

# Algorithm 716

## TSPACK: Tension Spline Curve-Fitting Package

R. J. RENKA  
University of North Texas

---

The primary purpose of TSPACK is to construct a smooth function which interpolates a discrete set of data points. The function may be required to have either one or two continuous derivatives. If the accuracy of the data does not warrant interpolation, a smoothing function (which does not pass through the data points) may be constructed instead. The fitting method is designed to avoid extraneous inflection points (associated with rapidly varying data values) and preserve local shape properties of the data (monotonicity and convexity), or to satisfy the more general constraints of bounds on function values or first derivatives. The package also provides a parametric representation for constructing general planar curves and space curves.

Categories and Subject Descriptors: G.1.1 [**Numerical Analysis**]: Interpolation; G.1.2 [**Numerical Analysis**]: Approximation; G.4 [**Mathematics of Computing**]: Mathematical Software

General Terms: Algorithms

Additional Key Words and Phrases: Convexity preserving, cubic spline, exponential spline, interpolation, monotonicity preserving, parametric curve, piecewise polynomial, shape preserving, smoothing, spline under tension, tension factor

---

### 1. INTRODUCTION

Among the most widely distributed software packages for curve fitting are deBoor's cubic spline codes [2], FITPACK [1] (available from its author), and PCHIP [4] which was selected for inclusion in *Numerical Methods and Software* [5] by virtue of being an excellent general purpose method. All three packages include methods designed to produce shape-preserving interpolants: deBoor's method, through appropriate selection of additional knots between the data abscissae, preserves convexity of the data [2, pgs. 299–316]; FITPACK uses exponential tension splines with uniform tension [9]; and PCHIP employs local knot-derivative estimates designed to preserve monotonicity of the data [3]. TSPACK combines the capabilities of the latter two methods by

---

This material is based upon work supported by the National Science Foundation under Award No. CCR-8921050.

Author's address: University of North Texas, Department of Computer Sciences, Denton, TX 76203-3886.

Permission to copy without fee all or part of this material is granted provided that the copies are not made or distributed for direct commercial advantage, the ACM copyright notice and the title of the publication and its date appear, and notice is given that copying is by permission of the Association for Computing Machinery. To copy otherwise, or to republish, requires a fee and/or specific permission.

© 1993 ACM 0098-3500/93/0300-0081 \$01.50

employing both monotonicity-constrained local derivative estimates and exponential tension splines.

The fitting function  $h(x)$  is defined locally, on each interval associated with a pair of adjacent abscissae (knots), by its values and first derivatives at the endpoints of the interval, along with a nonnegative tension factor  $\sigma$  associated with the interval ( $h$  is a Hermite interpolatory tension spline). With  $\sigma = 0$ ,  $h$  is the cubic function defined by the endpoint values and derivatives, and, as  $\sigma$  increases,  $h$  approaches the linear interpolant of the endpoint values. Since the piecewise linear interpolant of a set of data values preserves such data properties as positivity, monotonicity, and convexity,  $h$  can be forced to preserve these properties by choosing  $\sigma$  sufficiently large. Also, since  $\sigma$  varies with intervals, no more tension than necessary is used in each interval, resulting in a better fit and greater efficiency than is achieved with a single constant tension factor. The user is relieved of the burden of selecting tension factors by procedures for automatically computing the minimum tension required to preserve shape properties or to satisfy more general bounds on function values and first derivatives. TSPACK also includes extensions to smoothing splines and parametric curves.

The computational methods are briefly discussed in Section 2, the software package is described in Section 3, and some sample plots are presented in Section 4.

## 2. METHOD

The theoretical background and computational details of the methods are described in [8]. For completeness, however, we present here a brief discussion of the fundamental approach. For each knot interval  $[x_1, x_2]$  with endpoint data values  $y_1$  and  $y_2$ , and first derivatives  $y'_1$  and  $y'_2$ , along with a nonnegative tension factor  $\sigma$ , the restriction of the Hermite interpolatory tension spline  $h(x)$  to the interval is the unique linear combination of basis functions  $\{1, x, e^{\bar{\sigma}x}, e^{-\bar{\sigma}x}\}$  which takes on the four specified values, where  $\bar{\sigma} = \sigma/\Delta x$  for  $\Delta x = x_2 - x_1$ . The normalization of  $\sigma$  makes the interpolant independent of a scaling of the abscissae. Continuity of  $h$  and  $h'$  at the knots follows from the interpolatory conditions. Note that, in each interval,  $h$  satisfies

$$h^4(x) - \bar{\sigma}^2 h''(x) = 0.$$

Thus, if  $\sigma = 0$ ,  $h$  is the Hermite cubic interpolant defined by the data ( $h$  reduces to a linear combination of  $\{1, x, x^2, x^3\}$ ), and, as  $\sigma \rightarrow \infty$ ,  $h''(x) \rightarrow 0$  for all  $x$  in the open interval  $(x_1, x_2)$ ; i.e.,  $h$  uniformly approaches the linear interpolant of the endpoint data (and hence the term *tension factor*). In order to maintain numerical stability, the software actually employs the alternative set of basis functions  $\{1, x, \coshm(\sigma b), \sinhm(\sigma b)\}$ , where  $b = (x_2 - x)/\Delta x$ ,  $\coshm(z) = \cosh(z) - 1$ , and  $\sinhm(z) = \sinh(z) - z$ .

The formulation of  $h$  in terms of knot derivatives has an advantage over alternatives, such as a second derivative formulation, in terms of software flexibility. The knot derivatives may be selected in three ways.

- (1) They may be user-specified in an application for which they are known.
- (2) They may be computed by local methods, such as monotonicity-constrained quadratic fits to three data points, resulting in a  $C^1$  interpolant which often requires no tension to produce a visually pleasing curve.
- (3) They may be computed by solving the symmetric positive definite tridiagonal linear system associated with second derivative continuity at the interior knots (along with a pair of end conditions), resulting in a  $C^2$  interpolant.

The software also provides a  $C^2$  smoothing option in which both knot derivatives and knot function values are computed. This is more appropriate when significant measurement errors are present in the data values. The method is a modification of that of Reinsch [6, 7], in which a quadratic functional representing smoothness (linearized curvature of  $h$ ) is minimized subject to a user-specified bound on the weighted sum of squares of deviations from the data values. Test results demonstrate a significant advantage in the use of tension for smoothing as well as interpolation.

The means of automatically selecting tension will be described for the case of a bounds-constrained parametric planar curve (which is not discussed in [8]). The bounds might, for example, be chosen to avoid intersections between smooth contour curves. The other data fitting problems addressed by the software are treated by similar procedures. Given an ordered sequence of control points  $C_1 = (x_1, y_1), C_2 = (x_2, y_2), \dots, C_N = (x_N, y_N)$ , a smooth curve which passes through the points is obtained by constructing two Hermite interpolatory tension splines  $x(t) = h_1(t)$  and  $y(t) = h_2(t)$  such that  $C(t_i) = (x(t_i), y(t_i)) = C_i$ , where  $\{t_i\}_{i=1}^N$  is any strictly increasing sequence of parameter values. (The software provides for parameterization by cumulative polygon arc length.) For each knot interval  $[t_1, t_2]$  with associated control points  $C_1$  and  $C_2$ , and derivative vectors  $C'_1$  and  $C'_2$ , the problem is to find the minimum tension factor  $\sigma$  such that the signed orthogonal distance  $d(t)$  from the line defined by  $C_1$  and  $C_2$  to the curve segment  $C(t)$  is bounded above by a user-specified positive number  $\beta$  for all  $t \in [t_1, t_2]$  (and/or bounded below by a user-specified negative number). The signed distance is given by the vector cross product ( $z$  component):

$$d(t) = \Delta C \times (C(t) - C_1),$$

where  $\Delta C = (C_2 - C_1)/|C_2 - C_1|$  is a unit vector. Note that  $d(t_1) = d(t_2) = 0$ . Depending on the signs of  $\Delta C \times C'_1$  and  $\Delta C \times C'_2$ , there are one or two critical points  $t^*$  at which  $d'(t^*) = 0$ . Expressions for  $t^*$  are obtained from the roots of a quadratic equation: if  $\sigma = 0$ , then  $C(t)$  is cubic and  $d'(t)$  is quadratic in  $t$ ; for  $\sigma > 0$ ,  $e^{\sigma b} d'(t)$  is quadratic in  $e^{\sigma b}$ , where  $b = (t_2 - t)/(t_2 - t_1)$ . The maximum distance is  $d_{\max}(\sigma) = d(t^*)$ , where  $t^*$  is the appropriate root, or  $d_{\max}(\sigma) = 0$  if there is only one critical point  $t^* \in [t_1, t_2]$  and  $d(t^*) < 0$ . The optimal tension factor is taken to be  $\sigma = 0$  if  $d_{\max}(0) \leq \beta$ , or the zero of  $F(\sigma) = \beta - d_{\max}(\sigma)$  otherwise. Note that, since  $\beta > 0$ ,  $F(\sigma) > 0$  for sufficiently large  $\sigma$ , and  $F(0) < 0$  when zero tension is not sufficient to

satisfy the constraint. Thus, the nonlinear equation solver employs a bracketing interval which guarantees convergence. The procedure for evaluating  $d_{\max}(\sigma)$  is carefully coded to avoid cancellation error and overflow.

### 3. SOFTWARE

#### 3.1 Usage

TSPACK must be linked to a driver program which reserves storage, reads a data set, and calls the appropriate subprograms selected from those described below in Section 3.3. Header comments in the software modules provides details regarding the specification of input parameters and the work space requirements. It is recommended that curves be plotted in order to assess their appropriateness for the application. This requires a user-supplied graphics package.

#### 3.2 Code

The code is written in 1977 ANSI Standard Fortran. All variable and array names conform to the default typing convention: I-N for type INTEGER and A-H or O-Z for type REAL. (There are no DOUBLE PRECISION variables.) There are 32 modules, and they are ordered alphabetically in the package. Each consists of the following sections:

- the module name and parameter list with spaces separating the parameters into one to three subsets: input parameters, I/O parameters, and output parameters (in that order);
- type statements in which all parameters are typed and arrays are dimensioned;
- a heading with the name of the package, identification of the author, and date of the most recent modification to the module;
- a description of the module's purpose and other relevant information for the user;
- input parameter descriptions and output parameter descriptions in the same order as the parameter list;
- a list of other modules required (called either directly or indirectly);
- a list of intrinsic functions called, if any; and
- the code, including comments.

Note that it is assumed that floating point underflow results in assignment of the value zero. If not the default, this may be specified as either a compiler option or an operating system option. Also, overflow is avoided by restricting arguments to the exponential function EXP to have value at most SBIG = 85. SBIG, which appears in DATA statements in the evaluation functions, HVAL, HPVAL, HPPVAL, and TSINTL, must be decreased if it is necessary to accommodate a floating point number system with maximum exponent smaller than  $10^{37}$ . No other system dependencies are present in the code.

The modules that solve nonlinear equations, SIGS, SIGBP, SIG0, SIG1, SIG2, and SMCRV, include diagnostic print capability which allows the

iteration to be traced. This can be enabled by altering logical unit number LUN in a DATA statement in the relevant module.

### 3.3 Module Descriptions

The software modules are divided into three categories, referred to as level 1, level 2, and level 3, corresponding to the hierarchy of calling sequences: level 1 modules call level 2 modules which call level 3 modules. For most applications, the driver (the level 0 module) need only call two level 1 modules—one from each of groups (1) and (2). However, additional control over various options can be obtained by directly calling level 2 modules. Also, additional fitting methods, such as parametric smoothing, can be obtained by calling level 2 modules. Note that, in the case of a  $C^2$  fit with automatically selected tension, the use of level 2 modules requires that an iteration be placed around the computation of knot derivatives and tension factors.

3.3.1 *Level 1 Modules.* These are divided into two groups.

- (1) The following modules return knots (in the parametric case), knot derivatives, tension factors, and, in the case of smoothing, knot function values, which define the fitting function (or functions in the parametric case). The naming convention should be evident from the descriptions.

TSPSI Subroutine that constructs a shape-preserving or unconstrained interpolatory function. Refer to TSVAL1.

TSPSS Subroutine that constructs a shape-preserving or unconstrained smoothing spline. Refer to TSVAL1.

TSPSP Subroutine that constructs a shape-preserving or unconstrained parametric planar curve or space curve. Refer to TSVAL2 and TSVAL3.

TSPBI Subroutine that constructs a bounds-constrained interpolatory function. The constraints are defined by a user-supplied array containing upper and lower bounds on function values and first derivatives, along with required signs for the second derivative, for each interval. Refer to TSVAL1.

TSPBP Subroutine that constructs a bounds-constrained parametric planar curve. The constraints are defined by user-supplied arrays containing upper and lower bounds on the signed perpendicular distance between the smooth curve segment and the polygonal line segment associated with each knot interval. Refer to TSVAL2.

- (2) The following modules return values, derivatives, or integrals of the fitting function(s).

TSVAL1 Subroutine that evaluates a Hermite interpolatory tension spline or its first or second derivative at a user-specified set of points. Note that a smoothing curve constructed by TSPSS is the interpolant of the computed knot function values. The evaluation points need not lie in the interval defined by the knots, but care must be exercised in assessing the accuracy of extrapolation.

- TSVAL2 Subroutine that returns values or derivatives of a pair of Hermite interpolatory tension splines which form the components of a parametric planar curve. The output values may be used to construct unit tangent vectors, curvature vectors, etc.
- TSVAL3 Subroutine that returns values or derivatives of three Hermite interpolatory tension splines which form the components of a parametric space curve.
- TSINTL Function which returns the integral over a specified domain of a Hermite interpolatory tension spline. This provides an effective means of quadrature for a function defined only by a discrete set of values.

3.3.2 *Level 2 Modules.* These are divided into four groups.

- (1) The following modules are called by TSPSP and TSPBP to obtain a sequence of knots (parameter values) associated with a parametric curve. For some data sets, it might be advantageous to replace these with routines that implement an alternative method of parameterization.

ARCL2D Subroutine that computes the sequence of cumulative arc lengths associated with a sequence of points in the plane.

ARCL3D Subroutine that computes the sequence of cumulative arc lengths associated with a sequence of points in 3-space.

- (2) The following modules are called by the level 1, group (1) modules to obtain knot derivatives (and values in the case of SMCRV).

YPC1 Subroutine that employs a monotonicity-constrained quadratic interpolation method to compute locally defined derivative estimates, resulting in a  $C^1$  fit.

YPC1P Subroutine similar to YPC1 for the case of periodic end conditions. In the case of a parametric curve fit, periodic end conditions are necessary to obtain a closed curve.

YPC2 Subroutine that determines a set of knot-derivative estimates which result in a tension spline with two continuous derivatives and satisfying a user-specified choice among four types of end conditions.

YPC2P Subroutine similar to YPC2 for the case of periodic end conditions.

SMRCV Subroutine that given a sequence of abscissae with associated data values and tension factors, returns a set of function values and first derivative values defining a twice-continuously differentiable tension spline which smoothes the data and satisfies either natural or periodic end conditions.

- (3) The following modules are called by the level 1, group (1) modules to obtain tension factors associated with knot intervals.

SIGS Subroutine that given a sequence of abscissae, function values, and first derivative values, determines the set of minimum

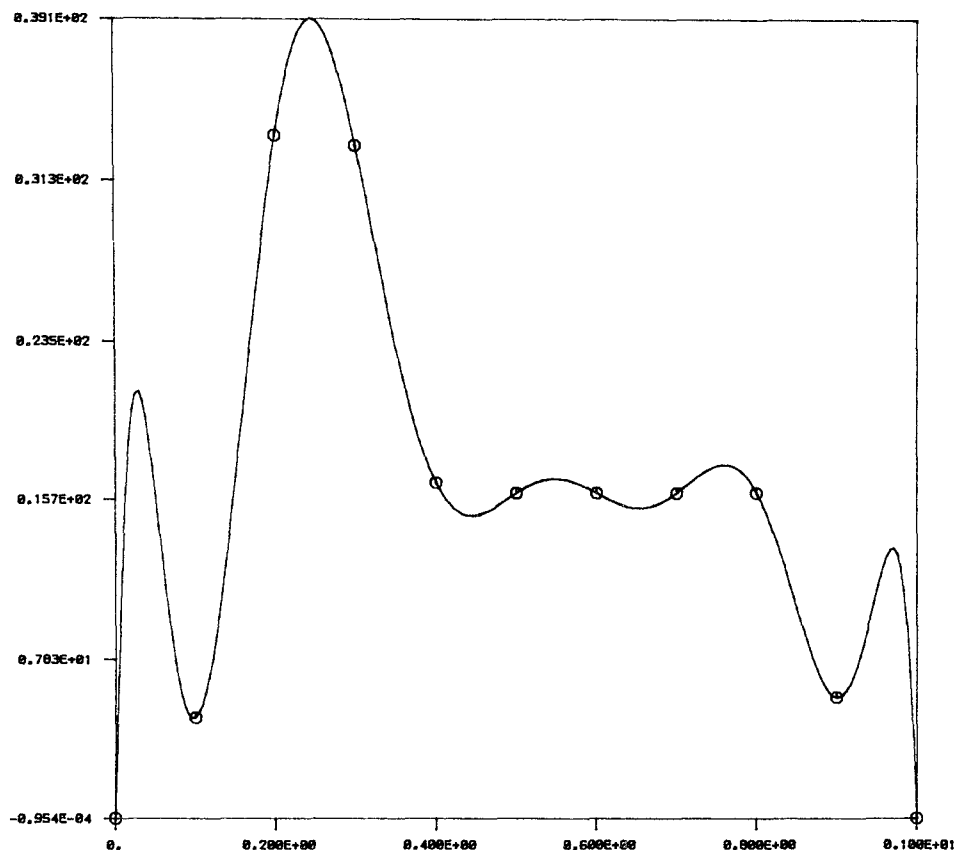


Fig. 1. Polynomial interpolation.

tension factors for which the Hermite interpolatory tension spline preserves local shape properties (monotonicity and convexity) of the data. SIGS is called by TSPSI, TSPSS, and TSPSP.

**SIGBI** Subroutine that given a sequence of abscissae, function values, and first derivative values, determines the set of minimum tension factors for which the Hermite interpolatory tension spline satisfies specified bounds constraints. SIGBI is called by TSPBI.

**SIGBP** Subroutine that given a sequence of points in the plane with associated derivative vectors, determines the set of minimum tension factors for which the parametric planar tension spline curve defined by the data satisfies specified bounds on the signed orthogonal distance between the parametric curve and the polygonal curve defined by the points. SIGBP is called by TSPBP.

- (4) The following functions are called by the level 1, group (2) modules to obtain values and derivatives. These provide a more convenient alternative to the level 1 routines when a single value is needed.

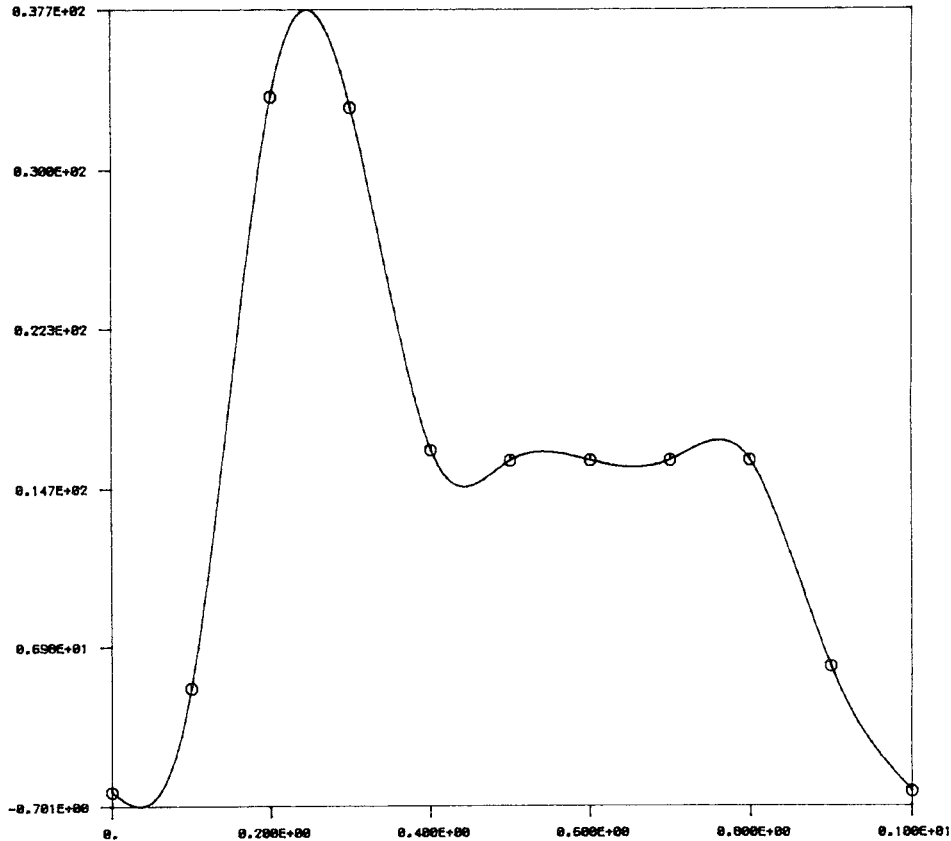


Fig. 2.  $C^2$  natural cubic spline interpolation.

HVAL Function that evaluates a Hermite interpolatory tension spline at a specified point.

HPVAL Function that evaluates the first derivative of a Hermite interpolatory tension spline at a specified point.

HPPVAL Function that evaluates the second derivative of a Hermite interpolatory tension spline at a specified point.

3.3.3 *Level 3 Modules.* These are divided into three groups.

- (1) The following functions are called by SIGBI to compute tension factors, and are convenient for obtaining an optimal tension factor associated with a single interval.

SIG0 Function that given a pair of abscissae, along with associated endpoint values and derivatives, determines the smallest tension factor for which the corresponding Hermite interpolatory tension spline satisfies a specified bound on function values in the interval.

SIG1 Function that given a pair of abscissae, along with associated

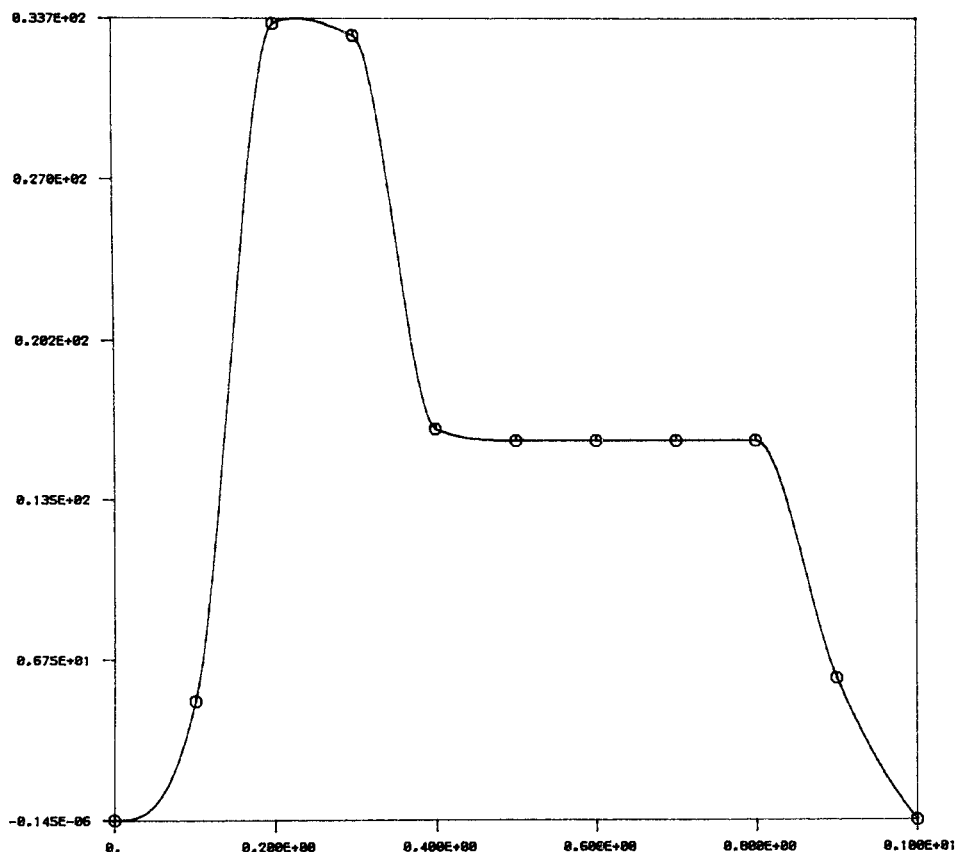


Fig. 3.  $C^1$  piecewise cubic interpolation.

endpoint data, determines the smallest tension factor for which the corresponding Hermite interpolatory tension spline satisfies a specified bound on first derivative values in the interval.

**SIG2** Function that given a pair of abscissae, along with associated endpoint data, determines the smallest tension factor for which the corresponding Hermite interpolatory tension spline preserves convexity (or concavity) of the data.

(2) The following modules are of general utility.

**INTRVL** Function that given an increasing sequence of abscissae, returns the index of an interval containing a specified point. INTRVL is called by the evaluation functions TSINTL, HVAL, HPVAL, and HPPVAL.

**SNHCSH** Subroutine called by several modules to compute accurate approximations to the modified hyperbolic functions which form a basis for exponential tension splines.

**STORE** Function used by SIGBP, SIGS, SIG0, SIG1, and SIG2 in

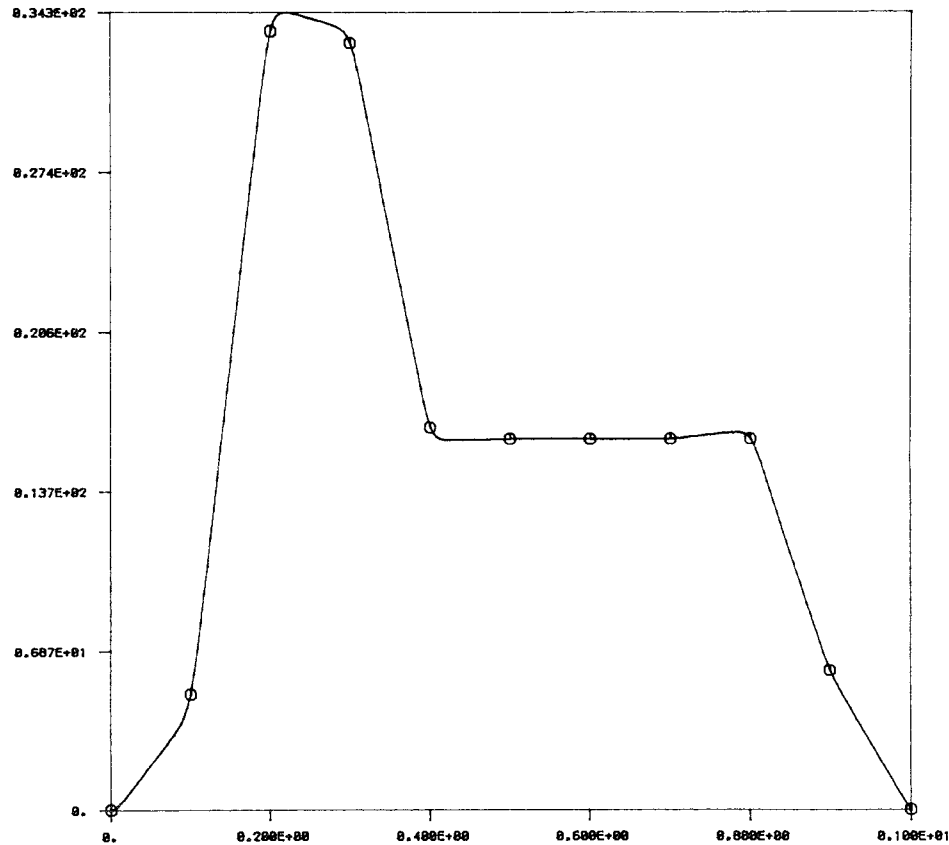


Fig. 4.  $C^2$  exponential tension spline interpolation with uniform tension.

computing the machine precision. STORE forces a value to be stored in main memory so that the precision of floating point numbers in memory locations rather than registers is computed.

(3) The remaining modules are listed below.

- B2TRI** Subroutine called by SMCRV to solve the symmetric positive definite block tridiagonal linear system associated with the gradient of the quadratic functional whose minimum corresponds to a smoothing curve with nonperiodic end conditions.
- B2TRIP** Subroutine similar to B2TRI for periodic end conditions.
- ENDSLP** Function that returns the derivative at a specified point X1 of a tension spline  $h(x)$  which interpolates three specified data points and has third derivative equal to zero at X1. ENDSLP is called by YPC2 when this choice of end conditions is selected by an input parameter.

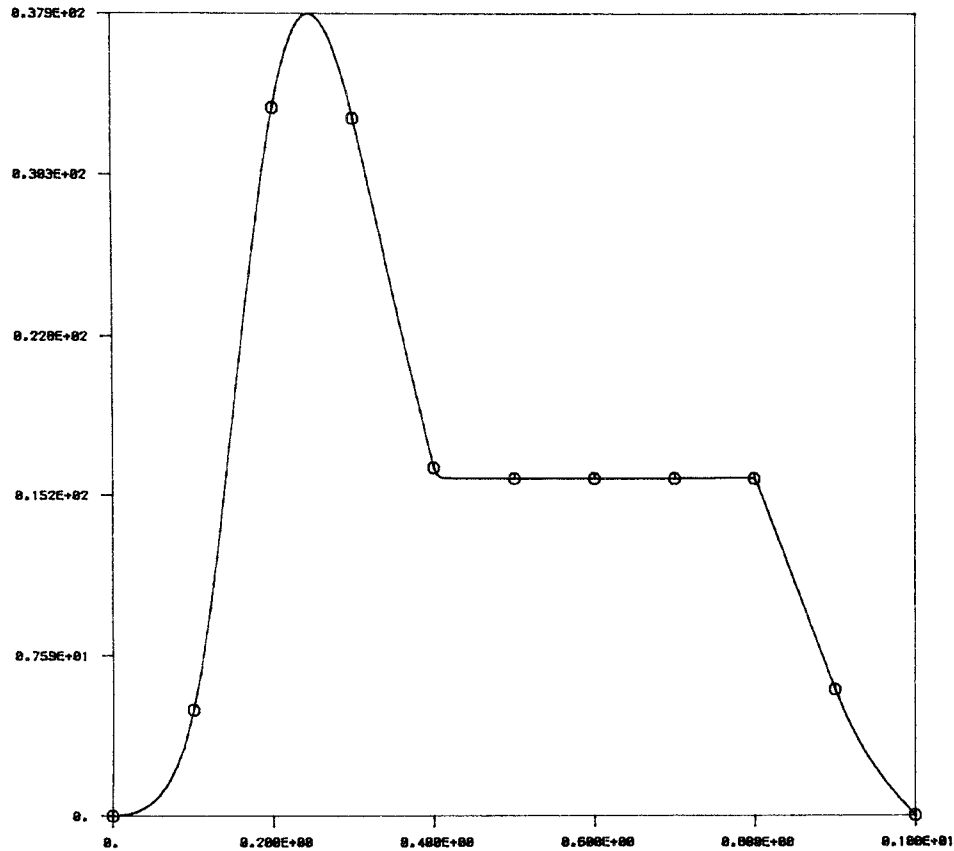
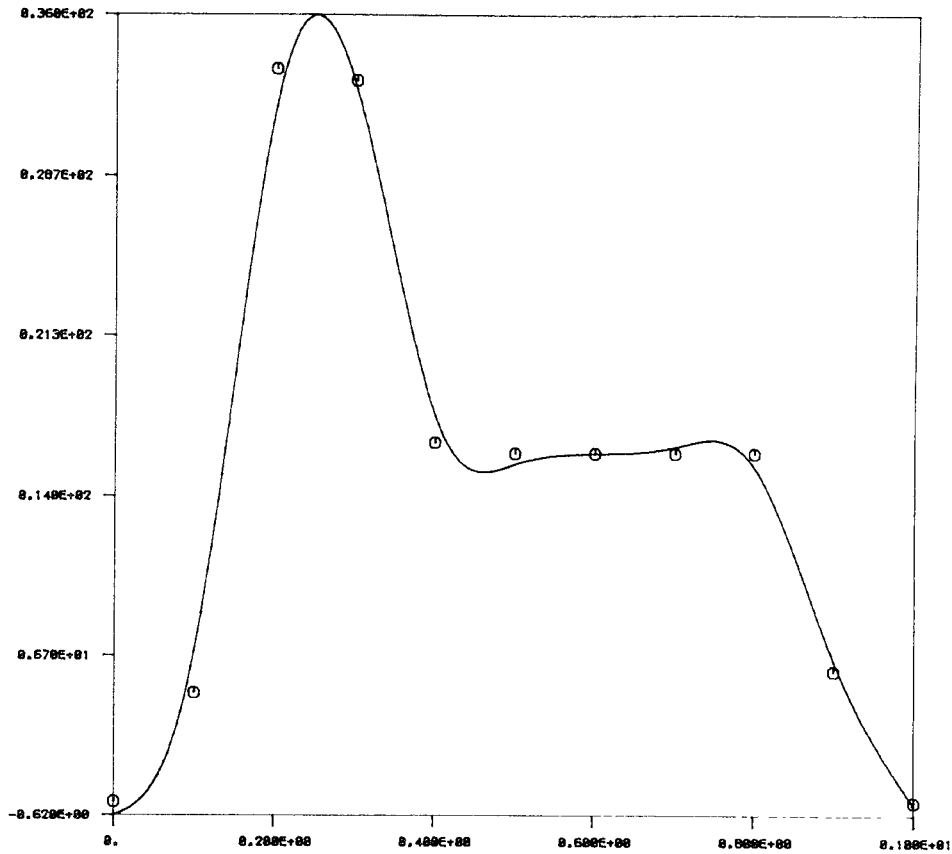


Fig. 5.  $C^2$  exponential tension spline interpolation with variable tension.

YPCOE Subroutine called by SMCRV, YPC2, and YPC2P to compute coefficients defining the linear system.

#### 4. SAMPLE PLOTS

The following figures depict functional fits to an 11-point data set (Data Set 6 in [8]), typical of those for which a shape-preserving method is required—one for which the underlying function has steep gradients. Figure 1 is a plot of the unique 10th degree polynomial interpolant and exhibits the oscillations that often characterize high-degree polynomial interpolation. Cubic splines are designed to limit these oscillations by reducing the extent to which each function value depends on relatively distant data points. For this data set, however, there remain extraneous inflection points in the cubic spline, as depicted in Figure 2. This fit employs natural end conditions (vanishing second derivatives at the endpoints), but alternative end conditions result in no noticeable difference in the interior where the undesirable oscillations

Fig. 6.  $C^2$  natural cubic spline smoothing.

occur. Figure 3 depicts a  $C^1$  piecewise cubic interpolant in which knot derivatives were computed by YPC1 (local monotonicity-constrained quadratic fits). This is very similar (but not identical) to the fit that would be produced by PCHIP. For this data set, the  $C^1$  fit with minimum tension required to preserve shape properties (SIGS) results in only two nonzero tension factors—small values in intervals 3 and 5. Since the plot is nearly identical to that of Figure 3, it is not included here. It is often the case that little or no tension is required to produce a visually pleasing  $C^1$  fit. The extraneous inflection points are eliminated by the local nature of the method. However, second derivative continuity, which may be required for some applications, has been sacrificed. The  $C^2$  exponential tension spline with tension factor  $\sigma = 15$  is depicted in Figure 4. This example demonstrates the weakness of uniform tension: a tension factor large enough to eliminate extraneous inflection points is often nearly piecewise linear in appearance. The final interpolant, depicted in Figure 5, is the shape-preserving  $C^2$  exponential tension spline computed by TSPSI. This fit has none of the drawbacks of the others, but is relatively expensive to compute, requiring 42 calls to YPC2 and SIGS. (Tests

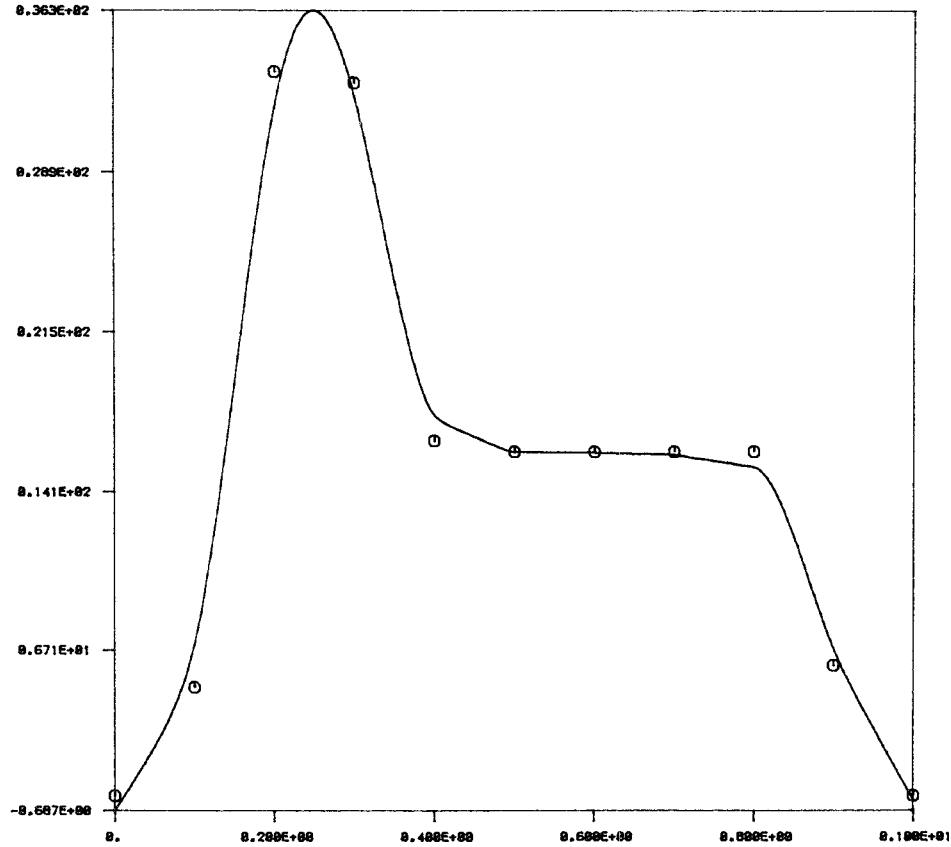


Fig. 7.  $C^2$  exponential tension spline smoothing with variable tension.

with six similar data sets, including the one used here, resulted in a maximum of 42 and an average of 17 YPC2/SIGS iterations to achieve effective convergence.)

The last two figures depict smoothing curves in which the data values are treated as measurements subject to error. It is assumed that the measurements have constant standard deviation  $\delta y = 1$ , and the bound on the weighted sum of squares of deviations from the data is taken to be the nominal value  $N = 11$ . The natural cubic spline fit and the shape-preserving tension spline (both computed by TSPSS) are displayed in Figures 6 and 7, respectively. These plots demonstrate that the advantage of tension for spline interpolation extends to smoothing curves as well.

#### ACKNOWLEDGMENTS

The author gratefully acknowledges helpful comments and suggestions from Alan Cline, Fred Fritsch, the anonymous referees, and Paul Boggs, who served as Algorithms Editor for this submission.

## REFERENCES

1. CLINE, A. K. Scalar- and planar-valued curve fitting using splines under tension. *Commun. ACM* 17, 4 (Apr. 1974), 218–223.
2. DE BOOR, C. *A Practical Guide to Splines*. Springer-Verlag, New York, 1978.
3. FRITSCH, F. N., AND BUTLAND, J. A method for constructing local monotone piecewise cubic interpolants. *SIAM J. Sci. Stat. Comput.* 5, 2 (June 1984), 300–304.
4. FRITSCH, F. N., AND CARLSON, R. E. Monotone piecewise cubic interpolation. *SIAM J. Numer. Anal.* 17, 2 (Apr. 1988), 238–246.
5. KAHANER, D., MOLER, C., AND NASH, S. *Numerical Methods and Software*. Prentice Hall, Englewood Cliffs, N.J., 1989.
6. REINSCH, C. H. Smoothing by spline functions. *Numer. Math.* 10 (1967), 177–183.
7. REINSCH, C. H. Smoothing by spline functions II. *Numer. Math.* 16 (1971), 451–454.
8. RENKA, R. J. Interpolatory tension splines with automatic selection of tension factors. *SIAM J. Sci. Stat. Comput.* 8, 3 (May 1987), 393–415.
9. SCHWEIKERT, D. G. An interpolatory curve using a spline in tension. *J. Math. Phys.* 45 (1966), 312–317.

Received August 1990; revised June 1991; accepted March 1992