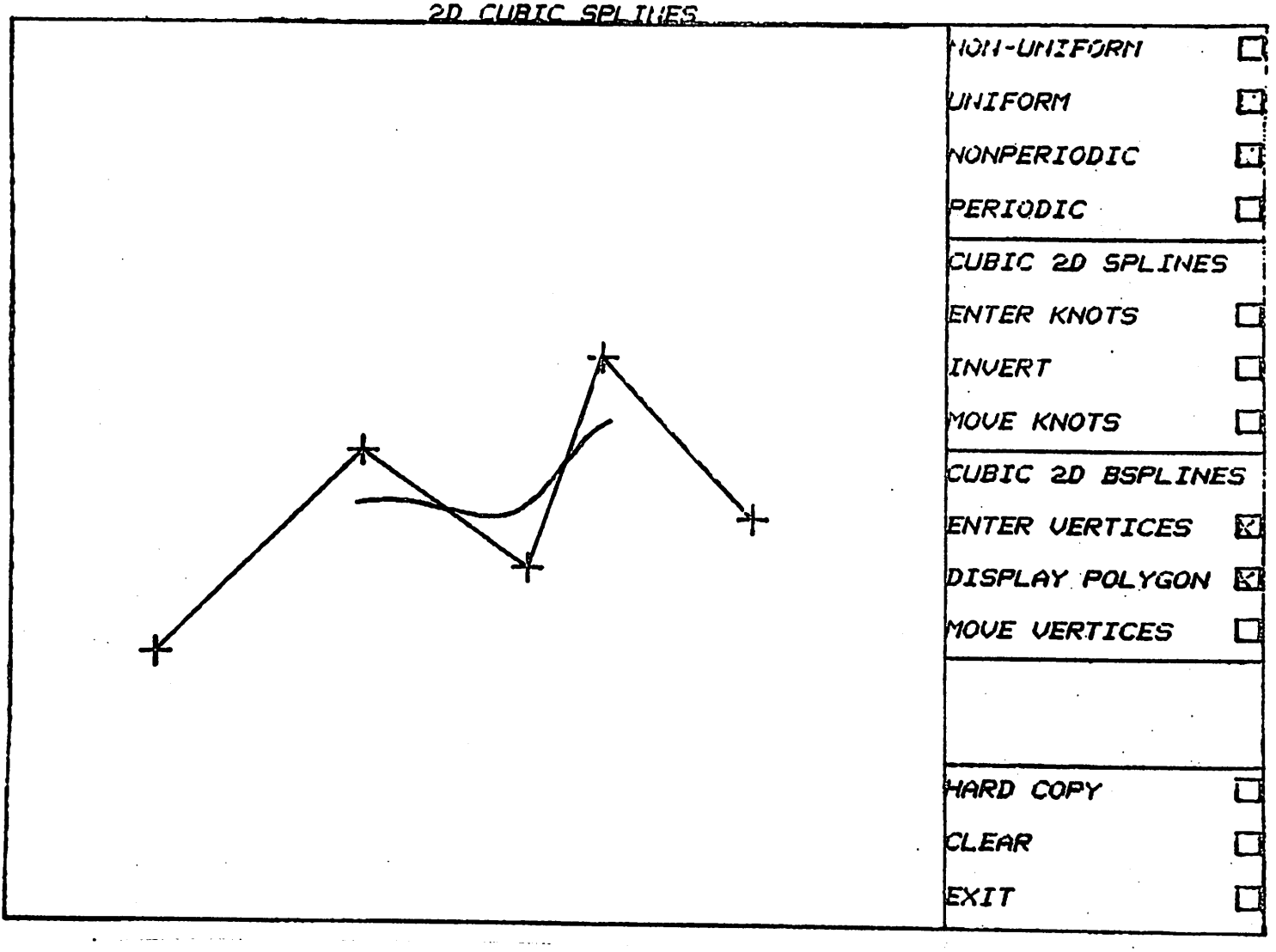


FIGURE D.6 SAME OPEN CUBIC B-SPLINE AS IN FIGURE D.5 WITH VERTICES OF DEFINING POLYGON CONNECTED

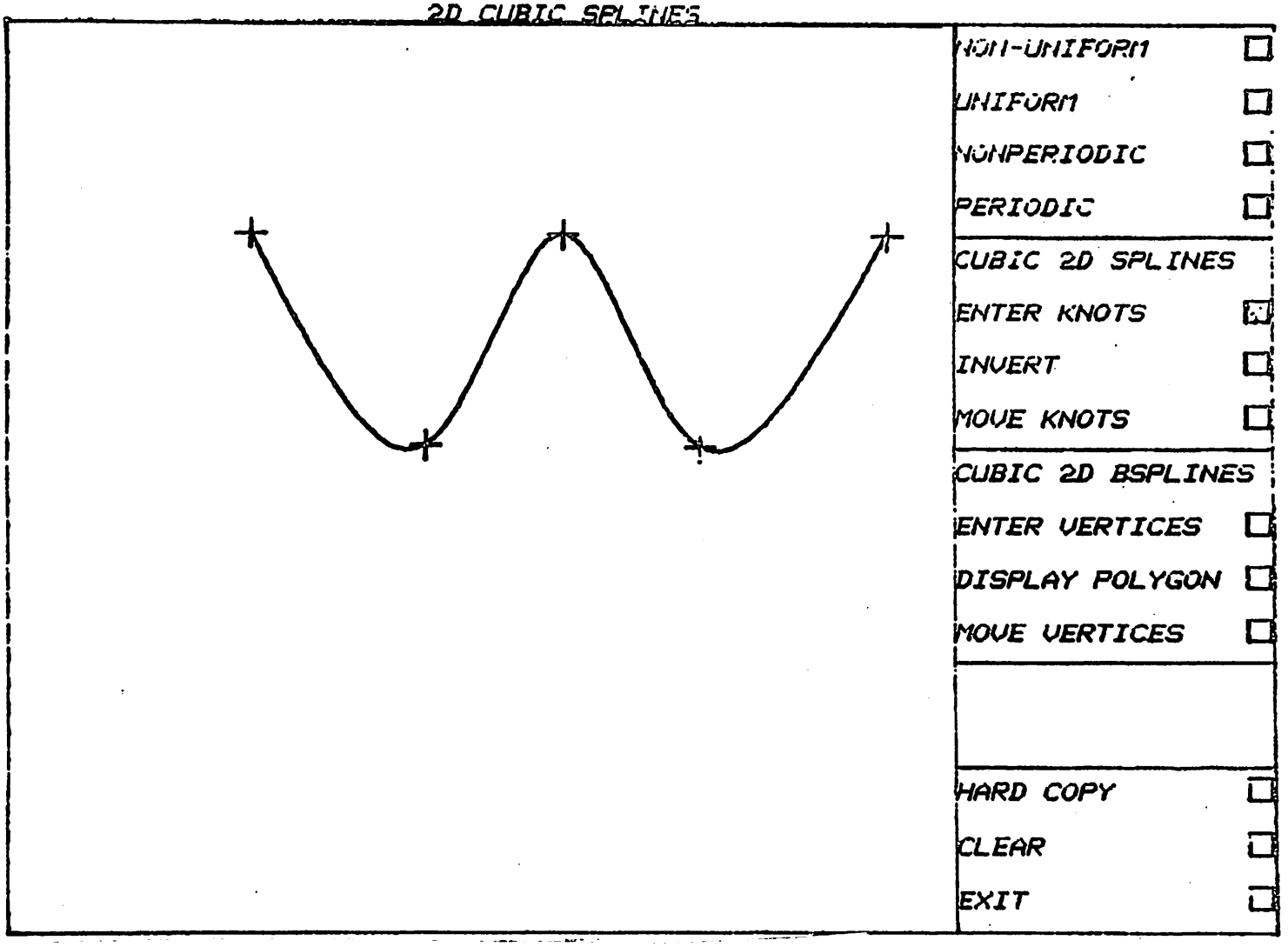


FIGURE D.7 OPEN CUBIC SPLINE THROUGH FIVE KNOTS OF MULTIPLICITY 1

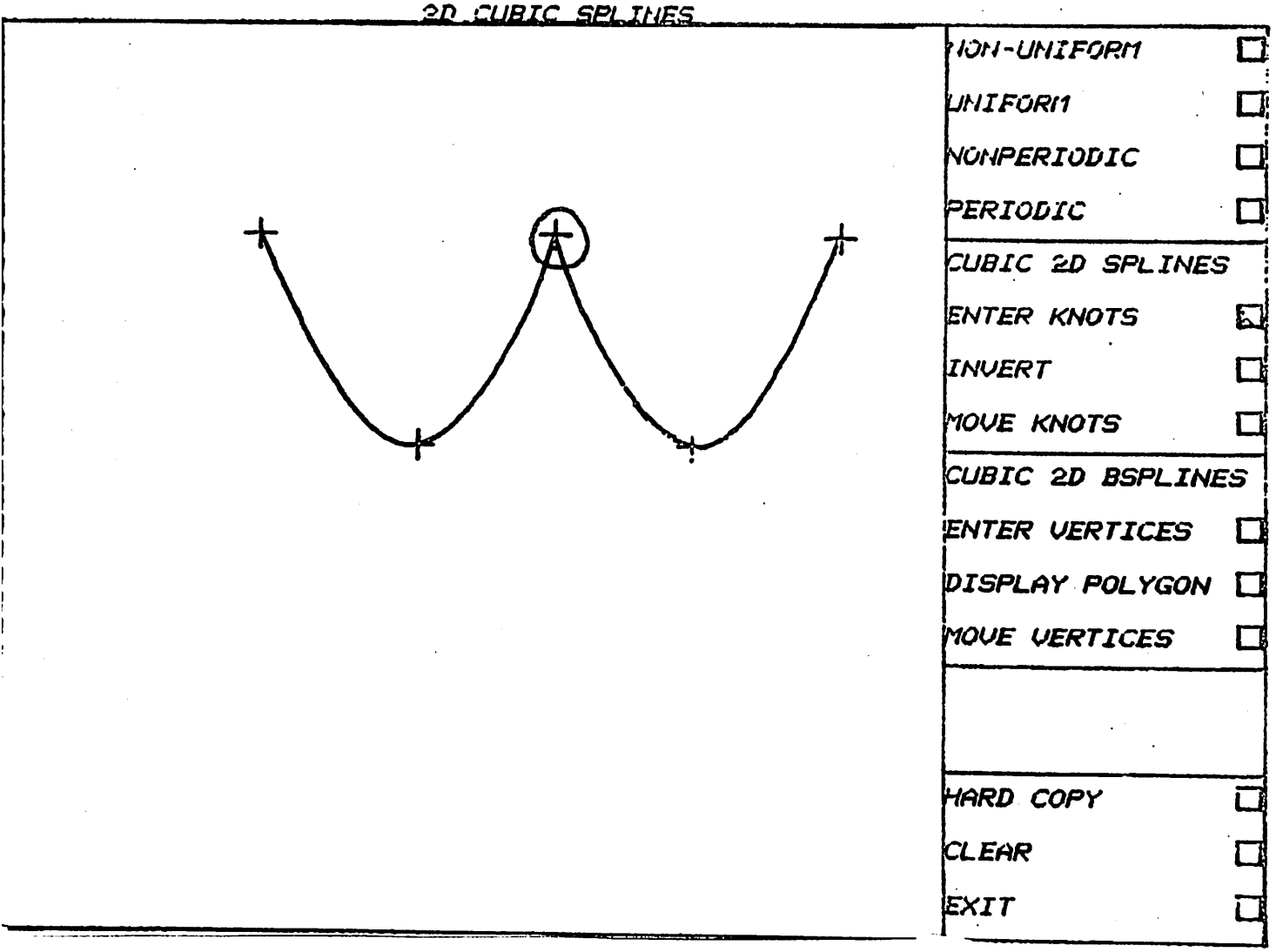


FIGURE D.8 OPEN CUBIC SPLINE THROUGH SAME FIVE KNOTS AS FIGURE D.7. CIRCLED KNOT HAS MULTIPLICITY 2. OTHER KNOTS ARE MULTIPLICITY 1. SPLINE HAS A CURVATURE DISCONTINUITY AT KNOT 3 (CIRCLED).

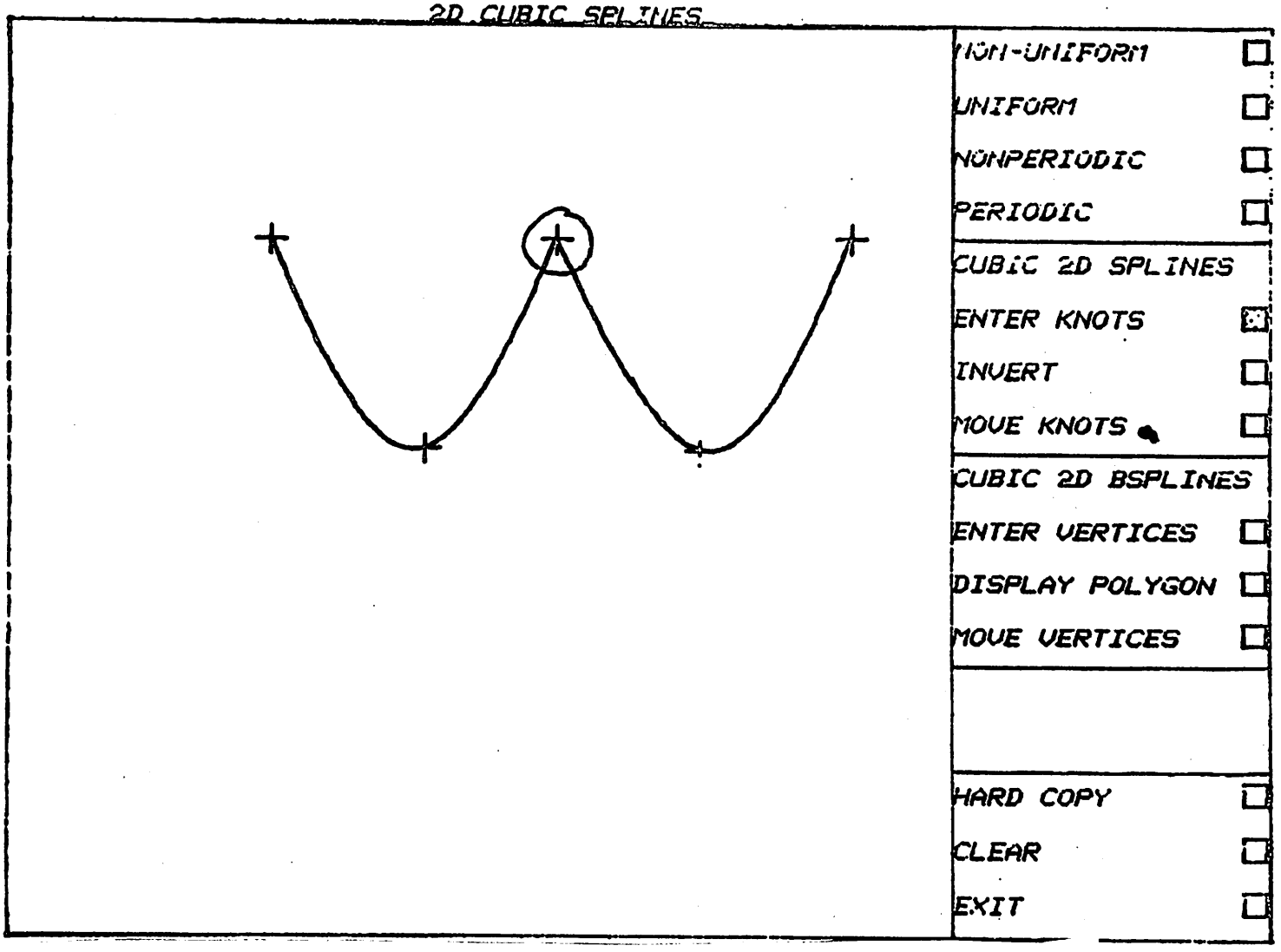
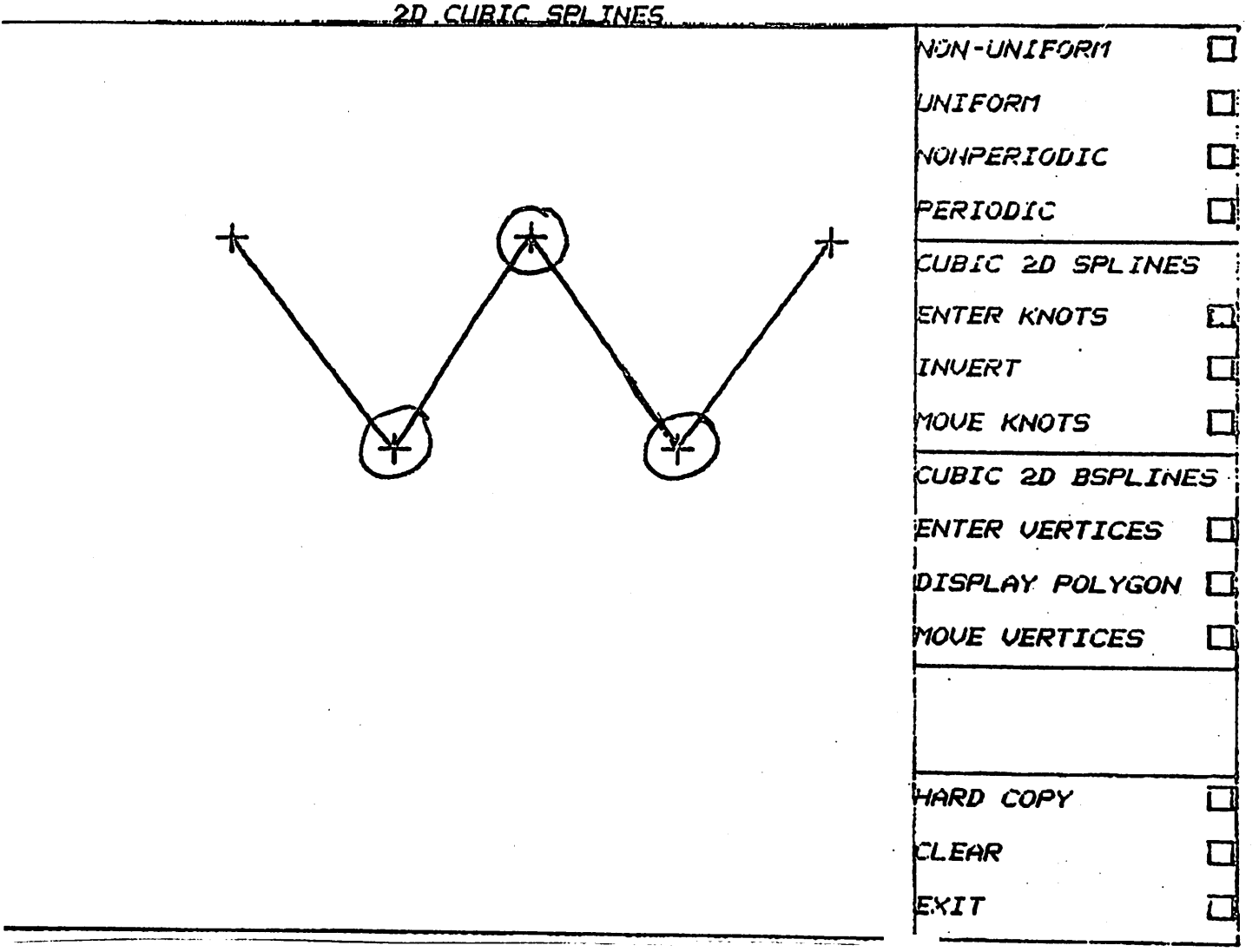


FIGURE D.9 OPEN CUBIC SPLINE THROUGH SAME FIVE KNOTS AS FIGURES D.7 AND D.8. KNOT 3 (CIRCLED) HAS MULTIPLICITY 3. OTHER KNOTS HAVE MULTIPLICITY 1. SPLINE ACTUALLY HAS A SLOPE DISCONTINUITY AT KNOT 3 (CIRCLED), EVEN THOUGH IT IS NOT APPARENT FROM PICTURE.



IGURE D.10 OPEN CUBIC SPLINE THROUGH SAME FIVE KNOT LOCATIONS AS FIGURES D.7, D.8, D.9. KNOTS 2, 3, 4 (CIRCLED) HAVE MULTIPLICITY 3. OTHER KNOTS HAVE MULTIPLICITY 1. SLOPE DISCONTINUITIES APPEAR AT KNOTS 2, 3, 4 (CIRCLED).

APPENDIX E

DERIVATION OF THE DEFINING EQUATIONS FOR CUBIC SPLINES

The following derivation proceeds along the lines of the derivation in Chapter 2 of [AHL67] with only slight notational changes to remain consistent with the text of this report.

First, let's consider the case of the non-periodic or open spline defined on the closed interval [a,b]. Given a knot vector (x_0,y_0), $(x_1,y_1),\ldots,(x_N,y_N)$ such that $a = x_0 < x_1 < x_2 < \ldots < x_N = b$, we seek to define a cubic spline function S(x) on the knot vector - i.e. S(x) must satisfy the following conditions:

(i) S(x) and its first two derivatives are continuous on [a,b]

(ii) S(x) is a cubic polynomial on each subinterval $[x_i, x_{i+1}]$ of the knot vector

(iii) $S(x_i) = y_i$ for $i = 0,1,\ldots,N$

Let M_j denote the second derivative of the spline function at the location x_j -- $S''(x_j)$. Because the second derivative of a cubic polynomial is a linear function, we derive the general second derivative equation (restricted to the interval $[x_{j-1},x_j]$) by linear interpolation between M_{j-1} and M_j. Thus, we have

$$S''(x) = M_{j-1}\frac{x_j - x}{h_j} + M_j\frac{x - x_{j-1}}{h_j} \qquad (E.1)$$

where $h_j = x_j - x_{j-1}$

If we integrate (E.1) and evaluate the constant of integration we get the equation of $S'(x)$ restricted to $[x_{j-1},x_j]$

$$S'(x) = -M_{j-1}\frac{(x_j - x)^2}{2h_j} + M_j \frac{(x - x_{j-1})^2}{2h_j} + \frac{y - y_{j-1}}{h_j} - \frac{M_j - M_{j-1}}{6} h_j \qquad (E.2.1)$$

The equation for $S'(x)$ restricted to $[x_j, x_{j+1}]$ is

$$S'(x) = -M_j\frac{(x_{j+1}-x)}{2h_{j+1}} + M_{j+1}\frac{(x - x_j)}{2h_{j+1}} + \frac{y - y_j}{h_{j+1}} - \frac{M_{j+1}-M_j}{6} h_j \qquad (E.2.2)$$

From (E.2.1) and (E.2.2) we obtain the following expressions for the one-sided limits of the derivatives by substituting $x = x_j$ into equations (E.2.1) and (E.2.2) respectively:

$$S'(x_j-) = \frac{h_j}{6} M_{j-1} + \frac{h_j}{3} M_j + \frac{y_j - y_{j-1}}{h_j} \qquad (E.3.1)$$

$$S'(x_j+) = -\frac{h_{j+1}}{3} M_j - \frac{h_{j+1}}{6} M_{j+1} + \frac{y_{j+1} - y_j}{h_{j+1}} \qquad (E.3.2)$$

The continuity of $S'(x)$ yields the condition $S'(x_j-) = S'(x_j+)$ or

$$\frac{h_j}{6} M_{j-1} + \frac{h_j}{3} M_j + \frac{y_j - y_{j-1}}{h_j} = \frac{-h_{j+1}}{3} M_j - \frac{h_{j+1}}{6} M_{j+1} + \frac{y_{j+1} - y_j}{h_{j+1}} \qquad (E.4)$$

Equation (E.4) may be rewritten as follows:

$$\frac{h_j}{6} M_{j-1} + \frac{h_j + h_{j+1}}{3} M_j + \frac{h_{j+1}}{6} M_{j+1} = \frac{y_{j+1} - y_j}{h_{j+1}} - \frac{y_j - y_{j-1}}{h_j} \qquad \text{(E.5)}$$

Notice that $\frac{y_{j+1} - y_j}{h_{j+1}}$ is actually the first divided difference, DDy_j, and $\frac{y_j - y_{j-1}}{h_j}$ is the first divided difference, DDy_{j-1}.

Thus the right hand side of equation (E.5) is actually a <u>multiple</u> of the second divided difference, DD^2y_{j-1}, since

$$DD^2y_{j-1} = \frac{DDy_j - DDy_{j-1}}{h_{j+1} + h_j} = \frac{DDy_j - DDy_{j-1}}{x_{j+1} - x_{j-1}}$$

Thus the right hand side of (E.5) is

$$DDy_j - DDy_{j-1} = (h_{j+1} + h_j) DD^2y_{j-1}$$

Now, if we multiply equation (E.5) by 6 and divide by $(h_{j+1} + h_j)$ we get the alternative form:

$$\frac{h_j}{h_{j+1} + h_j} M_{j-1} + 2M_j + \frac{h_{j+1}}{h_{j+1} + h_j} M_{j+1} = 6DD^2y_{j-1} \qquad \text{(E.6)}$$

Now (E.6) for j=1,...,N-1 are the defining equations for the values

$M_1,\ldots,M_{N-1}$. Note that we cannot solve for M_0 and M_N because DD^2y_0 and DD^2y_N are not defined - i.e. we would require values for the points (x_{-1},y_{-1}) and (x_{N+1},y_{N+1}) in order to define DD^2y_0 and DD^2y_N. These are called the "end conditions" and their values are supplied by the user. In the section III the most commonly used techniques for defining end conditions are described.

Let's define $\lambda_j = \dfrac{h_{j+1}}{h_{j+1}+h_j}$ and $\mu_j = 1 - \lambda_j$ $(j=1,\ldots,N-1)$

and let $d_j = 6DD^2y_j$, then the system of defining equation for the cubic spline may be written in matrix form as follows:

$$\begin{bmatrix} 2 & \lambda_0 & 0 & \ldots & 0 & 0 & 0 \\ \mu_1 & 2 & \lambda_1 & \ldots & 0 & 0 & 0 \\ 0 & \mu_2 & 2 & \ldots & 0 & 0 & 0 \\ \cdot & \cdot & \cdot & & & & \\ \cdot & \cdot & \cdot & \ldots & 2 & \lambda_{N-2} & 0 \\ 0 & 0 & 0 & \ldots & \mu_{N-1} & 2 & \lambda_{N-1} \\ 0 & 0 & 0 & \ldots & 0 & \mu_N & 2 \end{bmatrix} \begin{bmatrix} M_0 \\ M_1 \\ \cdot \\ \cdot \\ \cdot \\ M_{N-1} \\ M_N \end{bmatrix} = \begin{bmatrix} d_0 \\ d_1 \\ \cdot \\ \cdot \\ \cdot \\ d_{N-1} \\ d_N \end{bmatrix} \qquad (E.7)$$

where the user prespecifies the end conditions by giving values for λ_0, d_0, λ_N, d_N. In the case of "natural end conditions", it is only necessary to specify $M_0 = M_N = 0$. This has the effect of removing two equations from the system and allows one to solve the system of

N-1 equations for $M_1, M_2, \ldots, M_{N-1}$. The commonly used "end conditions" are defined in section III of this report.

The actual form of the cubic spline function for the given knot vector and second derivatives $M_0, M_1, \ldots, M_N$ may be obtained by integration of equation (E.2.1) and by evaluating the constant of integration.

$$S(x) = M_{j-1}\frac{(x_j - x)^3}{6h_j} + M_j\frac{(x - x_{j-1})^3}{6h_j} + \left(y_{j-1} - \frac{M_{j-1}h_j^2}{6}\right)\frac{x_j - x}{h_j} + \left(y_j - \frac{M_j h_j^2}{6}\right)\frac{x - x_{j-1}}{h_j} \qquad (E.8)$$

restricted to the interval $[x_{j-1}, x_j]$

With extensive algebraic manipulation, it can be shown that equation (E.8) is equivalent to the alternative spline form used throughout this report - namely,

$$S(x) = y_{j-1} + m_{j-1}(x - x_{j-1}) + M_{j-1}(x - x_{j-1})^2 + M_{j-1}(x - x_j)^3$$

where m_{j-1} = the first derivative at x_{j-1}

M_{j-1} = the second derivative at x_{j-1}

and M_{j-1} = the third derivative at x_{j-1}

The reader should take note of the following results:

(1) For the case of the periodic cubic spline where $x_0 = x_N$ and $y_0 = y_N$ we have $M_0 = M_N$ and $\lambda_N = h_1/(h_N + h_1)$, $\mu_N = 1 - \lambda_N$. We obtain the following system of defining equations

$$\begin{bmatrix} 2 & \lambda_1 & 0 & \cdots & 0 & \cdots & 0 \\ \mu_2 & 2 & \lambda_2 & & 0 & \cdots & 0 \\ 0 & \mu_3 & 2 & \lambda_3 & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & & \vdots & \vdots & \vdots \\ 0 & \cdots & \cdots & 0 & 2 & \lambda_{N-2} & 0 \\ 0 & \cdots & \cdots & 0 & \mu_{N-1} & 2 & \lambda_{N-1} \\ 0 & \cdots & \cdots & 0 & 0 & \mu_N & 2 \end{bmatrix} \begin{bmatrix} M_1 \\ M_2 \\ \vdots \\ M_{N-1} \\ M_N \end{bmatrix} = \begin{bmatrix} d_1 \\ d_2 \\ \vdots \\ d_{N-1} \\ d_N \end{bmatrix} \qquad \text{(E.9)}$$

(2) Systems (E.7) and (E.9) may be reformulated in terms of slopes at the knots. See [AHL67, p.13] for details.

(3) The defining equations for spline functions of any order may be obtained in the manner shown above. If the spline function is or order M (i.e. degree M-1), one merely uses the fact that the (M-2)nd derivative is linear to construct the analog of equation (E.1). Then successive integrations and evaluations of the

constants of integrations are performed to arrive at a spline function equation in terms of the (M-2)nd derivatives and the knot vectors. In particular the system of defining equations for <u>parabolic</u> splines is

$$\begin{bmatrix} \lambda_0 & 1 & 0 & \cdots & 0 \\ 0 & \lambda_1 & 1 & 0 & \cdots & 0 \\ 0 & 0 & \lambda_2 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 0 & \cdots & 1 & 0 \\ \vdots & \vdots & & & & \\ & & & & \lambda_{N-1} & 1 \\ 0 & 0 & \cdots & \cdots & 0 & \lambda_N \end{bmatrix} \begin{bmatrix} m_0 \\ m_1 \\ \vdots \\ m_{N-1} \\ m_N \end{bmatrix} = \begin{bmatrix} b_0 \\ b_1 \\ \vdots \\ b_{N-1} \\ b_N \end{bmatrix} \qquad \text{(E.10)}$$

where $\lambda_j = \dfrac{h_j}{h_{j-1}}$ and $b_j = (3y_j + 3y_{j-1})/h_{j-1} + (3y_j + y_{j-1})/h_j$

for $j=1,\ldots,N-1$

Again we supply the end conditions λ_0, b_0, λ_N, b_N and solve the system for the <u>slopes</u> $m_0, m_1, \ldots, m_N$

(4) When there is <u>uniform</u> spacing of the x_j i.e. $h_j = x_{j+1} - x_j = 1$ for all j then the system of defining equations becomes

$$\begin{bmatrix} 4 & 1 & 0 & 0 & & 0 & 0 \\ 1 & 4 & 1 & 0 & \cdots & 0 & 0 \\ 0 & 1 & 4 & 1 & & \vdots & \vdots \\ \vdots & \vdots & \vdots & & & 0 & 0 \\ 0 & \cdots & \cdots & 0 & 4 & 1 & 0 \\ \vdots & \vdots & \vdots & & 1 & 4 & 1 \\ 0 & \cdots & \cdots & 0 & 0 & 1 & 4 \end{bmatrix} \begin{bmatrix} M_0 \\ M_1 \\ \vdots \\ M_{N-1} \\ M_N \end{bmatrix} = \begin{bmatrix} 6\Delta^2 y_0 \\ 6\Delta^2 y_1 \\ \vdots \\ 6\Delta^2 y_{N-1} \\ 6\Delta^2 y_N \end{bmatrix} \quad \text{(E.11)}$$

There are stable methods for solution of these systems of equations - see references [AHL67, BAU77, deB79].

APPENDIX F

DERIVATION OF NEWTON'S FORWARD DIFFERENCE POLYNOMIAL INTERPOLATION FORMULA

Given a table of argument - function value pairs, we wish to derive the Newtion Forward Difference Polynomial Interpolation formula which may be used to interpolate values in the table. Assume that the arguments begin with a value of x=0 and they are equally spaced at an interval of h. Thus we have a table of the form:

argument (x)	functional value (f(x))
0	f(0)
h	f(h)
2h	f(2h)
.	.
.	.
.	.
nh	f(nh)

In order to use the Newton interpolation formula, we must make the assumption that the underlying function is analytic - i.e. it may be expanded in a Taylor series about any point. We will use the Taylor series expansion to obtain formulas for the derivative of a function in terms of discrete differences.

Let f(x) be an analytic function. We may find f(x+h) by expanding f(x) in a Taylor series about the point x:

$$f(x+h) = f(x) + hf'(x) + \frac{h^2}{2!} f''(x) + \frac{h^3}{3!} f'''(x) + \ldots \qquad (F.1)$$

Solving equation (F.1) for $f'(x)$ yields

$$f'(x) = f(x+h) - f(x) - \frac{h}{2!}f''(x) - \frac{h^2}{3!}f'''(x) + \ldots \qquad (F.2)$$

Using the order notation, we write

$$f'(x) = \frac{f(x+h) - f(x)}{h} + O(h). \qquad (F.3)$$

Note that the first term on the right hand side of equation (F.3) is the first ordinary difference divided by h (or alternatively the first divided difference). Thus equation (F.3) may be rewritten

$$f'(x) = \frac{\Delta f(x)}{h} + O(h) \qquad (F.4)$$

Now, by a similar expansion about x, we may find $f(x+2h)$ as

$$f(x+2h) = f(x) + 2hf'(x) + \frac{(2h)^2}{2!}f''(x) + \frac{(2h)^3}{3!}f'''(x) + \ldots \qquad (F.5)$$

Multiplying equation (F.1) by 2 and subtracting the result from equation (F.5) causes the $f'(x)$ term to drop out. We may solve for $f''(x)$ and get

$$f''(x) = \frac{f(x+2h) - 2f(x+h) + f(x)}{h^2} - hf'''(x) + \ldots \qquad (F.6)$$

Notice that the first term on the right hand side of equation (F.6) is the formula for the second ordinary difference of $f(x)$

divided by h^2. Thus equation (F.6) may be rewritten

$$f''(x) = \frac{\Delta^2 f(x)}{h^2} + O(h). \tag{F.7}$$

In this manner, we can obtain approximations to the derivatives of f(x) in terms of the corresponding finite differences and error terms. Now the derivation of Newton's forward difference formula follows in a straightforward manner.

Consider the Taylor series expansion of the function f(x) about the point x=0:

$$f(x) = f(0) + xf'(0) + \frac{x^2}{2!}f''(0) + \frac{x^3}{3!}f'''(0) + \ldots \tag{F.8}$$

None of the values of the derivatives are known; however, expressions for the derivatives may be obtained as in equations (F.4) and (F.7). Evaluate the expressions for the derivatives at x=0 and substitute the finite difference expressions into equation (F.8). Keep careful account of all the error terms and the following equation is obtained:

$$f(x) = f(0) + \frac{x}{h}\Delta f(0) + \frac{x(x-h)}{2!h^2}\Delta^2 f(0) + \frac{x(x-h)(x-2h)}{3!h^3}\Delta^3 f(0) + \ldots \tag{F.9}$$

Equation (F.9) is Newton's Forward Difference Polynomial Interpolation formula. It is clearly equivalent to equation (II.9) in the text when h=1.

APPENDIX G

deBoor - Cox Recursive Algorithm

To evaluate the B-spline function:

$$f(s) = \sum_i a_i B_{i,M}(s)$$

STEP 1: Find i such that $x_i \leq s \leq x_{i+1}$

STEP 2: $f(s) = a_i^{[M-1]}(s)$

STEP 3: Set j = i and k = M-1

$$a_j^{[k]}(s) = \begin{cases} a_j & \text{for } k = 0 \\ \lambda a_j^{[k-1]} + (1 - \lambda) a_{j-1}^{[k-1]} & \text{for } k > 0 \end{cases}$$

and $\lambda = \dfrac{s - x_j}{x_{j-k+M} - x_j}$ (j is the same as above)

deBoor also gives the following formula for the j^{th} derivative of f:

$$f^{(j)}(s) = (M-1)(M-2)\ldots(M-j)\sum_i b_i^{[j]} B_{i,M-j}(s)$$

where

$$b_i^{[k]} = \begin{cases} a_i & \text{for } k = 0 \\ \dfrac{b_i^{[k-1]} - b_{i-1}^{k-1}}{x_{i+M-k} - x_i} & \text{otherwise} \end{cases}$$

APPENDIX H

Yamaguchi Inversion Algorithm

Let K_i be the knots of a cubic interpolating spline.

V_i are the corresponding vertices of the vertex polygon.

STEP 1: Initialization:

For $i = 1,2,\ldots,n$

$$K_i \rightarrow V_i$$

then $V_1 \rightarrow V_0$, $V_n \rightarrow V_{n+1}$ (alternatively $V_1 \rightarrow V_{n+1}$, $V_n \rightarrow V_0$).

STEP 2: Computation:

For $i = 1,2,\ldots,n$

$$\delta_i = K_i - V_i + \tfrac{1}{2}\{K_i - \tfrac{1}{2}(V_{i-1} + V_{i+1})\}, \qquad \delta_i + V_i \rightarrow V_i,$$

then $V_1 \rightarrow V_0$, $V_n \rightarrow V_{n+1}$ (alternatively $V_1 \rightarrow V_{n+1}$, $V_n \rightarrow V_0$).

STEP 3: Halting Criterion:

If $\max \{\delta_i\} \leq \delta_s$, then halt

else go to STEP 2;

where δ_s is a predetermined tolerance value and the parentheses pertain to the closed case alternative.