Automatic Curve Fitting Using an Adaptive Local Algorithm

WON L. CHUNG
Korea Institute of Science and Technology

An adaptive approximation scheme which is based on interpolating piecewise cubic polynomials with continuous first derivatives is described It automatically compresses a large volume of numerical data for interactive display. The one-sided local procedures lead to efficient computation. The analysis of stability and approximation errors is also presented.

Key Words and Phrases adaptive approximation, automatic curve fitting, data compression, interactive display, L_2-approximation with continuity constraints, modeling and simulation systems, numerical stability, one-sided algorithm, piecewise cubic polynomials
CR Categories· 5.11, 5 13, 8.1, 8.2

1. INTRODUCTION

In this paper we present a new method of adaptive approximation which has been designed and implemented for the graphical display of the output from an interactive modeling and simulation system [4, 7].

The system generates approximations $f'(t_k) + \xi_k$ to a large number of functions $f'(t)$ at a large number of points t. The data are produced in the order of ascending t values. The problem is to reduce this large volume of data as it is generated to cut down the storage requirement and to process and display the data efficiently.

It is natural to choose the piecewise cubic polynomial and L_2-norm as the basis of the approximation because of their good properties for curve fitting [1–3, 5, 9, 11, 14]. Note that the stated problem resembles the problem of least-squares piecewise polynomial approximation with variable knots formulated as a nonlinear optimization problem [11]. However there is an important difference that requires the immediate processing of input data, necessitating an entirely different approach to the solution. The solution is provided by a new method of approximation which computes the piecewise cubic interpolant of class C^1 and the corresponding knots by means of one-sided local procedures using adap-

This work was supported in part by the Atomic Energy Commission under Grant AS AEC AT(11-1)2383, at the Department of Computer Science, University of Illinois at Urbana-Champaign, Urbana, Ill.
Author's address. Software Development Center, Korea Institute of Science and Technology, P.O. Box 131, Dong Dae Moon, Seoul, Korea.

tive criteria for formulating quantitative optimization and determining closeness of fit [4].

Section 2 presents the algorithm in detail and describes the idea of lookahead or extended interval, which is instrumental in making the algorithm numerically stable. The results of stability and error analysis are also described. In Section 3 the implementation aspects are discussed and the algorithm is shown to be computationally efficient and very effective in compressing a large volume of data. Section 3 concludes with a comparison of our method with other automatic curve fitting algorithms.

2. THE NEW METHOD

Let the input data be a sequence of points (t_i, f_i), $i = 0, 1, \ldots, K$, with $f_i = f(t_i)$ and $t_i < t_j$ if $i < j$. The output data are a set of ordered triples (x_j, y_j, m_j), $j = 0, 1, \ldots, N$, where the (x_j, y_j) are a small subset of the (t_i, f_i). The piecewise cubic polynomial interpolating through the knots (x_j, y_j) with slopes m_j at (x_j, y_j) is denoted by $Py(t)$. It is the collection of cubic segment $P_j(t)$ for each interval $I_j = (x_{j-1}, x_j]$, $j = 1, 2, \ldots, N$, given by

$$P_j(t) = m_{j-1}\alpha_j(t) + m_j\beta_j(t) + y_{j-1}\gamma_j(t) + y_j\delta_j(t) \tag{2.1}$$

using the standard basis for Hermite cubics. The interpolation condition is

$$y_j = f(x_j) = P_j(x_j) = P_{j+1}(x_j),$$

$$m_j = P'_j(x_j) = P'_{j+1}(x_j), \qquad j = 1, 2, \ldots, N-1;$$

$$x_0 = t_0, \qquad y_0 = f_0 = P_1(x_0), \qquad x_N = t_K, \qquad y_N = f_K = P_N(x_N).$$

The initial condition for the problem is given by $m_0 = f'(t_0) = P'_1(x_0)$.

Initially, given data (t_i, f_i), $i = 0, 1, \ldots$, initial condition m_0, and the maximum error tolerance $e > 0$, the index M_1 and the slope m_1 of the cubic segment $P_1(t)$ at t_{M_1} $(=x_1)$ for the first interval I_1 are determined so that M_1 is the largest index for which $\| f - P_1(t) \| \leqq e$, $t \in I_1$, with $\| f - P_1 \|$ denoting the local error criterion. The same process is repeated for successive cubic segments; i.e., for an interval I_j, the slope at the left endpoint m_{j-1} is fixed (equal to the computed slope at the right endpoint of I_{j-1}), and the index M_j and the slope m_j at t_{M_j} $(=x_j)$ are determined with the same criteria as above. The slope at the right endpoint is computed locally by minimizing a chosen error norm for the interval. The choice of an error norm is made adaptively to maximize the data reduction and determines the method for the given interval. For relatively horizontal segments, a linear least-squares approximation is used. But the approximation error measured in the ordinate direction is a reasonable criterion of closeness only for relatively horizontal curve segments. This has been confirmed by numerical experiments in which the linear method did not compress the data properly for almost vertical segments. This leads us to use a nonlinear least-squares error criterion for nearly vertical segments, namely, the *pseudodistance*, which is an approximation to the perpendicular distance between the data point and the cubic interpolant. This type of nonlinear least-squares approximation is reviewed in [10].

An inherent difficulty in this type of one-sided algorithm is the problem of numerical instability, as in the algorithms for solving initial value problems [6].

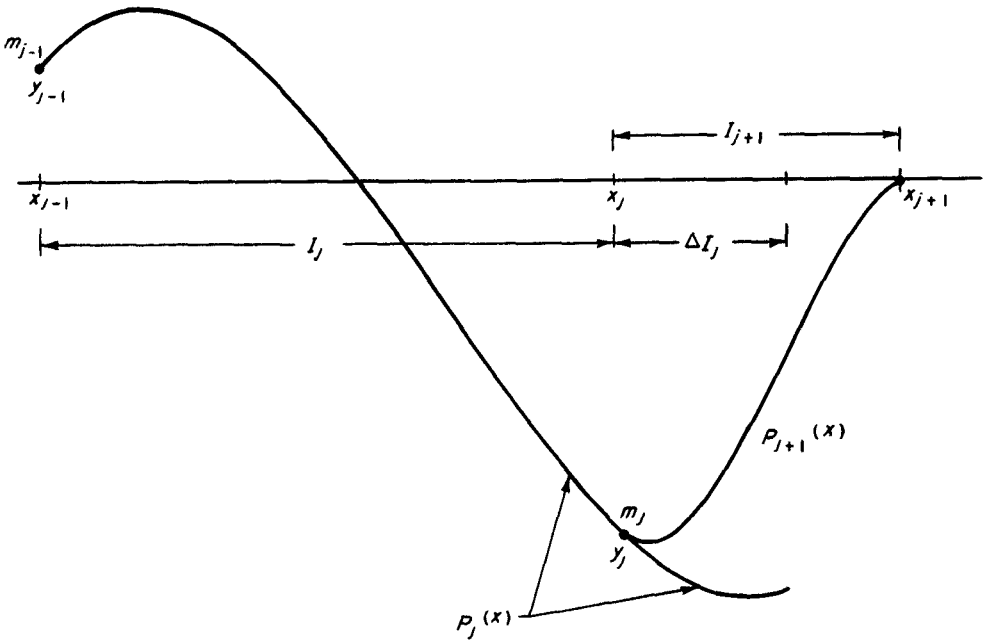


Fig. 1. Piecewise cubic interpolant and extended interval.

Our method is no exception since a computed value of the slope at the left endpoint is used to compute the slope at the right endpoint, and this causes a potential amplification of error. The stability problem is solved by the *extended interval*, denoted by ΔI_j, which is the extension of an interval $I_j = (x_{j-1}, x_j]$ to the right of x_j, for computation of each cubic segment (see Figure 1). The idea is to incorporate the information regarding future data which is provided by extra data points beyond the current interval into the computation of the slope m_j at x_j, thereby adjusting the slope at the right endpoint to allow the next curve segment, for which m_j is prescribed, to fit the data over the interval I_{j+1} better. For the moment, we shall view ΔI_j as a fixed-length extension of I_j. Later we discuss the choice of ΔI_j. This extension of the interval is used primarily for the linear method because experiments showed that it did not affect the performance of the nonlinear method. The nonlinear method instead uses a correction term for the slope to achieve the same effect.

2.1 Linear Method

The linear method computes each slope $m_j = P_j'(x_j)$ as a solution of a least-squares approximation with equality constraints on a finite point set for the interval $I_j + \Delta I_j$. The slope minimizes the weighted discrete L_2-norm of error with respect to m_j:

$$\| E_j \|_2^2 = \sum_{t_k \in I_j + \Delta I_j} \omega_k [f(t_k) - P_j(t_k)]^2 \tag{2.2}$$

with continuity constraints given by

$$P_j(x_{j-1}) = y_{j-1}, \qquad P_j'(x_{j-1}) = m_{j-1}, \qquad P_j(x_j) = y_j$$

where

$$\omega_k = t_{k+1} - t_{k-1} \qquad \text{for} \quad t_k \in I_j + \Delta I_j.$$

This weighting is chosen to compensate for the adverse effect of data point distribution (see Rice [11, vol. 1, p. 42]). Since m_j is the only parameter to be determined, we obtain

$$m_j = \frac{\sum_{t_k \in I_j + \Delta I_j} \omega_k \beta_j(t_k)\{f_k - m_{j-1}\alpha_j(t_k) - y_{j-1}\gamma_j(t_k) - y_j\delta_j(t_k)\}}{\sum_{t_k \in I_j + \Delta I_j} \omega_k\{\beta_j(t_k)\}^2}. \tag{2.3}$$

We search for the knot x_j such that $h_j = x_j - x_{j-1}$ is maximized while satisfying the error criterion $\| f - P_j \|_\infty \le e$ for I_j. Note that as x_j varies, the intervals I_j and ΔI_j vary.

Since the computation of m_j using eq. (2.3) is based on the inexact value of m_{j-1}, which was computed from m_{j-2}, etc., we must make sure that the amplification of errors in m_j is bounded. We loosely define this concept as the *stability* of our method. We also need to consider a realistic assumption on the input data errors ξ_k. The approximation error comes from two sources: the error of the interpolation based on eqs. (2.1) and (2.3) and the error due to the inexact input data. We first analyze the stability and approximation errors for the case where the input data are assumed to represent the function exactly, and next consider the effect of errors in the input data. Then follows the discussion of the choice of ΔI_j, which is based on the mathematical analysis and computational experience.

Let $\bar{m}_j$ denote the slope of $P_j(t)$ at x_j, which is computed by eq. (2.3) with m_{j-1} replaced by $y'_{j-1} = f'(x_{j-1})$, the true value of the first derivative at the previous knot. Define the local (truncation) error $\bar{\sigma}_j = y'_j - \bar{m}_j$, which is the amount by which the y'_j do not satisfy eq. (2.3), and the global error $\sigma_j = y'_j - m_j$, which includes the propagated error. Then, from these definitions and eq. (2.3), we obtain the linear difference equation for the σ_j:

$$\sigma_j = -\lambda_j \sigma_{j-1} + \bar{\sigma}_j$$

where the *stability parameter* λ_j is given by

$$\lambda_j = \frac{\sum_{t_k \in I_j + \Delta I_j} \omega_k \alpha_j(t_k)\beta_j(t_k)}{\sum_{t_k \in I_j + \Delta I_j} \omega_k[\beta_j(t_k)]^2}. \tag{2.4}$$

We may show that (see [4]), assuming $f \in C^4$, and with $H = \max_{i \le j} h_i$,

$$|\bar{\sigma}_j| \le \tfrac{1}{24} \| f^{(4)} \|_\infty H^3 \equiv D.$$

Let σ_0 be the initial error in the derivative. Then if $\lambda = \max_{i \le j} |\lambda_i|$,

$$|\sigma_j| \le \lambda^j |\sigma_0| + \frac{1 - \lambda^j}{1 - \lambda} D \tag{2.5}$$

which describes the error propagation for computed values of the m_j. We may include in σ_0 the error of m_0 when $f'(t_0)$ is not available as an initial condition. If $|\sigma_0|$ and D are bounded, the stability condition $\lambda < 1$ implies the boundedness of $|\sigma_j|$. The graph of λ_j as a function of $\theta_j \equiv |\Delta I_j| / |I_j|$ for uniform data is shown in Figure 2.

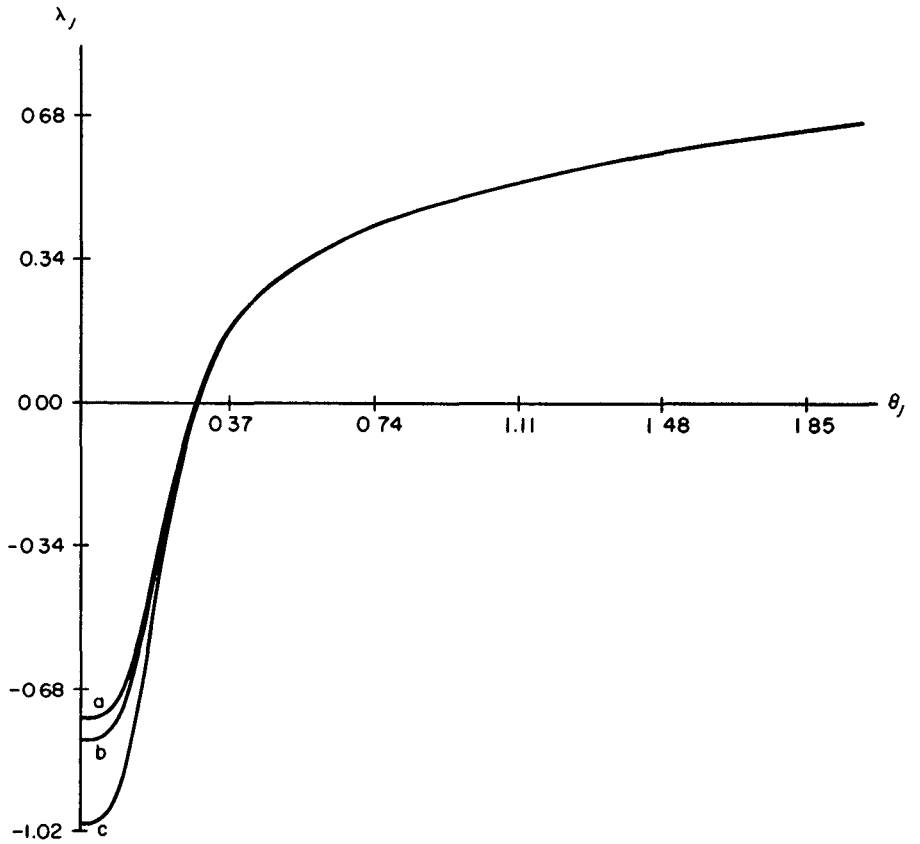


Fig. 2. Stability parameters for equidistant data points. Number of data points: curve a = 100, curve b = 50, and curve c = 2.

It is simple to show that for sufficiently smooth f the pointwise error of the piecewise cubic polynomial interpolation based on the linear method is given by

$$\|f^{(r)} - Py^{(r)}\|_\infty \leq B_r \|f^{(4)}\|_\infty H^{4-r}, \qquad r = 0, 1, \ldots, 3 \tag{2.6}$$

where the B_r are error bound constants [4]. A pragmatic error bound B_0 obtained by numerical evaluation is shown in Figure 3 as a function of θ_j for the limiting case which uses the continuous L_2-norm. The values of B_0 for real data point distributions exhibit the same behavior as this graph in that B_0 becomes minimum at the θ_j, where λ_j is zero.

We now consider the stability and approximation error in the presence of input data errors, which are assumed to be random and bounded. Specifically, we examine the errors in the m_j and Py due to perturbed input data. Let $P_j(t)$ be a cubic polynomial segment computed from exact function values $f_k = f(t_k)$, and let $\hat{P}_j(t)$ denote such a segment computed from data with errors, $f_k + \xi_k$, $|\xi_k| \leq E$. We include in ξ_k and $\hat{m}_j$ the errors caused by numerical integration and roundoff.

Let $\rho_j = m_j - \hat{m}_j$ denote the error in m_j due to input data errors. Then, by an

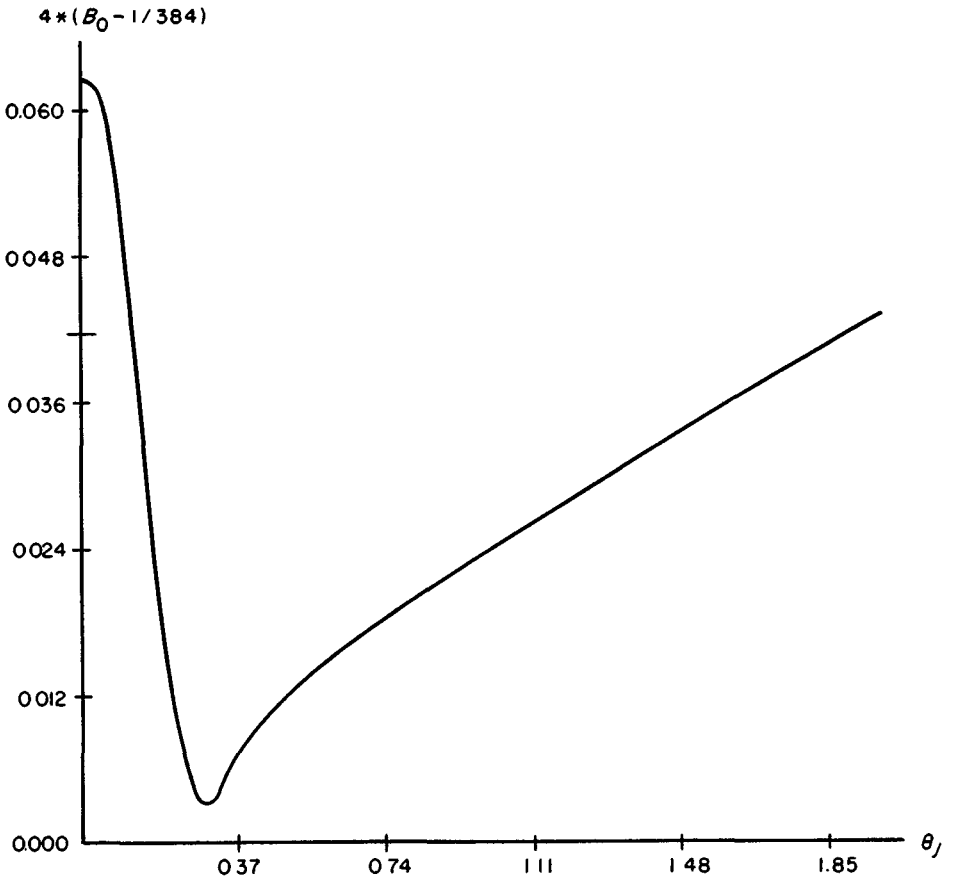


Fig. 3. Error bound constant B_0

analysis similar to σ_j and from eq. (2.3), we obtain

$$|\rho_j| \leq \frac{ER}{1-\lambda} + \lambda^j - \left[|\rho_0| - \frac{ER}{1-\lambda}\right] \tag{2.7}$$

where ρ_0 is defined in the same manner as σ_0 in eq. (2.5) and

$$\begin{aligned} R &= \Big\| \Big[\Big| \sum_{t_k \in I_j + \Delta I_j} \omega_k \beta_j(t_k) \Big| + |\textstyle\sum \omega_k \beta_j(t_k) \gamma_j(t_k)| \\ &\qquad + |\textstyle\sum \omega_k \beta_j(t_k) \delta_j(t_k)| \Big] / \textstyle\sum \omega_k [\beta_j(t_k)]^2 \Big\|_\infty \\ &= \mathcal{O}(h_{\min}^{-1}) \end{aligned}$$

and $h_{\min} = \min_{i \leq j} h_i$. Thus $|\rho_j|$ will be bounded as long as $\lambda < 1$ and R is bounded. Note that input data errors do not affect the values of the λ_j since the λ_j are independent of function values. From eqs. (2.5) and (2.7) we conclude that the condition of stability is satisfied if $\lambda < 1$ and R is bounded.

Considering the difference $\epsilon_j(t)$ between $P_j(t)$ and $\hat{P}_j(t)$, we obtain

$$|\epsilon_j(t)| \equiv |P_j(t) - \hat{P}_j(t)| \leq |\rho_j|[|\alpha_j(t)| + |\beta_j(t)|] + E[|\gamma_j(t)| + |\delta_j(t)|]. \tag{2.8}$$

If $\lambda < 1$ and $\rho_0 = \mathcal{O}(E/h_{\min})$, then from eqs. (2.7) and (2.8) we obtain the approximation error due to the input data errors:

$$\|Py - \hat{P}y\|_\infty = \|\epsilon_j(t)\|_\infty \leq [c(H/h_{\min}) + 1]E \tag{2.9}$$

where c is a positive constant related to R. $\|\epsilon_j(t)\|_\infty$ will be bounded if c and $H/h_{\min}$ are bounded. c is bounded if R is bounded, which is a condition for stability as indicated in the above. Boundedness of $H/h_{\min}$ results from the algorithm.

Combining eqs. (2.6) and (2.9), we finally obtain the following error formula which includes both types of errors:

$$\|f - \hat{P}y\|_\infty \leq B_0\|f^{(4)}\|_\infty H^4 + [c(H/h_{\min}) + 1]E. \tag{2.10}$$

As long as the first term on the right-hand side of eq. (2.10) is a dominant one (say, for "well-posed problems"), random errors in the input data do not adversely affect the interpolation if the stability condition is satisfied. This is the usual case in our class of applications since the maximum error E of input data given as the numerical solution of a differential equation is many orders smaller in magnitude than e, the prescribed error tolerance of the fit. However, if e is set too large, the fit could produce undesirable undulations, particularly where not enough data points are located. An error bound similar to eq. (2.10) can be obtained for the approximation to the first derivative f' at points other than knots. The linear method is also effective in approximating f' simultaneously with f. However, approximation to the higher derivatives is not recommended.

Now we consider the practical problem of achieving stability and discuss the choice of θ_j, the size of the extended interval ΔI_j relative to h_j. It is seen from eqs. (2.5), (2.7), and (2.10) that four parameters affect the stability and approximation errors—E, λ, R, and $H/h_{\min}$. We may modify the definitions of $\bar{\sigma}$ and (D and R accordingly) to include the roundoff error. Then eqs. (2.5) and (2.7) also take care of bounding the accumulated roundoff error. We note from eqs. (2.4) and (2.7) that λ_j and R depend on θ_j and data-point distribution. It is essential to choose a suitable size of θ_j, and for uniform data and limiting approximation $\theta_j \simeq 0.3$ is the optimal choice since $\lambda_j = 0.0$ and B_0 is minimum simultaneously [4]. This is seen from Figures 2 and 3. R behaves just like B_0 but takes its minimum at $\theta_j \simeq 3.7$. Experiments indicates that the θ_j chosen to make $\lambda_j = 0.0$, i.e., $\theta_j = 0.3$, gives best results for uniform data. For nonuniform data, which is normally the case in practice, an optimal θ_j may be chosen such that $\lambda_j = 0.0$ and B_0 is minimum at the same time. But this does not necessarily lead to the best results. For larger values of θ_j for which R becomes minimum, performance of the algorithm deteriorates. We used a single point for ΔI_j successfully in most cases. The average of θ_j used was around 0.2. It seems to be desirable to choose a θ_j which is smaller than 0.3. We have already discussed the ratio $H/h_{\min}$ in connection with eqs. (2.9) and (2.10). In practice, $H/h_{\min}$ was easily bounded and errors in the input data hardly

caused any problem, as illustrated in the next section ($E = 10^{-6} \cdot \|f\|_\infty$ in our experiments).

2.2 Nonlinear Method

A *pseudodistance* d_k at a data point (t_k, f_k) is defined by

$$d_k = \left\{ \frac{[f(t_k) - P_j(t_k)]^2}{[1 + (P_j'(t_k))^2]} \right\}^{1/2}. \tag{2.11}$$

The nonlinear method computes the value of m_j by the following criterion:

$$\frac{\partial}{\partial m_j} \left\{ \sum_{t_k \in I_j} |d_k|^2 + w(m_j - s)^2 \right\} = 0 \tag{2.12}$$

where $0 < w < 1$ and s is the first-order divided-difference approximation to $f'(x_j)$. The second term in the bracket is used to adjust the value of m_j in the same spirit as the extended interval is used in the linear method. This leads to a nonlinear equation for m_j:

$$m_j = s - \sum_{t_k \in I_j} \{f_k - P_j(t_k)\} \times [-\beta_j(t_k)\{1 + (P_j'(t_k))^2\} - \{f_k - P_j(t_k)\}P_j'(t_k)\beta_j'(t_k)]/[w\{1 + (P_j'(t_k))^2\}^2]. \tag{2.13}$$

Simplification of eq. (2.13) leads to a fifth-degree polynomial in m_j which can be solved by an iterative method, e.g., Laguerre's method. We also use a different local error criterion, i.e., $\max_k d_k^2 \leq e$ for I_j.

Observations from the experiments give evidence that the nonlinear method is definitely superior to the linear method for nearly vertical segments. The extra computational effort required by the nonlinear scheme is offset by its capability to produce a more compact function description. Although we cannot give a mathematical analysis of the nonlinear scheme, the fact that the nonlinear scheme is rather sparsely used and the stability of the linear scheme based on eqs. (2.5) and (2.7) result in a stable adaptive approximation.

3. IMPLEMENTATION AND TEST RESULTS

3.1 Implementation

Two major implementation problems are discussed: (1) computational complexity, and (2) criterion for method selection.

Maximum efficiency was obtained by the following:

(a) Efficient coding of eqs. (2.3) and (2.13): For instance, each iteration for eq. (2.3) can be computed with 20 arithmetic operations.

(b) Constant extended intervals: The extended interval of fixed relative size, $\theta_j = 0.3$, can be used in the case of uniform data. It was observed that even one or two extra points could be used without degrading the effectiveness of the algorithm for nonuniform data.

(c) Knot prediction heuristics: A simple-minded but inefficient algorithm for finding the knot and cubic polynomial segment for I_{j-1} is to successively

compute the cubic polynomial approximation to (t_i, f_i), $i = j_1, j_2, \ldots, j_L$, and increase L until the error tolerance is violated. Instead, we use a sequence of prediction heuristics to select a small number of tentative knots until the optimal knot is determined. Three heuristics are used for this purpose:

(i) The initial prediction is computed from the past interval lengths by linear extrapolation, i.e., $x_{j+1}^{(1)}$ is chosen such that

$$h_{j+1}^{(1)} = \frac{1}{2}\left[\frac{\sum_{i=1}^{j-1} h_i}{(j-1)} + h_j\right].$$

The superscripts denote the number of predictions.

(ii) The second prediction is based on the reasoning that $\epsilon = \|f - P_y\| = \mathcal{O}(h^4)$ and the assumption that $\|f^{(4)}\|_\infty$ and an error bound constant do not vary drastically over the interval; i.e., $x_{j+1}^{(2)}$ is chosen such that

$$h_{j+1}^{(2)} = (e/\epsilon^{(1)})^{1/4} h_{j+1}^{(1)}$$

where $\epsilon^{(1)}$ is the approximation error for $x_{j+1}^{(1)}$.

(iii) The third and subsequent predictions are based on the bisection of the data-point indices: for $k = 1, 2, 3, \ldots$, if $(e - \epsilon^{(k)})(e - \epsilon^{(k+1)}) > 0$, then

$$h_{j+1}^{(k+2)} = \frac{[h_{j+1}^{(k)}(\epsilon^{(k+1)} - e) - h_{j+1}^{(k+1)}(\epsilon^{(k)} - e)]}{(\epsilon^{(k+1)} - \epsilon^{(k)})};$$

otherwise, choose the index of the predicted knot such that

$$i^{(k+2)} = \tfrac{1}{2}[i^{(k+1)} + i^{(k)}].$$

If an actual count of arithmetic operations is used as an efficiency measure, more than 70 percent of the computational effort is saved on the average owing to knot prediction heuristics [4].

To illustrate the overall efficiency of the implementation, the linear method is compared with running orthogonalization [12]. They share the same goal but use different techniques. The former does this by knot prediction, and the latter by orthogonalization. Unfortunately, the use of the extended interval and continuity constraints makes it infeasible to make direct use of the simplicity and power of running orthogonalization whose complexity is $\mathcal{O}(M)$, where M is the number of data points contained in an interval. On the one hand, we can derive the recursive computation formula for eq. (2.3), which is similar to running orthogonalization and therefore has the complexity of $\mathcal{O}(M)$. On the other hand, our method based on knot prediction results in better efficiency for small M because the coefficient for the operation count is very small although its complexity is $\mathcal{O}(M^2)$. As a result, numerical stability is obtained at the expense of efficiency.

A local static criterion is used for switching between linear and nonlinear methods. Selection of a method for each interval is based on the comparison of a fixed threshold THRESH with the maximum value of the first-order divided-difference approximation to $f'(t_k)$ adjusted by a scale factor. The latter is defined by

$$\text{SMAX}_j = \left[\frac{\max_{t_k \in I_j} |f_k - f_{k-1}|}{(t_k - t_{k-1})}\right] * \text{SCL}$$

Example 1

$$y' = -10^4[y - \sin x - \tanh 10^4(x - \pi/2)] + \cos x + \operatorname{sech} 10^4(x - \pi/2)$$
$$y(0) = 0, \qquad t_K = 5.0.$$

Example 2

$$y' = 7.48(1 - y) - 22.3(0.325\cos 22.3t - \sin 22.3t)e^{-7.48t}, \qquad t < 2$$
$$y' = -14.96(0.0728 + y) + 44.6(0.325\cos 44.6(t - 2) - \sin 44.6(t - 2))e^{-14.96(t-2)}, \qquad t > 2$$
$$y(0) = 0, \qquad t_K = 5.0.$$

Example 3

$$y' = -5.0 \times 10^5 y + V/6 \times 10^{-5}$$
$$V = 0.0, \qquad t < 0.1$$
$$= +6.0, \qquad t > 0.1$$
$$= -6.0, \qquad t > 0.2$$
$$y(0) = 0, \qquad t_K = 3.0.$$

Example 4

$$y' = 0.89[-161y - 158.8\cos 100t + 103.542\sin 100t - 123(0.16e^{-38t} + 1.16e^{-284t})]$$
$$y(0) = 0, \qquad t_K = 5.0.$$

Example 5

$$y' = -10y + 10^3(1 - 10t)e^{-10t}$$
$$y(0) = 0; \qquad t_K = 2.0.$$

Fig. 4. Differential equations.

Table I. Comparison of Number of Knots and Compression Ratios

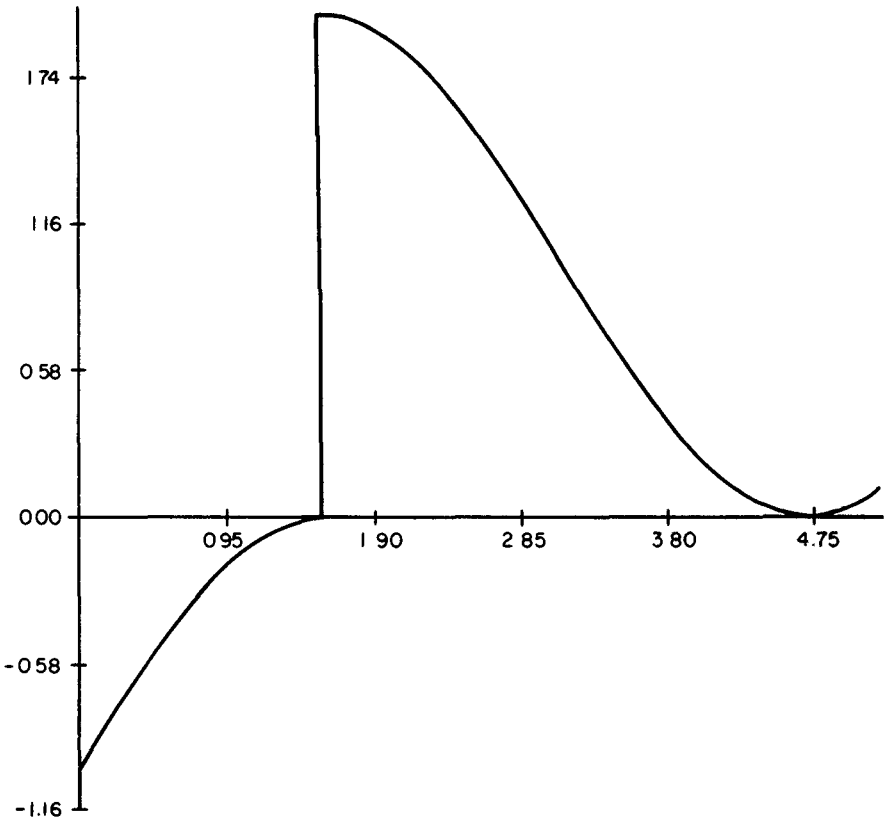
Example number	Number of data points	Linear scheme	Nonlinear scheme	Adaptive scheme
1	209	32 (15.3)	24 (11.5)	10 (4.8)
2	367	32 (8.7)	37 (10)	22 (6.0)
3	398	28 (7)	6 (1.5)	6 (1.5)
4	358	60 (16.8)	26 (7.3)	33 (9.2)
5	120	12 (10)	22 (18.3)	14 (11.6)

Note. Numbers within parentheses are the compression ratios, defined as N/K, expressed in percentages.

where SCL = XSCL/YSCL and XSCL and YSCL are scaling factors for x and y coordinates, respectively. YSCL is updated each time a new maximum or minimum of f_k is detected, and is determined globally. We choose the linear method for I_j if $\text{SMAX}_j < \text{THRESH}$. The nonlinear method is used otherwise. THRESH = 100 was effectively used in most cases. It is effective in the sense that the original and approximated curves are visually indistinguishable when they are plotted within an 8- by 8-inch window.

3.2 Examples and Test Results

We have experimented extensively with many examples which range from highly smooth functions to square-wave functions. The input data were generated by the ODE integrator DIFSUB [6] with EPS = 10^{-6} and for $t \in [0, t_K]$. Differential equations used to generate the data for Examples 1 through 5 are listed in Figure 4. Test results for these examples are summarized in Table I. The percentages



	$X(J)$	$Y(J)$	$M(J)$
(1)	0.0	0.0	−0.10000E 05
(2)	0.15855D−02	−0.99841E 00	0.10346E 01
(3)	0.16868D 00	−0.83212E 00	0.97231E 00
(4)	0.72215D 00	−0.33900E 00	0.74039E 00
(5)	0.11813D 01	−0.74905E−01	0.37851E 00
(6)	0.15704D 01	0.17480E−02	0.23917E−01
(7)	0.15762D 01	0.20000E 01	−0 60439E−02
(8)	0 25861D 01	0.15273E 01	−0.85951E 00
(9)	0.41263D 01	0.16688E 00	−0.55114E 00
(10)	0.51908D 01	0.11225E 00	0 48089E 00

Fig. 5. Example 1.

are the compression ratio which is defined as N/K. Figure 5 shows the output from our algorithm and the reproduced curve for Example 1 to illustrate the merit of our method.

3.3 Comparison with Other Methods

It is instructive to compare our method with other automatic curve fitting algorithms. This will not be an exhaustive comparison, but we shall consider a number of recently published methods—Akima [2], Dierckx [5], Ichida et al. [8],

and Rice [13]—which are based on piecewise polynomials. Comparison is based on a set of criteria: (a) approximation, i.e., smoothing, interpolation, adaptive approximation; (b) basis and smoothness of approximating functions; (c) computational complexity, i.e., time and storage, local versus global procedures; (d) stability; (e) applications. However, the diversity of these criteria makes quantitative comparison impossible. Thus we shall emphasize relative merits and differences in approach and intended applications.

All of the studies cited in this paper except for [5] use local procedures: The methods in [2] and [8] and our method are based on piecewise cubic polynomials of C^1; the method of [13] is based on piecewise polynomials of user-specified degrees; reference [5] uses cubic splines. When the parameters of an approximating function are determined by minimization of error norms, choice of the form, i.e., basis, of an approximating function affects the computational complexity and stability of the algorithm: [5] uses B-splines, [8] uses a triangular basis of the form $(x - x_j)^i$, $i = 0, 1, 2, 3$, [13] uses divided difference representation, and we use a standard Hermite basis.

All but the method of [2] involve some form of data compression. References [5] and [8] deal with smoothing of experimental data explicitly, whereas our method and those of [2] and [13] are based on interpolation. In general, interpolation leads to a smaller number of parameters and therefore is cheaper than approximation, but it can only be applied to highly accurate data. (For example, our method needs to compute one parameter for each cubic segment, but the method of [8] requires four.)

The adaptive algorithm of [13] is designed for the functions $f(x)$ whose values and derivatives are available at any x and thus not directly applicable to discrete data. Our method and that of [8] are also adaptive and one-sided (or one-pass), like that of [13], but are intended for discrete data. All three methods rely on one knot prediction scheme or another to improve efficiency, with [8] and [13] using interval bisection. One-sided algorithms which approximate derivative (and function) values need to resolve the stability problem. Our method and that of [8] improve stability by employing the past and future data for each subinterval, respectively. Both works discuss the analysis of stability.

ACKNOWLEDGMENT

The author wishes to express sincere thanks to Professors C. W. Gear and D. S. Watanabe for valuable guidance during the preparation of this work. He also thanks the referees for constructive criticism which lead to an improved paper.

REFERENCES

1. AHLBERG, J H , NILSON, E.N., AND WALSH, J.L. *The Theory of Splines and Their Applications.* Academic Press, New York, 1967.
2. AKIMA, D.V. A new method of interpolation and smooth curve fitting based on local procedures. *J ACM 17*, 4 (Oct. 1970), 589–602.
3. BIRKHOFF, G., AND DE BOOR, C. Piecewise polynomial interpolation and approximation. In *Approximation of Functions*, H. Garabedian, Ed., American Elsevier, New York, 1965, pp. 164–190.
4. CHUNG, W.L. Automatic curve fitting for interactive display. Ph.D. Dissertation, UIUCDCS-R-75-713, Dep Comput. Sci., Univ. Illinois at Urbana-Champaign, Urbana, Ill. May 1975.

5. DIERCKX, P. An algorithm for smoothing, differentiation and integration of experimental data using spline functions. *J. Comput. App. Math 1*, 3 (1975), 165–184.
6. GEAR, C W. *Numerical Initial Value Problems in Ordinary Differential Equations.* Prentice-Hall, Englewod Ciffs, N J., 1971.
7. GEAR, C.W., ET AL. The simulation and modeling system—A snapshot view. DCS File No. 824, Univ. Illinois at Urbana-Champaign, Urbana, Ill., 1970.
8. ICHIDA, K., YOSHIMOTO, F., AND KIYONO, T. Curve fitting by a one-pass method with a piecewise cubic polynomial *ACM Trans. Math. Software 3*, 2 (June 1977), 164–174.
9. POWELL, M.J.D. Curve fitting by splines in one variable In *Numerical Approximation to Functions and Data*, J G. Hayes, Ed. The Athlone Press, University of London, London, England, 1970, pp. 65–83.
10. POWELL, M J.D., AND MACDONALD, J.R A rapidly convergent iterative method for the solution of the generalized nonlinear least squares problem. *Comput. J. 15*, 2 (May 1972), 148–155.
11. RICE, J.R. *The Approximation of Functions*, 2 vols. Addison-Wesley, Reading, Mass., 1964 and 1969.
12. RICE, J R. Running orthogonalization. *J. Approx. Theory 4* (1971), 332–338.
13. RICE, J.R Algorithm 525. ADAPT, adaptive smooth curve fitting. *ACM Trans. Math. Software 4*, 1 (March 1978), 82–94.
14. SCHULTZ, M. *Spline Analysis* Prentice-Hall, Englewood Cliffs, N J , 1973.

Received February 1976, revised July 1979