PREPRINT UCRL-80997

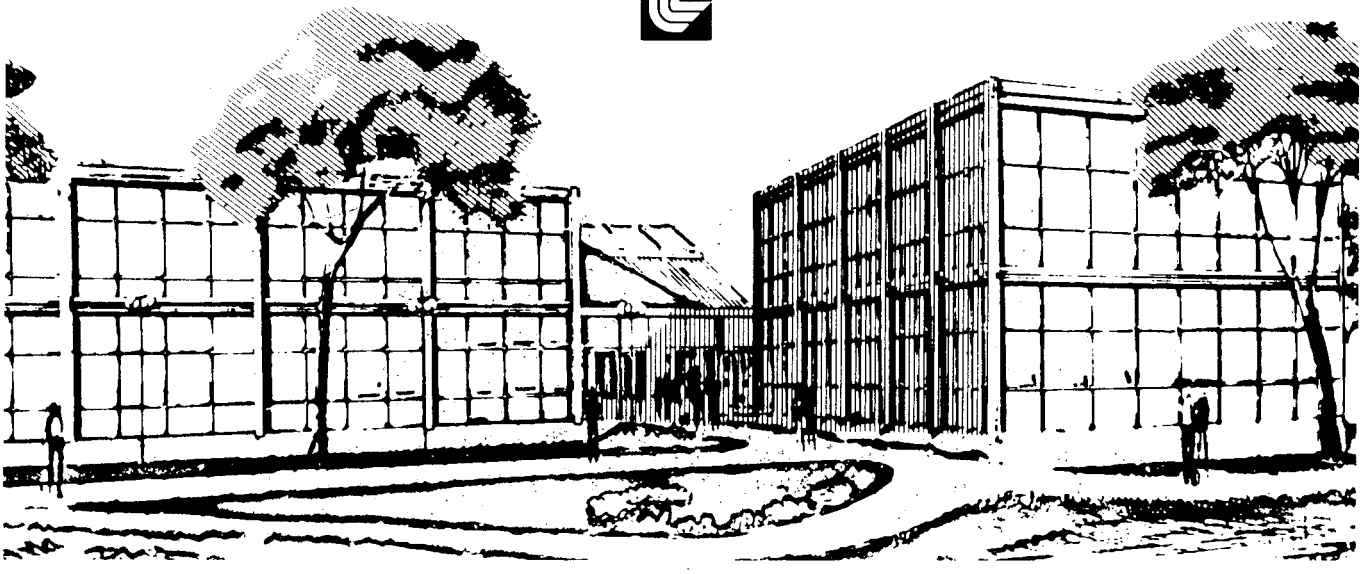
Lawrence Livermore Laboratory

SOME EXPERIMENTS WITH PIECEWISE CUBIC INTERPOLATION

F.N. Fritsch

June 1978

Prepared for presentation at NASIG'78, Argonne National Laboratory, 12-13 April 1978.



DISCLAIMER

SOME EXPERIMENTS WITH PIECEWISE CUBIC INTERPOLATION

Table of Contents

ABSTRACT

An iterative refinement process for adjusting derivative values in the Hermite representation of a piecewise cubic function to produce visually pleasing interpolants is described. The difficulties encountered at various stages in the development of the algorithm are outlined, and future research directions are indicated.

1. Introduction.

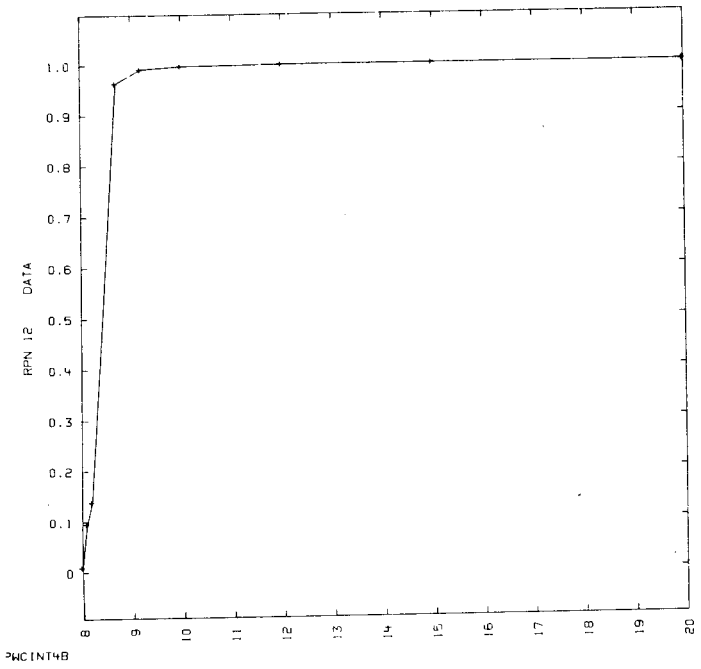
This study was motivated by the failure of standard cubic spline interpolation procedures* to provide acceptable interpolants for certain data sets used in radiochemical calculations. These data are typically reasonably smooth (free from significant experimental error) and bounded between zero and one. The number of data points is generally fairly small (10 to 30). In certain cases, however, the spacing of the independent variable is extremely non-uniform and the dependent variable ranges over many orders of magnitude. Several typical data sets are depicted in Figure 1.1.

Because of the drastic changes in slope in such data sets, the cubic spline interpolant frequently exhibits unphysical "bumps" or "wiggles" between the data values. In particular, the interpolant may badly violate the physical constraints $0 \leq f(x) \leq 1$, even though the data do not. (Some cubic spline interpolants are shown in Section 3.1.)

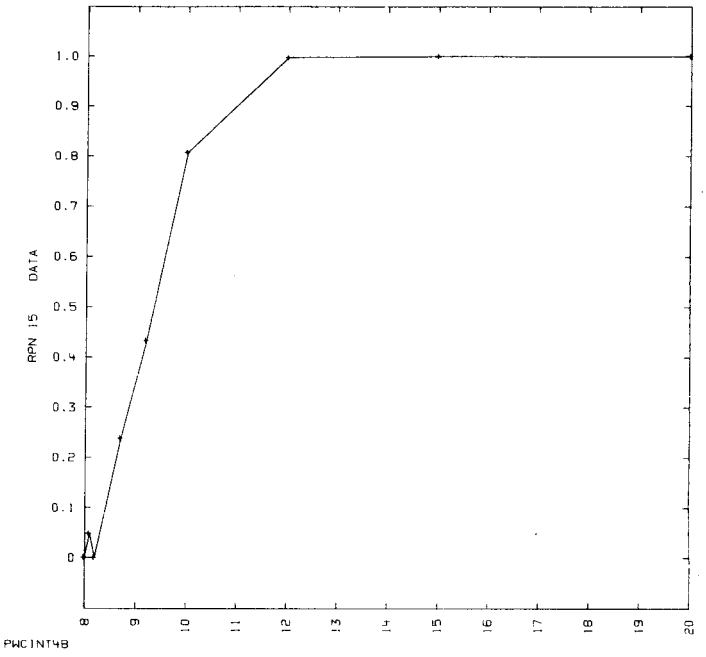
The object of this study was to see if it is possible to produce piecewise cubic interpolants that are more visually pleasing (hence, hopefully more "physical") than cubic splines for such data sets. After some mathematical preliminaries and a discussion of previously existing methods, problems with even the best method are described. There follows a step-by-step treatment of the development of an iterative refinement procedure for producing "improved"

---

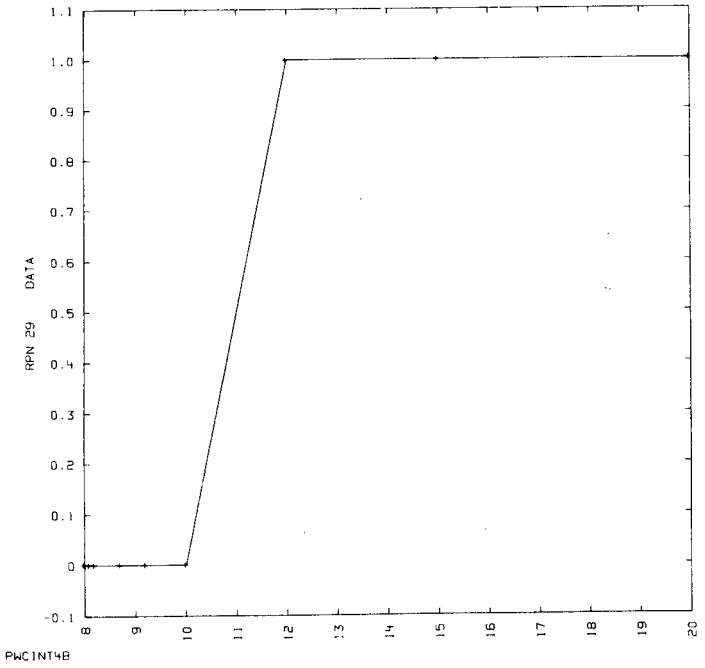
*Such as IMSL Subroutine ICSICU[5]. [Notice: Reference to a company or product name does not imply approval or recommendation of the product by the University of California or the U.S. Department of Energy to the exclusion of others that may be suitable.]



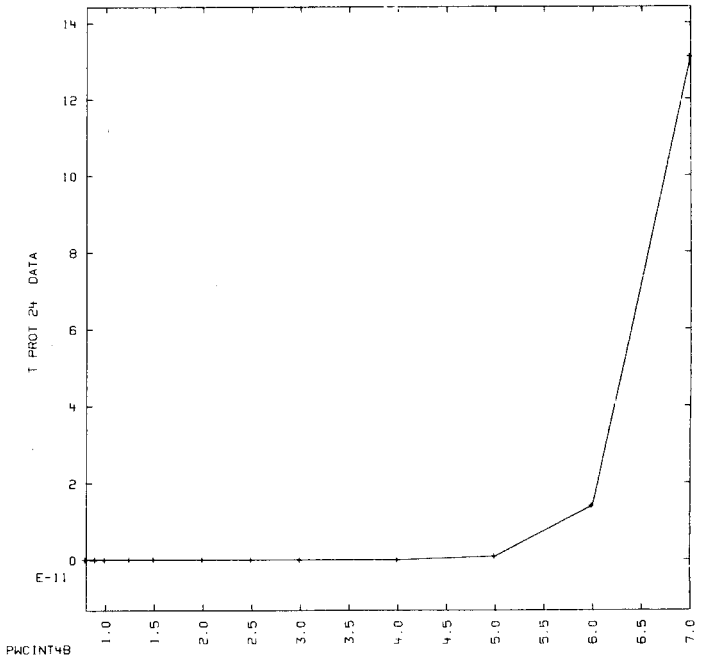
(a)



(b)



(c)



(d)

Figure 1.1 Some Typical Data Sets

interpolants. We conclude with an outline of the overall process, a discussion of pros and cons for the method, and an indication of possible future developments. For completeness, we give in an appendix numerical values for all data sets shown in this report.

2. Mathematical Preliminaries.

This section contains basic definitions and introduces the mathematical notation to be used throughout this report.

2.1. Piecewise Cubic Interpolants. We assume that we are given n data points (x_i,f_i), $i = 1(1)n$, where $x_1 < x_2 < \ldots < x_n$. A piecewise cubic function $p(x)$ with knot sequence $\{x_1, \ldots, x_n\}$ has the form

(2.1) $p(x) = c_i(x)$, $x_i \leq x < x_{i+1}$,

where $c_i(x)$ is a cubic polynomial. Such a function is said to be a piecewise cubic interpolant to the data (x_i, f_i) if it satisfies

(2.2) $p(x_i) = f_i$, $i = 1(1)n$.

(Such a piecewise cubic function is assumed to be continuous at the data points, or knots.) See Figure 2.1, for example.

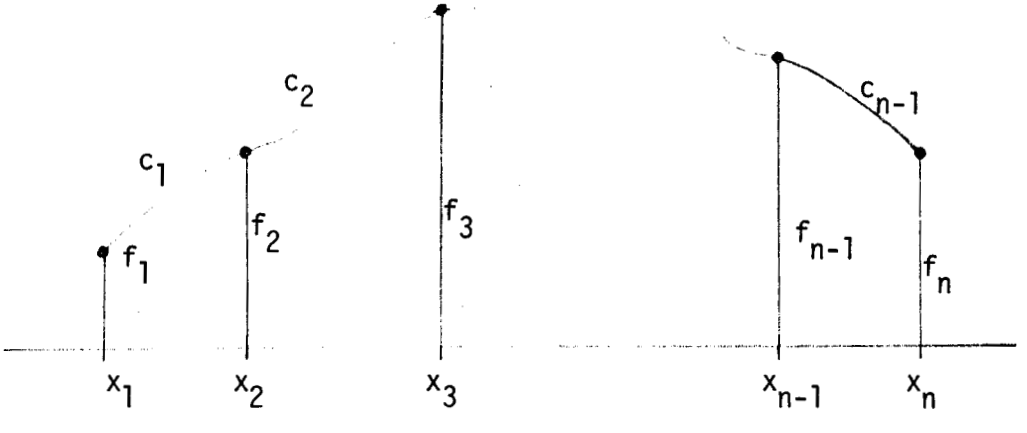


Figure 2.1 Piecewise Cubic Interpolant

2.2. The Hermite Representation. A cubic polynomial $c(x)$ is uniquely determined by its function and derivative values at two distinct points. Let $x_1 \neq x_2$, $c(x_1) = f_1$, $c(x_2) = f_2$, $c'(x_1) = d_1$, $c'(x_2) = d_2$. Then

$$c(x) = f_1H_1(x) + f_2H_2(x) + d_1H_3(x) + d_2H_4(x), \tag{2.3}$$

where the H_j are the cubic polynomials with the properties

$$\begin{aligned}
&H_1(x_1) = 1;\ H_1(x_2) = H_1'(x_1) = H_1'(x_2) = 0;\\
&H_2(x_2) = 1;\ H_2(x_1) = H_2'(x_1) = H_2'(x_2) = 0;\\
&H_3'(x_1) = 1;\ H_3(x_1) = H_3(x_2) = H_3'(x_2) = 0;\\
&H_4'(x_2) = 1;\ H_4(x_1) = H_4(x_2) = H_4'(x_1) = 0.
\end{aligned} \tag{2.4}$$

The polynomials $H_j(x)$, which are illustrated in Figure 2.2, are referred to as the Hermite basis functions for $[x_1,x_2]$. Equation (2.3) is the Hermite representation of $c(x)$.

If $p(x)$ is a continuously differentiable piecewise cubic function with knot sequence $\{x_1,\ldots,x_n\}$, $x_j \neq x_i$ when $j \neq i$, then $p(x)$ is uniquely determined by its function and derivative values at the knots. If $f_i = p(x_i)$, $d_i = p'(x_i)$, then the Hermite representation of $p(x)$ is given by (2.1) with

$$c_i(x) = f_iH_1^{(i)}(x) + f_{i+1}H_2^{(i)}(x) + d_iH_3^{(i)}(x) + d_{i+1}H_4^{(i)}(x), \tag{2.5}$$

where the $H_j^{(i)}(x)$ are the Hermite basis functions for the i-th subinterval $[x_i,x_{i+1}]$, $i = 1(1)n-1$. (See Figure 2.3.) We shall use this Hermite representation throughout this report.

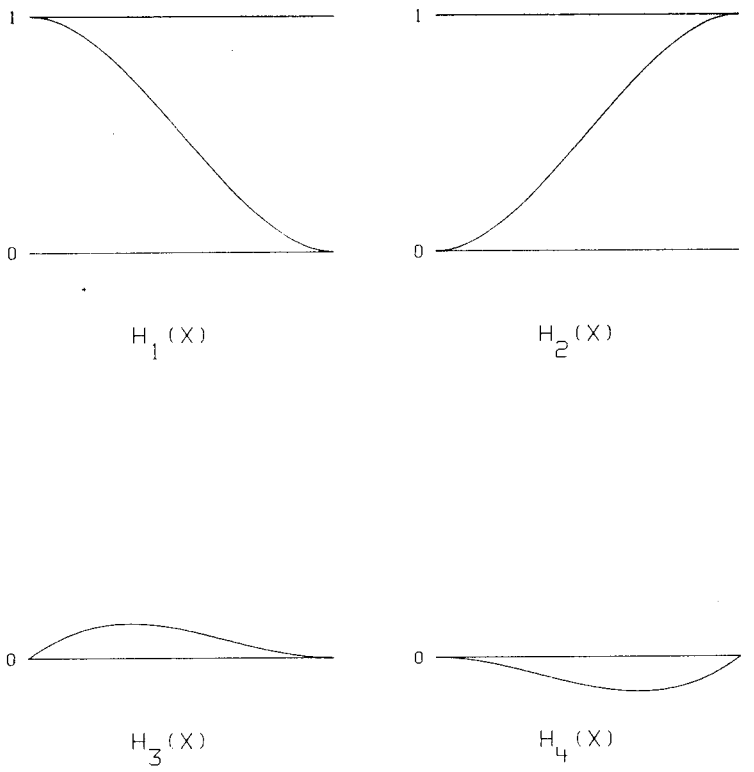


Figure 2.2. The Hermite Basis Functions

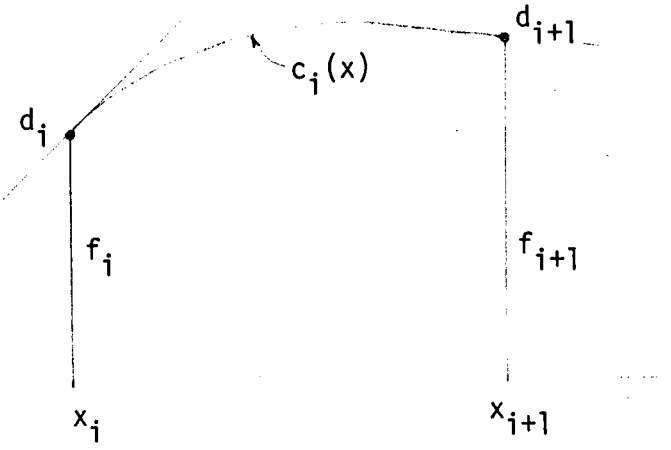


Figure 2.3 The Hermite Representation

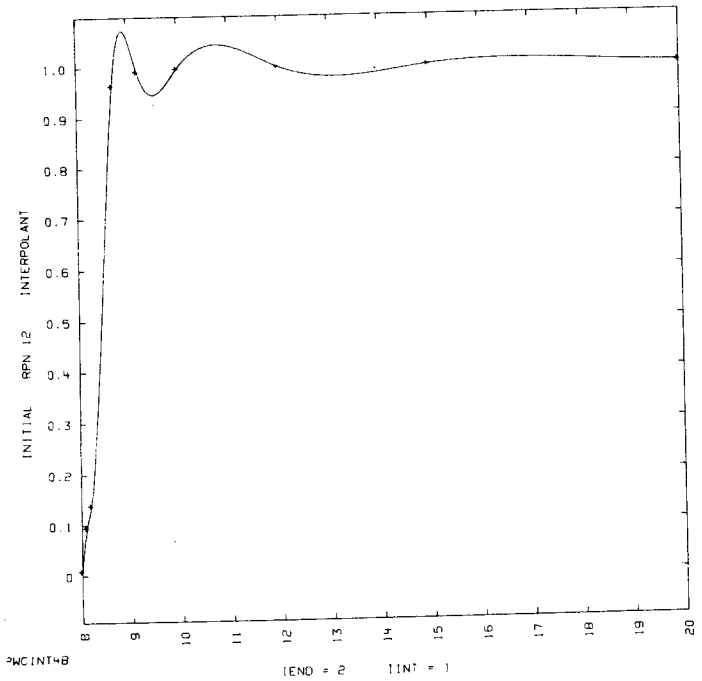
3. Existing Interpolation Methods.

If we are given both function values f_i and derivative values d_i at the data points x_i, then the piecewise cubic Hermite interpolant is uniquely determined by (2.1) and (2.5). In most applications, however, derivative values are not available. We shall assume that the derivative values d_i are at our disposal, and will study various ways to approximate them to produce the "best" interpolant.

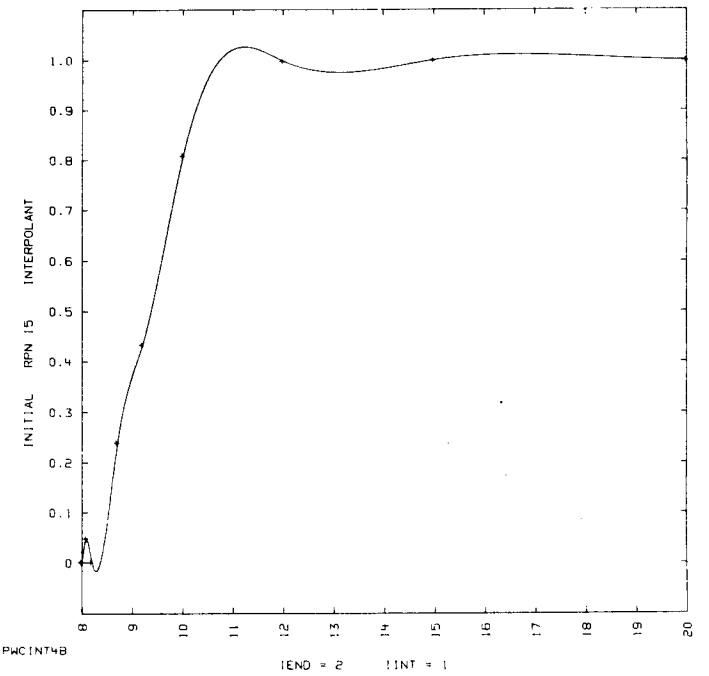
3.1 Cubic Splines. A cubic spline that interpolates the data (x_i,f_i) is a piecewise cubic function with knot sequence $\{x_1,\ldots,x_n\}$ that satisfies (2.2) and has continuous first and second derivatives at the interior knots $x_2, \ldots, x_{n-1}$. By counting parameters, one can see that there are two free parameters required to completely determine a cubic spline interpolant (see [2]). These are generally determined by specifying the first or second derivative values at the boundary points x_1, x_n.

Cubic splines have become quite popular in recent years. However, they can exhibit quite unphysical oscillations for certain types of problems, as illustrated in Figure 3.1. While different endpoint conditions give different interpolants, there simply is not enough freedom available in a cubic spline to eliminate the oscillations. To do this, we shall have to give up second derivative continuity.

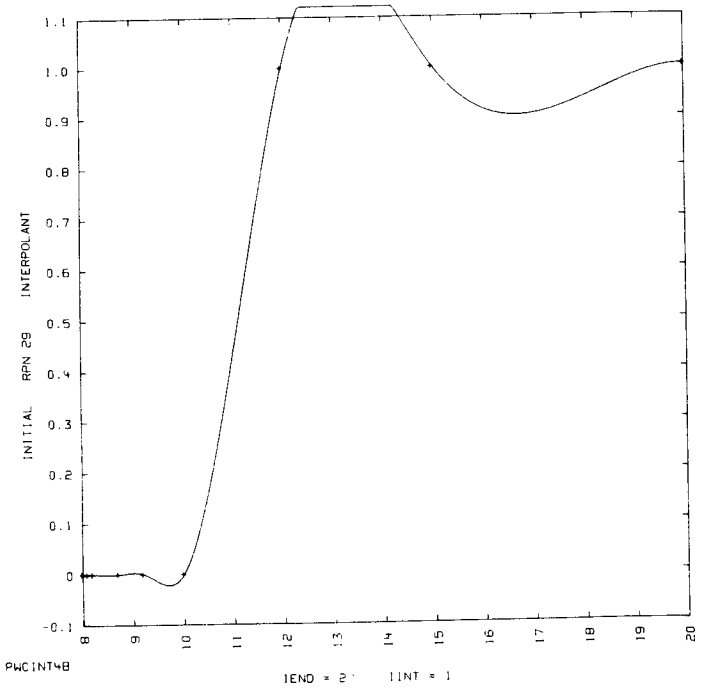
3.2. Finite Difference Approximations. One commond method for approximating the derivatives d_i in (2.5) is to use finite difference formulas. The result of approximating d_i in terms of f_{i-1}, f_i, f_{i+1} (quadratic approximation) is sometimes called osculatory interpolation. While bumps and wiggles are still present, they tend to be localized in regions where the data exhibits rapid changes in slope. (See Figure 3.2)



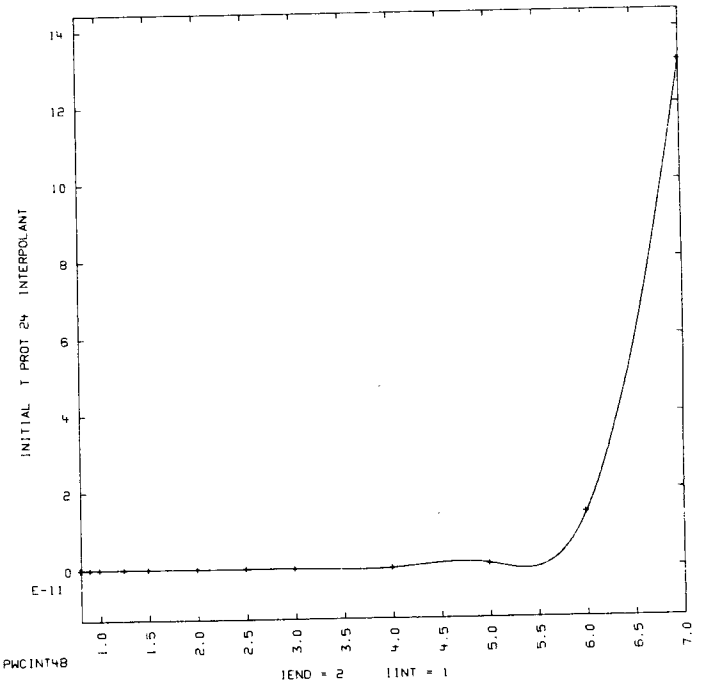
(a)



(b)

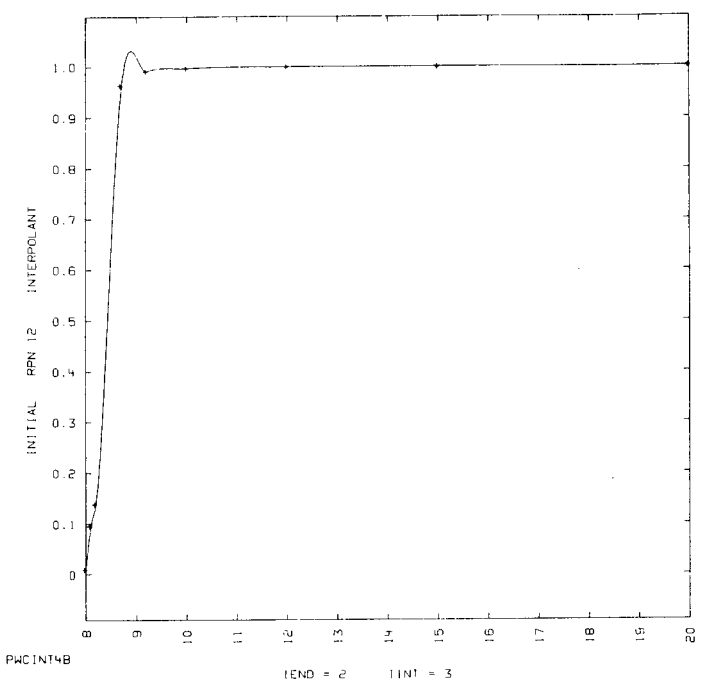


(c)

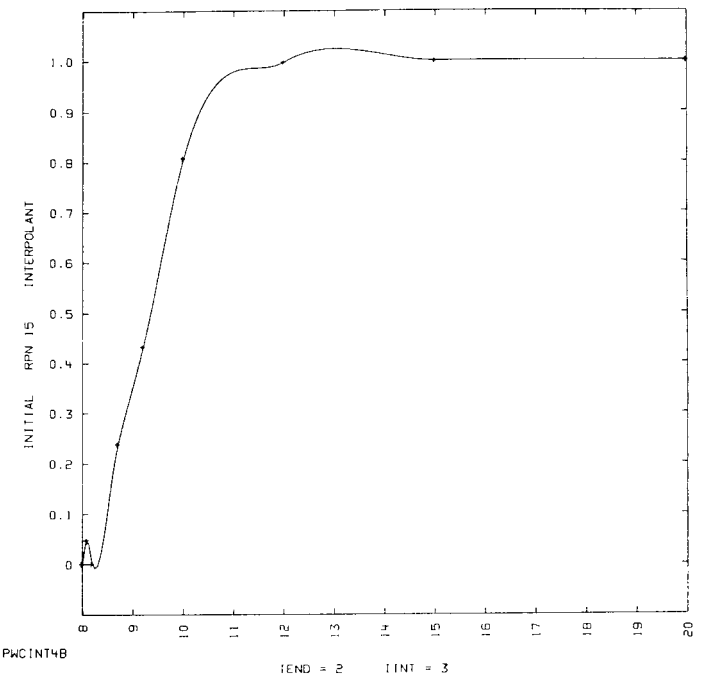


(d)

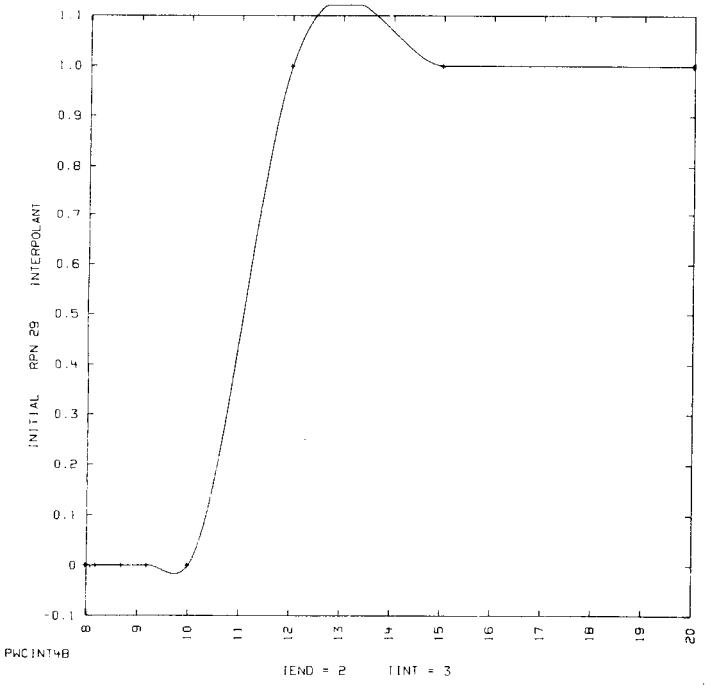
Figure 3.1. Cubic Spline Interpolants



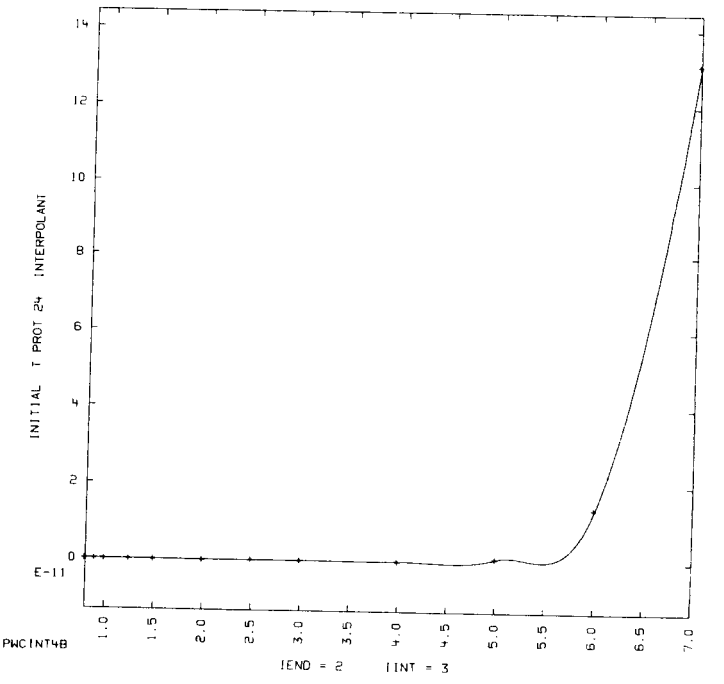
(a)



(b)



(c)



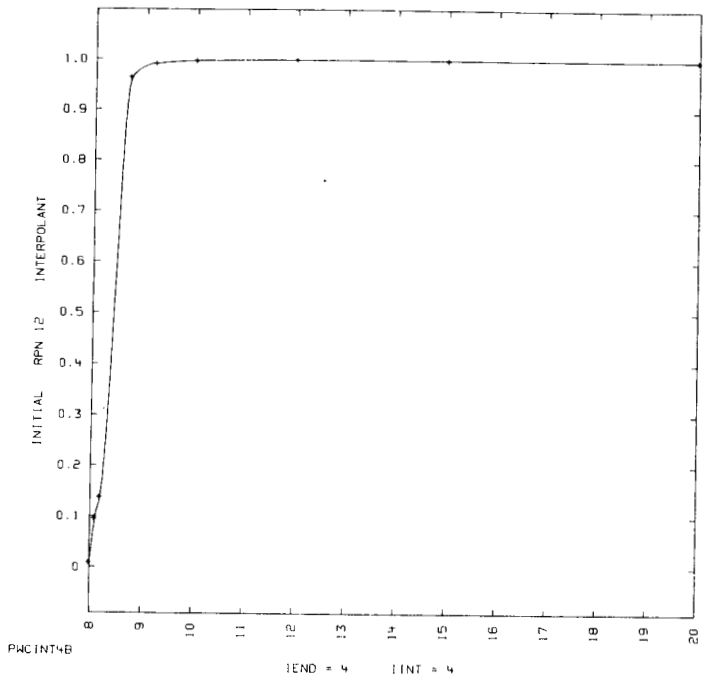
(d)

Figure 3.2. Interpolants from Three Point Finite Difference Approximations

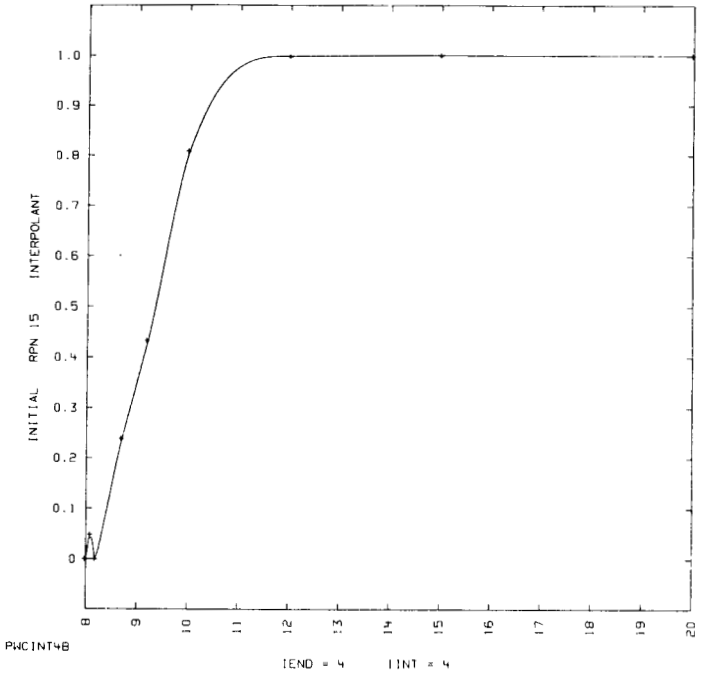
3.3. Akima's Formulas. In 1970, H. Akima [1] introduced a new method for determining the derivatives d_i that is intended to approximate the sort of curve a trained draftsman might draw through the given data points. This is implemented, for example, in IMSL subroutine IQHSCU [5], where the method is referred to as quasi-Hermite interpolation. This method does a rather good job on most of the data sets under consideration. Figure 3.3 shows the Akima interpolants for the data we have been considering. Note that the step-function of Figure 3.3(c) illustrates the behavior of Akima's formulas in an extreme case, where no other method tested to date gives acceptable results.

3.4. Other Methods. We mention briefly two other methods that were tried on these data sets and abandoned. One method is to use splines under tension [3], which are piecewise functions whose definition involves a "tension parameter". When this parameter is zero (no tension), the spline under tension reduces to an ordinary cubic spline; when the tension approaches infinity the spline under tension converges to a piecewise linear interpolant. The idea is to choose the tension parameter large enough to eliminate extraneous bumps and wiggles. Unfortunately, for these data a tension parameter large enough to do the job tended to make the curve nearly piecewise linear in regions where the ordinary spline was "good". Furthermore, the spline under tension is not a piecewise polynomial, so is much more expensive to evaluate than an ordinary spline.

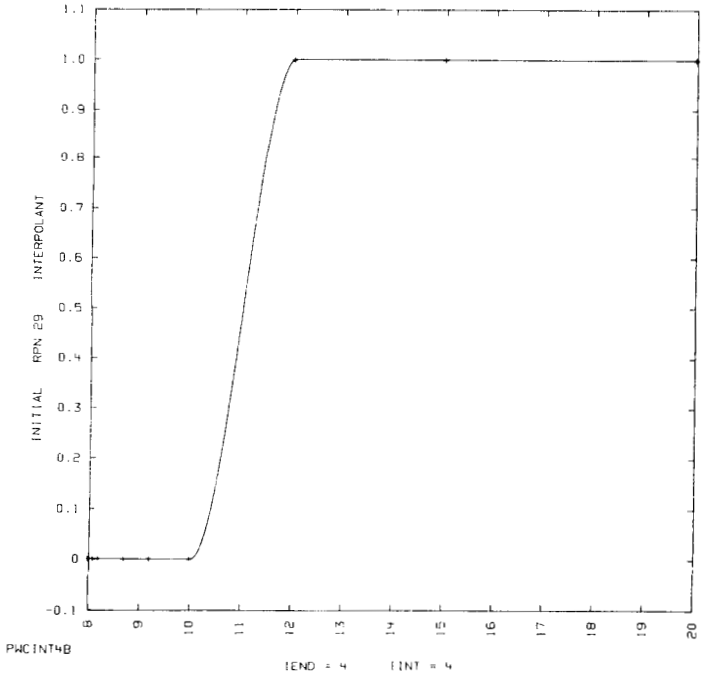
The second method tried was a data smoothing process whereby a cubic spline is allowed to depart somewhat from the interpolation conditions (2.2) in order to produce a "smoother" curve. This method is implemented, for example, in IMSL subroutine ICSMOU [5]. In this case, it was necessary to allow the spline to depart significantly from the data in order to smooth out the extraneous bumps and wiggles. The problem is not that the data has noise, and this was simply the application of an inappropriate method.



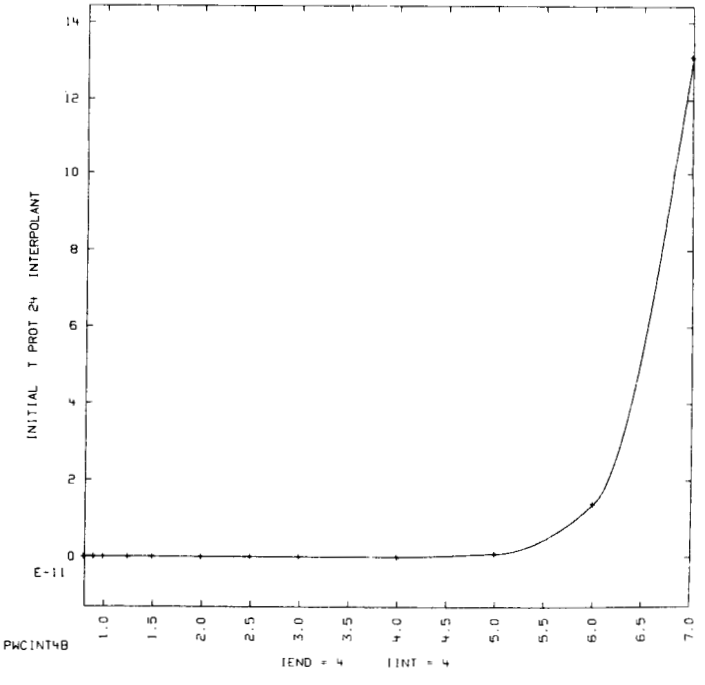
(a)



(b)



(c)



(d)

Figure 3.3. Interpolants from Akima's Formulas

4. Problems with Akima's Method.

As we have seen, the Akima method does a good job on a great variety of data sets. It was not until it had been applied to several hundred such data sets that it was found that it is not the universal answer. Several types of problems were encountered, and they are discussed separately. But first we need to explicitly display the formulas that are used.

4.1. Akima's Formulas. Let m_1, m_2, m_3, m_4 denote the slopes of the chords formed by five successive data points. (See Figure 4.1.)

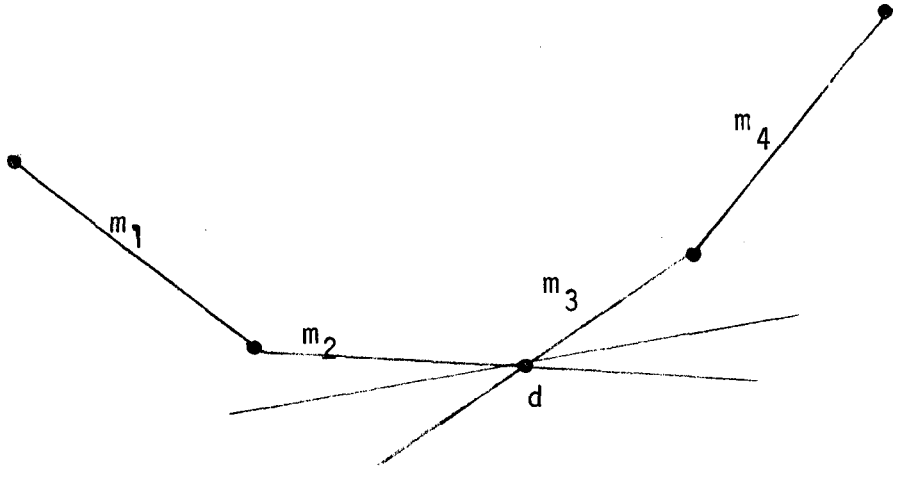


Figure 4.1. Akima's Notation

We wish to approximate the derivative d at the center point by a convex combination of the surrounding slopes:

$$d = \frac{\Delta_2 m_2 + \Delta_3 m_3}{\Delta_2 + \Delta_3}, \quad \Delta_2 + \Delta_3 \neq 0. \tag{4.1}$$

Akima defines the weights Δ_k by differences of slopes:

$$\Delta_2 = |m_4 - m_3|, \ \Delta_3 = |m_2 - m_1|. \tag{4.2}$$

This choice is justified geometrically in [1]. Unfortunately, (4.2) can produce cases in which $\Delta_2 = \Delta_3 = 0$, so that (4.1) cannot be used. In this case, Akima uses

(4.3) $$d = \frac{1}{2}(m_2+m_3), \quad \Delta_2 + \Delta_3 = 0 \ .$$

To completely define all of the derivatives, something special needs to be done for the end derivatives d_1, d_2, d_{n-1}, d_n (where there do not exist two data points on either side of the one at which the derivative is to be approximated.) Akima computes four fictitious data points at x_{-1}, x_0, x_{n+1}, x_{n+2}. The x-values x_{-1}, x_0, x_2, x_3 are located symmetrically about x_1, and the f-values at x_{-1} and x_0 are computed from the quadratic that passes through (x_i, f_i), $i = 1, 2, 3$. A similar quadratic extrapolation procedure is applied at the right end. This turns out to be equivalent to setting fictitious slopes according to

(4.4)
$$m_0 = 2m_1 - m_2 \ ; \ m_{-1} = 2m_0 - m_1 = 3m_1 - 2m_2 \ ;$$
$$m_n = 2m_{n-1} - m_{n-2} \ ; \ m_{n+1} = 2m_n - m_{n-1} = 3m_{n-1} - 2m_{n-2} \ ;$$

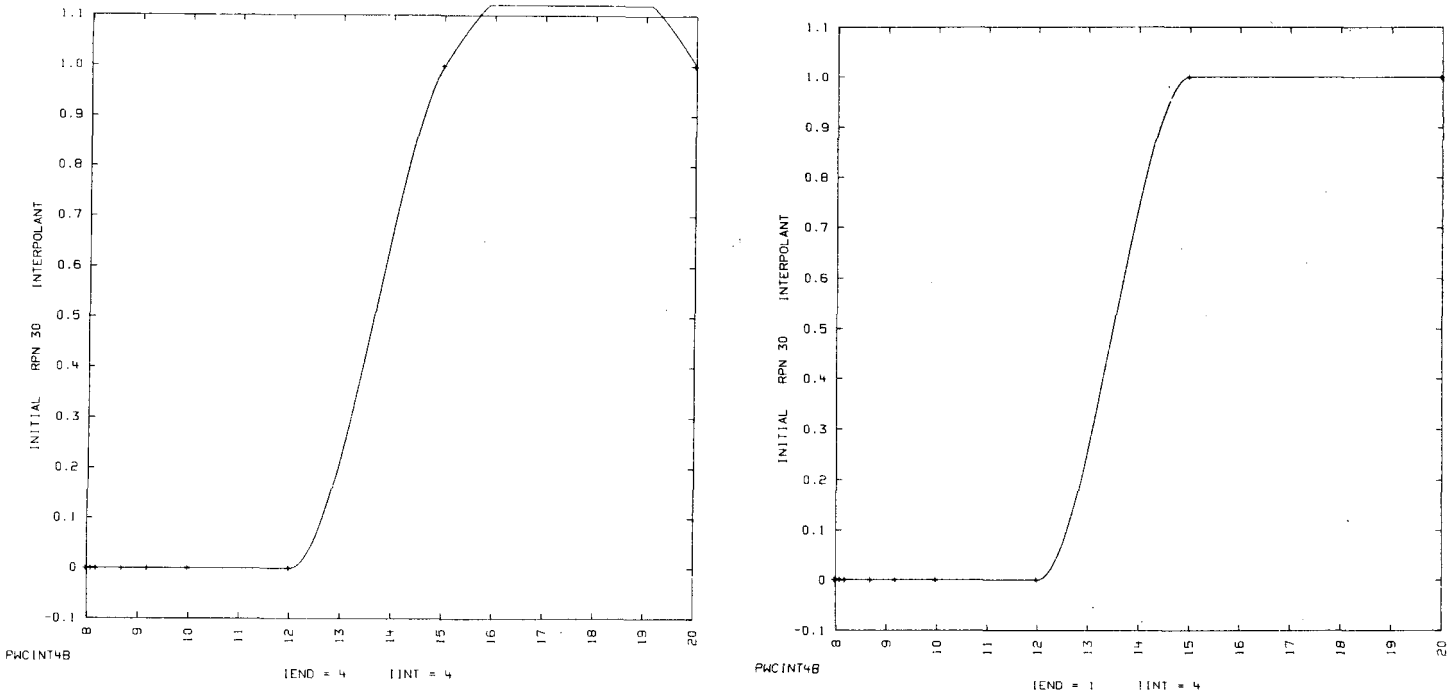
and then applying (4.1) - (4.3) at all n data points $x_1, \ldots, x_n$.

4.2 Endpoint Problems. The first problem discovered with the Akima method was an endpoint problem. This is most strikingly illustrated in Figure 4.2(a), where we have attempted to interpolate a step function similar to the one in Figure 3.3(c) except that there are now only two data points at the top of the "cliff". The difficulty here is that the quadratic extrapolant is not a good approximation of the actual behavior of the data. (We expect a monotonic function.)

This problem can be eliminated by using linear extrapolation. This is achieved by using

(4.5) $$m_{-1} = m_0 = m_1 \ ; \ m_{n+1} = m_n = m_{n-1}$$

instead of (4.4). The resulting interpolant is shown in Figure 4.2(b).

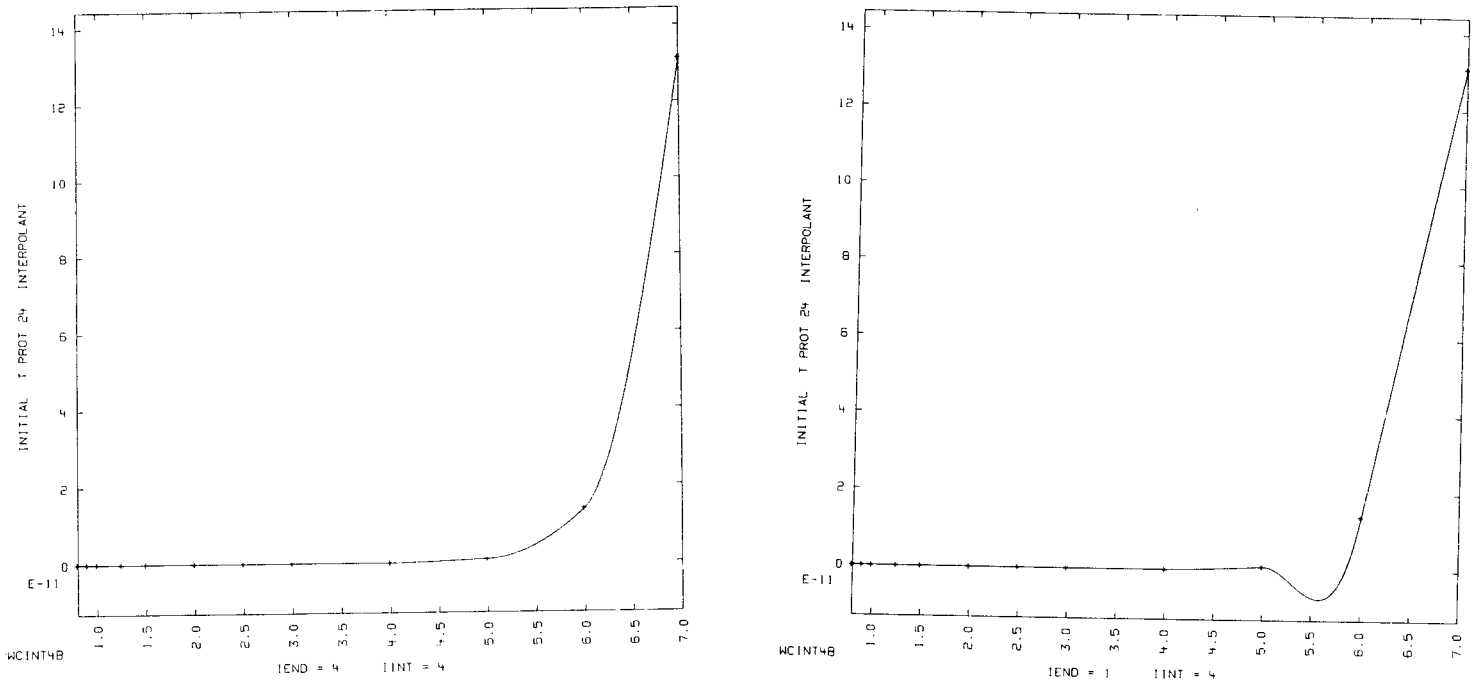


(a) (b)

Figure 4.2 Endpoint Problem with Fix

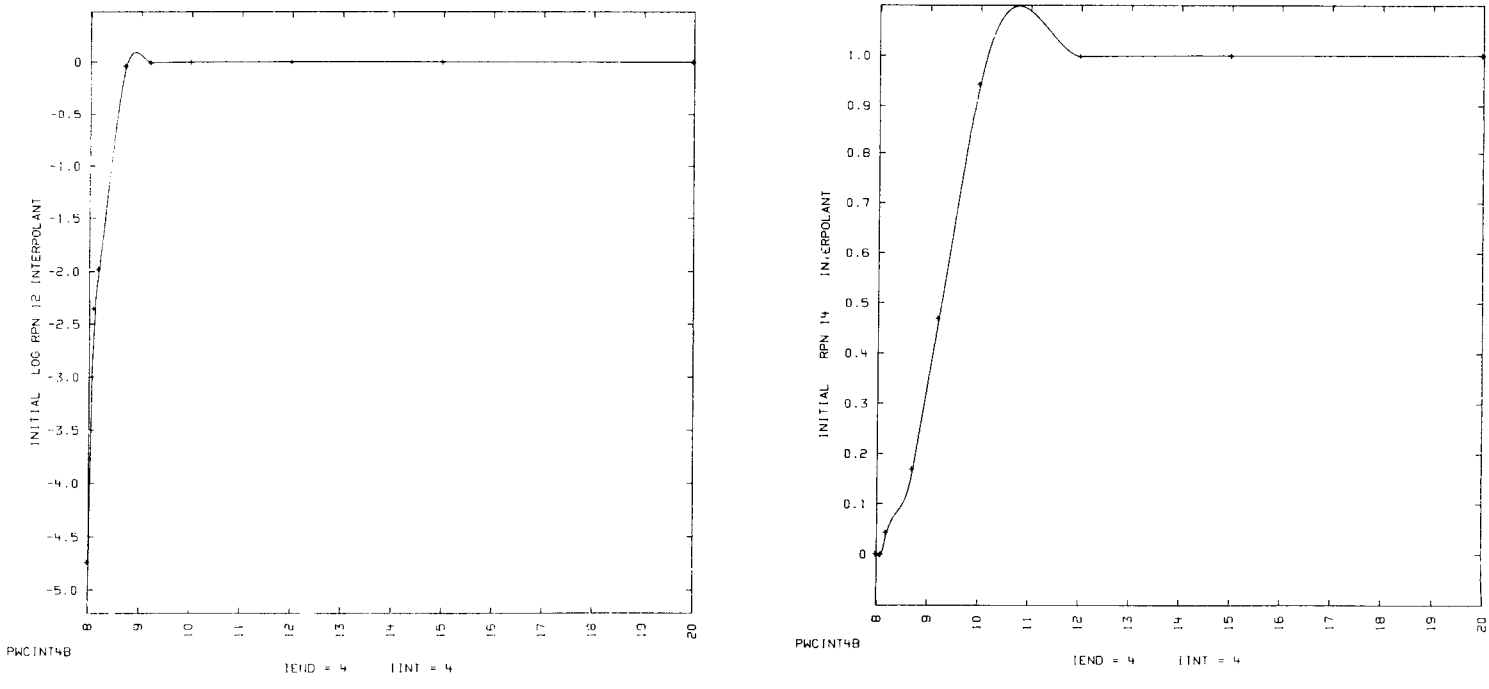
As it turns out, however, (4.5) is not a universal fix either. In figure 4.3 we have an example in which the original Akima formulas (4.4) produce a much more acceptable interpolant (a) than do (4.5), where a "bump" has appeared (b). Hence, more study is needed to produce appropriate end conditions.

4.3. Bumps. Our first example of an Akima interpolant with a bump was Figure 4.3(b). While we have seen that the Akima method is much less prone to extraneous bumps and wiggles than the other piecewise cubic interpolation methods considered here, Figure 4.4 gives two examples to demonstrate that bumps are not strictly an endpoint problem, but can occur whenever a computed derivative value at one of the endpoints of a subinterval is very much larger than the slope of the cord.



(a) (b)

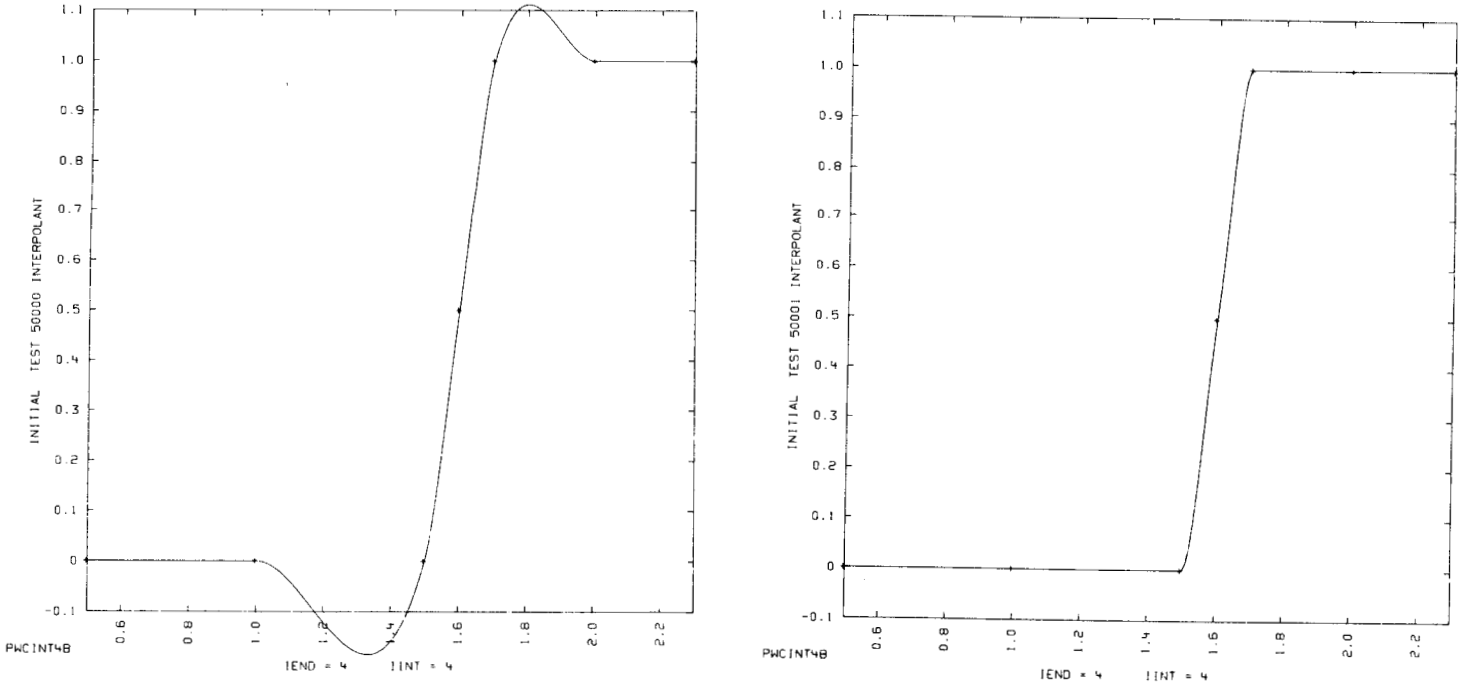
Figure 4.3 Fix Creates Endpoint Problem



(a) (b)

Figure 4.4. Akima Interpolants with Bumps

4.4. Discontinuous Behavior. In attempting to construct sample data sets which exhibit bumps, another problem with the Akima formulas was encountered. The idea was to add a data point partway up the cliff of a step function such as in Figure 3.3(c). As the point gets closer to the top (or bottom) we expected a bump to appear. To our surprise, however, when the point was exactly half-way up, two bumps appeared (Figure 4.5(a)). The mystery deepened when it was discovered that any change at all in the location of the point again produced an acceptable interpolant (Figure 4.5(b)).



(a) (b)

Figure 4.5. Discontinuous Behavior of Interpolant

The source of this difficulty can be discovered by examining formulas (4.1) and (4.3). It turns out that (4.1), considered as a function of Δ_2 and Δ_3, has an essential singularity at $(\Delta_2,\Delta_3) = (0,0)$. Suppose the point (Δ_2,Δ_3) approaches the origin along a line with slope β/α $(\alpha,\beta\neq 0)$. Substituting $\alpha\Delta_3 = \beta\Delta_2$ into (4.1) and assuming $\Delta_2 \neq 0$ we obtain

(4.6)
$$d(\Delta_2,\Delta_3)\Big|_{\alpha\Delta_3=\beta\Delta_2} = \frac{\alpha m_2 + \beta m_3}{\alpha + \beta}$$

The quantity on the right in (4.6) is independent of Δ_2, so is the limit as $(\Delta_2,\Delta_3) \to (0,0)$ along the line $\alpha\Delta_3 = \beta\Delta_2$. Since this clearly has different values for different values of β/α, (4.1) has an essential singularity at (0,0). Akima's formula (4.3) corresponds to the choice $\beta/\alpha = 1$, which happens to be very inappropriate for the particular set of data in Figure 4.5. While this discontinuous change of interpolant behavior with changes in the data is a very serious defect mathematically, it is extremely unlikely that it will cause difficulty in practice. (As it turns out, this problem and the endpoint problem are both taken care of by the process designed to eliminate bumps.)

5. An Idea and Its Refinement.

5.1. The Idea — Iterative Derivative Improvement. The germ of an idea is generated by study of Figure 5.1, where a close look is taken at an interval containing a bump.

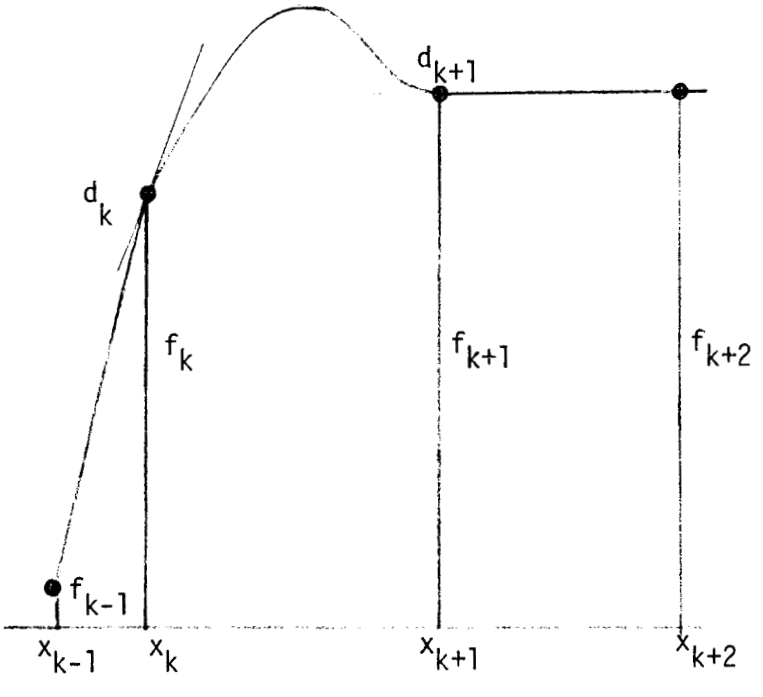


Figure 5.1. Close-up of a Bump

If we view the slope d_k as a "knob" that can be turned to alter the shape of the curve, it is intuitively appealing to consider what would happen as we "turn down" the value of d_k. It seems that the bump should become smaller. If d_k is too small (say, zero), we will introduce a wiggle. Thus it appears that an acceptable interpolant should be achievable for some intermediate value of d_k.

This heuristic discussion suggests that we use the Akima formulas only to generate initial guesses for an iterative procedure. Given some measure of badness for such a piecewise cubic interpolant, use some optimization procedure to adjust the derivatives to minimize the "badness" of the interpolant. While the idea seems simple, there were many hurdles to be leaped before a successful implementation was achieved. These form the subject matter for the rest of this section.

5.2. Measures of Badness. The most difficult problem, and the most interesting, is to determine a quantitative model for the badness of a curve. We need to be able to compute some measure that varies continuously with the derivative values d_i and which becomes larger when a human observer declares that the curve has gotten "worse". (No, we are not willing to plug a human into the computer for this function evaluation.) Since "badness" appears to be a (relatively) local property of a piecewise cubic interpolant, we choose to assign a badness measure $b_i(d_i,d_{i+1})$ to each subinterval $[x_i,x_{i+1}]$, $i = 1(1)n-1$. Note that b_i depends only on two of the d's. The overall badness of a piecewise cubic interpolant is then some norm of the vector $b = (b_1,\ldots,b_{n-1})$. We write this norm in the form

$$(5.1) \qquad b(\underset{\sim}{d}) = \sum_{i=1}^{n-1}{}^{*} w_i \left[b_i(d_i,d_{i+1}) \right]^{\alpha},$$

where the w_i are some weights which are at our disposal and $\alpha \geq 1$. We assume $b_i \geq 0$ for all i. (Thus, $w_i \equiv 1$ and $\alpha = 2$ gives the ordinary Euclidean norm; $w_i \equiv 1$ and $\alpha = 1$, the L_1-norm.)

Thus we have to not only choose the form for the individual badness measures b_i, but also the weights w_i and the power α in (5.1). After a good deal of experimentation it was determined that the L_1-norm,

$$b(d) = \sum_{i=1}^{n-1} b_i(d_i d_{i+1}), \tag{5.2}$$

works as well as any, and is easier to compute than the general form (5.1). (The reader is invited to try other norms if they seem appropriate for his/her application.)

Many measures of badness were tried, and we are not yet satisfied that our "final" choice is "right". Among these were:

- Area between chord and curve.
- Fraction of interval over which sign of derivative disagrees with sign of chord.
- Arc length of curve.
- Ratio of length of curve to length of chord.

The choice that finally proved successful was

$$b_i = \frac{s_i - c_i}{c_i} = \frac{s_i}{c_i} - 1 \quad , \tag{5.3}$$

where s_i is the length of the cubic over $[x_i, x_{i+1}]$ and c_i is the length of the chord.

5.3. Local Minimization. The procedure discussed above, using a general unconstrained minimization routine, experienced severe convergence difficulties. It was then decided to try an iterative local minimization. If the k-th interval is "worst" (that is, $b_k \geq b_i$ for all i), we minimize the sum of badness measures over the three intervals whose badness is affected by d_k and d_{k+1}:

$$(5.4) \qquad \min_{d_k, d_{k+1}} \sum_{i=k-1}^{k+1} b_i(d_i, d_{i+1}),$$

where we omit the first term if k = 1, the last if k = n-1. Here d_{k-1} and d_{k+2} are held fixed during this two-dimensional optimization. We then select the interval that is now worst. If it is still the k-th, we try to improve the next-worst before giving up. This procedure, with the measure in (5.3), was moderately successful, but something was still wrong.

5.4 Scaling. The measure (5.3) identified the interval with the obvious bump in Figure 4.4(b) as worst, all right, but not by a significant margin over other obviously "good" intervals. A closer examination of this situation revealed a scaling problem. Since the independent variable in this data set ranges over [0,12], while the dependent variable is constrained to [0,1], the bump truly is small on the original scale. On the scale of our plots, however, the bump is significant. The solution is to scale the x- and y-values needed to compute b_i in (5.3) as follows.

$$(5.5) \qquad \hat{x} = x/\Delta x; \quad \hat{f} = f/\Delta f$$

where the scale factors are given by

$$(5.6) \qquad \Delta x = x_{max} - x_{min} = x_n - x_1; \quad \Delta f = f_{max} - f_{min}.$$

($\Delta f = 1$ for most data sets under consideration here.) A sample calculation to illustrate this effect is given in Figure 5.2, for the worst interval in Figure 4.4(b). Here we approximate the arc length by the sum of the lengths of four chords. Note that the smallest possible value for the ratio s_i/c_i is 1 in either case.

With this modification, the procedure outlined in the previous section is quite successful in eliminating undesirable bumps and wiggles from a piecewise cubic interpolant. Some examples are given in Figure 5.3. These

x_j	$c(x_j) \equiv f_j$	Δf_j	Δx_j^2	Δf_j^2	Δs_j^2	Δs_j
10.0	0.94374					
		0.14368	0.25	0.02064	0.27064	0.52023
10.5	1.08742					
		0.00378	0.25	0.00001	0.25001	0.50001
11.0	1.09120					
		-0.05622	0.25	0.00316	0.25316	0.50313
11.5	1.03498					
		-0.03634	0.25	0.00312	0.25132	0.50132
12.0	0.99864					

$$s_i \approx 2.02471 = \sum \Delta s_j$$

$$c_i = \text{chord (orig.)} = \sqrt{(12.-10.)^2 + (0.99864-0.94374)^2} \doteq \sqrt{4.+0.00301} \doteq 2.00075$$

$$s_i/c_i \approx 1.01198; \quad b_i \approx 0.01198$$

Figure 5.2(a). Calculation in original coordinates

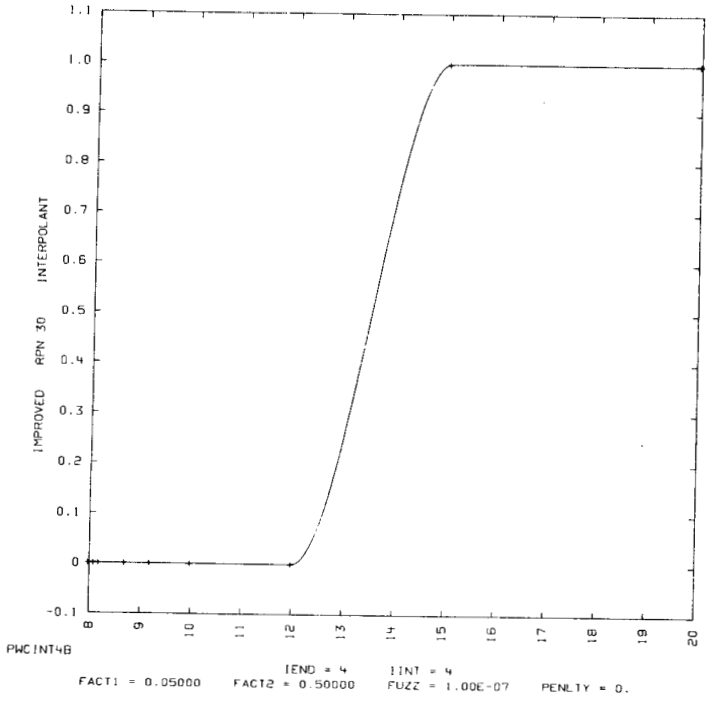
$\Delta \hat{x}_j^2$	$\Delta \hat{f}_j^2$	$\Delta \hat{s}_j^2$	$\Delta \hat{s}_j$
0.00174	0.02064	0.02238	0.14939
0.00174	0.00001	0.00175	0.04179
0.00174	0.00316	0.00490	0.06997
0.00174	0.00312	0.00306	0.05528

$$\hat{s}_i \approx 0.31663 = \sum \Delta \hat{s}_j$$

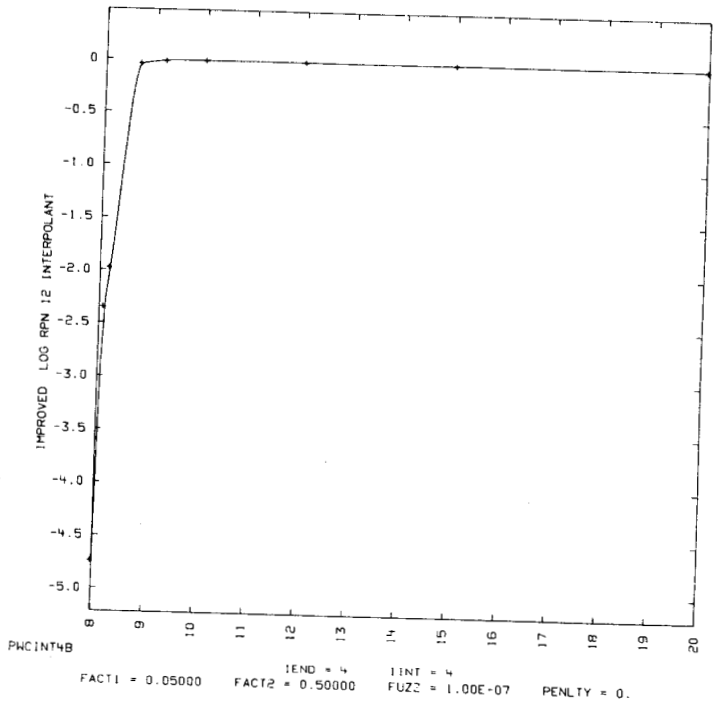
$$\hat{c}_i = \text{chord (modif.)} = \sqrt{(2\ /12)^2 + (0.99864-0.94374)^2} \doteq \sqrt{0.02778+0.00301} \doteq 0.17348$$

$$\hat{s}_i/\hat{c}_i \approx 1.80437; \quad \hat{b}_i \approx 0.80437$$

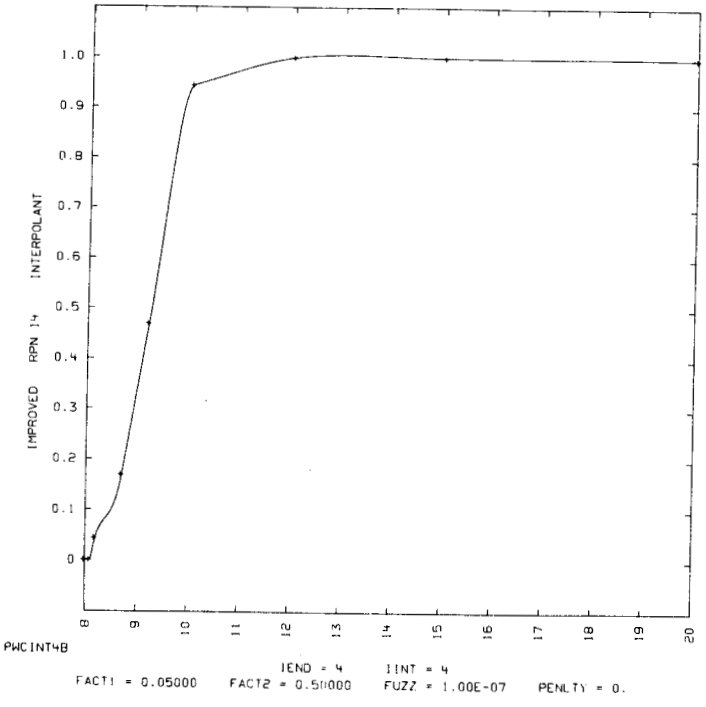
Figure 5.2(b). Calculation in Scaled Coordinates, $\Delta x = 12$, $\Delta f = 1$.



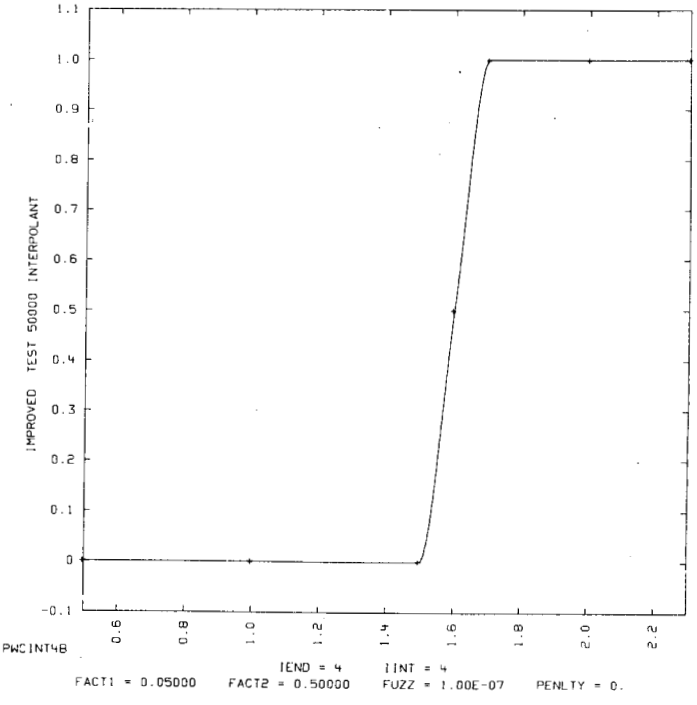
(a)



(b)



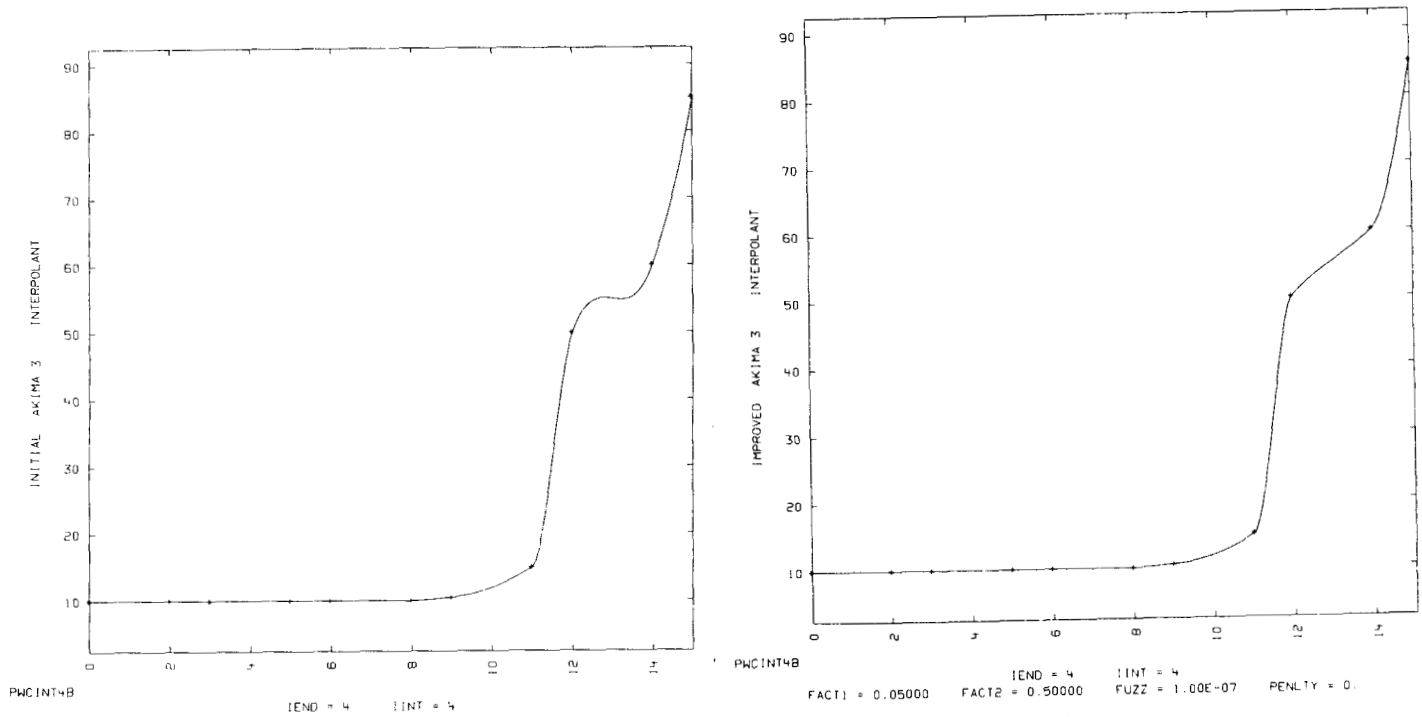
(c)



(d)

Figure 5.3 Results of Iterative Improvement

correspond to Figures 4.2(a), 4.4(a), 4.4(b), and 4.5(a), respectively. This procedure was even able to improve on Akima's method for the third sample data set in Akima's paper [1]. This is illustrated by the before-after sequence in Figure 5.4.



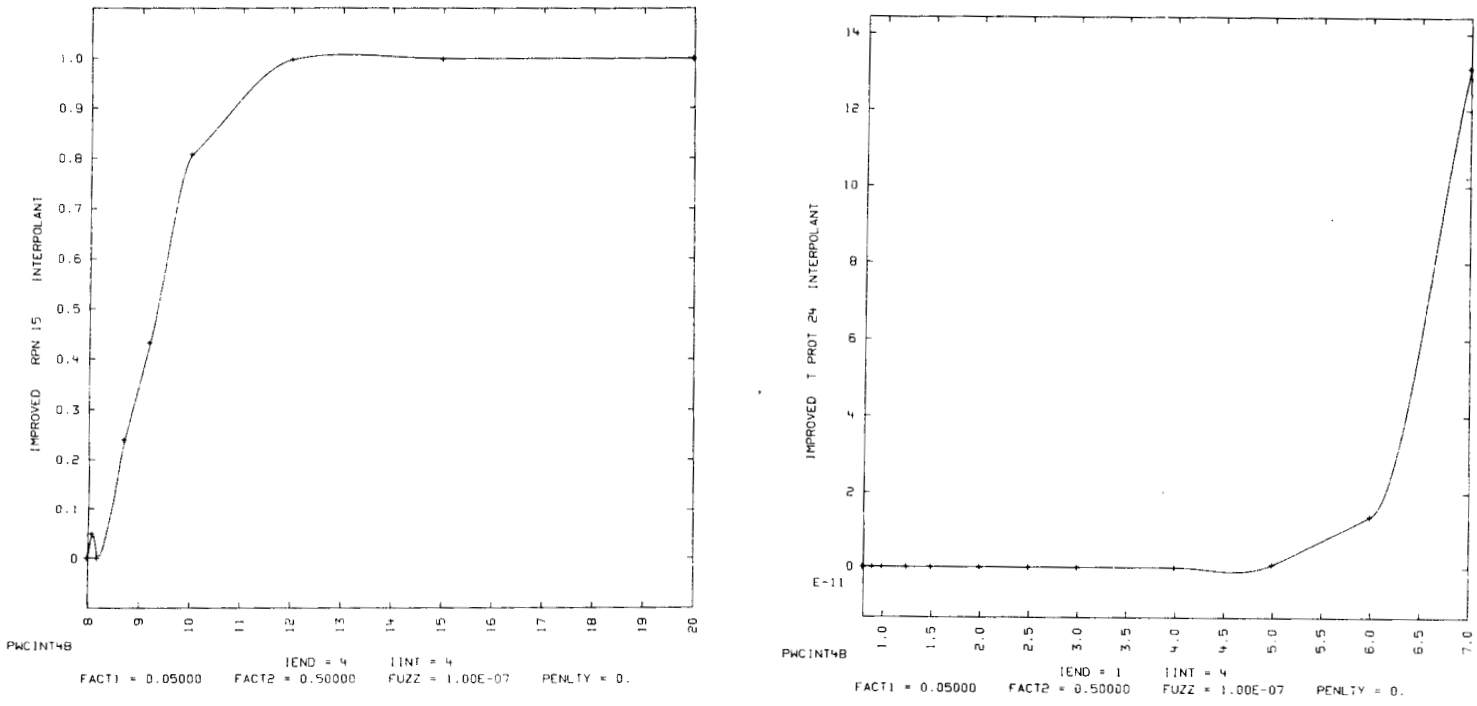
(a) (b)

Figure 5.4. Akima's Third Data Set

Note: Since much of the experimentation reported earlier that resulted in the decision to use local optimization with (5.3) was done before scaling was introduced, it may well be that some of these other options should be re-examined.

5.5. Standard Curve. Just as we had become hopeful for the end to our search for an acceptable piecewise cubic interpolation scheme, the examples in Figure 5.5 were discovered. Here the original Akima interpolants (Figure 3.3(b) and 3.3(d)) appear to be "better" than the "improved" ones. A careful examination suggests that we have succeeded in making the curves too flat on the worst interval by using the chord as standard curve. We must search for a "rounder" standard curve whose arc length is still easy to compute. Its arc length will replace

c_i in (5.3). Since the numerator can now become negative, must we also introduce an absolute value, to penalize curves that are too flat.



(a) (b)

Figure 5.5. "Improved" Interpolant Looks Worse

Our choice of standard curve for the k-th subinterval is as follows:

(a) If we are in an end interval (k=1 or k=n-1) or if the four points $P_i = (x_i, f_i)$, i = k-1(1)k+2 do <u>not</u> form a convex quadrilateral, use the cord;

(b) If $P_{k-1}P_kP_{k+1}P_{k+2}$ is a convex quadrilateral, then use that circle that passes through P_k and P_{k+1}, which is tangent to one of the segments $P_{k-1}P_k$ or $P_{k+1}P_{k+2}$, and which has the larger radius. See Figure 5.6, where the solid circle is chosen as the standard curve.

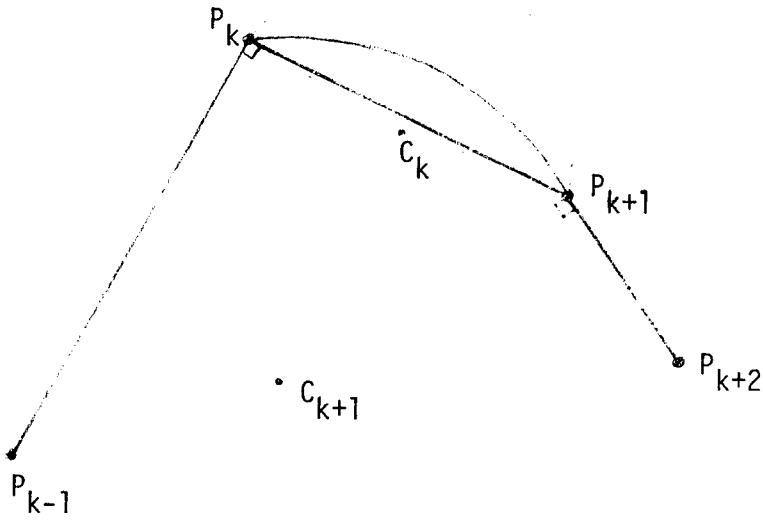
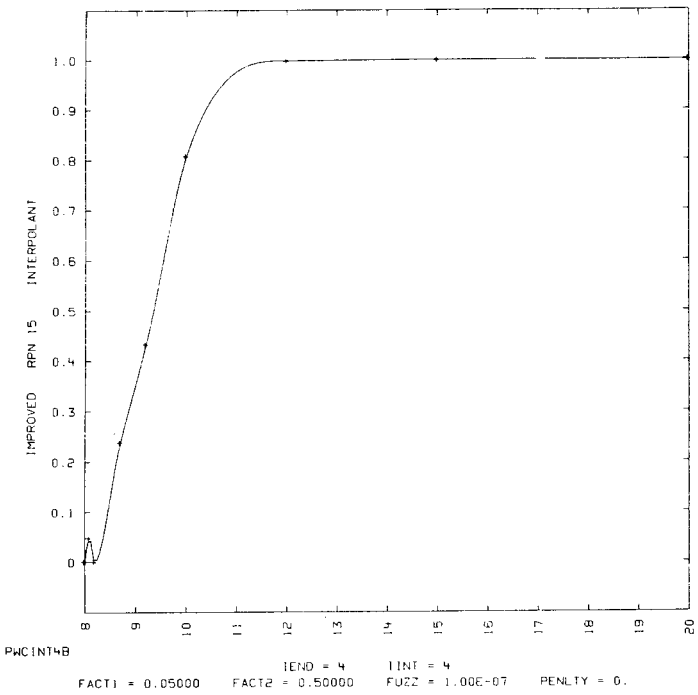


Figure 5.6. The Tangent Circles

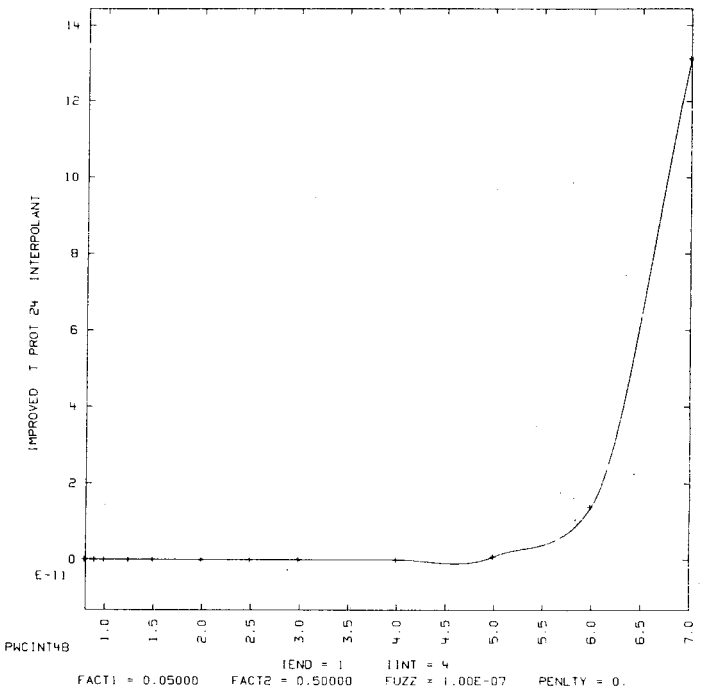
C_j is the center of the circle tangent at P_j.

This proves to be not too complicated a calculation. Use of the length of this modified standard curve in (5.3) results in the interpolants plotted in Figure 5.7. These correspond to Figures 5.5(a), 5.5(b), 5.3(b), and 5.3(c), respectively. The new interpolants corresponding to Figures 5.3(a), 5.3(d) and 5.4 are indistinguishable from the earlier plots.

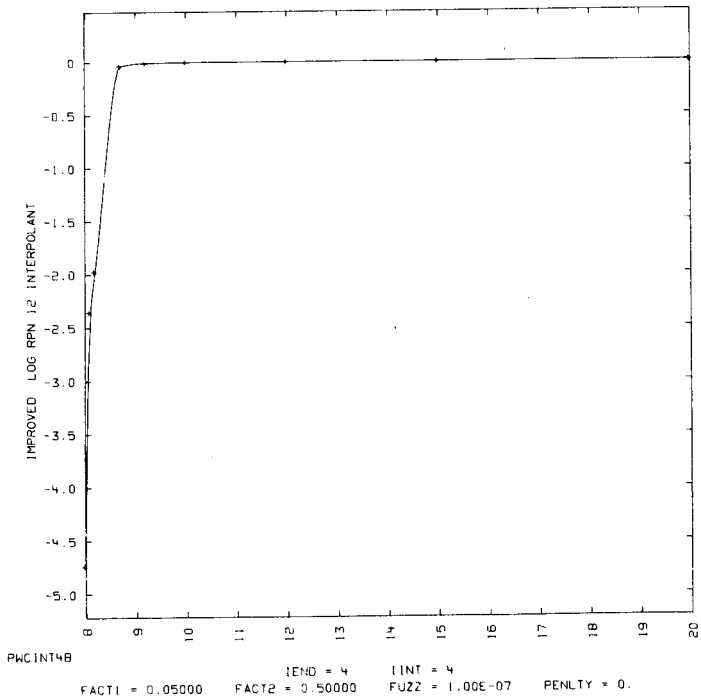
5.6 Constraints. One final adjustment was needed before Figure 5.7(a) could be produced. On the first attempt an interpolant was produced which wiggles about the chord, rather than staying above it. The difficulty is that with the length of some curve other than the chord in (5.3) the objective (5.4) can have more than one local minimum. Our unconstrained optimization procedure had simply converged to the wrong one.



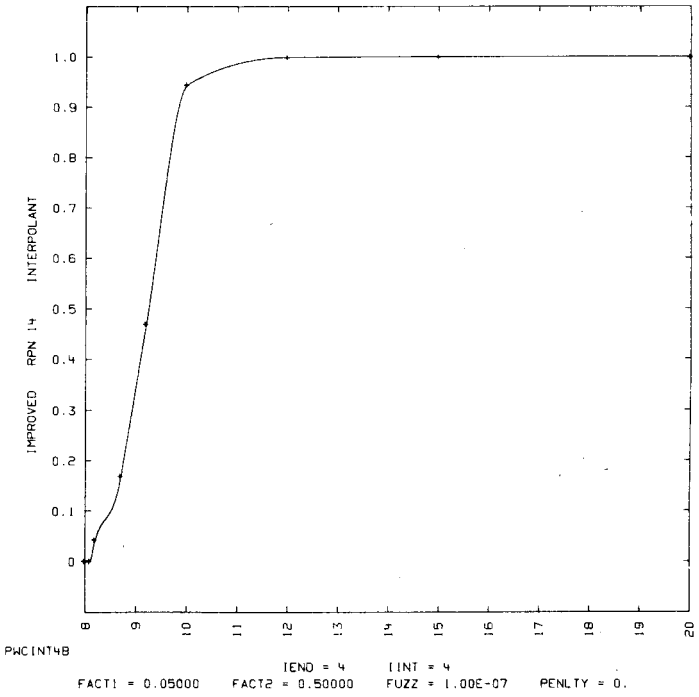
(a)



(b)



(c)



(d)

Figure 5.7. Results of Iterative Improvement Using Tangent Circle

Adding the simple bound constraints

(5.7) $$|m_{k-1}| < |d_k| < |m_k|, \quad k = 2(1)n-1,$$

tends to force the cubic to lie in the triangle determined by P_kP_{k+1} and the intersection of the extended segments $P_{k-1}P_k$, $P_{k+1}P_{k+2}$ as depicted in Figure 5.8. (Unfortunately, these are only necessary, not sufficient conditions.) Rather than fixing the end derivatives, we (quite arbitrarily) require

(5.8) $$\tfrac{1}{2}|m_1| \leq |d_1| \leq 2|m_1|; \quad \tfrac{1}{2}|m_{n-1}| \leq |d_n| \leq 2|m_{n-1}|.$$

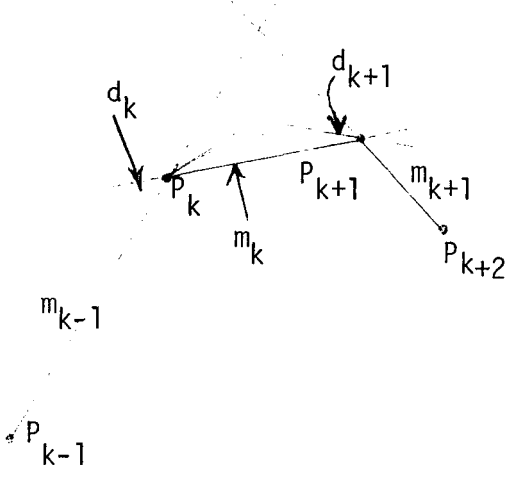


Figure 5.8. Constraint Justification.

6. Summary of the Overall Iterative Improvement Process.

In Figure 6.1 is a flowchart that summarizes the overall iterative improvement process. A few observations are in order.

6.1. Initial Guesses. While we have been thinking in terms of using the Akima formulas to generate initial values for the derivatives, the process in Figure 6.1 is really independent of the source of the initial guesses. For example, one obtains results such as those in Figure 6.2 if the cubic spline interpolants of Figure 3.1 are used for the initial guesses. Here the left-hand picture in each pair is the result of adjusting the spline derivatives to satisfy the constraints, while the right-hand picture is the final interpolant. The initial curves were given in Figures 3.1(a) and 3.1(b).

6.2 Iteration Parameters. As is usual with such an iteration procedure, there are a number of iteration parameters that the user (or code designer) can adjust to "tune" the algorithm to a particular class of problems. Among these are the following. The default values (given in brackets []) have been determined by experimenting with many data sets of the type shown in this report.

FACT1: Cutoff b_i-value for "bad" intervals (see first test in flowchart) [0.05].

FACT2: Minimum relative reduction in objective (5.4) required in order for improvement to be considered successful [0.5].

FUZZ: Fuzz to be used in tests against zero in computation of the length of the standard curve [1.0E-7].

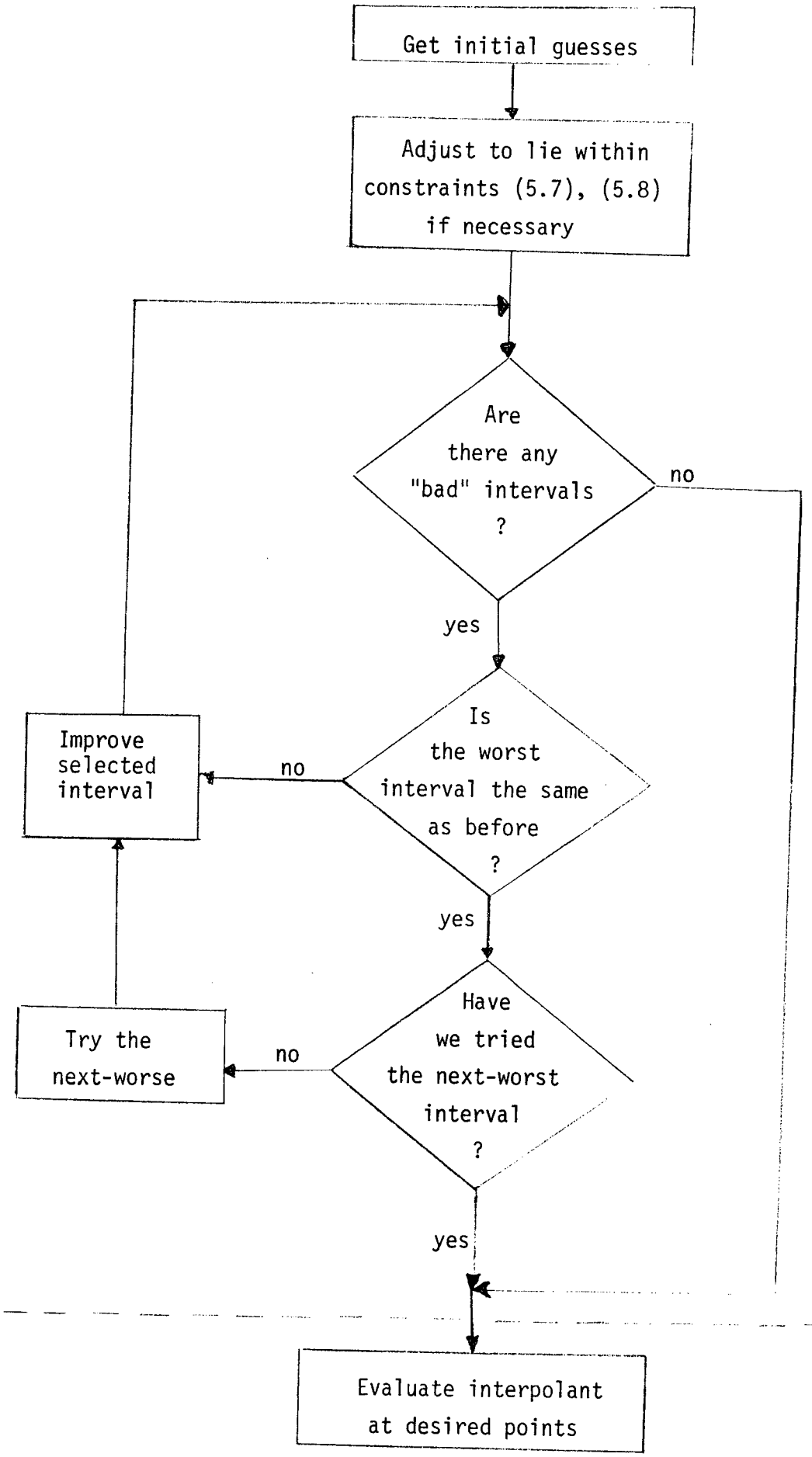
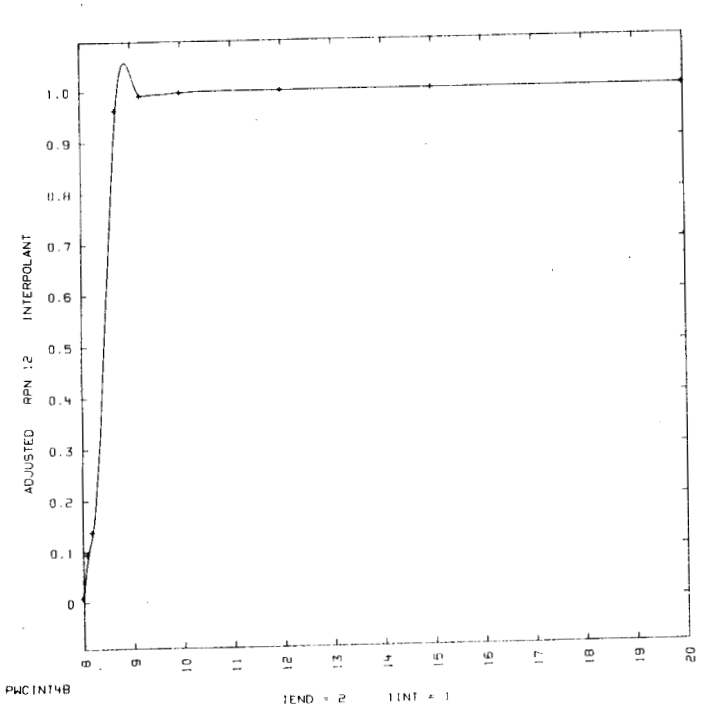
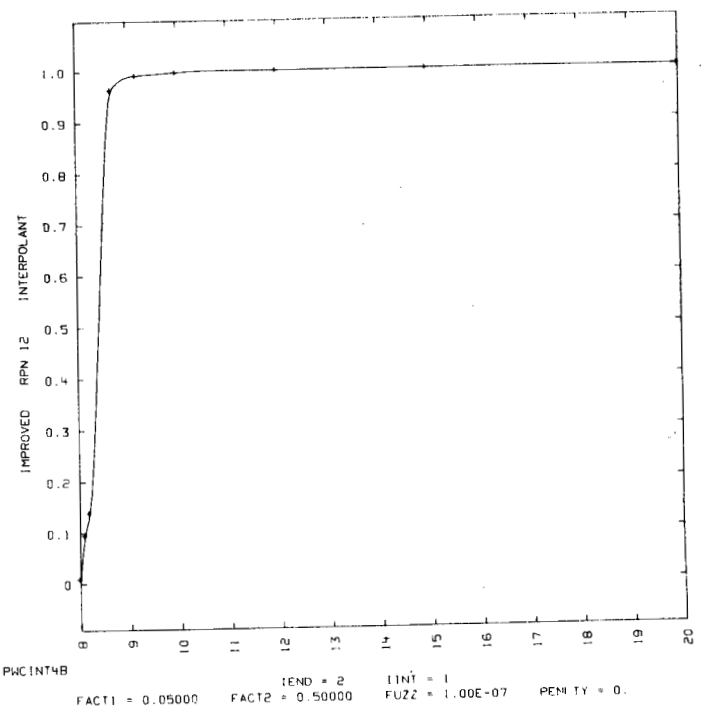


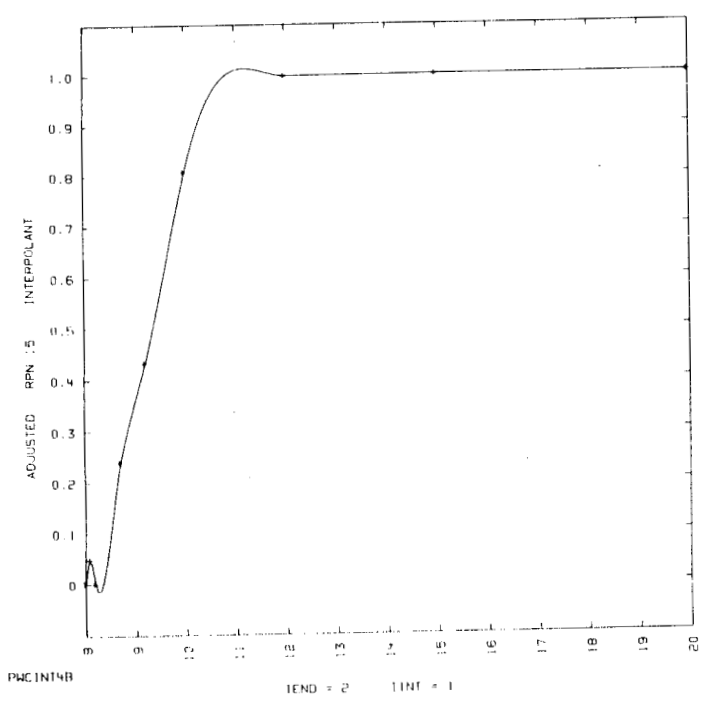
Figure 6.1. The Overall Iterative Improvement Process



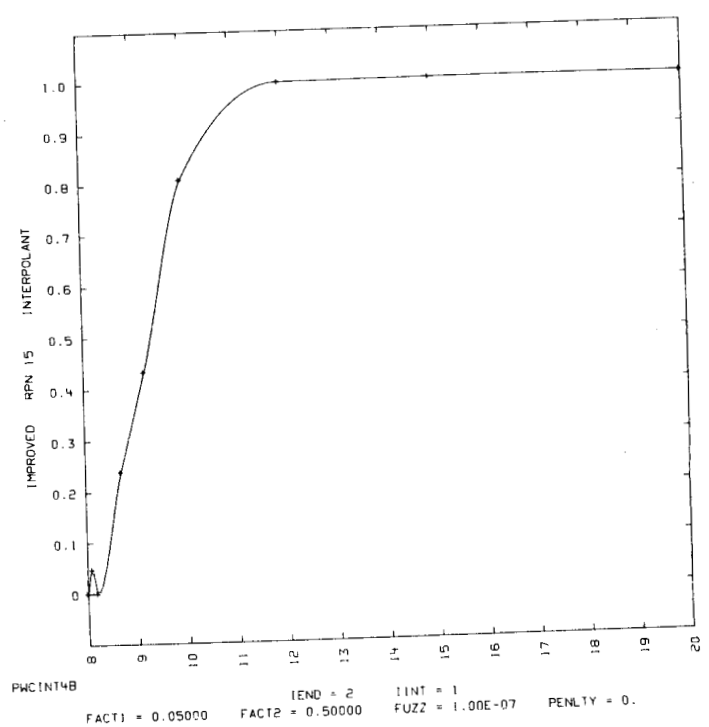
(a)



(b)



(c)



(d)

Figure 6.2. Improvement of Cubic Spline Interpolant

PENLTY: Factor to be applied to a penalty term*, which is an approximation to the area of the curve outside the range $f_{min} \leq p(x) \leq f_{max}$, added to the objective (5.4) [0.].

CONVC: Convergence criterion used in the two-dimensional minimization procedure [1.0E-3].

MAXFN: Maximum number of objective evaluations allowed in any application of the two-dimensional minimization [200].†

---

* This is an attempt (not overly successful) at imposing the constraint $f_{max} \leq (p(x) \leq f_{min}$ on the interpolant.

† Actually, the number of evaluations used is generally in the range 10-30.

7. Pros and Cons.

We attempt to list here a number of the good and bad features of the method described here.

Pros:

- Produces visually pleasing pictures.
- Interpolant does not exhibit singular behavior with small changes in data values.

Cons:

- We have given up second derivative continuity.
- The algorithm is rather complicated.
- The algorithm is quite expensive in terms of code required, storage space, and execution time.

The last objection is mitigated somewhat by two observations. First of all, if the Akima formulas are used for the initial guesses, the large amount of extra expense implied by executing the procedure in Figure 6.1 occurs only for those exceptional data sets on which the Akima method fails. (Otherwise, there are no "bad" intervals.) The only extra expense in this case is the computation of the initial badness measures b_i. Even in those exceptional cases there are generally only one or two intervals that require improvement.

Secondly, the procedure can be conveniently separated into two parts as indicated by the dashed line in Figure 6.1. In many applications the calculation of the derivative values (interpolation coefficients) can be done in a set-up phase, with only the final derivative values passed on to the production code that evaluates the interpolant at thousands of points. In this case, the extra expense involved may be insignificant.

8. Future Developments

We conclude by indicating a number of areas in which further research might be worthwhile.

1. Is it possible to derive bounds on the derivative values that guarantee:
   a. That the interpolant stays within the triangle of Figure 5.8?
   b. That the interpolant is monotone when the data is?
   c. That the interpolant is convex (concave) when the data is?
   d. That the interpolant stays within prescribed bounds when the data does?
2. Is (5.3) the "right" measure of badness? Is the tangent circle the "right" standard curve?
3. Can the minimization be done analytically?
4. If not, can a more efficient minimization procedure be designed?
5. Given appropriate assumptions about the function from which the data were generated, can we establish any error bounds for the final interpolant?

Affirmative answers to some of these questions should simplify the procedure outlined here, or even make the iterative improvement unnecessary.

Acknowledgements.

The author wishes to thank the following people for their contributions to this project: Don Gardner and Sandy Taylor for providing the data sets that motivated this study; Bob Dickinson and Ralph Carlson for introducing me to the Hermite representation and for providing the code UNI [4] which served as a starting point for this work; Paul Dubois for the many stimulating discussions that helped me clear some of the hurdles described in Section 5.

REFERENCES

1. H. Akima, A New Method of Interpolation and Smooth Curve Fitting Based on Local Procedures, J. ACM 17, 4 (Oct. 1970), 589-602.

2. R.E. Carlson, Piecewise Polynomial Functions, Lawrence Livermore Laboratory Report UCID-17059 (Jan. 1976).

3. A.K. Cline, Scalar- and Planar-Valued Curve Fitting Using Splines Under Tension, Comm. ACM, 17, 4 (April 1974), 218-223.

4. R.P. Dickinson, Jr., and R.E. Carlson, UNI: Univariate Hermite and Spline Interpolation Code, Lawrence Livermore Laboratory Report UCID-30140 (July 1976).

5. International Mathematical and Statistical Libraries, Inc. IMSL Library 3 Reference Manual, IMSL LIB03-0006, IMSL. (Houston, TX. July 1977).

"Work performed under the auspices of the U.S. Department of Energy by the Lawrence Livermore Laboratory under contract number W-7405-ENG-48."

Appendix. Listing of the Data Values

Identification: LOG RPN 12

I	X(I)	Y(I)
1	7.99	-4.7435660
2	8.09	-2.3515516
3	8.19	-1.9778674
4	8.7	-3.628957E-02
5	9.2	-8.317490E-03
6	10.	-2.865100E-03
7	12.	-6.562200E-04
8	15.	-1.100100E-04
9	20.	-1.200000E-05

Identification: RPN 12

I	X(I)	Y(I)
1	7.99	8.707540E-03
2	8.09	9.522130E-02
3	8.19	0.138364
4	8.7	0.964361
5	9.2	0.991717
6	10.	0.997139
7	12.	0.999344
8	15.	0.999890
9	20.	0.999988

Identification: RPN 14

I	X(I)	Y(I)
1	7.99	0.
2	8.09	2.764290E-05
3	8.19	4.374980E-02
4	8.7	0.169183
5	9.2	0.469428
6	10.	0.943740
7	12.	0.998636
8	15.	0.999919
9	20.	0.999994

Identification: RPN 15

I	X(I)	Y(I)
1	7.99	0.
2	8.09	4.785400E-02
3	8.19	5.558730E-05
4	8.7	0.238983
5	9.2	0.433176
6	10.	0.808165
7	12.	0.997843
8	15.	0.999952
9	20.	0.999998

Identification: RPN 29

I	X(I)	Y(I)
1	7.99	0.
2	8.09	0.
3	8.19	0.
4	8.7	0.
5	9.2	0.
6	10.	0.
7	12.	1.
8	15.	1.
9	20.	1.

Identification: RPN 30

I	X(I)	Y(I)
1	7.99	0.
2	8.09	0.
3	8.19	0.
4	8.7	0.
5	9.2	0.
6	10.	0.
7	12.	0.
8	15.	1.
9	20.	1.

Identification: TEST 50000

I	X(I)	Y(I)
1	0.5	0.
2	1.0	0.
3	1.5	0.
4	1.6	0.50000
5	1.7	1.
6	2.0	1.
7	2.3	1.

Identification: TEST 50001

I	X(I)	Y(I)
1	0.5	0.
2	1.0	0.
3	1.5	0.
4	1.6	0.50001
5	1.7	1.
6	2.0	1.
7	2.3	1.

Identification: T PROT 24

I	X(I)	Y(I)
1	0.8	0.
2	0.9	0.
3	1.0	0.
4	1.25	0.
5	1.5	0.
6	2.0	0.
7	2.5	0.
8	3.	0.
9	4.	0.
10	5.	8.8397E-13
11	6.	1.4016E-11
12	7.	1.3125E-10

Identification: AKIMA 3

I	X(I)	Y(I)
1	0.	10.
2	2.	10.
3	3.	10.
4	5.	10.
5	6.	10.
6	8.	10.
7	9.	10.5
8	11.	15.
9	12.	50.
10	14.	60.
11	15.	85.