General Disclaimer

One or more of the Following Statements may affect this Document

- This document has been reproduced from the best copy furnished by the organizational source. It is being released in the interest of making available as much information as possible.

- This document may contain data, which exceeds the sheet parameters. It was furnished in this condition by the organizational source and is the best copy available.

- This document may contain tone-on-tone or color graphs, charts and/or pictures, which have been reproduced in black and white.

- This document is paginated as submitted by the original source.

- Portions of this document are not fully legible due to the historical nature of some of the material. However, it is the best reproduction available from the original submission.

DOE /ER /03077-171

Courant Mathematics and Computing Laboratory

U.S. Department of Energy

Theory, Computation, and Application of Exponential Splines

Brian J. McCartin

Research and Development Report

Prepared under Contract DE-AC02-76ER03077 with the U. S. Department of Energy, Division of Basic Energy Sciences, Applied Mathematical Sciences Program; and NASA Grants Nos. NGT-33-016-800 and NGT-33-016-201.

Mathematics and Computing
October 1981

New York University

UNCLASSIFIED

Courant Mathematics and Computing Laboratory

New York University

Mathematics and Computing DOE/ER/03077-171

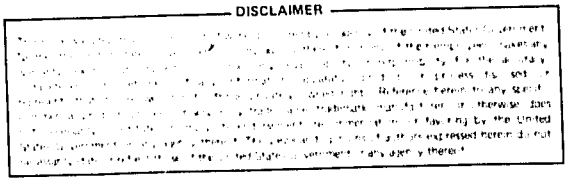
Theory, Computation, and Application of Exponential Splines

Brian J. McCartin

October 1981

Prepared under Contract DE-AC02-76ER03077 with the U. S. Department of Energy, Division of Basic Energy Sciences, Applied Mathematical Sciences Program; and NASA Grants Nos. NGT-33-016-800 and NGR-33-016-201.

UNCLASSIFIED



MKH!

Table of Contents

ABSTRACT

Herein, we discuss a generalization of the semiclassical cubic spline known in the literature as the exponential spline. In actuality, the exponential spline represents a continuum of interpolants ranging from the cubic spline to the linear spline. A particular member of this family is uniquely specified by the choice of certain "tension" parameters.

We first outline the theoretical underpinnings of the exponential spline. This development roughly parallels the existing theory for cubic splines. The primary extension lies in the ability of the exponential spline to preserve convexity and monotonicity present in the data.

We next discuss the numerical computation of the exponential spline. A variety of numerical devices are employed to produce a stable and robust algorithm. An algorithm for the selection of tension parameters that will produce a shape preserving approximant is developed. A sequence of selected curve-fitting examples are presented which clearly demonstrate the advantages of exponential splines over cubic splines.

We conclude with a consideration of the broad spectrum of possible uses of exponential splines in the applications. Our primary emphasis is on computational fluid dynamics although the imaginative reader will recognize the wider generality of the techniques developed.

Notation

$a = x_1 < \ldots < x_{N+1} = b$

$h_i = x_{i+1} - x_i \quad (i = 1,\ldots,N)$

$p_i =$ tension parameter $\quad (i = 1,\ldots,N)$

end conditions: $\tau'(a) = f'(a), \quad \tau'(b) = f'(b)$

$f =$ data

$\tau =$ exponential spline interpolant

$s =$ cubic spline interpolant

$$b_1 = \frac{(f_2 - f_1)}{h_1} - f'(a)$$

$$b_i = \frac{f_{i+1}-f_i}{h_i} - \frac{f_i - f_{i-1}}{h_{i-1}} \quad (i = 2,\ldots,N)$$

$$b_{N+1} = f'(b) - \frac{(f_{N+1}-f_N)}{h_N}$$

$$\left.\begin{aligned} e_i &= \left[\frac{1}{h_i} - \frac{p_i}{S_i}\right] / p_i^2 \\ d_i &= \left[p_i \frac{C_i}{S_i} - \frac{1}{h_i}\right] / p_i^2 \\ S_i &= \sinh(p_i h_i) \\ C_i &= \cosh(p_i h_i) \end{aligned}\right\} \quad (i = 1,\ldots,N)$$

Preface

A preoccupation with problems of interpolation and approximation may be traced back to the dawn of mathematical enterprise. The primary impetus for these investigations was the desire to determine the motions of the heavenly bodies. Indeed, the historical record [P1] contains many practical computations performed by the Babylonian scribes. As an example, one such calculation employs linear interpolation in a table to predict the risings and settings of Mercury. Furthermore, other orbital calculations made during this period reveal that higher order polynomial interpolation was also in use.

An early example of an approximation method that did not reduce to straightforward interpolation was Archimedes' estimation of π [P2]. His technique utilized the approximation of a circle by a sequence of inscribed and circumscribed polygons. Moreover, he provided error bounds for this procedure, which may very well be a "first" in approximation theory. In modern parlance, we would dub this approximation by linear splines (see below).

The piecewise-linear approximation scheme reappeared in Euler's work [P3] on the numerical solution of ordinary differential equations (ODE's). This in turn was a key ingredient of the Cauchy-Peano existence theorem for ODE's [P4].

The eighteenth century also witnessed the full blossoming of polynomial interpolation [P5,P6] culminating in the Newton and Lagrange forms for the interpolating polynomial.

In spite of its theoretical appeal, polynomial interpolation suffers from several serious drawbacks. First of all, since polynomials are analytic functions, any local modification to the data entails a global effect on the interpolant. Also, since N+1 points require an $N^{\underline{th}}$ degree polynomial interpolant, very frequently the result is a curve with undesired undulations. A problem of a much more serious nature is the Runge phenomenon [P7]. Runge's example illustrates the disturbing fact that as the mesh width vanishes, the polynomial interpolant does not necessarily converge to the function being approximated.

One solution to these problems is to use a polynomial of degree m (m small) for the first m+1 data points, another polynomial for the next m+1 data points, and so on. Unfortunately, this procedure, called piecewise-polynomial interpolation, produces an interpolant that belongs only to $C^0[X_1, X_{N+1}]$.

Such was the state of affairs until Schoenberg introduced polynomial spline functions [P3] in the mid-1940's. His idea can be condensed as follows. Locally, i.e. between two consecutive data points, the interpolant is to be a polynomial of degree n while the global interpolant is

only required to belong to $C^{n-1}[x_1, x_{N+1}]$. This results in $N(n+1) - (n-1)$ conditions for the determination of $N(n+1)$ polynomial coefficients. That is to say, we have an $(n-1)$-dimensional space of polynomial spline interpolants to the data. Extra constraints are typically provided in terms of end conditions. Note that for $n = 1$ we obtain the previously alluded to linear splines.

A popular choice is the case $n = 3$, i.e. cubic spline interpolation. The use of such low degree polynomials reduces the risk of wiggles, while second derivative continuity is sufficient in many applications. An additional attraction of the cubic spline is that it possesses a direct analogue in beam theory. This is the draftsman's spline, whence comes the name of this mode of approximation.

Since their introduction, splines have been studied intensively. Convergence of interpolatory splines has been established, as well as convergence of higher derivatives provided the function being approximated is sufficiently smooth. In general, the rate of convergence depends on the degree of smoothness of this underlying function.

The practical utility of cubic splines is quite evident from their widespread use as finite element basis functions, in collocation approximations to differential equations, and in geometric and data-fitting applications. At first glance it would seem that many issues in practical

problems of interpolation and approximation have been resolved by their introduction.

This is true to a limited extent. However, cubic splines can and do produce spurious oscillations in the interpolant. In some cases this is merely a nuisance but in others it can prove to be detrimental. For example, in combustion calculations it could produce an unrealistic detonation, or in computational aerodynamics it could result in the generation of a nonphysical shock wave.

Späth [47] first proposed the exponential spline as a remedy to these difficulties. We arrive at the exponential spline by returning to beam theory. We add a tensile force which has the effect of pulling the beam taut between the support points. The resulting interpolant includes exponentials in place of higher order monomials. Pruess [35,37] has rigorously established that for sufficiently great tensile forces the exponential spline so produced mimics both convexity and monotonicity properties present in the data.

The work at hand gathers together the principal existing results related to exponential spline interpolation and introduces a number of new contributions to the state-of-the-art in the theory and computation of exponential splines. Some novel applications are then considered, in particular the treatment of compressible fluid flows, where it is shown that exponential splines permit the simulation of shock waves with negligible overshoots in the computed solution.

In the development of the theory of exponential splines, we first detail the boundary value problem for the spline under tension [42] followed by its explicit solution in terms of second derivatives [35]. The translation of this second derivative formulation into the corresponding first derivative formulation is then developed. The exponential spline is then placed within the more general framework of piecewise L-splines. This leads us to define higher order tension splines. The possibility of Hermite interpolation by exponential splines is then considered. Next we derive certain extremal properties of exponential splines. Convergence properties of exponential splines originally obtained by Pruess [35] are established via a modified procedure. In addition, these results are extended to include third derivatives. Next, a review of Pruess' [35] results on shape preservation properties of exponential spline interpolation is given. The cardinal spline basis on finite knot sets is introduced, various properties are established by extending the arguments of Birkhoff and de Boor [7], and further properties are obtained by a new procedure. Finally, the B-spline basis is derived together with certain ancillary considerations.

The consideration of computational matters begins with the direct solution of the spline equations [2]. Bounds for the condition number of the tridiagonal matrix are derived.

For the second derivative formulation this was supplied by Pruess [35] while for the first derivative formulation this is new. There follows an analysis of the iterative solution of the spline equations, including a discussion of the optimum relaxation factor. Next is a treatment of spline end conditions. Both the techniques used and the results obtained are unavailable elsewhere. An alternative power series representation, as suggested by Pruess [35], is derived for small hyperbolic function arguments. New parameter selection algorithms are given which produce co-convex and/or co-monotone interpolants. This complements the non-constructive existence proofs previously noted. The periodic exponential spline is then detailed. A variety of numerical topics is next considered. Finally, we present a sequence of examples illustrating the inherent superiority of exponential splines to cubic splines.

The subject of the application of exponential splines has largely been neglected in the literature [17,39,45]. We commence with a broad spectrum of geometric applications in computational fluid dynamics. We then take up the approximate solution of the Laplace, heat, and wave equations using exponential splines. After this we embark on an investigation of the possibility of using the properties of exponential splines to inhibit the appearance of wiggles and overshoots in the numerical simulation of flows with shock waves,

while retaining a high order of accuracy. We commence with the numerical solution of a one dimensional model problem and conclude with the simulation of two dimensional inviscid fluid flow in a channel. We develop a scheme which uses exponential splines to approximate spatial derivatives, while employing a fourth order Runge-Kutta time stepping procedure. Along the way, we generalize to our scheme a result of Lax and Wendroff [28] concerning the computation of weak solutions of nonlinear hyperbolic conservation laws. There follows a discussion of the stability and accuracy of the proposed scheme and a number of alternative schemes. Artificial viscosity is included via the coupling of Flux Vector Splitting [50] with the upwinding of derivatives [24]. We note that our treatment of Flux Vector Splitting contains a new result concerning the nature of the split Jacobians. Boundary conditions are enforced in a consistent fashion by performing a local analysis of the characteristic variables [19,22]. The significant achievement here is that the proposed scheme yields a numerical solution that is third order accurate (fourth order accurate on a uniform mesh) in smooth regions of the flow, while accurately capturing any discontinuities that arise without significant overshoots/undershoots.

References

[P1] Neugerbauer, O., The Exact Sciences in Antiquity, Dover, 1969.

[P2] Heath, T. L., The Work of Archimedes, Cambridge, 1897.

[P3] Schoenberg, I. J., Cardinal Spline Interpolation, CBMS Vol. 12, SIAM, 1973.

[P4] Coddington, E. A., and Levinson, N., Theory of Ordinary Differential Equations, McGraw-Hill, 1955.

[P5] Bell, E. T., Men of Mathematics, Simon and Schuster, 1937.

[P6] Newman, J. R., The World of Mathematics: Vol. 1, Simon and Schuster, 1956.

[P7] Davis, P. J., Interpolation and Approximation, Dover, 1975.

Part I: Theory of Exponential Splines

I.1. Overview

In this first part we discuss exponential splines from a theoretical viewpoint. Starting from the analogy of a cubic spline to a beam we add a tension term to the governing differential equation thus giving rise to the exponential spline. The solution to this boundary value problem expresses the exponential spline in terms of its second derivatives at the knots. This is the form we prefer since it leads to simpler expressions than a representation in terms of the first derivatives. However, since it will be useful to us later, we next derive this other system of equations. The exponential spline is then inspected in the context of generalized splines. This leads us to consider higher order tension splines and higher degree interpolation. Certain extremal properties are then derived. Convergence of the approximating spline is next studied through the proximity of the interpolating cubic and exponential splines with the same end conditions. The shape preservation capabilities of exponential splines are then reviewed. Finally, we add to the number of possible representations by introducing the cardinal spline and B-spline bases.

I.2. Derivation of Exponential Spline Equations in Terms of Second Derivatives

The cubic spline is well known to have the following analogue in beam theory [2]. Consider a simply supported beam with supports $\{(x_i,f_i)\}_{i=1}^{N+1}$. Then $s(x)$, the deflection of the beam, is a solution to

$$[E\cdot I\cdot D^2]s = M$$

between successive supports. Here

E = Young's modulus

I = cross sectional moment of inertia

M = bending moment.

Under the assumption of weightlessness, M is a piecewise linear continuous function with break points at the supports. Thus, differentiating the above twice we arrive at the two-point boundary value problem on $[x_i,x_{i+1}]$ $(i = 1,\ldots,N)$:

$$[D^4]s = 0\ , \quad s(x_i) = f_i\ , \quad s(x_{i+1}) = f_{i+1}\ ,$$

$$s''(x_i) = s_i''\ , \quad s''(x_{i+1}) = s_{i+1}''\ ,$$

where s_i'' and s_{i+1}'' are chosen to ensure $s \in C^2[a,b]$ when $s'(a)$ and $s'(b)$ are given. Note that $[D^4]s = 0$ on $[x_i,x_{i+1}] \Rightarrow s$ is a cubic there.

The cubic spline so defined has a tendency to exhibit unwanted undulations. Correspondingly, the above analogy suggests that the application of uniform tension between

supports might remedy the problem [42,43].* The beam equation on $[x_i, x_{i+1}]$ then becomes

$$[E \cdot I \cdot D^2 - t_i I]s = M \ .$$

Letting $p_i^2 = (t_i / E \cdot I)$ leads us to define the exponential spline [47,48,49] as the solution to the boundary value problem on $[x_i, x_{i+1}]$ $(i = 1,\ldots,N)$:

$$[D^4 - p_i^2 D^2]\tau = 0 \ , \quad \tau(x_i) = f_i \ , \quad \tau(x_{i+1}) = f_{i+1} \ ,$$

$$\tau''(x_i) = \tau_i'' \ , \quad \tau''(x_{i+1}) = \tau_{i+1}'' \ ,$$

with τ_i'' $(i = 1,\ldots,N+1)$ as yet undetermined.
Let us pause to note that

(i) $p_i \to 0 \Rightarrow [D^4]\tau = 0$, i.e. the cubic spline,

(ii) $p_i \to \infty \Rightarrow [\frac{1}{p_i^2} D^4 - D^2]\tau = 0 \Rightarrow [D^2]\tau = 0$,

i.e. the polygonal

("broken line") interpolant.

We now solve this boundary value problem (BVP) for the exponential spline. Let $t(x) = \tau''(x)$, then $t(x)$ is a solution to the BVP

$$t'' - p_i^2 t = 0 \ , \quad t(x_i) = \tau_i'' \ , \quad t(x_{i+1}) = \tau_{i+1}'' \ .$$

Therefore

---

* In the following paragraphs and throughout the thesis, $s(x)$ will represent the cubic spline and $\tau(x)$ the exponential spline.

$$t(x) = \frac{1}{S_i}\left\{\tau''_i \sinh p_i(x_{i+1}-x) + \tau''_{i+1} \sinh p_i(x-x_i)\right\}.$$

Hence $t(x)$ satisfies the BVP

$$\tau''(x) = \frac{1}{S_i}\left\{\tau''_i \sinh p_i(x_{i+1}-x) + \tau''_{i+1} \sinh p_i(x-x_i)\right\}$$

$$\tau(x_i) = f_i\ , \quad \tau(x_{i+1}) = f_{i+1}.$$

Therefore

$$\tau(x) = \frac{1}{p_i^2 S_i}\left\{\tau''_i \sinh p_i(x_{i+1}-x) + \tau''_{i+1} \sinh p_i(x-x_i)\right\}$$
$$+ \left(f_i - \frac{\tau''_i}{p_i^2}\right)\frac{x_{i+1}-x}{h_i} + \left(f_{i+1} - \frac{\tau''_{i+1}}{p_i^2}\right)\frac{x-x_i}{h_i}$$
$$\left(= A_i + B_i x + C_i e^{p_i x} + D_i e^{-p_i x}\right).$$

The requirements of first derivative continuity at the points of interpolation yield expressions for the determination of τ''_i $(i = 1,\ldots,N+1)$. Specifically, for $i = 2,\ldots,N$ we have

$$\tau'(x_i^+) = \frac{1}{p_i S_i}\left\{-\tau''_i C_i + \tau''_{i+1}\right\} + \frac{1}{h_i}\left\{f_{i+1} - f_i + \frac{\tau''_i - \tau''_{i+1}}{p_i^2}\right\}$$

$$\tau'(x_i^-) = \frac{1}{p_{i-1} S_{i-1}}\left\{-\tau''_{i-1} + \tau''_i C_{i-1}\right\} + \frac{1}{h_{i-1}}\left\{f_i - f_{i-1} + \frac{\tau''_{i-1} - \tau''_i}{p_{i-1}^2}\right\}.$$

Also

$$\tau'(x_1) = \frac{1}{p_1 S_1}\left\{-\tau''_1 C_1 + \tau''_2\right\} + \frac{1}{h_i}\left\{f_2 - f_1 + \frac{\tau''_1 - \tau''_2}{p_1^2}\right\}$$

$$\tau'(x_{N+1}) = \frac{1}{p_N S_N}\left\{-\tau''_N + \tau''_{N+1} C_N\right\} + \frac{1}{h_N}\left\{f_{N+1} - f_N + \frac{\tau''_N - \tau''_{N+1}}{p_N^2}\right\}.$$

Hence, τ_i'' $(i = 1,\ldots,N+1)$ are the solution of the tri-diagonal system

$$\begin{cases} d_1\tau_1'' + e_1\tau_2'' = b_1 \ , \\ e_{i-1}\tau_{i-1}'' + (d_{i-1}+d_i)\tau_i'' + e_i\tau_{i+1}'' = b_i \ , \quad (i=2,\ldots,N), \\ e_N\tau_N'' + d_N\tau_{N+1}'' = b_{N+1} \ , \end{cases}$$

where $e_i \equiv [\frac{1}{h_i} - \frac{p_i}{S_i}]/p_i^2$, $d_i \equiv [p_i \frac{C_i}{S_i} - \frac{1}{h_i}]/p_i^2$ with $S_i \equiv \sinh(p_i h_i)$, $C_i = \cosh(p_i h_i)$. τ is uniquely defined once τ_i'' $(i = 1,\ldots,N+1)$ are determined.

I.3. Derivation of Exponential Spline Equations in Terms of First Derivatives.

Recall that on $[x_i, x_{i+1}]$ the exponential spline is the solution to the boundary value problem

$$(*) \quad [D^4 - p_i^2 D^2]\tau(x) = 0 \text{ subject to}$$

$$\tau(x_i) = f_i \ , \ \tau(x_{i+1}) = f_{i+1} \ , \ \tau'(x_i) = \tau'_i \ , \ \tau'(x_{i+1}) = \tau'_{i+1}.$$

The general solution of (*) is

$$(**) \quad \begin{cases} \tau(x) = K_1^{(i)} e^{p_i(x_{i+1}-x)} + K_2^{(i)} e^{p_i(x-x_i)} + K_3^{(i)} x + K_4^{(i)} \Rightarrow \\ \tau'(x) = -p_i K_1^{(i)} e^{p_i(x_{i+1}-x)} + p_i K_2^{(i)} e^{p_i(x-x_i)} + K_3^{(i)}. \end{cases}$$

The determination of $K_j^{(i)}$ $(j = 1,2,3,4)$ then consists of substituting into (**) and enforcing the boundary conditions.

However, the solution of the resulting 4×4 system of equations becomes quite tedious thus making an alternative route desirable. We instead use previously obtained results to derive the desired tridiagonal system.

Recall that on $[x_i, x_{i+1}]$ $(i = 1,\ldots,N)$

$$\tau'(x) = \frac{1}{p_i S_i}\left\{-\tau''_i \cosh p_i(x_{i+1}-x) + \tau''_{i+1} \cosh p_i(x-x_i)\right\}$$
$$+ \frac{1}{h_i}\left\{(f_{i+1}-f_i) - \frac{\tau''_{i+1}-\tau''_i}{p_i^2}\right\}. \Rightarrow$$

$$\tau'_{i-1} = -d_{i-1}\tau''_{i-1} - e_{i-1}\tau''_i + \frac{f_i - f_{i-1}}{h_{i-1}}$$

$$\tau'_i = e_{i-1}\tau''_{i-1} + d_{i-1}\tau''_i + \frac{f_i - f_{i-1}}{h_{i-1}}$$

$$\tau_i' = -d_i\tau_i'' - e_i\tau_{i+1}'' + \frac{f_{i+1}-f_i}{h_i}$$

$$\tau_{i+1}' = e_i\tau_i'' + d_i\tau_{i+1}'' + \frac{f_{i+1}-f_i}{h_i} .$$

$$\Rightarrow \tau_i'' = \left[\frac{d_{i-1}}{d_{i-1}^2-e_{i-1}^2}\right] \cdot \left\{\tau_i' + \frac{e_{i-1}}{d_{i-1}}\tau_{i-1}' - \left[\frac{e_{i-1}}{d_{i-1}} + 1\right] \cdot \left[\frac{f_i-f_{i-1}}{h_{i-1}}\right]\right\}$$

and

$$\tau_i'' = \left[\frac{d_i}{e_i^2 - d_i^2}\right] \cdot \left\{\tau_i' + \frac{e_i}{d_i}\tau_{i+1}' - \left[\frac{e_i}{d_i} + 1\right] \cdot \left[\frac{f_{i+1}-f_i}{h_i}\right]\right\} .$$

Equating these results yields

$$\left[\frac{e_{i-1}}{d_{i-1}^2-e_{i-1}^2}\right]\tau_{i-1}' + \left[\frac{d_{i-1}}{d_{i-1}^2-e_{i-1}^2} + \frac{d_i}{d_i^2-e_i^2}\right]\tau_i' + \left[\frac{e_i}{d_i^2-e_i^2}\right]\tau_{i+1}'$$

$$= \left[\frac{1}{d_{i-1}-e_{i-1}}\right] \cdot \left[\frac{f_i-f_{i-1}}{h_{i-1}}\right] + \left[\frac{1}{d_i-e_i}\right] \cdot \left[\frac{f_{i+1}-f_i}{h_i}\right] .$$

This of course must be modified at the boundaries.

The special case $p_i = p$, $h_i = h$ $\forall i$ yields

$$\left[\frac{e}{2(d+e)}\right]\tau_{i-1}' + \left[\frac{d}{d+e}\right]\tau_i' + \left[\frac{e}{2(d+e)}\right]\tau_{i+1}' = \frac{f_{i+1}-f_{i-1}}{2h} .$$

Note that $\left[\frac{e}{2(d+e)}\right] + \left[\frac{d}{d+e}\right] + \left[\frac{e}{2(d+e)}\right] = 1$.

Furthermore, if $p_i = 0$ $\forall i$ then we have $e_i = h_i/6$ and $d_i = h_i/3$. Consequently, the above relations reduce to the familiar cubic spline equations.

If we wish to specify $\tau''_1 = f''(a)$ and/or $\tau''_{N+1} = f''(b)$ we may use the relations

$$\left[\frac{d_1}{d_1^2-e_1^2}\right]\tau'_1 + \left[\frac{e_1}{d_1^2-e_1^2}\right]\tau'_2 = \left[\frac{1}{d_1-e_1}\right]\cdot\left[\frac{f_2-f_1}{h_1}\right] - f''(a)$$

$$\left[\frac{e_N}{d_N^2-e_N^2}\right]\tau'_N + \left[\frac{d_N}{d_N^2-e_N^2}\right]\tau'_{N+1} = \left[\frac{1}{d_N-e_N}\right]\cdot\left[\frac{f_{N+1}-f_N}{h_N}\right] + f''(b) \ .$$

If we want the weights in front of τ'_{i-1} , τ'_i, τ'_{i+1} to sum to unity we may multiply the $i^{\underline{th}}$ equation by

$$\frac{(d_{i-1}-e_{i-1})(d_i-e_i)}{(d_{i-1}-e_{i-1})+(d_i-e_i)}$$

producing

$$\frac{e_{i-1}(d_i-e_i)}{(d_{i-1}+e_{i-1})[(d_{i-1}-e_{i-1})+(d_i-e_i)]}\cdot\tau'_{i-1}$$

$$+\left\{\frac{d_{i-1}(d_i-e_i)}{(d_{i-1}+e_{i-1})[(d_{i-1}-e_{i-1})+(d_i-e_i)]}\right.$$

$$\left.+\frac{d_i(d_{i-1}-e_{i-1})}{(d_i+e_i)[(d_{i-1}-e_{i-1})+(d_i-e_i)]}\right\}\cdot\tau'_i$$

$$+\frac{e_i(d_{i-1}-e_{i-1})}{(d_i+e_i)[(d_{i-1}-e_{i-1})(d_i-e_i)]}\cdot\tau'_{i+1} =$$

$$\frac{d_i - e_i}{(d_{i-1}-e_{i-1})+(d_i-e_i)}\cdot\frac{f_i-f_{i-1}}{h_{i-1}} + \frac{d_{i-1}-e_{i-1}}{(d_{i-1}-e_{i-1})+(d_i-e_i)}\cdot\frac{f_{i+1}-f_i}{h_i}.$$

We may now use our relations between $\{\tau''_k\}_{k=1}^{N+1}$ and $\{\tau'_k\}_{k=1}^{N+1}$ to establish

$$\tau(x) = f_i \cdot \frac{x_{i+1}-x}{h_i} + f_{i+1} \cdot \frac{x-x_i}{h_i} + \frac{1}{e_i-d_i} \cdot \frac{f_{i+1}-f_i}{h_i}$$

$$\cdot \left[\frac{\sinh p_i(x-x_i) - \sinh p_i(x_{i+1}-x)}{p_i^2 S_i} + \frac{x_{i+1}-2x+x_i}{p_i^2 h_i}\right]$$

$$+ \tau'_i \left\{\frac{d_i}{e_i^2-d_i^2}\left[\frac{\sinh p_i(x_{i+1}-x)}{p_i^2 S_i} - \frac{x_{i+1}-x}{p_i^2 h_i}\right]\right.$$

$$\left. - \frac{e_i}{e_i^2-d_i^2}\left[\frac{\sinh p_i(x-x_i)}{p_i^2 S_i} - \frac{x-x_i}{p_i^2 h_i}\right]\right\}$$

$$+ \tau'_{i+1} \left\{\frac{e_i}{e_i^2-d_i^2}\left[\frac{\sinh p_i(x_{i+1}-x)}{p_i^2 S_i} - \frac{x_{i+1}-x}{p_i^2 h_i}\right]\right.$$

$$\left. - \frac{d_i}{e_i^2-d_i^2}\left[\frac{\sinh p_i(x-x_i)}{p_i^2 S_i} - \frac{x-x_i}{p_i^2 h_i}\right]\right\}.$$

I.4. Generalized Splines

In deriving the equation for the exponential spline we used the factorization $D^4-p_i^2D^2 = (D^2 - p_i^2I)D^2$. The symmetric factorization $D^4-p_i^2D^2 = (D^2+p_iD)(D^2-p_iD)$, however, permits an interpretation in a generalized spline context [2,3,4,21,33,34,40,41,51].

In general, let L be a linear differential operator of order m:

$$L = p_0(x)D^m + p_1(x)D^{m-1} + \ldots + p_{m-1}(x)D + p_m(x)$$

with $p_k(x) \in C^m[a,b]$; $k = 0,1,\ldots,m$. Let L^* be the formal adjoint of L:

$$L^* = (-1)^m D^m(p_0) + (-1)^{m-1} D^{m-1}(p_1) + \ldots - D(p_{m-1}) + p_m.$$

A generalized (interpolatory) spline, s, associated with L is defined as $s \in C^{2m-2}[a,b]$; $s \in C^{2m}(x_i,x_{i+1})$ where it satisfies $L^*Ls = 0$, $i = 1,\ldots,N$; and $s(x_i) = f_i$, $i = 1,\ldots,N+1$. (Also known as L-splines.)

We note the following special cases:

(i) $m = 2$, $L = D^2 \Rightarrow L^* = D^2$; thus $L^*L = D^4$, i.e. cubic spline interpolation.

(ii) $m = 2$, $L = D^2 - pD \Rightarrow L^* = D^2+pD$; thus $L^*L = D^4 - p^2D^2$, i.e. hyperbolic spline interpolation [6,42,43].

We may instead insist that s satisfy $L_i^* L_i s = 0$ on (x_i, x_{i+1}). Such splines are called piecewise L-splines [33]. If we let $L_i = D^2 - p_i D$ then $L_i^* = D^2 + p_i D$ $\Rightarrow L_i^* L_i = D^4 - p_i^2 D^2$, i.e. exponential spline interpolation.

The principal advantage of this framework is that the functional $\sum_{i=1}^{N} \int_{x_i}^{x_{i+1}} (L_i f)^2 \, dx$ is minimized by the piecewise L-spline fulfilling the end conditions $(L_1 \tau)^{(k-1)} = 0$ $(k = 1, \ldots, m-1)$ at x_1 and $(L_N \tau)^{(k-1)} = 0$ at x_{N+1}. In a subsequent section we specialize this result to the exponential spline and use it to produce further extremal properties.

I.5. Higher Order Tension Splines

In this section we generalize the exponential spline. The starting point for this discussion is the characterization of the exponential spline as being in the null space of $E \equiv D^4 - p_i^2 D^2$ between knots. One possible extension would be to consider piecewise solutions of

$$[D^4 + \alpha \cdot p_i D^3 - p_i^2 D^2]\, t(x) = 0 \ .$$

This has the desirable feature of reducing to $[D^4]\, t = 0$ for $p_i = 0$ and $[D^2]\, t = 0$ for $p_i = \infty$. However this operator does not permit a factorization as $L^* L$ since $\alpha \neq 0$ implies a differential operator that is not self-adjoint. As such it does not produce piecewise L-splines. In fact the most general fourth order homogeneous differential operator with real constant coefficients (lead coefficient = 1) that permits such a decomposition is precisely the exponential spline operator E.

Hence if we want a generalization using constant coefficients that produce L-splines we must increase the order of the operator. Thus, consider the sixth order ODE

$$[D^6 + \alpha D^5 + \beta D^4 + \gamma D^3 + \delta D^2]\, t(x) = 0 \ .$$

Let $L = D^3 + \mu D^2 + \eta D \Rightarrow L^* = - D^3 + \mu D^2 - \eta D$. Thus $L^* L = - D^6 + (\mu^2 - 2\eta) D^4 - \eta^2 D^2$. So that multiplying the ODE by -1 we have

$$[L^*L]t(x) = 0$$

where $\alpha = \gamma = 0$ (by self-adjointness of the differential operator), $\delta = \eta^2$, $\beta = 2\eta - \mu^2$. I.e.

$$[D^6 + (2\eta-\mu^2)D^4 + \eta^2D^2]t(x) = 0$$

or

$$[D^2(D^2+\mu D+\eta I)(D^2-\mu D+\eta I)]t(x) = 0 .$$

The characteristic roots will then determine the basis functions for the null space. The double root of zero admits 1 and x. The other roots are

$$\lambda \in \left\{ \frac{-\mu \pm \sqrt{\mu^2-4\eta}}{2}, \frac{\mu \pm \sqrt{\mu^2-4\eta}}{2} \right\}.$$

If $\eta^2 = 0$ we admit x^2, x^3, $e^{-\mu x}$, $e^{\mu x}$. If $\mu^2-4\eta = 0$ we admit $e^{-\mu x}$, $xe^{-\mu x}$, $e^{\mu x}$, $xe^{\mu x}$. Otherwise we have four distinct λ's and the corresponding basis functions. Note that for $\mu^2 > 4\eta$ we obtain hyperbolic functions while if $\mu^2 < 4\eta$ we obtain trigonometric functions.

All this leads us to the following definition of tension splines of order 2m (degree 2m-1). Let

$$T \equiv D^{2m} + \alpha_{2m-2}D^{2m-2} + \ldots + \alpha_4D^4 + \alpha_2D^2$$

have the factorization $T = (-1)^m L^*L$ where

$$L \equiv D^m + \beta_{m-1}D^{m-1} + \ldots + \beta_2D^2 + \beta_1D \Rightarrow$$

$$L^* = (-1)^m D^m + (-1)^{m-1}\beta_{m-1}D^{m-1} + \ldots + \beta_2D^2 - \beta_1D.$$

Then t(x) is a tension spline if it is a piecewise solution of

$$[T]t = [(-1)^m L^*L]t = 0 .$$

Therefore

$$D^2[D^{m-1} + \beta_{m-1}D^{m-2} + \ldots + \beta_2 D + \beta_1] \cdot [D^{m-1} - \beta_{m-1}D^{m-2} + \ldots + (-1)^{m-2}\beta_2 D + (-1)^{m-1}\beta_1]t = 0 .$$

Again the double root admits 1 and x as basis functions for $N(L^*L)$. The other basis functions are determined by the remaining roots of the characteristic equation.

This generalization allows us to pursue one of two routes. First we could require a greater degree of smoothness at the knots. The sixth order operator discussed above would produce a quintic tension spline in this context. On the other hand we could require higher order interpolation at the knots. Our previous example then amounts to a quintic Hermite interpolant under tension matching function values together with first and second derivatives.

I.6. Hermite Interpolants

The spline fit by its very nature is a global scheme as it requires the solution of a tridiagonal system. On the other hand osculatory, or Hermite, interpolation provides a local means of interpolation [15]. For this reason Hermite interpolation is many times preferred over spline interpolation.

Hermite interpolation requires the specification of a certain number of consecutive derivatives at each knot. The particular number may vary from knot to knot. The local nature of this approximation comes to us at the expense of smoothness. For example if we specify first derivatives we have a cubic Hermite interpolant which is only C^1 as opposed to the C^2 smoothness provided by the cubic spline. Moreover, the required derivatives are typically not available and must themselves be approximated.

With these provisos duly noted, we now proceed to discuss Hermite interpolation by exponential splines. As pointed out in another section a higher degree of contact may be achieved by resorting to higher order tension splines.

For ease of presentation, we restrict $x \in [0,1]$ with $f(0) = f_0$, $f(1) = f_1$, $f'(0) = f_0'$, $f'(1) = f_1'$ given. In this setting exponential Hermite interpolation is effected by

$$h(x) = f_0\phi_0(x) + f_1\phi_1(x) + f_0'\bar{\phi}_0(x) + f_1'\bar{\phi}_1(x)$$

where ϕ_0, ϕ_1, $\bar{\phi}_0$, $\bar{\phi}_1$ are the cardinal functions. These functions are defined as follows.

$$\phi_0(x) = a_0+b_0x+c_0e^{px}+d_0e^{-px}; \quad \phi_0(0)=1,\ \phi_0(1)=\phi_0'(0)=\phi_0'(1)=0$$

$$\phi_1(x) = a_1+b_1x+c_1e^{px}+d_1e^{-px}; \quad \phi_1(1)=1,\ \phi_1(0)=\phi_1'(0)=\phi_1'(1)=1$$

$$\bar{\phi}_0(x) = \bar{a}_0+\bar{b}_0x+\bar{c}_0e^{px}+\bar{d}_0e^{-px}; \quad \bar{\phi}_0'(0)=1,\ \bar{\phi}_0(0)=\bar{\phi}_0(1)=\bar{\phi}_0'(1)=0$$

$$\bar{\phi}_1(x) = \bar{a}_1+\bar{b}_1x+\bar{c}_1e^{px}+\bar{d}_1e^{-px}; \quad \bar{\phi}_1'(1)=1,\ \bar{\phi}_1(0)=\bar{\phi}_1(1)=\bar{\phi}_1'(0)=0.$$

This leads to

$$\phi_0(x) = \Big\{(2 - 2\cosh p + 2p\sinh p) + (-2p\sinh p)x + (1 - e^{-p})e^{px} + (1-e^{p})e^{-px}\Big\}/\Big\{4-4\cosh p + 2p\sinh p\Big\}$$

$$\phi_1(x) = \Big\{(2p\sinh p)x + (1 - e^{p})e^{px} + (1 - e^{-p})e^{-px}\Big\} / \Big\{4 - 4\cosh p + 2p\sinh p\Big\}$$

$$\bar{\phi}_0(x) = \Big\{p(2 - 2\cosh p)x + (-1 - p + e^{p})e^{px} + (1-p-e^{-p})e^{-px}\Big\} / \Big\{p(4 - 4\cosh p + 2p\sinh p)\Big\}$$

$$\bar{\phi}_1(x) = \Big\{(2\sinh p - 2p) + p(2 - 2\cosh p)x + (-1+p+e^{-p})e^{px} + (1+p-e^{p})e^{-px}\Big\} / \Big\{p(4 - 4\cosh p+2p\sinh p)\Big\}.$$

Notice that

$$\phi_1(x) = \phi_0(1-x)$$
$$\bar{\phi}_1(x) = \bar{\phi}_0(1-x).$$

Note that all four basis functions involve e^{px} and e^{-px}. The computational complexity may be reduced by the following construction [30].

Let

$$h(x) = f_0\psi_1(x) + f_1\hat{\psi}_1(x) + A\psi_2(x) + B\psi_3(x)$$

where

$$\psi_1(x) = 1-x$$

$$\hat{\psi}_1(x) = x = \psi_1(1-x)$$

$$\psi_2(x) = a_2+b_2x+c_2e^{px};\ \psi_2(0)=0,\ \psi_2(1)=0,\ \psi_2'(0)=\alpha$$

$$\psi_3(x) = a_3+b_3x+c_3e^{-px};\ \psi_3(0)=0,\ \psi_3(1)=0,\ \psi_3'(0)=\beta$$

and A,B are as yet undetermined.

This produces

$$\psi_2(x) = \alpha \cdot \frac{-1+(1-e^{p})x+e^{px}}{1+p-e^{p}}$$

$$\psi_3(x) = \beta \cdot \frac{-1+(1-e^{-p})x+e^{-px}}{1-p-e^{-p}} .$$

Therefore

$$\psi_2'(1) = \psi_2'(0) \cdot \frac{1-e^{p}+pe^{p}}{1 + p - e^{p}}$$

$$\psi_3'(1) = \psi_3'(0) \cdot \frac{1-e^{-p}-pe^{-p}}{1 - p - e^{-p}} = \frac{-e^{p}+1+p}{1-e^{p}+pe^{p}} \cdot \psi_3'(0)$$

$$\Rightarrow \psi_3'(0) = \frac{1 - e^{p} + pe^{p}}{1 + p - e^{p}}\ \psi_3'(1).$$

Letting $q \equiv \frac{1-e^{p}+pe^{p}}{1+p-e^{p}}$ we have

$$\psi_2'(1) = q\cdot\psi_2'(0)$$

$$\psi_3'(0) = q\cdot\psi_3'(1) \ .$$

Now $f_0\psi_1 + f_1\hat{\psi}_1$ has a slope of $m = f_1 - f_0$. Hence we require that

$$A\psi_2'(0) + B\psi_3'(0) = f_0' - m$$

and

$$A\psi_2'(1) + B\psi_3'(1) = f_1' - m \ .$$

I.e.,

$$A\psi_2'(0) + Bq\psi_3'(1) = f_0' - m$$

and

$$Aq\psi_2'(0) + B\psi_3'(1) = f_1' - m$$

or

$$\begin{bmatrix} \psi_2'(0) & q\psi_3'(1) \\ q\psi_2'(0) & \psi_3'(1) \end{bmatrix} \begin{bmatrix} A \\ B \end{bmatrix} = \begin{bmatrix} f_0'-m \\ f_1'-m \end{bmatrix}$$

$$\Rightarrow \begin{bmatrix} A \\ B \end{bmatrix} = \frac{1}{(1-q^2)\psi_2'(0)\psi_3'(1)} \begin{bmatrix} \psi_3'(1) & -q\psi_3'(1) \\ -q\psi_2'(0) & \psi_2'(0) \end{bmatrix} \begin{bmatrix} f_0'-m \\ f_1'-m \end{bmatrix} .$$

Now let

$$\psi_2'(0) = \frac{1}{1+q} = \frac{1+p-e^p}{2(1-e^p)+p(1+e^p)}$$

and

$$\psi_3'(1) = \frac{1}{1-q} = \frac{1+p-e^p}{p(1-p)}$$

$$\Rightarrow \begin{bmatrix} A \\ B \end{bmatrix} = \begin{bmatrix} \frac{1}{1-q} & \frac{-q}{1-q} \\ \frac{-q}{1+q} & \frac{1}{1+q} \end{bmatrix} \begin{bmatrix} f_0'-m \\ f_1'-m \end{bmatrix} .$$

Equally as simple would be to let

$$\psi_2'(0) = \psi_3'(1) = 1$$

$$\Rightarrow \quad \begin{bmatrix} A \\ \\ B \end{bmatrix} = \begin{bmatrix} \frac{1}{1-q^2} & \frac{-q}{1-q^2} \\ \frac{-q}{1-q^2} & \frac{1}{1-q^2} \end{bmatrix} \begin{bmatrix} f_0'-m \\ \\ f_1'-m \end{bmatrix} .$$

What is important here is not so much the result as the technique. Specifically, suppose we are provided with two new functions $e^{\sigma x}$ and $e^{-\sigma x}$. Together with $1, x, e^{px}, e^{-px}$ we are now required to match f, f', f" at x = 0,1. The cardinal spline approach would have us construct the six new basis functions

$$\phi_0(x) = a_0+b_0x+c_0e^{px}+d_0e^{-px}+e_0e^{\sigma x}+f_0e^{-\sigma x};$$

$$\phi_0(0) = 1, \ \phi_0(1) = \phi_0'(0) = \phi_0'(1) = \phi_0''(0) = \phi_0''(1) = 0$$

$$\phi_1(x) = \phi_0(1-x)$$

$$\bar{\phi}_0(x) = \bar{a}_0+\bar{b}_0x+\bar{c}_0e^{px}+\bar{d}_0e^{-px}+\bar{e}_0e^{\sigma x}+\bar{f}_0e^{-\sigma x};$$

$$\bar{\phi}_0'(0) = 1, \ \bar{\phi}_0(0) = \bar{\phi}_0(1) = \bar{\phi}_0'(1) = \bar{\phi}_0''(0) = \bar{\phi}_0''(1) = 0$$

$$\bar{\phi}_1(x) = \bar{\phi}_0(1-x)$$

$$\bar{\bar{\phi}}_0(x) = \bar{\bar{a}}_0+\bar{\bar{b}}_0x+\bar{\bar{c}}_0e^{px}+\bar{\bar{d}}_0e^{-px}+\bar{\bar{e}}_0e^{\sigma x}+\bar{\bar{f}}_0e^{-\sigma x} \ ;$$

$$\bar{\bar{\phi}}_0''(0) = 1, \ \bar{\bar{\phi}}_0(0) = \bar{\bar{\phi}}_0(1) = \bar{\bar{\phi}}_0'(0) = \bar{\bar{\phi}}_0'(1) = \bar{\bar{\phi}}_0''(1) = 0$$

$$\bar{\bar{\phi}}_1(x) = \bar{\bar{\phi}}_0(1-x) \ .$$

On the other hand our alternate technique would have us calculate h"(0) and h"(1). We would then define

$$\psi_4(x) = a_4 + b_4x + c_4e^{px} + d_4e^{-px} + e_4e^{\sigma x} ;$$

$$\psi_4(0) = \psi_4(1) = \psi_4'(0) = \psi_4''(1) = 0, \psi_4''(0) = \alpha$$

$$\psi_5(x) = a_5 + b_5x + c_5e^{px} + d_5e^{-px} + e_5e^{-\sigma x} ;$$

$$\psi_5(0) = \psi_5(1) = \psi_5'(0) = \psi_5'(1) = 0 , \psi_5''(0) = \beta.$$

Finally, we would set

$$H(x) = h(x) + A\psi_4(x) + B\psi_5(x)$$

where

$$A\psi_4''(0) + B\psi_5''(0) = f_0'' - h''(0)$$

and

$$A\psi_4''(1) + B\psi_5''(1) = f_1'' - h''(1) .$$

I.7. Extremal Properties

Results in the literature on piecewise L-splines [33] provide certain extremal properties associated with either of the functionals $\int_a^b (f'' \pm pf')^2\, dx$. In this section we derive a third functional, $\int_a^b [(f'')^2 + p^2(f')^2]\, dx$, and then establish the associated extremal properties.

We begin with the following

Lemma. $$\int_\alpha^\beta \left\{v(u'' \pm pu') - u(v'' \mp pv')\right\} dx = \left\{u'v - uv' \pm puv\right\}\Big|_\alpha^\beta$$

Proof:

$$\int_\alpha^\beta \{\ldots\}\, dx = \int_\alpha^\beta \left\{vu'' \pm pvu' - uv'' \pm puv'\right\} dx$$

$$= \int_\alpha^\beta \left\{vu'' - uv'' \pm p(uv)'\right\} dx$$

$$= vu'\Big|_\alpha^\beta - \int_\alpha^\beta v'u'\, dx - uv'\Big|_\alpha^\beta + \int_\alpha^\beta u'v'\, dx \pm puv\Big|_\alpha^\beta$$

$$= \left\{u'v - uv' \pm puv\right\}\Big|_\alpha^\beta \qquad \text{Q.E.D.}$$

Now let $p(x)$ be a step function $\ni p(x) = p_i$ on $[x_i, x_{i+1}]$; $i = 1,\ldots,N$. In addition let f have an absolutely continuous first derivative and a square integrable second derivative. Then notice that

$$\int_a^b (f'' \pm pf')^2\,dx = \int_a^b [f''-\tau'' \pm p(f'-\tau')]^2\,dx$$

$$+ \int_a^b (\tau'' \pm p\tau')^2\,dx$$

$$+ 2\int_a^b [f''-\tau'' \pm p(f'-\tau')]\cdot(\tau'' \pm p\tau')\,dx.$$

Inspect this last term,

$$\int_a^b [f''-\tau'' \pm p(f'-\tau')]\cdot(\tau'' \pm p\tau')\,dx\ ;$$

by letting $u = f-\tau$ and $v = \tau'' \pm p\tau'$ in our lemma. This yields

$$\sum_{i=1}^{N} \int_{x_i}^{x_{i+1}} (\tau'' \pm p\tau')[f''-\tau'' \pm p(f'-\tau')] - (f-\tau)[\tau^{(iv)} \pm p\tau'''$$

$$\mp p(\tau''' \pm p\tau'')]\,dx$$

$$= \sum_{i=1}^{N} \left\{(f'-\tau')(\tau'' \pm p\tau') - (f-\tau)(\tau''' \pm p\tau'') \pm p(f-\tau)(\tau'' \pm p\tau')\right\}_{x_i}^{x_{i+1}}.$$

Letting τ interpolate to f at $\{x_i\}_{i=1}^{N+1}$ produces

$$\sum_{i=1}^{N} \int_{x_i}^{x_{i+1}} (\tau'' \pm p\tau')[f''-\tau'' \pm p(f'-\tau')]$$

$$= \sum_{i=1}^{N} \left\{(f'-\tau')(\tau'' \pm p\tau')\right\}_{x_i}^{x_{i+1}}$$

$$= (f'(b)-\tau'(b))\cdot(\tau''(b) \pm p\tau'(b)) - (f'(a)-\tau'(a))(\tau''(a) \pm p\tau'(a)).$$

Therefore

$$\int_a^b (f''\pm pf')^2 dx = \int_a^b [f''-\tau''\pm p(f'-\tau')]^2 dx + \int_a^b (\tau''\pm p\tau')^2 dx$$

$$+ 2\Big\{(f'(b)-\tau'(b))\cdot(\tau''(b) \pm p\tau'(b)) - (f'(a) - \tau'(a))\cdot(\tau''(a)\pm p\tau'(a))\Big\}.$$

Adding these two relations then leads to an extension of Holladay's theorem:

Theorem.

$$\int_a^b [(f'')^2+p^2(f')^2]\ dx = \int_a^b [(f''-\tau'')^2+ p^2(f'-\tau')^2]\ dx$$

$$+ \int_a^b [(\tau'')^2+p^2(\tau')^2]dx + 2\Big\{(f'(b)-\tau'(b))\cdot\tau''(b)-(f'(a)-\tau'(a))\cdot\tau''(a)\Big\}.$$

This leads us to define [32] the inner product $(f,g) \equiv \int_a^b f''g''\ dx + p^2 \int_a^b f'g'\ dx$ together with the induced pseudonorm

$$\|f\|_e \equiv \Big\{ \int_a^b [(f'')^2+ p^2(f')^2]\ dx\Big\}^{1/2}.$$

This allows us to restate our previous result as

$$\|f\|_e^2 = \|f-\tau\|_e^2+\|\tau\|_e^2+ 2\Big\{(f'(b)-\tau'(b))\cdot\tau''(b)-(f'(a)-\tau'(a))\cdot\tau''(a)\Big\}.$$

So that if $\tau''(a) = \tau''(b) = 0$, or $\tau'(a) = f'(a)$, $\tau'(b) = f'(b)$ we have $\|f\|_e^2 = \|f-\tau\|_2^2 + \|\tau\|_e^2$. We thus have arrived at the

following

Theorem (Minimum Norm Property). Let $g \in C^2[a,b]$ and interpolate to f at $\{x_i\}_{i=1}^{N+1}$. Then $\|g\|_e$ is a minimum when $g = \tau \ni \tau''(a) = \tau''(b) = 0$. If we restrict $g'(a) = f'(a)$ and $g'(b) = f'(b)$ then $\|g\|_e$ is a minimum when $g = \tau \ni \tau'(a) = f'(a)$, $\tau'(b) = f'(b)$. In both cases this τ is unique.

Proof (of uniqueness): Suppose there is also a $\bar{\tau}(x)$ with the minimum property. Then

$$\|\bar{\tau}\|_e^2 = \|\bar{\tau}-\tau\|_e^2 + \|\tau\|_e^2$$

and likewise

$$\|\tau\|_e^2 = \|\tau-\bar{\tau}\|_e^2 + \|\bar{\tau}\|_e^2 .$$

Therefore

$$\|\bar{\tau}-\tau\|_e^2 = \|\bar{\tau}\|_e^2 - \|\tau_e\|^2 \geq 0$$

and

$$\|\tau-\bar{\tau}\|_e^2 = \|\tau\|_e^2 - \|\bar{\tau}\|_e^2 \geq 0 \Rightarrow$$

$$\|\tau-\tau\|_e^2 = \int_a^b [(\tau''-\bar{\tau}'')^2 + p^2(\tau'-\bar{\tau}')^2]\, dx = 0$$

$$\Rightarrow \quad \tau''-\bar{\tau}'' = 0 \Rightarrow \tau - \bar{\tau} = Ax + B.$$

But $(\tau-\bar{\tau})(a) = (\tau-\bar{\tau})(b) = 0 \Rightarrow A = B = 0$. Therefore $\tau \equiv \bar{\tau}$.

Q.E.D.

This leads us to the

Theorem (Best Approximation Property). Let f have an absolutely continuous first derivative and a square integrable second derivative. Let τ be the interpolatory exponential spline with $\tau'(a) = f'(a)$ and $\tau'(b) = f'(b)$. Let $\bar{\tau}$ be any exponential spline with knots at $\{x_i\}_{i=1}^{N+1}$ and tension $p(x)$. Under these conditions τ is the best approximation by these exponential splines. I.e. $\|f-\bar{\tau}\|_e \geq \|f-\tau\|_e$.

Proof: Let $g \equiv f-\bar{\tau}$ and $\hat{\tau} = \tau-\bar{\tau}$. Hence $\hat{\tau}$ is an exponential spline $\ni \hat{\tau}(x_i) = \tau(x_i)-\bar{\tau}(x_i) = f(x_i)-\bar{\tau}(x_i) = g(x_i)$ and $\hat{\tau}'(a) = \tau'(a)-\bar{\tau}'(a) = f'(a)-\bar{\tau}'(a) = g'(a)$ and $\hat{\tau}'(b) = \tau'(b)-\bar{\tau}'(b) = f'(b)-\bar{\tau}'(b) = g'(b)$. Our Extended Holladay's Theorem $\Rightarrow$

$$\|g\|_e^2 = \|g-\hat{\tau}\|_e^2 + \|\hat{\tau}\|_e^2$$

$$\Rightarrow \|f-\bar{\tau}\|_e^2 = \|f-\bar{\tau}-\tau+\bar{\tau}\|_e^2 + \|\tau-\bar{\tau}\|_e^2$$

or

$$\|f-\bar{\tau}\|_e^2 = \|f-\tau\|_e^2 + \|\tau-\bar{\tau}\|_e^2 \ . \qquad \text{Q.E.D.}$$

Note that we have strict inequality unless $\|\tau-\bar{\tau}\|_e^2 = 0 \Rightarrow \tau'' = \bar{\tau}''$ and $\tau' = \bar{\tau}' \Rightarrow \bar{\tau} = \tau+Ax+B$ $\Rightarrow \bar{\tau}' = \tau'+A \Rightarrow A = 0$, i.e. τ is unique up to an additive constant.

I.8. Convergence Results

In this section we establish rates of convergence for the exponential spline in the limit of vanishing mesh width. Convergence rates for higher derivatives are also given for functions possessing a certain degree of smoothness.

We begin by studying the proximity of cubic and exponential splines with identical end conditions. In what follows $\|\cdot\|$ is the max-norm for continuous functions,

$$h = \max_i h_i \ , \quad \text{and} \quad p_{max} = \max_i p_i \ .$$

The following result has been obtained by Pruess [35].

Theorem 1.

$$\|D^i(s-\tau)\| \leq (26/3)\ p_{max}^2\ h^{4-i} \max_j |s_j''| \quad (i=0,1,2),$$

where, as before, s and τ are the cubic and exponential splines, respectively.

We present a companion result.

Theorem 2.

$$\|D^i(s-\tau)\| \leq (26/3)\ p_{max}^2\ h^{4-i} \max_j |\tau_j''| \quad (i=0,1,2).$$

Proof: Define $\delta \equiv s-\tau$. We treat the case of specified end slopes. We then have

$$\delta_1'' + \frac{1}{2}\,\delta_2'' = \left[\frac{3d_1}{h_1} - 1\right]\tau_1'' + \left[\frac{3e_1}{h_1} - \frac{1}{2}\right]\tau_2''$$

$$\frac{h_{i-1}}{2(h_{i-1}+h_i)}\,\delta''_{i-1} + \delta''_i + \frac{h_i}{2(h_{i-1}+h_i)}\,\delta''_{i+1}$$

$$= \left[\frac{6e_{i-1}-h_{i-1}}{2(h_{i-1}+h_i)}\right]\tau''_{i-1} + \left[\frac{3(d_{i-1}+d_i)}{h_{i-1}+h_i} - 1\right]\tau''_i + \left[\frac{6e_i-h_i}{2(h_{i-1}+h_i)}\right]\tau''_{i+1} \qquad (i=2,\ldots,N)$$

$$\frac{1}{2}\,\delta''_N + \delta''_{N+1} = \left[\frac{3e_N}{h_N} - \frac{1}{2}\right]\tau''_N + \left[\frac{3d_N}{h_N} - 1\right]\tau''_{N+1} .$$

Now let $\theta_i \equiv p_i h_i$. Power series expansions then verify the identities

$$(1) \quad 0 \le \frac{3e_i}{h_i} - \frac{1}{2} \le -\frac{\theta_i^2}{2}$$

$$(2) \quad 0 \le \frac{3d_i}{h_i} - 1 \le -\frac{\theta_i^2}{2}$$

$$(3) \quad 0 \le \frac{6e_i-h_i}{2(h_{i-1}+h_i)} \le -\frac{\theta_i^2}{2}$$

$$(4) \quad 0 \le \frac{6e_{i-1}-h_{i-1}}{2(h_{i-1}+h_i)} \le -\frac{\theta_{i-1}^2}{2}$$

$$(5) \quad 0 \le \frac{3(d_{i-1}+d_i)}{h_{i-1}+h_i} - 1 \le -p_{max}^2\, h^2$$

$$(6) \quad 0 < e_i/d_i \le \frac{1}{2}$$

$$(7) \quad 0 < \frac{e_{i-1}+e_i}{d_{i-1}+d_i} \le \frac{1}{2} .$$

Next let $\Delta \equiv [\delta''_1,\ldots,\delta''_{N+1}]^T$. Then we have

$$(I+E)\Delta = r$$

with the straightforward definitions. We thus have

$$\|E\|_\infty = \frac{1}{2} \quad \text{and} \quad \|r\|_\infty = \max\Big(|r_1|, \max_{1\le i\le N}|r_i|, |r_{N+1}|\Big)$$

However, using the above bounds, we have

$$\left.\begin{aligned} |r_1| &\le p_{max}^2 h^2 \max_i |\tau_i''| \\ |r_{N+1}| &\le p_{max}^2 h^2 \max_i |\tau_i''| \\ |r_i| &\le 2\, p_{max}^2 h^2 \max_i |\tau_i''| \end{aligned}\right\} \Rightarrow$$

$$\|r\|_\infty \le 2\, p_{max}^2 h^2 \max_i |\tau_i''| \quad \Rightarrow$$

$$\|\Delta\|_\infty \le \|(I+E)^{-1}\|_\infty \|r\|_\infty \le \|r\|_\infty/(1-\|E\|_\infty) \le 4\, p_{max}^2 h^2 \max_i |\tau_i''|.$$

We can now use this to attain our goal. Specifically, $\delta(x) \in C^2[a,b]$ and the conditions $\delta(x_i) = \delta(x_{i+1}) = 0$ $(i = 1,\ldots,N)$ together with the Mean Value Theorem $\Rightarrow \exists \xi \in (x_i, x_{i+1}) \ni \delta'(\xi_i) = 0 \Rightarrow \delta(x) = \delta(x)-\delta(x_i) = \int_{x_i}^{x} \delta'(t)\, dt$ and $\delta'(x) = \delta'(x)-\delta'(\xi_i) = \int_{\xi_i}^{x} \delta''(t)\, dt$.

However,

$$\delta''(x) = s''(x)-\tau''(x) = s_{i+1}'' \frac{x-x_i}{h_i} + s_i'' \frac{x_{i+1}-x}{h_i} - \tau_{i+1}'' \cdot \frac{\sinh p_i(x-x_i)}{\sinh p_i h_i} - \tau_i'' \cdot \frac{\sinh p_i(x_{i+1}-x)}{\sinh p_i h_i}$$

$$= \delta_{i+1}'' \frac{x-x_i}{h_i} + \delta_i'' \cdot \frac{x_{i+1}-x}{h_i} + \tau_{i+1}'' \left[\frac{x-x_i}{h_i} - \frac{\sinh p_i(x-x_i)}{\sinh p_i h_i}\right]$$

$$+ \tau_i'' \left[\frac{x_{i+1}-x}{h_i} - \frac{\sinh p_i(x_{i+1}-x)}{\sinh p_i h_i}\right] .$$

Each of the bracketed terms can be expanded to reveal that they are bounded by $p_i^2 h_i^2 / 3$. Hence

$$|\delta''(x)| \le |\delta_{i+1}''| + |\delta_i''| + |\tau_{i+1}''| \cdot \frac{p_i^2 h_i^2}{3} + |\tau_i''| \cdot \frac{p_i^2 h_i^2}{3}$$

$$\le 2 \cdot 4 \cdot p_{max}^2 h^2 \max_i |\tau_i''| + 2 \cdot \max_i |\tau_i''| \cdot \frac{p_{max}^2 h^2}{3} = \frac{26}{3}\, p_{max}^2 h^2 \max_i |\tau_i''|$$

so that

$$\|\delta''(x)\| \leq \frac{26}{3} p_{max}^2 h^2 \max_i |\tau_i''| \Rightarrow$$

$$\|\delta'(x)\| \leq \frac{26}{3} p_{max}^2 h^3 \max_i |\tau_i''| \Rightarrow$$

$$\|\delta(x)\| \leq \frac{26}{3} p_{max}^2 h^4 \max_i |\tau_i''| . \qquad \text{Q.E.D.}$$

Note that $\max_i|\tau_i''|$ is bounded as shown by the following argument. Divide both sides of each equation in the tridiagonal system by the diagonal term. The resulting system may be written as $(I+E)\tau'' = \bar{b}$ with the obvious definitions. Hence $\|\tau''\|_\infty \leq \|\bar{b}\|_\infty / (1-\|E\|_\infty)$. Now (6) and (7) above imply that $\|E\|_\infty \leq 1/2$ and thus that $\|\tau''\|_\infty \leq 2\|\bar{b}\|_\infty$. $2\|\bar{b}\|_\infty$ is bounded as $p_i h_i \to 0$ and in fact $\to 3\|f''\|$. These theorems may be used to obtain results for $\|D^i(f-\tau)\|$ from known bounds for $\|D^i(f-s)\|$. E.g.

Corollary 1. If $f \in C^4[a,b]$ then $\exists\, k(p_{max}, \|D^2 f\|, \|D^4 f\|)$ independent of $h \ni \|D^i(f-\tau)\| \leq k h^{4-i}$ $(i = 0,1,2)$.

We now take up the convergence of the third derivative. We have on $[x_i, x_{i+1}]$

$$\delta'''(x) = \frac{\delta_{i+1}''-\delta_i''}{h_i} + \tau_{i+1}''\left[\frac{1}{h_i} - \frac{p_i}{s_i}\cosh p_i(x-x_i)\right] + \tau_i''\left[-\frac{1}{h_i} + \frac{p_i}{s_i}\cosh p_i(x_{i+1}-x)\right]$$

$$\Rightarrow |\delta'''(x)| \leq \frac{2}{h_i}\|\Delta\|_\infty + \left[\frac{p_i c_i}{s_i} - \frac{1}{h_i}\right] \cdot 2 \max_i |\tau_i''|.$$

However,

$$\frac{p_i c_i}{s_i} - \frac{1}{h_i} \leq \frac{p_{max}^2 h_i}{3} .$$

Therefore

$$|\delta'''(x)| \leq 8\, p_{max}^2 \frac{h^2}{h_i} \max_i |\tau_i''| + \frac{2}{3} p_{max}^2 h_i \cdot \max_i |\tau_i''| \Rightarrow$$

$$|\delta'''(x)| \leq \frac{26}{3} p_{max}^2 h \max_i |\tau_i''| \cdot \frac{h}{h_{min}} .$$

Known results for the cubic spline now establish the O(h) convergence of the third derivative.

I.9. Shape Preserving Interpolation

In this section we review the results of Pruess [35,37] concerning the behavior of exponential splines in the limit of infinite tension. Our main interest here is in the shape preservation properties of exponential spline interpolation. In what follows we assume a fixed set of data and uniformly bounded tension ratios, $p_{max}/p_{min} \leq \rho$.

Let $\lambda(x)$ denote the linear spline of interpolation. Then we have

Theorem 1. Given a sequence of exponential splines $\ni p_i \to \infty$ for some i; then in any closed subinterval of (x_i,x_{i+1}), $\tau''(x) \to 0$ and $\tau'(x) \to \lambda'(x)$ uniformly while $\tau(x) \to \lambda(x)$ uniformly in $[x_i,x_{i+1}]$.

This theorem gives us hope that we can produce co-convex and co-monotone interpolants using exponential splines with sufficiently high tension. The fulfillment of this expectation is the subject of

Theorem 2. If b_i,b_{i+1} are positive (negative), then for p_{i-1}, p_i, p_{i+1} sufficiently large $\tau''(x)$ is positive (negative) in $[x_i,x_{i+1}]$. If $\lambda'(x)$ is positive (negative) in (x_{i-1},x_{i+2}), then for p_{i-1}, p_i, p_{i+1} sufficiently large $\tau'(x)$ is positive (negative) in $[x_i,x_{i+1}]$.

We note that at the knots we have

$$\tau'(x_i) \to \frac{1}{2}\left[\frac{f_i - f_{i-1}}{h_{i-1}} + \frac{f_{i+1} - f_i}{h_i}\right]$$

$$\tau''(x_i) \to \frac{b_i}{d_{i-1} + d_i} \sim \min(p_{i-1}, p_i) \ .$$

The cubic spline many times exhibits unwanted oscillations due to the emergence of overshoots and/or extraneous inflection points. The above results assure us that the exponential spline can remedy this situation for appropriately chosen tension parameters. We subsequently construct such shape preserving interpolants.

I.10. Cardinal Spline Basis

The form of the Lagrange interpolating polynomial suggests the following basis for the exponential splines with given knots and tensions [6,7,14,34].

$$C_0'(x_1) = 1,\ C_0(x_j) = 0 \quad (j=1,\ldots,N+1),\ C_0'(x_{N+1}) = 0$$

$$C_i'(x_1) = 0,\ C_i(x_j) = \delta_{ij} \quad (j=1,\ldots,N+1),\ C_i'(x_{N+1}) = 0;$$

$$i = 1,\ldots,N+1;$$

$$C_{N+2}'(x_1) = 0,\ C_{N+2}(x_j) = 0 \quad (j=1,\ldots,N+1),\ C_{N+2}'(x_{N+1}) = 1.$$

These conditions uniquely define the cardinal spline basis $\{C_i(x)\}_{i=0}^{N+2}$. Clearly

$$\tau(x) = f_1' \cdot C_0(x) + \sum_{i=1}^{N+1} f_i \cdot C_i(x) + f_{N+1}' \cdot C_{N+2}(x) .$$

In what follows we restrict ourselves to uniform mesh and tension.

We begin by generalizing the arguments of Birkhoff and de Boor [7]. Let $t(x)$ be the function on $[x_1, x_{N+1}]$ of the form

$$t(x) = a + bx + ce^{px} + de^{-px}$$

satisfying

$$t(x_1)=f(x_1),\ t(x_{N+1})=f(x_{N+1}),\ t'(x_1)=f'(x_1),\ t'(x_{N+1})=f'(x_{N+1}).$$

Then given the exponential spline fit to $g(x) \equiv f(x) - t(x)$ with zero slope end conditions, we can simply add $t(x)$ to it to obtain the fit for f. Hence, WLOG we consider only functions $\ni f(x_1) = f(x_{N+1}) = f'(x_1) = f'(x_{N+1}) = 0$. Hence we need only consider

$$\tau(x) = \sum_{i=2}^{N} f_i \cdot C_i(x) \ .$$

Lemma 1. Any function of the form $e(x) = a+bx+ce^{px}+de^{-px}$ which satisfies $e(0) = 0$ and $e(h) = 0$ also satisfies

$$\begin{bmatrix} e'(h) \\ e''(h) \end{bmatrix} = \begin{bmatrix} \frac{phc-s}{ph-s} & \frac{2(1-c)+phs}{p(ph-s)} \\ \frac{p^2hs}{ph-s} & \frac{phc-s}{ph-s} \end{bmatrix} \begin{bmatrix} e'(0) \\ e''(0) \end{bmatrix} .$$

Proof:

$$e(x) = -\frac{e_0''}{p^2} + x\cdot\left[\frac{e_0''(1-c)-e_0'(ps)}{p(ph-s)}\right] + \sinh px\cdot\left[\frac{e_0''(c-1)+p^2he_0'}{p^2(ph-s)}\right] + \cosh px\cdot\left[\frac{e_0''}{p^2}\right]$$

Q.E.D.

Corollary 1. For $i \neq j+1, j$, $C_i(x)$ satisfies

$$\begin{bmatrix} C_i'(x_{j+1}) \\ C_i''(x_{j+1}) \end{bmatrix} = \begin{bmatrix} \frac{phC-s}{ph-S} & \frac{2(1-c)+phS}{p(ph-s)} \\ \frac{p^2hS}{ph-S} & \frac{phC-S}{phS} \end{bmatrix} \begin{bmatrix} C_i'(x_j) \\ C_i''(x_j) \end{bmatrix} .$$

Proof: $C_i(x)$ with i so restricted satisfies the hypotheses of the lemma. Q.E.D.

Note. All the above matrix elements are < 0, with, in particular,

$$-2 \leq \frac{\text{phc-s}}{\text{ph-s}} < 0 .$$

Corollary 2. For $i = 2,\ldots,N$, $C_i(x)$ satisfies

$$C_i'(x_j)C_i''(x_j) \geq 0 , \quad \text{for} \quad j < i$$

$$C_i'(x_j)C_i''(x_j) \leq 0 , \quad \text{for} \quad j > i .$$

Proof: For $j = 0$ we have $C_i'(x_j) = 0$; for $j = 1,\ldots,i-1$ the result is a consequence of the negativity of the above matrix; for $j > i$ changing x to -x reverses the sign of $C_i'(x)C_i''(x)$. Q.E.D.

Corollary 3. For $i = 2,\ldots,N$, $C_i(x)$ satisfies

$$|C_i'(x_j)| < \frac{1}{2} |C_i'(x_{j+1})| , \quad j < i-1$$

$$|C_i'(x_{j+1})| < \frac{1}{2} |C_i'(x_j)| , \quad j > i .$$

Proof: For $j < i-1$ the inequality follows from Corollaries 1 and 2 with strict inequality the consequence of $C_i''(x_1) \neq 0$ (otherwise $C_i(x) \equiv 0$) which implies that $C_i''(x_j) \neq 0$ for $j < i$; for $j > i$ we appeal to symmetry about x_i. Q.E.D.

Note. This implies an exponential decay of $|C_i(x)|$ away from x_i.

The following results can be established by arguments nearly identical to those in [7]. Hence their proofs are omitted.

Lemma 2. Let $\tau(x)$ be any exponential spline with knots $\{x_i\}_{i=1}^{N+1}$ (uniform mesh and tension), which satisfies

$$\tau_{i-1} = \tau_{i+1} = 0 \ , \quad \tau_i = v > 0 \ , \ \tau'_{i-1}\cdot\tau''_{i-1} \geq 0, \ \tau'_{i+1}\cdot\tau''_{i+1} \leq 0.$$

Then

$$\tau''_i < 0, \ \tau''_{i-1} \geq 0, \ \tau'_{i+1} \leq 0 \ , \text{ and } \tau(x) \geq 0 \text{ on } [x_{i-1},x_{i+1}].$$

Lemma 3. Let $\tau(x)$ be such that

$$\tau_{i-1} = \tau_i = \tau_{i+1} = 0 \ , \ \tau''_{i-1} \leq 0 \ , \ \tau'_{i+1} \geq 0 \ .$$

Then $\tau(x) \geq 0$ in $[x_{i-1},x_i]$.

Corollary For $i = 1,\ldots,n-1$ we have

(a) $|C_i(x)| \leq |C'_i(x_j)|\cdot h$ on $[x_j,x_{j+1}]$, $j > i$

(b) $|C_i(x)| \leq |C'_i(x_j)|\cdot h$ on $[x_{j-1},x_j]$, $j < i-1$.

Let us now consider the "natural" cardinal splines defined by the conditions

$$N_i(x_j) = \delta_{ij} \quad (j = 1,\ldots,N+1) \ ,$$
$$N''_i(x_1) = N''_i(x_{N+1}) = 0, \quad i = 1,\ldots,N+1.$$

Let $\bar{N}(x)$ be any exponential spline with $\bar{N}'_1 = \bar{N}''_{N+1} = 0$. Then the remaining $\{\bar{N}''_i\}_{i=2}^{N}$ are determined from

$$\begin{bmatrix} a & 1 & & & \\ 1 & a & 1 & & \\ & 1 & a & 1 & \\ & & 1 & a & \end{bmatrix} \begin{bmatrix} \bar{N}_2'' \\ \bar{N}_3'' \\ \vdots \\ \bar{N}_{N-1}'' \\ \bar{N}_N'' \end{bmatrix} = \begin{bmatrix} b_2/e \\ b_3/e \\ \vdots \\ b_{N-1}/e \\ b_N/e \end{bmatrix}$$

or

$$A\bar{N}'' = \bar{b}.$$

Here $a \equiv 2d/e$.

Now A can be factored as

$$A = LU$$

where

$$L \equiv \begin{bmatrix} \rho_2 & & & \\ 1 & & & \\ & \ddots & & \\ & & 1 & \rho_N \end{bmatrix} \quad ; \quad U \equiv \begin{bmatrix} 1 & 1/\rho_2 & & \\ & \ddots & & \\ & & & 1/\rho_{N-1} \\ & & & 1 \end{bmatrix}$$

with

$$\rho_2 = a$$

$$\rho_i = a - \frac{1}{\rho_{i-1}} \qquad (i = 3,\ldots,N).$$

It is evident that

$$\frac{2d}{e} = a = \rho_2 > \rho_3 > \ldots > \rho_{N-1} > \rho_N > \frac{a + \sqrt{a^2-4}}{2} =$$

$$= \frac{d}{e} + \sqrt{\left(\frac{d}{e}\right)^2 - 1} > 2 + \sqrt{3}$$

$$\Rightarrow \frac{e}{2d} < \frac{1}{\rho_i} < \frac{d}{e} - \sqrt{\left(\frac{d}{e}\right)^2 - 1} < 2 - \sqrt{3}\ .$$

In particular, letting $\bar{N}(x) = N_j(x)$ we have (after some lengthy computations)

$$\left.\begin{aligned}
\bar{N}_i'' &= -\frac{1}{\rho_i}\bar{N}_{i+1}'' \ ; && i = 2,\ldots,j-2\\
\bar{N}_i'' &= -\frac{1}{\rho_{N+1-i}}\bar{N}_{i-1}'' \ ; && i = j+2,\ldots,N\\
\bar{N}_{j-1}'' &= \frac{1}{\rho_{i-1}}\left[\frac{1}{he} - \bar{N}_j''\right]\\
\bar{N}_{j+1}'' &= \frac{1}{\rho_{N-j}}\cdot\left[\frac{1}{he} - \bar{N}_j''\right]\\
\bar{N}_j'' &= -\frac{1}{he}\left[\frac{2 + \frac{1}{\rho_{j-1}} + \frac{1}{\rho_{N-j}}}{4 - \frac{1}{\rho_{j-1}} - \frac{1}{\rho_{N-j}}}\right]
\end{aligned}\right\} \quad j \neq 1,2,N,N+1 .$$

We then have the following

Theorem. For $\bar{N}(x) = N_j(x)$; $j = 3,\ldots,N-1$ and either $x \in [x_i,x_{i+1}]$; $j+1 \leq i \leq N$ or $x \in [x_{i-1},x_i]$; $2 \leq i \leq j-1$ we have

$$|\bar{N}(x)| \leq \left(1 - \frac{1}{\sqrt{3}}\right)\cdot h^2 \cdot |N_i''| .$$

Proof: We take the case $x \in [x_{i-1},x_i]$; $2 \leq i \leq j-1$. Then $\bar{N}_{i-1}'' = \frac{-1}{\rho_{i-1}}\bar{N}_i''$ and

$$\bar{N}(x) = \frac{\bar{N}_i''}{p^2}\left\{\left[\frac{\sinh p(x-x_{i-1})}{\sinh ph} - \frac{1}{\rho_{i-1}}\cdot\frac{\sinh p(x_i-x)}{\sinh ph}\right]\right.$$

$$\left. - \left[\frac{x-x_{i-1}}{h} - \frac{1}{\rho_{i-1}}\quad\frac{x_i-x}{h}\right]\right\}$$

$$\Rightarrow \quad |\bar{N}(x)| \leq h^2\left(1 - \frac{1}{\sqrt{3}}\right)\cdot|N_i''| .$$

Similar considerations apply to the other case. Q.E.D.

Corollary. For the above case we have

$$|\bar{N}(x)| \leq \frac{2}{3} \cdot \frac{h}{e} (\sqrt{3} - 1)(2 - \sqrt{3})^{j-i}$$

with a similar result for the other case.

Proof: Our recurrence relations for $\{\bar{N}_k''\}_{k=2}^{j} \Rightarrow$

$$|\bar{N}_i''| \leq \frac{2}{\sqrt{3}\ he} \cdot (2 - \sqrt{3})^{j-i} . \qquad \text{Q.E.D.}$$

The cardinal spline formulation is not useful for calculations because of the need to calculate and store a basis function for each data point. It does however provide great insight into many numerical aspects of splines. For example, the above considerations allow us to determine the effect of a change in some data value, say $f_i \to f_i + \varepsilon_i$. We simply add the term $\varepsilon_i \cdot C_i(x)$ to the spline. Similarly, a study of $C_0(x)$ and $C_{N+2}(x)$ allows one to discuss the global effect of the end conditions.

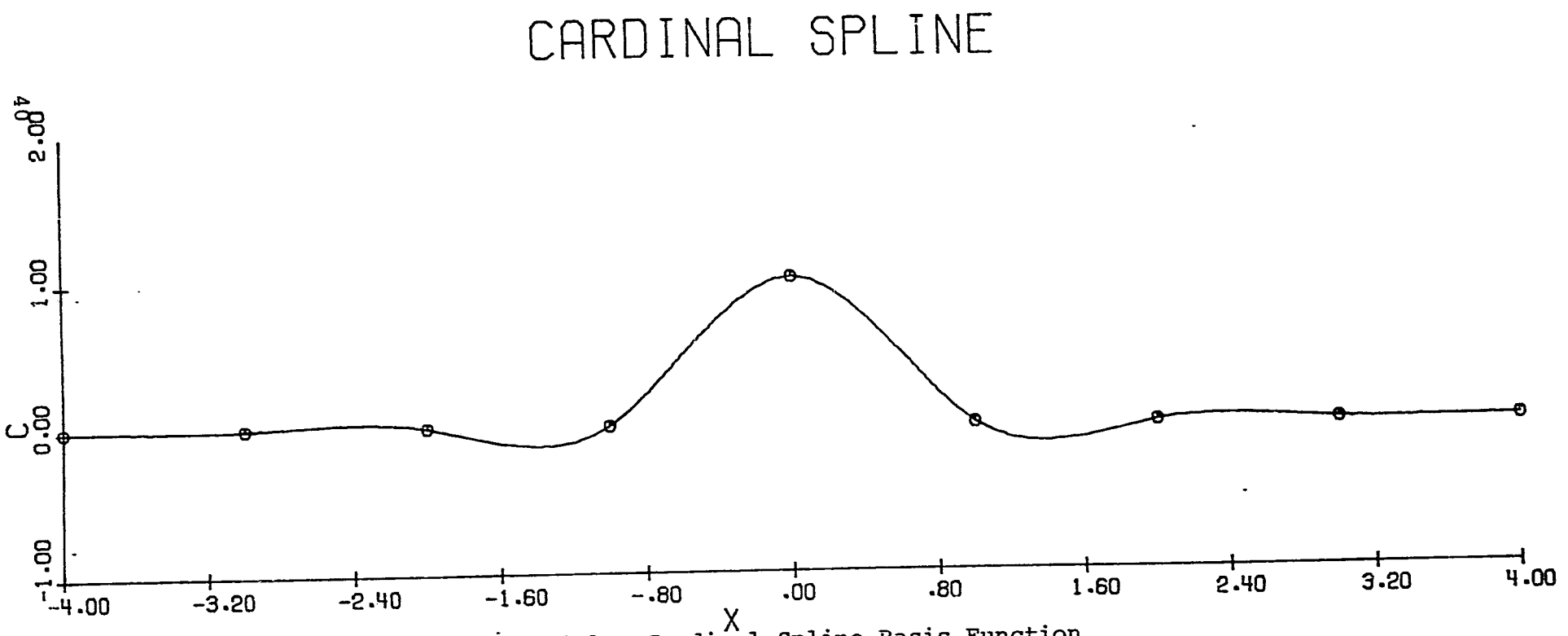


Figure I.10.1. Cardinal Spline Basis Function

I.11. B-Spline Basis

We now construct another basis for the space of exponential splines with given tensions. This basis will be constructed so as to have minimal support [5,9,34,36]. We have the following

Lemma. Any exponential spline, B(x), with a support of fewer than four intervals is identically zero (assuming that there are at least five knots).

Proof: At the end points of the support of B(x) we must have $B = B' = B'' = 0$ since $B(x) \in C^2$. These six conditions and the spline continuity constraints allow us to explicitly calculate B(x) for the cases of support of one, two and three intervals. In all cases, a direct calculation reveals that $B(x) \equiv 0$. If we assume uniqueness of the splines so defined we can obtain this result in the following more elegant fashion. (Just show that the required matrices are invertible for uniqueness.) For the case of one interval we have four undetermined coefficients and six end conditions, hence only the zero function satisfies these constraints. For two intervals we have eight coefficients, six end conditions and three continuity conditions and again the zero function is the solution. For three intervals we have 12 coefficients, six end conditions and six continuity conditions. Once again we lack the extra coefficient by which to specify a nonzero

function value. Finally, with four intervals we have sixteen coefficients, six end conditions and nine continuity constraints. We may use the extra degree of freedom to specify $B(x) \neq 0$ at some point. Q.E.D.

We now proceed to define such a "B-spline". To simplify the computations we restrict our attention to uniform mesh and tension. Given five points $\{x_0+ih\}_{i=-2}^{2}$ we require that

$$B(x_0-2h) = B'(x_0-2h) = B''(x_0-2h) = B(x_0+2h) = B'(x_0+2h) = B''(x_0+2h) = 0.$$

We normalize so that $B(x_0) = 1$. Taking our cue from the cubic B-spline we set $B'(x_0) = 0$, i.e. we take into account the symmetry inherent in the definition of $B(x)$. This allows us to solve for $B(x)$ on $[x_0, x_0+2h]$ and then reflect this about the line $x = x_0$.

We set

$$B(x) = \begin{cases} a_1+b_1(x-x_0)+c_1e^{p(x-x_0)}+d_1e^{-p(x-x_0)}, & x_0 \leq x \leq x_0+h \\ a_2+b_2(x-x_0-2h)+c_2e^{p(x-x_0-2h)}+d_2e^{-p(x-x_0-2h)}, & x_0+h \leq x \leq x_0+2h \end{cases}$$

$$\Rightarrow$$

$$B'(x) = \begin{cases} b_1+pc_1e^{p(x-x_0)}-pd_1e^{-p(x-x_0)}, & x_0 \leq x \leq x_0+h \\ b_2+pc_2e^{p(x-x_0-2h)}-pd_2e^{-p(x-x_0-2h)}, & x_0+h \leq x \leq x_0+2h \end{cases}$$

$$B''(x) = \begin{cases} p^2c_1e^{p(x-x_0)}+p^2d_1e^{-p(x-x_0)}, & x_0 \leq x \leq x_0+h \\ p^2c_2e^{p(x-x_0-2h)}+p^2d_2e^{-p(x-x_0-2h)}, & x_0+h \leq x \leq x_0+2h. \end{cases}$$

The end conditions and continuity conditions $\Rightarrow$

$$a_2 + c_2 + d_2 = 0$$

$$b_2 + pc_2 - pd_2 = 0$$

$$c_2 + d_2 = 0$$

$$a_1 + c_1 + d_1 = 1$$

$$b_1 + pc_1 - pd_1 = 0$$

$$a_1 + b_1 h + c_1 e^{ph} + d_1 e^{-ph} = a_2 - b_2 h + c_2 e^{-ph} + d_2 e^{ph}$$

$$b_1 + pc_1 e^{ph} - pd_1 e^{-ph} = b_2 + pc_2 e^{-ph} - pd_2 e^{ph}$$

$$c_1 e^{ph} + d_1 e^{-ph} = c_2 e^{-ph} + d_2 e^{ph}$$

$\Rightarrow$ $a_2 = 0$, $c_2 = -b_2/2p$, $d_2 = b_2/2p$, $b_2 = p/2(phc-s)$,

$$a_1 = \frac{phc}{phc-s}, \quad b_1 = \frac{p}{2} \cdot \left[\frac{c(c-1) + s^2}{(phc-s)(1-c)} \right],$$

$$c_1 = \frac{1}{4} \left[\frac{e^{-ph}(1-c)+s(e^{-ph}-1)}{(phc-s)(1-c)} \right], \quad d_1 = \frac{1}{4} \left[\frac{e^{ph}(c-1)+s(e^{ph}-1)}{(phc-s)(1-c)} \right] \Rightarrow$$

$$B(x) = \begin{cases} a_1 + b_1(x-x_0) + c_1 e^{p(x-x_0)} + d_1 e^{-p(x-x_0)}, & x_0 \le x \le x_0+h \\ b_2[(x-x_0-2h) - \frac{1}{p}\sinh p(x-x_0-2h)], & x_0+h \le x \le x_0+2h \end{cases}$$

$$B'(x) = \begin{cases} b_1 + pc_1 e^{p(x-x_0)} - pd_1 e^{-p(x-x_0)}, & x_0 \le x \le x_0+h \\ b_2[1-\cosh p(x-x_0-2h)], & x_0+h \le x \le x_0+2h \end{cases}$$

$$B''(x) = \begin{cases} p^2 c_1 e^{p(x-x_0)} + p^2 d_1 e^{-p(x-x_0)}, & x_0 \le x \le x_0+h \\ -b_2 p \sinh p(x-x_0-2h), & x_0+h \le x \le x_0+2h. \end{cases}$$

Hence,

$$B''(x_0) = \frac{-p^2 s}{phc-s} \to \frac{-3}{h^2} \quad \text{as} \quad p \to 0$$

$$B(x_0+h) = \frac{1}{2} \cdot \frac{s-ph}{phc-s} \to \frac{1}{4} \quad \text{as} \quad p \to 0$$

$$B'(x_0+h) = \frac{p}{2} \cdot \frac{1-c}{phc-s} \to -\frac{3}{4h} \quad \text{as} \quad p \to 0$$

$$B''(x_0+h) = \frac{p^2 s}{2(phc-s)} \to \frac{3}{2h^2} \quad \text{as} \quad p \to 0$$

which are the correct cubic spline limits.

Now add the points x_{-2}, x_{-1}, x_0, x_{N+2}, x_{N+3}, x_{N+4} to our set of knots in the obvious fashion. Denote by $B_i(x)$ the B-spline centered at x_i $(i = 0,\ldots,N+2)$. All the B_i are simply translates of one another. We now show in a constructive fashion that $\{B_i(x)\}_{i=0}^{N+2}$ forms a basis for the exponential splines on this mesh with tension p.

Let

$$\tau(x) = \sum_{i=0}^{N+2} a_i B_i(x) ,$$

then

$$\tau(x_j) = \sum_{i=0}^{N+2} a_i B_i(x_j) = \frac{s-ph}{2(phc-s)} a_{j-1} + a_j + \frac{s-ph}{2(phc-s)} a_{j+1}$$
$$j = 1,\ldots,N+1.$$

Also

$$\tau'(x) = \sum_{i=0}^{N+2} a_i B_i'(x) \Rightarrow$$

$$\tau'(x_j) = \sum_{i=0}^{N+2} a_i B_i'(x_j) = \frac{p(1-c)}{2(phc-s)} a_{j-1} + \frac{p(c-1)}{2(phc-s)} a_{j+1}.$$

Finally,

$$\tau''(x) = \sum_{i=0}^{N+2} a_i B_i''(x) \Rightarrow$$

$$\tau''(x_j) = \sum_{i=0}^{N+2} a_i B_i''(x_j) = \frac{p^2 s}{2(phc-s)} a_{j-1} - \frac{p^2 s}{phc-s} a_j + \frac{p^2 s}{2(phc-s)} a_{j+1}.$$

Let's first take slope end conditions. We then have

$$\frac{p(1-c)}{2(phc-s)}\, a_0 + \frac{p(c-1)}{2(phc-s)}\, a_2 = f_1'$$

$$\frac{s-ph}{2(phc-s)}\, a_{j-1} + a_j + \frac{s-ph}{2(phc-s)}\, a_{j+1} = f_j \;;\; j = 1,\ldots,N+1$$

$$\frac{p(1-c)}{2(phc-s)}\, a_N + \frac{p(c-1)}{2(phc-s)}\, a_{N+2} = f_{N+1}'$$

or, in matrix form,

$$\begin{bmatrix} \frac{p(1-c)}{2(phc-s)} & 0 & \frac{p(c-1)}{2(phc-s)} & & \\ \frac{s-ph}{2(phc-s)} & 1 & \frac{s-ph}{2(phc-s)} & & \\ & \frac{s-ph}{2(phc-s)} & 1 & \frac{s-ph}{2(phc-s)} & \\ & \frac{p(1-c)}{2(phc-s)} & 0 & \frac{p(c-1)}{2(phc-s)} & \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ \vdots \\ a_{N+1} \\ a_{N+2} \end{bmatrix} = \begin{bmatrix} f_1' \\ f_1 \\ \vdots \\ f_{N+1} \\ f_{N+1}' \end{bmatrix}.$$

This system is diagonally dominant and hence $\tau(x)$ is uniquely expressible in this form.

Note that we can reduce this system to tridiagonal form. Multiply the first equation by $\frac{s-ph}{p(c-1)}$ and add it to the second; multiply the last equation by $\frac{s-ph}{p(1-c)}$ and add it to the equation above it.

Now take the case of second derivative end conditions. We then have

$$\frac{p^2s}{2(phc-s)}\, a_0 - \frac{p^2s}{phc-s}\, a_1 + \frac{p^2s}{2(phc-s)}\, a_2 = f_1''$$

$$\frac{s-ph}{2(phc-s)}\, a_{j-1} + a_j + \frac{s-ph}{2(phc-s)}\, a_{j+1} = f_j \;;\; j=1,\ldots,N+1$$

$$\frac{p^2s}{2(phc-s)}\, a_N - \frac{p^2s}{phc-s}\, a_{N+1} + \frac{p^2s}{2(phc-s)}\, a_{N+2} = f_{N+1}'',$$

or, in matrix form,

$$\begin{bmatrix} \frac{p^2s}{2(phc-s)} & -\frac{p^2s}{phc-s} & \frac{p^2s}{2(phc-s)} & & & \\ \frac{s-ph}{2(phc-s)} & 1 & \frac{s-ph}{2(phc-s)} & & & \\ & \frac{s-ph}{2(phc-s)} & 1 & \frac{s-ph}{2(phc-s)} & & \\ & & \frac{p^2s}{2(phc-s)} & \frac{-p^2s}{phc-s} & \frac{p^2s}{2(phc-s)} & \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ \vdots \\ a_{N+1} \\ a_{N+2} \end{bmatrix} = \begin{bmatrix} f''_1 \\ f_1 \\ \vdots \\ f_{N+1} \\ f''_{N+1} \end{bmatrix}.$$

Again diagonal dominance implies a unique representation of this form for $\tau(x)$.

We can also reduce this sytem to tridiagonal form. Multiply the first equation by $\frac{ph-s}{p^2s}$ and add it to the second; multiply the last equation by $\frac{ph-s}{p^2s}$ and add it to the equation above it.

Note that in either case no more than four basis functions contribute to the value of $\tau(x)$ at any point.

In addition to the value of the B-spline representation as a theoretical tool and a computational device, it is also of much utility to interactive computer graphics. This utility stems from the local nature of the basis. A spline created as a linear combination of B-splines can be displayed with the user then being able to alter $\{a_i\}_{i=0}^{N+2}$. As each change affects only four intervals the user can experiment and design a pleasing curve.

There is even a more subtle aspect of such interactive design. Recall that

$$\tau_j = \frac{s-ph}{2(phc-s)}\, a_{j-1} + a_j + \frac{s-ph}{2(phc-s)}\, a_{j+1} \; ; \; j=k-1,k,k+1.$$

If we now change τ_k to $\tau_k+\delta$ while simultaneously changing τ_{k-1} to $\tau_{k-1} + \frac{s-ph}{2(phc-s)}\,\delta$ and τ_{k+1} to $\tau_{k+1} + \frac{s-ph}{2(phc-s)}\,\delta$ then this simply amounts to redefining $a_k \to a_k+\delta$. Hence, we can move groups of points without having to compute a new spline!

It is anticipated that many other aspects of cubic B-spline theory can be extended to this more general context.

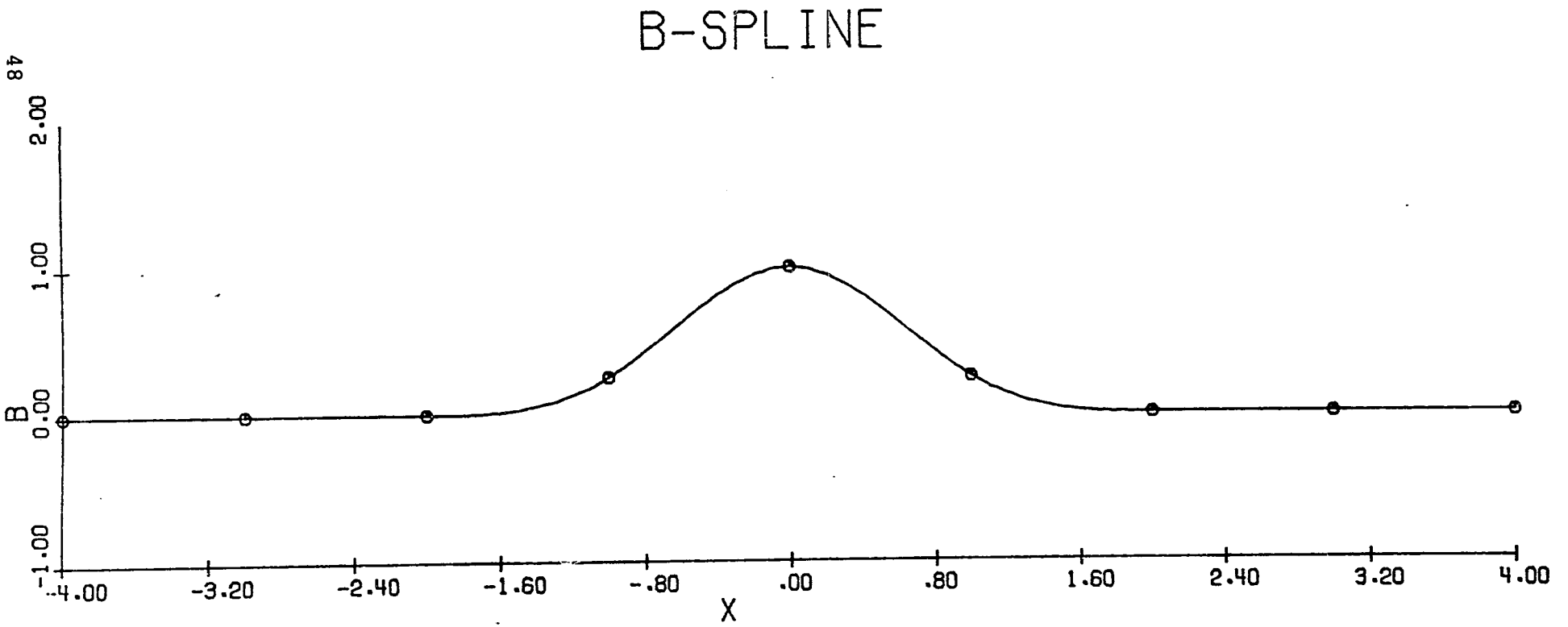


Figure I.11.1. B-Spline Basis Function

Part II: Computation of Exponential Splines

II.1. Overview

In this second part we concern ourselves with matters related to the computation of exponential splines. First we detail the direct solution of the spline equations including a discussion of matrix condition numbers. The iterative solution of the spline equations is then considered. Next is a thorough discussion of spline end conditions. We then produce a power series representation that avoids the possible loss of accuracy in the evaluation of the hyperbolic functions for small arguments. This is followed by the development of tension parameter selection algorithms that produce the shape preserving interpolants previously shown to exist. The periodic exponential spline is thenintroduced and its computational aspects are surveyed. A variety of numerical considerations next occupy our attention. Finally, a sequence of examples is presented which demonstrates the inherent superiority of exponential splines to cubic splines.

II.2. Direct Solution of the Spline Equations

As we have seen, the formulation of the spline equations in terms of either first or second derivatives leads to a (symmetric) tridiagonal system. These equations share the property with their cubic spline counterparts that they can be solved in O(N) arithmetic operations.

The following algorithm [2] may be used to solve for τ'_i or τ''_i $(i = 1,\ldots,N+1)$: let

$$\begin{aligned} PK_1 &= DIAG_1 \\ Q_1 &= -E_1/PK_1 \\ U_1 &= B_1/PK_1 \end{aligned}$$

then set

$$\left.\begin{aligned} PK_i &= E_{k-1} \cdot Q_{i-1} + DIAG_i \\ Q_i &= -E_i/PK_i \\ U_i &= (B_i - E_{i-1}\cdot U_{i-1})/PK_i \end{aligned}\right\} \qquad (i = 2,\ldots,N+1)$$

and finally

$$\begin{aligned} \tau''_{N+1} &= U_{N+1} \\ \tau''_i &= Q_i \cdot \tau''_{i+1} + U_i \,, \qquad (i = N,\ldots,1). \end{aligned}$$

In the above, $DIAG_i$ $(i=1,\ldots,N+1)$ are the diagonal elements, E_i $(i=1,\ldots,N)$ are the off-diagonal elements, $E_{N+1} = 0$, and B_i are the elements of the right hand side vector.

A quantity of interest in the present context is the condition number of the spline matrix [35]. Let us first take the first derivative formulation. Gerschgorin's theorem reveals that the largest eigenvalue of A will be

less than or equal to

$$\frac{d_{i-1}}{d_{i-1}^2-e_{i-1}^2}+\frac{d_i}{d_i^2-e_i^2}+\frac{e_{i-1}}{d_{i-1}^2-e_{i-1}^2}+\frac{e_i}{d_i^2-e_i^2}$$

$$=\frac{p_{i-1}^2 s_{i-1} h_{i-1}}{p_{i-1}h_{i-1}c_{i-1}-2s_{i-1}+p_{i-1}h_{i-1}}+\frac{p_i^2 s_i h_i}{p_i h_i c_i-2s_i+p_i h_i}$$

and that the smallest eigenvalue of A will be greater than or equal to

$$\frac{d_{i-1}}{d_{i-1}^2-e_{i-1}^2}+\frac{d_i}{d_i^2-e_i^2}-\frac{e_{i-1}}{d_{i-1}^2-e_{i-1}^2}-\frac{e_i}{d_i^2-e_i^2}=\frac{p_{i-1}s_{i-1}}{c_{i-1}-1}+\frac{p_i s_i}{c_i-1} \quad .$$

Hence,

$$\|A\|_2 \leq 2 \max_i \frac{p_i^2 s_i h_i}{p_i h_i c_i-2s_i+p_i h_i}$$

and

$$\|A^{-1}\|_2 \leq \frac{1}{2} \max_i \frac{c_i-1}{p_i s_i} \quad .$$

For the second derivative formulation the same line of reasoning yields

$$\|A\|_2 \leq 2 \max_i \frac{c_i-1}{p_i s_i}$$

and

$$\|A^{-1}\|_2 \leq \frac{1}{2} \max_i \frac{p_i^2 s_i h_i}{p_i h_i c_i-2s_i+p_i h_i} \quad .$$

Power series expansions establish

$$\frac{c_{i-1}}{p_i s_i} \leq \frac{h_i}{2}$$

$$\frac{p_i^2 h_i s_i}{p_i h_i c_i-2s_i+p_i h_i} \leq \frac{6}{h_i} \quad .$$

We thus have the following situation:

(i) first derivative formulation -

$$\|A\|_2 \leq \frac{12}{h_{min}}$$

$$\|A^{-1}\|_2 \leq \frac{h}{4}$$

(ii) second derivative formulation -

$$\|A\|_2 \leq h$$

$$\|A^{-1}\|_2 \leq \frac{3}{h_{min}}.$$

Hence in both cases we have the matrix condition number

$$K(A) \equiv \|A\|_2\|A^{-1}\|_2 \leq \frac{3h}{h_{min}} .$$

II.3. Iterative Solution of the Spline Equations

The spline equations in terms of second derivatives with slope end conditions are

$$\begin{bmatrix} d_1 & e_1 & & & & \\ e_1 & d_1+d_2 & e_2 & & & \\ & \ddots & \ddots & \ddots & & \\ & & e_{N-1} & d_{N-1}+d_N & e_N & \\ & & & e_N & d_N & \end{bmatrix} \begin{bmatrix} \tau''_1 \\ \tau''_2 \\ \vdots \\ \tau''_N \\ \tau''_{N+1} \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_N \\ b_{N+1} \end{bmatrix}$$

or $A\tau'' = b$ with the obvious definitions.

Let $A = D + O$ where D is diagonal and $O_{ii} = 0 \ \forall i$. Then $(D+O)\tau'' = b$ which implies that $\tau'' = -D^{-1}O\tau'' + D^{-1}b$. Define $R \equiv D^{-1}O$ and $r \equiv D^{-1}b$. Therefore,

$$\tau'' = -R\tau'' + r.$$

In any iterative scheme [20] derived from this relation due consideration must be given to the eigenvalues of R.

Explicitly

$$R = \begin{bmatrix} 0 & e_1/d_1 & & & \\ e_1/(d_1+d_2) & 0 & e_2/(d_1+d_2) & & \\ & \ddots & \ddots & \ddots & \\ & & e_{N-1}/(d_{N-1}+d_N) & 0 & e_N/(d_{N-1}+d_N) \\ & & & e_N/d_N & 0 \end{bmatrix}.$$

Note that, although R is not symmetric, we do have

$$D^{1/2}{}_R\, D^{-1/2} = D^{-1/2}{}_O\, D^{-1/2}$$

which is symmetric. Hence, R is similar to a symmetric matrix thus implying that its eigenvalues are real. Moreover,

$$Rx = \lambda x \Rightarrow$$
$$R\bar{x} = -\lambda\bar{x}$$

where $\bar{x}$ is obtained from x by altering the sign of every other element.

Gerschgorin's theorem then allows us to conclude that all of R's eigenvalues satisfy

$$|\lambda| \leq \max\left\{\frac{e_1}{d_1}, \frac{e_{i-1}+e_i}{d_{i-1}+d_i}, \frac{e_N}{d_N}\right\} \equiv \mu \leq \frac{1}{2}.$$

Young's theory then allows us to use simultaneous over-relaxation

$$[\tau'']^{(n+1)} = \omega[r - R[\tau'']^{(n)}] - (\omega-1)[\tau'']^{(n)}$$

and the optimum relaxation factor will be given by

$$\omega^* = \frac{2}{1 + \sqrt{1-\lambda^2_{max}}}.$$

As an estimate of λ_{max} we use μ. To see that this is a reasonable choice, consider the case of uniform mesh width and tension. In this case

$$\mu = \frac{e}{d}$$

and $Rx = \mu x$ with $x = [1, \text{———}, 1]^T$. I.e. $\mu = \lambda_{max}$ and this produces an exact value of ω^* ! In this instance,

the special case of the cubic spline yields $\mu = 1/2$ and $\omega^* = 4(2 - \sqrt{3})$.

A possible starting point for the iterative process would be

$$[\tau'']^{(0)} = \frac{2b_i}{h_{i-1}+h_i} .$$

The spline equations in terms of first derivatives with second derivative end conditions are

$$\left[\frac{d_1}{d_i^2-e_i^2}\right]\tau_1' + \left[\frac{e_1}{d_i^2-e_i^2}\right]\tau_2' = \left[\frac{1}{d_1-e_1}\right]\cdot\left[\frac{f_2-f_1}{h_1}\right] - \tau_1''$$

$$\left[\frac{e_{i-1}}{d_{i-1}^2-e_{i-1}^2}\right]\tau_{i-1}' + \left[\frac{d_{i-1}}{d_{i-1}^2-e_{i-1}^2} + \frac{d_i}{d_i^2-e_i^2}\right]\tau_1' + \left[\frac{e_i}{d_i^2-e_i^2}\right]\tau_{i+1}'$$

$$= \left[\frac{1}{d_{i-1}-e_{i-1}}\right]\cdot\left[\frac{f_i-f_{i-1}}{h_{i-1}}\right] + \left[\frac{1}{d_i-e_i}\right]\cdot\left[\frac{f_{i+1}-f_i}{h_i}\right] , \quad i=2,\ldots,N$$

$$\left[\frac{e_N}{d_N^2-e_N^2}\right]\tau_N' + \left[\frac{d_N}{d_N^2-e_N^2}\right]\tau_{N+1}' = \tau_{N+1}'' + \left[\frac{1}{d_N-e_N}\right]\cdot\left[\frac{f_{N+1}-f_N}{h_N}\right]$$

or $A\tau' = b$ with the obvious definitions.

The iterative treatment of this system proceeds along precisely the same lines as before. However, now

$$|\lambda| \leq \max\left\{\frac{e_1}{d_1}, \frac{e_{i-1}(d_i^2-e_i^2) + e_i(d_{i-1}^2-e_{i-1}^2)}{d_{i-1}(d_i^2-e_i^2) + d_i(d_{i-1}^2-e_{i-1}^2)}, \frac{e_N}{d_N}\right\} \equiv \mu \leq \frac{1}{2} .$$

Once again, μ is used as our estimate of λ_{max} with the familiar justification.

II.4. End Conditions

In our previous discussions we have always assigned one boundary condition at each end of the interval of interpolation, i.e. a boundary value problem. It occurs to us however that we could rightfully assign both end conditions at one end, i.e. an initial value problem. This would allow us to concatenate sequentially defined splines in a C^2 fashion. To facilitate clarity we will restrict ourselves to uniform mesh and tension.

If we choose to specify $\tau_1' = \phi'$ and $\tau_1'' = \phi''$ then we can write the resulting equations as

$$\begin{bmatrix} 1 & & & & \\ d & e & & & \\ e & 2d & e & & \\ & \ddots & \ddots & \ddots & \\ & & e & 2d & e \end{bmatrix} \begin{bmatrix} \tau_1'' \\ \tau_2'' \\ \tau_3'' \\ \vdots \\ \tau_{N+1}'' \end{bmatrix} = \begin{bmatrix} \phi'' \\ (f_2-f_1)/h_1-\phi' \\ b_2 \\ \vdots \\ b_N \end{bmatrix} .$$

Now let

$$a_i \equiv \frac{\partial \tau_i''}{\partial \phi'} , \qquad A_i \equiv \frac{\partial \tau_i''}{\partial \phi''} \Rightarrow$$

$$a_1 = 0 , \qquad a_2 = -\frac{1}{e}$$

$$A_1 = 1 , \qquad A_2 = -\frac{d}{e}$$

$$ea_{i-2}+2da_{i-1}+ea_i = 0$$

$$eA_{i-2}+2dA_{i-1}+eA_i = 0 .$$

Thus we consider the difference equation

$$z_{k-1} + 2\mu z_k + z_{k+1} = 0 \; ; \quad z_0 \, , \, z_1 \quad \text{given,}$$

with $\mu \equiv d/e$. Assuming a solution of the form $z_k = \alpha q^k \Rightarrow 1+2\mu q+q^2 = 0 \Rightarrow q_1 = -\mu-\sqrt{\mu^2-1} < -1$, $q_2 = -\mu + \sqrt{\mu^2-1} < 1$. We introduce the basis sequences defined by

$$X_{k-1} + 2\mu X_k + X_{k+1} = 0 \; ; \quad X_0 = 1, \quad X_1 = 0$$

$$Y_{k-1} + 2\mu Y_k + Y_{k+1} = 0 \; ; \quad Y_0 = 0, \quad Y_1 = 1 \quad \Rightarrow$$

$$X_n = \frac{q_2}{q_2-q_1} q_1^n - \frac{q_1}{q_2-q_1} q_2^n$$

$$Y_n = -\frac{1}{q_2-q_1} q_1^n + \frac{1}{q_2-q_1} q_2^n .$$

Hence,

$$Z_n = Z_0 \cdot X_n + Z_1 \cdot Y_n$$

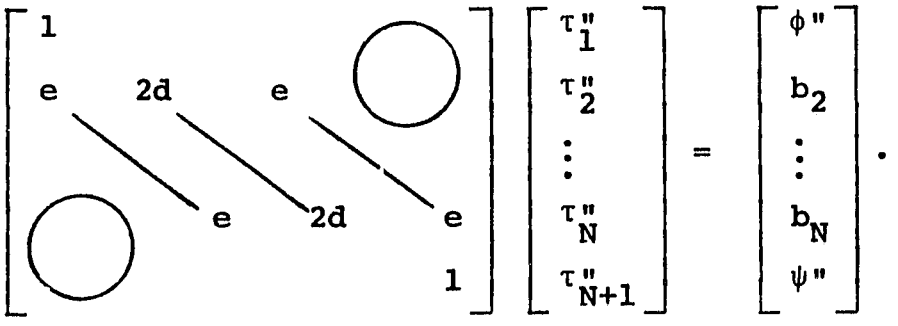
and therefore

$$a_n = -\frac{1}{e} Y_{n-1}$$

$$A_n = X_{n-1} - \frac{d}{e} Y_{n-1} \, .$$

We now see why such a specification is not suitable. As $|q_1| > 1$ we have the effect of a perturbation in the initial conditions increasing exponentially. In this sense these end conditions are ill-conditioned.

Let us use this technique to study the specification of second derivatives at the two ends. The system of equations is



Letting $a_i \equiv \frac{\partial \tau''_i}{\partial \phi''}$, $A_i \equiv \frac{\partial \tau''_i}{\partial \psi''}$ we have

$$a_1 = 1,\ a_{N+1} = 0,\ A_1 = 0,\ A_{N+1} = 1 \quad \text{and}$$

$$a_{i-1} + 2\mu a_i + a_{i+1} = 0$$

$$A_{i-1} + 2\mu A_i + A_{i+1} = 0 \ .$$

Therefore

$$a_n = c_1 q_1^{n-1} + c_2 q_2^{n-1}$$

$$A_n = k_1 q_1^{n-1} + k_2 q_2^{n-1} \ .$$

We demonstrate the determination of the coefficients for a_n.

The boundary conditions give us

$$\left.\begin{aligned} c_1 + c_2 &= 1 \\ c_1 q_1^N + c_2 q_2^N &= 0 \end{aligned}\right\} \Rightarrow c_1 = -\frac{q_2^N}{q_1^N - q_2^N}\ ,\quad c_2 = \frac{q_1^N}{q_1^N - q_2^N}$$

$\Rightarrow$

$$a_n = \frac{-q_1^{n-1} q_2^N + q_1^N q_2^{n-1}}{q_1^N - q_2^N} \ .$$

Similarly

$$A_n = \frac{-q_1^{N+1-n} q_2^N + q_1^N q_2^{N+1-n}}{q_1^N - q_2^N} \ .$$

Note that since $q_1 q_2 = 1$ these may be rewritten as

$$a_n = \frac{q_1^{N-n+1} - q_2^{N-n+1}}{q_1^N - q_2^N}$$

$$A_n = \frac{q_1^{n-1} - q_2^{n-1}}{q_1^N - q_2^N}$$

which are clearly well behaved.

Now consider the specification of first derivatives at both ends. The system of equations is

$$\begin{bmatrix} d & e & & & \\ e & 2d & e & & \\ & \ddots & \ddots & \ddots & \\ & & e & 2d & e \\ & & & e & d \end{bmatrix} \begin{bmatrix} \tau_1'' \\ \tau_2'' \\ \vdots \\ \tau_N'' \\ \tau_{N+1}'' \end{bmatrix} = \begin{bmatrix} (f_2-f_1)/h_1 - \phi' \\ b_2 \\ \vdots \\ b_N \\ \psi' - (f_{N+1}-f_N)/h_N \end{bmatrix}.$$

Letting $a_i \equiv \frac{\partial \tau_i''}{\partial \phi'}$, $A_i \equiv \frac{\partial \tau_i''}{\partial \psi'}$ we have

$$da_1 + ea_2 = -1, \quad ea_N + da_{N+1} = 0, \quad dA_1 + eA_2 = 0, \quad eA_N + dA_{N+1} = 1$$

$$\Rightarrow \mu a_1 + a_2 = -1/e, \quad a_N + \mu a_{N+1} = 0, \quad \mu A_1 + A_2 = 0, \quad A_N + \mu A_{N+1} = 1/e$$

$$a_{i-1} + 2\mu a_i + a_{i+1} = 0$$

$$A_{i-1} + 2\mu A_i + A_{i+1} = 0 .$$

Once again,

$$a_n = c_1 q_1^{n-1} + c_2 q_2^{n-1}$$

$$A_n = k_1 q_1^{n-1} + k_2 q_2^{n-1} .$$

Therefore

$$\left.\begin{aligned} a_1 &= c_1 + c_2 \\ a_2 &= c_1 q_1 + c_2 q_2 \end{aligned}\right\}$$

together with $\mu a_1 + a_2 = -1/e \Rightarrow$

$$c_1(\mu+q_1) + c_2(\mu+q_2) = -1/e \ .$$

Also

$$\left.\begin{aligned} a_N &= c_1 q_1^{N-1} + c_2 q_2^{N-1} \\ a_{N+1} &= c_1 q_1^N + c_2 q_2^N \end{aligned}\right\}$$

together with $a_N + \mu a_{N+1} = 0 \Rightarrow$

$$c_1(q_1^{N-1}+\mu q_1^N) + c_2(q_2^{N-1}+\mu q_2^N) = 0 \ .$$

Hence,

$$c_1 = \frac{1}{e} \cdot \frac{q_2^{N-1} + \mu q_2^N}{[(\mu+q_1)(q_2^{N-1}+\mu q_2^N) + (\mu+q_2)(q_1^{N-1}+\mu q_1^N)]}$$

$$c_2 = -\frac{1}{e} \cdot \frac{q_1^{N-1} + \mu q_1^N}{[(\mu+q_1)(q_2^{N-1}+\mu q_2^N) + (\mu+q_2)(q_1^{N-1}+\mu q_1^N)]} \ .$$

In a similar fashion we obtain

$$k_1 = -\frac{1}{e} \cdot \frac{\mu + q_2}{[(\mu+q_1)(q_2^{N-1}+\mu q_2^N) - (\mu+q_2)(q_1^{N-1}+\mu q_1^N)]}$$

$$k_2 = \frac{1}{e} \cdot \frac{\mu + q_1}{[(\mu+q_2)(q_2^{N-1}+\mu q_2^N) - (\mu+q_2)(q_1^{N-1}+\mu q_1^N)]} \ .$$

These formulae are of great utility in the following application. Suppose that we spline fit a collection of data and then decide that the end conditions that were used were inappropriate. Rather than computing a new spline

we may use the above to update $\{\tau_i''\}_{i=1}^{N+1}$.

For example, consider the case of second derivatives specified at both ends. The old spline equations are

$$A\tau'' = b$$

and the new spline equations are

$$A(\tau''+\delta\tau'') = b + \delta b$$

where $\delta b = [\delta\phi'' \quad 0 \quad \ldots \quad 0 \quad \delta\psi'']^T$. So we may solve $A\cdot\delta\tau'' = \delta b$ and add the result to τ''. Here $\delta\tau''$ is simply a linear combination of the previously derived quantities. Specifically $(\delta\tau'')_i = \delta\phi''\cdot a_i + \delta\psi''\cdot A_i$.

Similar comments apply to the case of first derivatives specified. Moreover, all the preceding analysis may be extended in a straightforward fashion to the case of mixed end conditions.

It is interesting to note that if we are confronted with a spline code that only accepts either $\tau_1' = \tau_{N+1}' = 0$ or $\tau_1'' = \tau_{N+1}'' = 0$ as end conditions we can use the following auxiliary functions [5].

$$A_1(x) = f(x) - \frac{1}{2}\left[\frac{(x-a)^2}{b-a} f'(b) - \frac{(b-x)^2}{b-a} f'(a)\right]$$

$$A_2(x) = f(x) - \frac{1}{6}\left[\frac{(x-a)^3}{b-a} f''(b) + \frac{(b-x)^3}{b-a} f''(a)\right].$$

We then calculate A_i at the knots, spline fit these values using the appropriate end conditions and then set

$$f(x) \approx \tau(x) + \frac{1}{2}\left[\frac{(x-a)^2}{b-a} f'(b) - \frac{(b-x)^2}{b-a} f'(a)\right]$$

or

$$f(x) \approx \tau(x) + \frac{1}{6}\left[\frac{(x-a)^3}{b-a} f''(b) + \frac{(b-x)^3}{b-a} f''(a)\right]$$

depending on which end conditions were used.

We conclude this section with a treatment of the important practical problem of supplying end conditions when only data points themselves are available. We choose to consider $f(x) \in C^4[a,b]$ as our paradigm.

Let $f \in C^4[x_1, x_{N+1}]$ with $M \equiv \sup|f^{(4)}|$. A theorem of Pruess [35] states that

$$\|D^i(s-\tau)\| \leq (26/3)\, p_{max}^2\, h^{4-i} \max_j |s_j''| \; ; \quad i=0,1,2.$$

A result of Kershaw [25] states that

$$\|D^i(f-s)\| \leq c_i \cdot h^{2-i}[h^2 M + 8 \max_j |f_j''-s_j''|]; \; i=0,1,2.$$

Hence, we may conclude that

$$\|D^i(\tau-f)\| \leq (26/3)\, p_{max}^2\, h^{4-i} \max_j |s_j''| + c_i h^{4-i} \cdot M + c_i h^{2-i} \cdot 8 \max_j |f_j''-s_j''| \; ; \qquad i=0,1,2.$$

Thus, in order to maintain uniform $O(h^{4-i})$ accuracy, we need only ensure that $\max_j |f_j''-s_j''| = O(h^2)$. This is well known to be the case when exact slope end conditions are specified. We now investigate the effect of using approximate rather than exact slope end conditions thus extending known results [44] for the cubic spline to the exponential spline. Recall that

$$As'' = 6b$$

where

$$b_1 = \frac{f_2 - f_1 - h_1 f_1'}{h_1^2}$$

$$b_k = \frac{(1-\alpha_k) f_{k-1} - f_k + \alpha_k f_{k+1}}{h_{k-1} h_k} \; ; \qquad k=2,\ldots,N$$

$$b_{N+1} = -\frac{f_{N+1} - f_N - h_N f_{N+1}'}{h_N^2}$$

$$\alpha_k = \frac{h_{k-1}}{h_{k-1} + h_k} \; ; \qquad k=2,\ldots,N$$

and

$$A = \begin{bmatrix} 2 & 1 & & & \\ \alpha_2 & 2 & 1-\alpha_2 & & \\ & \ddots & \ddots & \ddots & \\ & & \alpha_N & 2 & 1-\alpha_N \\ & & & 1 & 2 \end{bmatrix} .$$

We then have

$\|\frac{1}{2} A - I\| = \frac{1}{2}$ (since the sup vector-norm induces the max row sum matrix-norm)

$\Rightarrow \|A^{-1}\| = \frac{1}{2} \|(\frac{1}{2} A)^{-1}\| \leq (1/2)/[1-\|\frac{1}{2} A-I\|] = 1.$

Now consider

$$A \bar{s}'' = 6\bar{b}$$

where $\bar{b}$ is obtained from b by replacing f_1' and f_{N+1}' by their estimated values $\bar{f}_1'$ and $\bar{f}_{N+1}'$.

This results in

$$A(s'' - \bar{s}'') = 6(b - \bar{b})$$

$$\Rightarrow s'' - \bar{s}'' = 6A^{-1}(b - \bar{b})$$

$$\Rightarrow \|s'' - \bar{s}''\| \leq 6 \cdot \|A^{-1}\| \cdot \|b - \bar{b}\|$$

$$\leq 6 \cdot \|b - \bar{b}\|$$

$$= 6 \max\left\{ |b_1 - \bar{b}_1| , |b_{N+1} - \bar{b}_{N+1}| \right\}$$

$$= 6 \max\left\{ \frac{|f_1' - \bar{f}_1'|}{h_1} , \frac{|f_{N+1}' - \bar{f}_{N+1}'|}{h_N} \right\}.$$

If we have

$$\bar{f}_1' = f_1' + O(h^3) \quad \text{and} \quad \bar{f}_{N+1}' = f_{N+1}' + O(h^3)$$

then

$$\max_j |f_j'' - \bar{s}_j''| \leq \max_j |f_j'' - s_j''| + \max_j |s_j'' - \bar{s}_j''| = O(h^2)$$

as desired.

So we see that we need only supply a third order accurate approximation to f'(a) and f'(b) in order to retain the interior error bounds. This can be achieved by using the slope predicted by the four-point one-sided difference formula derived from Lagrange interpolation.

This gives us the following end conditions.

Left-hand end:

$$\bar{f}_1' = c_1 f_1 + c_2 f_2 + c_3 f_3 + c_4 f_4$$

with

$$c_1 = - \frac{[(h_1+h_2)(h_1+h_2+h_3) + (h_1)(h_1+h_2+h_3) + (h_1)(h_1+h_2)]}{(h_1)(h_1+h_2)(h_1+h_2+h_3)}$$

$$c_2 = \frac{(h_1+h_2)(h_1+h_2+h_3)}{(h_1)(h_2)(h_2+h_3)}$$

$$c_3 = - \frac{(h_1)(h_1+h_2+h_3)}{(h_1+h_2)(h_2)(h_3)}$$

$$c_4 = \frac{(h_1)(h_1+h_2)}{(h_1+h_2+h_3)(h_2+h_3)(h_3)} \quad .$$

Note that

$$c_1 = - (c_2 + c_3 + c_4) \quad .$$

Right-hand end:

$$\bar{f}'_{N+1} = - d_4 f_{N-2} - d_3 f_{N-1} - d_2 f_N - d_1 f_{N+1}$$

with

$$d_1 = - \frac{[(h_N+h_{N-1})(h_N+h_{N-1}+h_{N-2}) + h_N(h_N+h_{N-1}+h_{N-2}) + h_N(h_N+h_{N-1})]}{(h_N)(h_N+h_{N-1})(h_N+h_{N-1}+h_{N-2})}$$

$$d_2 = \frac{(h_N+h_{N-1})(h_N+h_{N-1}+h_{N-2})}{(h_N)(h_{N-1})(h_{N-1}+h_{N-2})}$$

$$d_3 = - \frac{(h_N)(h_N+h_{N-1}+h_{N-2})}{(h_N+h_{N-1})(h_{N-1})(h_{N-2})}$$

$$d_4 = \frac{(h_N)(h_N+h_{N-1})}{(h_N+h_{N-1}+h_{N-2})(h_{N-1}+h_{N-2})(h_{N-2})} \quad .$$

Note that

$$d_1 = - (d_2 + d_3 + d_4) \quad .$$

We have the following error estimate for the left hand end condition

$$f_1' - \bar{f}_1' = - \frac{f^{(4)}(\xi)}{4!} (h_1)(h_1+h_2)(h_1+h_2+h_3) \quad , \quad \xi \in [x_1, x_4].$$

The corresponding estimate for the right hand end condition is

$$f_{N+1}' - \bar{f}_{N+1}' = \frac{f^{(4)}(\xi)}{4!} (h_N)(h_N+h_{N-1})(h_N+h_{N-1}+h_{N-2}),$$

$$\xi \in [x_{N-2}, x_{N+1}].$$

We establish the result for $\bar{f}_1'$. Let $f \in C^4[x_1, x_4]$ and define the linear functional $L[f] \equiv f'(x_1) - (c_1 f_1 + c_2 f_2 + c_3 f_3 + c_4 f_4)$. Now L annihilates all polynomials of degree 3 and hence Peano's theorem guarantees that

$$L[f] = \int_{x_1}^{x_4} f^{(4)}(t)\ K(t)\ dt$$

where

$$K(t) = \frac{1}{3!} L[(x-t)_+^3] \quad \text{(considered as a function of x)}$$

and

$$(x-t)_+^3 = \begin{cases} (x-t)^3 , & x \geq t \\ 0 , & x < t \end{cases} .$$

Now

$$L_x[(x-t)_+^3] = 3(x_1-t)_+^2 - [c_1(x_1-t)_+^3 + c_2(x_2-t)_+^3 + c_3(x_3-t)_+^3 + c_4(x_4-t)_+^3]$$

$$= 6 \cdot K(t) \ .$$

Therefore

$$6 \cdot K(t) = \begin{cases} -[c_2(x_2-t)^3 + c_3(x_3-t)^3 + c_4(x_4-t)^3] ; & x_1 \leq t \leq x_2 \\ -[c_3(x_3-t)^3 + c_4(x_4-t)^3] ; & x_2 \leq t \leq x_3 \\ -[c_4(x_4-t)^3] ; & x_3 \leq t \leq x_4 . \end{cases}$$

A tedious calculation verifies that

$$K(t) \leq 0 \; ; \qquad t \in [x_1, x_4].$$

A strong form of Peano's theorem then allows us to conclude that

$$L[f] = \frac{f^{(4)}(\xi)}{4!} L[x^4] \; ; \; \xi \in [x_1, x_4] \; .$$

An equally arduous computation then shows that

$$L[x^4] = -h_1(h_1+h_2)(h_1+h_2+h_3) \; .$$

Therefore

$$L[f] = -\frac{f^{(4)}(\xi)}{4!} (h_1)(h_1+h_2)(h_1+h_2+h_3) \; ,$$

i.e.

$$f'(x_1) - \bar{f}'(x_1) = -\frac{f^{(4)}(\xi)}{4!} (h_1)(h_1+h_2)(h_1+h_2+h_3) \; , \xi \in [x_1, x_4].$$

On a uniform mesh with constant tension there is an increase in the order of approximation. Hence, in this instance we should provide higher order estimates of the end conditions. We simply use

$$\hat{f}'_1 = \frac{1}{24h} [-50f_1 + 96f_2 - 72f_3 + 32f_4 - 6f_5]$$

$$\hat{f}'_{N+1} = \frac{1}{24h} [6\, f_{N-3} - 32f_{N-2} + 72f_{N-1} - 96f_N + 50f_{N+1}]$$

which derive from a quartic Lagrange interpolation [1]. The error estimates are

$$f'_1 - \hat{f}'_1 = \frac{1}{5} h^4 f^{(5)}(\xi) \; , \qquad \xi \in [x_1, x_5]$$

and

$$f'_{N+1} - \hat{f}'_{N+1} = \frac{1}{5} h^4 f^{(5)}(\xi), \quad \xi \in [x_{N-3}, x_{N+1}] \; .$$

The above considerations for first derivative end conditions with second derivatives as unknowns are easily extendable to second derivative end conditions. Furthermore, both of these cases may also be applied to the equations with first derivatives as unknowns with similar results.

II.5. Alternative Power Series Representation

The next problem to be treated is the loss of significance when evaluating $\sinh(p_i h_i)$ for small values of $(p_i h_i)$. Specifically, since $\sinh x = \frac{e^x - e^{-x}}{2}$ for $|x| << 1$ we have $e^x = e^{-x} = 1$, an invitation to "catastrophic cancellation." Such small arguments would arise naturally but the scaling suggested below makes these considerations even more critical. Hence, for small values of $(p_i h_i)$ (say < 0.07), we would like to resort to a power series representation of the hyperbolic functions. We note that such measures are unnecessary on IBM hardware. However, their inclusion enhances software portability. Moreover, the resulting formulae allow the use of $p_i = 0 \quad \forall i$; thus the cubic spline may be used as the zeroth iterate.

We proceed as follows:

$$e_i = \frac{h_i}{6}\left(1 - \frac{7}{60}\,p_i^2 h_i^2 + \frac{31}{2520}\,p_i^4 h_i^4\right) + O(p_i^6 h_i^7) ,$$

$$d_i = \frac{h_i}{3}\left(1 - \frac{1}{15}\,p_i^2 h_i^2 + \frac{2}{315}\,p_i^4 h_i^4\right) + O(p_i^6 h_i^7) .$$

Note that $p_i = 0 \quad \forall i \Rightarrow$

$$e_i = \frac{h_i}{6} ,$$

$$d_i = \frac{h_i}{3} ,$$

which are seen to produce the system of linear algebraic equations for the cubic spline with specified derivative end conditions [2]. That is,

$$\frac{h_1}{3}\tau_1'' + \frac{h_1}{6}\tau_2'' = \frac{f_2 - f_1}{x_2 - x_1} - f'(a) \equiv b_1 ,$$

$$\frac{h_{i-1}}{6}\tau_{i-1}'' + \frac{h_{i-1}+h_i}{3}\tau_i'' + \frac{h_i}{6}\tau_{i+1}'' \equiv b_i , \quad (i=2,\ldots,N),$$

$$\frac{h_N}{6}\tau_N'' + \frac{h_N}{3}\tau_{N+1}'' = f'(b) - \frac{f_{N+1} - f_N}{x_{N+1} - x_N} \equiv b_{N+1} .$$

We have the following expansion for $\tau(x)$:

$$\begin{aligned}\tau(x) = {} & f_i\left(\frac{x_{i+1}-x}{h_i}\right) + f_{i+1}\left(\frac{x-x_i}{h_i}\right) \\ & - \frac{h_i}{6}(x_{i+1}-x)\,\tau_i''\left(1 - \frac{7}{60}p_i^2h_i^2 + \frac{31}{2520}p_i^4h_i^4\right. \\ & + \frac{1}{6}p_i^2(x_{i+1}-x)^2 - \frac{1}{h_i^2}(x_{i+1}-x)^2 \\ & \left. - \frac{7}{195}p_i^4h_i^2(x_{i+1}-x)^2 - \frac{p_i^2}{20h_i^2}(x_{i+1}-x)^4 + \frac{p_i^4}{120}(x_{i+1}-x)^4\right) \\ & - \frac{h_i}{6}(x-x_i)\,\tau_{i+1}''\left(1 - \frac{7}{60}p_i^2h_i^2 + \frac{31}{2520}p_i^4h_i^4\right. \\ & + \frac{1}{6}p_i^2(x-x_i)^2 - \frac{1}{h_i^2}(x-x_i)^2 - \frac{7}{195}p_i^4h_i^2(x-x_i)^2 \\ & \left. - \frac{p_i^2}{20h_i^2}(x-x_i)^4 + \frac{p_i^4}{120}(x-x_i)^4\right).\end{aligned}$$

Note that, when $p_i = 0 \quad \forall i$, we have

$$\begin{aligned}\tau(x) = {} & f_i\left(\frac{x_{i+1}-x}{h_j}\right) + f_{i+1}\left(\frac{x-x_i}{h_i}\right) - \frac{1}{6}h_i(x_{i+1}-x)\,\tau_i''\left(1 - \frac{(x_{i+1}-x)^2}{h_i^2}\right) \\ & - \frac{1}{6}h_i(x-x_i)\,\tau_{i+1}''\left(1 - \frac{(x-x_i)^2}{h_i^2}\right).\end{aligned}$$

which is the equation for the cubic spline on the $i^{\underline{th}}$

interval [2]. Hence, a convenient choice for initial tension parameter values is $p_i = 0 \quad \forall i$.

Many applications require the derivatives of the interpolant. For completeness, we include the formulae for $\tau'(x)$, $\tau''(x)$, $\tau'''(x)$, in terms of both hyperbolic functions and the alternative power series representation. Hyperbolic function representation:

$$\tau'(x) = \frac{1}{p_i \sinh(p_i h_i)} \left(\tau''_{i+1} \cosh p_i(x-x_i) - \tau''_i \cosh p_i(x_{i+1}-x)\right)$$

$$+ \frac{1}{h_i}\left((f_{i+1}-f_i) + \frac{1}{p_i^2}(\tau''_i-\tau''_{i+1})\right).$$

$$\tau''(x) = \tau''_{i+1} \cdot \frac{\sinh p_i(x-x_i)}{\sinh(p_i h_i)} + \tau''_i \cdot \frac{\sinh p_i(x_{i+1}-x)}{\sinh(p_i h_i)} .$$

$$\tau'''(x) = p_i \tau''_{i+1} \cdot \frac{\cosh p_i(x-x_i)}{\sinh(p_i h_i)} - p_i \tau''_i \frac{\cosh p_i(x_{i+1}-x)}{\sinh(p_i h_i)} .$$

Power series representation:

$$\tau'(x) = \frac{1}{h_i}(f_{i+1}-f_i) - c_1 h_i \cdot \left\{\tau''_{i+1}(T_1 + 3 \cdot T_2 \cdot x_2^2 + 5 \cdot T_3 \cdot x_2^4) - \tau''_i(T_1 + 3 \cdot T_2 \cdot x_1^2 + 5 \cdot T_3 \cdot x_1^4)\right\}.$$

$$\tau''(x) = c_1 h_i \cdot \left\{\tau''_i \cdot x_1 (6 \cdot T_2 + 20 \cdot T_3 \cdot x_1^2) - \tau''_{i+1} \cdot x_2 (6 \cdot T_2 + 20 \cdot T_3 \cdot x_2^2)\right\}$$

$$\tau'''(x) = -c_1 h_i \cdot \left\{6 \cdot \tau''_i (T_2 + 10 \cdot T_3 x_1^2) + 6 \cdot \tau''_{i+1}(T_2 + 10 \cdot T_3 \cdot x_2^2)\right\}$$

where

$$T_1 = 1 + c_3 \cdot p_i^2 h_i^2 + c_4 \cdot p_i^4 h_i^4$$

$$T_2 = c_1 \cdot p_i^2 - \frac{1}{h_i^2} + c_7 \cdot p_i^4 h_i^2$$

$$T_3 = \left(c_8 \cdot \frac{1}{h_i^2} + c_9 \cdot p_i^2\right) \cdot p_i^2$$

$$x_1 = x_{i+1} - x \ , \quad x_2 = x - x_i$$

$c_1 = 1/6$, $c_3 = -7/60$, $c_4 = 31/2520$, $c_7 = -7/195$, $c_8 = -1/20$, $c_9 = 1/120$.

Note that in the presence of scaling these formulae must be modified accordingly.

II.6. Parameter Selection Algorithm for Co-Convex Interpolation

In this and the following section we take up the task of tension parameter selection. This is the key problem which needs to be solved if exponential splines are to be useful in practice. A satisfactory treatment is not available in the literature.

A previous section assures us that for large enough tension parameters the exponential spline interpolant is free from extraneous inflection points. The question now arises as to how to choose p_i $(i=1,\ldots,N)$ which are sufficiently but not excessively large. Excessively large estimates will produce an interpolant which is kinky in appearance since $\tau_i'' , \tau_{i+1}'' \to \infty$ as $p_k \to \infty$ $(k=i-1,i,i+1)$. In this section we present a tension parameter selection scheme that answers this question in both a theoretical and practical sense. (Note: Assume that $b_i \neq 0$; $i = 1,\ldots,N+1$.)

Assuming $\tau_i'' \neq 0$ $(i=1,\ldots,N+1)$ then $\tau_i'' b_i > 0$ $(i=1,\ldots,N+1)$ is a necessary and sufficient condition for no extraneous inflection points. Hence, we will iteratively alter p_i $(i = 1,\ldots,N)$ so as to enforce $\tau_i'' b_i > 0$ $(i=1,\ldots,N+1)$. Before proceeding we need the following easily established facts:

(a) $p_i > 0 \Rightarrow d_i > 0$

(b) $ab > 0$ iff $|a-b| < \max(|a|,|b|)$.

For purposes of illustration, assume that for some choice, $p_i^{(n)}$ $(i=1,\ldots,N)$, we have $\tau''_k b_k < 0$ for some k between 2 and N (a similar analysis will subsequently be given for the end intervals). We then have

$$|e_{k-1}\tau''_{k-1} + e_k\tau''_{k+1}| = |b_k - (d_{k-1}+d_k)\tau''_k| \ .$$

Now

$$b_k\tau''_k > 0 \text{ iff } |b_k - (d_{k-1}+d_k)\tau''_k| < \max(|b_k|,\ (d_{k-1}+d_k)|\tau''_k|\ )$$

since

$$d_i > 0 \quad \text{for} \quad p_i > 0 \quad (i = 1,\ldots,N) \ .$$

We therefore define $\tilde{p}_{k-1}$, $\tilde{p}_k$ so that $\tilde{e}_{k-1}$, $\tilde{e}_k$ produce

$$|\tilde{e}_{k-1}\tau''_{k-1} + \tilde{e}_k\tau''_{k+1}| < \max(|b_k|,\ (d_{k-1}+d_k)|\tau''_k|\ );$$

i.e., after freezing the τ'''s, we vary the p's so that all these inequalities are satisfied.

Letting

$$\tilde{e}_i < \frac{\max(|b_k|, (d_{k-1}+d_k)|\tau''_k|)}{2\ \max\ (|\tau''_{k-1}|,\ |\tau''_{k+1}|)} \quad ; \quad i = k-1,\ k$$

produces the desired result.

Define

$$\lambda = \frac{\max(|b_k|,\ (d_{k-1}+d_k)|\tau''_k|)}{2\ \max(|\tau''_{k-1}|, |\tau''_{k+1}|)} \ .$$

Now, since

$$e_i = [\frac{1}{h_i} - \frac{p_i}{\sinh(p_i h_i)}]/\ p_i^2 \ ,$$

we seek to satisfy the inequality

$$g_i(x) < 0 \ , \qquad (i = k-1,\ k),$$

where

$$g_i(x) = \left[\frac{1}{h_i} - \frac{x}{\sinh(h_i x)}\right] / x^2 - \bar{\lambda} \ , \qquad (i=k-1,k).$$

These considerations lead directly to

$$\frac{\frac{1}{h_i} - \frac{p_i}{\sinh(p_i h_i)}}{p_i^2} < \bar{\lambda} \ ,$$

$$\Rightarrow \ \frac{1}{h_i} - \frac{p_i}{\sinh(p_i h_i)} - p_i^2 \bar{\lambda} < 0 \ ,$$

$$\Rightarrow \ \frac{1}{h_i} - p_i^2 \bar{\lambda} < \frac{p_i}{\sinh(p_i h_i)} \ .$$

This is certainly the case if

$$\frac{1}{h_i} - \bar{\lambda} p_i^2 = 0 \ ,$$

i.e.

$$p_i = (\bar{\lambda} h_i)^{-1/2}.$$

Consequently, we define

$$\tilde{p}_i = (\bar{\lambda} h_i)^{-1/2} \quad \text{(or perhaps } \tilde{p}_i = \max((\bar{\lambda} h_i)^{-1/2}, \ p_i^{(n)})); \ i=k-1,k$$

(i.e. we always increase the p's).

However, the new tension parameters ($\tilde{p}$'s) will alter the τ"'s thus requiring an iterative procedure. The suggested procedure is

$$p^{(n+1)} = p^{(n)} + \omega(\tilde{p} - p^{(n)}) \ .$$

It was previously assumed that $\tau_i'' \neq 0$ $(i=1,\ldots,N+1)$. It can be shown [42,43] that if $\tau_i'' = 0$ for any $2 \leq i \leq N$ then $\tau_{i-1}''\tau_{i+1}'' \geq 0$ is necessary and sufficient for the

prior analysis to hold. Now we have

$$(d_{i-1} + d_i)\tau_i'' = b_i - e_{i-1}\tau_{i-1}'' - e_i\tau_{i+1}'' ,$$

and, since we still assume $b_k \neq 0 \quad \forall k$, we cannot have $\tau_{i-1}'' = \tau_{i+1}'' = 0$ (i.e. either e_{i-1} or e_i are actually present in the equation). So to make $\tau_i'' \neq 0$ when $\tau_{i-1}''\tau_{i+1}'' < 0$, we perturb e_{i-1} and e_i by incrementing p_{i-1} and p_i by some small amount, ε.

It remains to remove the assumption $b_k \neq 0 \quad \forall k$. If $b_i = 0$ then the points (x_{i-1},y_{i-1}), (x_i,y_i) and (x_{i+1},y_{i+1}) should be joined by a straight line (with a similar statement true in the end intervals). The two remaining portions should then be fit separately using slope end conditions at x_{i-1}^{-} and x_{i+1}^{+} derived from the slope of the straight line segment. This may be accomplished implicitly by the following alteration of the coefficient matrix, A. Set as many of the set $\{A_{i-1,i}, A_{i,i+1}, A_{i+1,i}, A_{i,i-1}\}$ as exist equal to zero and then proceed as usual. This produces the desired τ_k'' $(k=1,\ldots,i-1,i+1,\ldots,N+1)$. The interval $[x_{i-1},x_{i+1}]$ is then fit separately with a linear function. The points (x_{i-1},y_{i-1}) and (x_{i+1}, y_{i+1}) are now to be treated as the right and left hand endpoints, respectively, of two distinct exponential splines.

In summary, we note that the tension parameter selection problem is inherently nonlinear (by virtue of the nonlinear occurrence of p_i in the interpolant). It is then hardly surprising that an iterative procedure should suggest itself.

We now return to the problem of parameter selection for the end intervals. The relevant equations are

$$d_1\tau_1'' + e_1\tau_2'' = b_1$$
$$e_N\tau_N'' + d_N\tau_{N+1}'' = b_{N+1} .$$

$\tau_1''b_1 < 0$: $|e_1\tau_2''| = |b_1 - d_1\tau_1''|$.

Now $b_1\tau_1'' > 0$ iff $|b_1 - d_1\tau_1''| < \max(|b_1|, d_1|\tau_1''|)$.
Therefore define $\tilde{p}_1 \ni \tilde{e}_1$ produces $|\tilde{e}_1\tau_2''| < \max(|b_1|, d_1|\tau_1''|)$.
Letting $\tilde{e}_1 < \max(|b_1|, d_1|\tau_1''|) / |\tau_2''|$ produces the desired result. Hence, define $\bar{\lambda} = \max(|b_1|, d_1|\tau_1''|) / |\tau_2''|$.

$\tau_{N+1}''b_{N+1} < 0$: $|e_N\tau_N''| = |b_{N+1} - d_N\tau_{N+1}''|$.

Now $b_{N+1}\tau_{N+1}'' > 0$ iff $|b_{N+1} - d_N\tau_{N+1}''| < \max(|b_{N+1}|, d_N|\tau_{N+1}''|)$.
Therefore define $\tilde{p}_N \ni \tilde{e}_N$ produces $|\tilde{e}_N\tau_N''| < \max(|b_{N+1}|, d_N|\tau_{N+1}''|)$.
Letting $\tilde{e}_N < \max(|b_{N+1}|, d_N|\tau_{N+1}''|) / |\tau_N''|$ produces the desired result. Hence, define $\bar{\lambda} = \max(|b_{N+1}|, d_N|\tau_{N+1}''|)/|\tau_N''|$.

II.7. Parameter Selection Algorithm for Co-Monotone Interpolation

In this section we develop a tension parameter selection algorithm that preserves any monotonicity present in the data [23]. Specifically, if the polygonal interpolant has slope of constant sign in three successive intervals then we select the tension parameters in these intervals so that $\tau'(x)$ has this same sign in the middle interval. Similar considerations apply at the end intervals.

We begin by rewriting the exponential spline as

$$\tau(x) = A_i + B_i x + C_i e^{p_i x} + D_i e^{-p_i x} \quad \text{on} \quad [x_i, x_{i+1}]$$

where

$$A_i \equiv \frac{1}{h_i}\left\{x_{i+1}\left[f_i - \frac{\tau''_i}{p_i^2}\right] - x_i\left[f_{i+1} - \frac{\tau''_{i+1}}{p_i^2}\right]\right\}$$

$$B_i = \left[\frac{f_{i+1}-f_i}{h_i}\right] - \left[\frac{\tau''_{i+1}-\tau''_i}{p_i^2 h_i}\right]$$

$$C_i \equiv \frac{1}{2p_i^2 s_i}\left[-\tau''_i\, e^{-p_i x_{i+1}} + \tau''_{i+1}\, e^{-p_i x_i}\right]$$

$$= \frac{e^{-p_i x_i}}{2p_i^2 s_i}\left[-\tau''_i e^{-p_i h_i} + \tau''_{i+1}\right]$$

$$D_i \equiv \frac{1}{2p_i^2 s_i}\left[\tau''_i\, e^{p_i x_{i+1}} - \tau''_{i+1}\, e^{p_i x_i}\right]$$

$$= \frac{e^{p_i x_i}}{2p_i^2 s_i}\left[\tau''_i\, e^{p_i h_i} - \tau''_{i+1}\right] .$$

Hence,

$$\tau'(x) = B_i + p_i C_i e^{p_i x} - p_i D_i e^{-p_i x}$$

$$\tau''(x) = p_i^2 [C_i e^{p_i x} + D_i e^{-p_i x}]$$

$$\tau'''(x) = p_i^3 [C_i e^{p_i x} - D_i e^{-p_i x}] \quad \text{on} \quad (x_i, x_{i+1}) .$$

Assume that $\tau_i' \tau_{i+1}' \geq 0$ with both having the "correct" sign.

Therefore, any extremum of $\tau'(x)$ interior to $[x_i, x_{i+1}]$ is characterized by $\tau''(x^*) = 0 \Rightarrow$

$$e^{2p_i x^*} = - \frac{D_i}{C_i} .$$

Now if $D_i/C_i \geq 0$ there is no interior extremum and this spline segment is monotonicity-preserving. If $D_i/C_i < 0$ then $e^{p_i x^*} = \sqrt{-D_i/C_i} \Rightarrow$

$$\tau'(x^*) = B_i + p_i \sqrt{-C_i D_i} \, [\operatorname{sgn}(C_i) - \operatorname{sgn}(D_i)]$$

$$= B_i + 2 \operatorname{sgn}(C_i) \, p_i \sqrt{-C_i D_i}$$

$$= B_i - 2 \operatorname{sgn}(D_i) \, p_i \sqrt{-C_i D_i} .$$

Also

$$\tau'''(x^*) = p_i^3 \sqrt{-C_i D_i} \, [\operatorname{sgn}(C_i) - \operatorname{sgn}(D_i)]$$

$$= 2 \operatorname{sgn}(C_i) \cdot p_i^3 \sqrt{-C_i D_i} = -2 \operatorname{sgn}(D_i) \, p_i^3 \sqrt{-C_i D_i} .$$

Therefore

(i) $C_i > 0$, $D_i < 0 \Rightarrow \tau'''(x^*) > 0 \Rightarrow \tau'(x^*)$ is a (local) minimum;

(ii) $C_i < 0$, $D_i > 0 \Rightarrow \tau'''(x^*) < 0 \Rightarrow \tau'(x^*)$ is a (local) maximum.

If $\tau'(x^*)$ has the correct sign then $\tau(x)$ is locally monotonicity preserving. If $\tau'(x^*)$ is of the "wrong" sign we must do something about it if $x^* \in [x_i, x_{i+1}]$. Since

$$x^* = x_i + \frac{1}{2p_i} \ell n \left[\frac{\tau''_{i+1} - \tau''_i \, e^{p_i h_i}}{\tau''_{i+1} - \tau''_i \, e^{-p_i h_i}} \right]$$

we thus have two cases:

(a) if $\tau''_i < e^{p_i h_i} \tau''_{i+1}$ then $x^* \in [x_i, x_{i+1}]$ iff $\tau''_{i+1} \geq 0$

(b) if $\tau''_i > e^{p_i h_i} \tau''_{i+1}$ then $x^* \in [x_i, x_{i+1}]$ iff $\tau''_{i+1} \leq 0$.

We now come to the case in which $\tau'(x^*)$ has the wrong sign. In this event we iteratively modify the tension parameter, p_i, so as to enforce the requirement that $\tau'(x^*)$ should have the correct sign.

Consequently, consider once again our special cases, with

$$m_i \equiv \frac{f_{i+1} - f_i}{h_i} \quad \text{and} \quad \gamma \equiv B_i + p_i \cdot 2 \, \text{sgn}(C_i) \cdot \sqrt{-C_i D_i} \, ;$$

(i) $C_i > 0$, $D_i < 0 \Rightarrow \tau'(x^*)$ is a minimum

(a) if $m_i \leq 0$ then do not alter p_i

(b) if $m_i > 0$ then, since we want $\gamma > 0$, set

$$p_i^{(n+1)} = - B_i / 2\sqrt{-C_i D_i} + \delta$$

(ii) $C_i < 0$, $D_i > 0 \Rightarrow \tau'(x^*)$ is a maximum

(a) if $m_i \geq 0$ then do not alter p_i

(b) if $m_i < 0$ then, since we want $\gamma < 0$, set

$$p_i^{(n+1)} = B_i / 2\sqrt{-C_i D_i} + \delta.$$

The only remaining issue is if the computed τ'_i should be of the wrong sign i.e. $\tau'_i m_{i-1} < 0$ and $\tau'_i m_i < 0$. We proceed as follows.

$$\tau'(x_i^+) = \frac{1}{p_i^2 s_i}\left[-p_i\tau_i'' \cosh(p_i h_i) + p_i\tau_{i+1}''\right] + \left[\frac{1}{h_i}(f_{i+1}-f_i) - \frac{\tau_{i+1}''-\tau_i''}{p_i^2}\right]$$

$$\tau'(x_i^-) = \frac{1}{p_{i-1}^2 s_{i-1}}\left[-p_{i-1}\tau_{i-1}'' + p_{i-1}\tau_i'' \cosh(p_{i-1}h_{i-1})\right] + \frac{1}{h_i}\left[(f_i - f_{i-1}) - \frac{\tau_i''-\tau_{i-1}''}{p_{i-1}^2}\right]$$

As we know, subtracting these relations yields the tridiagonal system for τ_i'' $(i = 1,\ldots,N+1)$.

However, we choose to average them which produces

$$\tau_i' = \frac{1}{2}\left\{[e_{i-1}\tau_{i-1}'' + (d_{i-1}-d_i)\tau_i'' - e_i\tau_{i+1}''] + [m_{i-1}+m_i]\right\}.$$

We choose to enforce

$$e_{i-1}\cdot|\tau_{i-1}''| + (d_{i-1}+d_i)\cdot|\tau_i''| + e_i\cdot|\tau_{i+1}''| < |m_{i-1}+m_i|.$$

Now,

$$e_{i-1}\cdot|\tau_{i-1}''|+(d_{i-1}+d_i)\cdot|\tau_i''|+e_i\cdot|\tau_{i+1}''|$$
$$\leq (e_{i-1} + d_{i-1} + e_i + d_i)\max(|\tau_{i-1}''|,|\tau_i''|, |\tau_{i+1}''|).$$

We need the following

<u>Lemma</u>. (a) $e_i \leq \frac{1}{p_i^2 h_i}$

(b) $d_i \leq \frac{1}{p_i}$

<u>Proof:</u> (a) $e_i = \left[\frac{1}{h_i} - \frac{p_i}{s_i}\right] / p_i^2 < \frac{1}{p_i^2 h_i}$ since we know $e_i > 0$

$$\text{(b)} \quad d_i \leq \frac{1}{p_i} \quad \text{iff} \quad \frac{c_i}{s_i} - \frac{1}{\rho_i h_i} \leq 1$$

but $\frac{\cosh x}{\sinh x} - \frac{1}{x} \leq 1$ since $f(x) \equiv x \sinh x + \sinh x - x \cosh x \geq 0$

which follows from $f(0) = 0$ and

$$f'(x) = \cosh x + x(\cosh x - \sinh x) \geq 0 \quad \text{for } x \geq 0.$$

Q.E.D.

We are thus led to enforcing

$$\left[\frac{1}{p_{i-1}^2 h_{i-1}} + \frac{1}{p_{i-1}} + \frac{1}{p_i^2 h_i} + \frac{1}{p_i}\right] \max\left(|\tau''_{i-1}|, |\tau''_i|, |\tau''_{i+1}|\right) < |m_{i-1} + m_i|$$

or, alternatively

$$\frac{1}{p_{i-1}^2 h_{i-1}} + \frac{1}{p_{i-1}} < \frac{|m_{i-1} + m_i|}{2 \max\left(|\tau''_{i-1}|, |\tau''_i|, |\tau''_{i+1}|\right)}$$

together with

$$\frac{1}{p_i^2 h_i} + \frac{1}{p_i} < \frac{|m_{i-1} + m_i|}{2 \max\left(|\tau''_{i-1}|, |\tau''_i|, |\tau''_{i+1}|\right)} .$$

This ultimately leads to

$$p_i^{(n+1)} = \frac{1+p_i h_i}{p_i h_i} \cdot \frac{2 \max\left(|\tau''_{i-1}|, |\tau''_i|, |\tau''_{i+1}|\right)}{|m_{i-1} + m_i|} + \delta$$

and

$$p_{i-1}^{(n+1)} = \frac{1+p_{i-1} h_{i-1}}{p_{i-1} h_{i-1}} \cdot \frac{2 \max\left(|\tau''_{i-1}|, |\tau''_i|, |\tau''_{i+1}|\right)}{|m_{i-1} + m_i|} + \delta .$$

II.8. Periodic Exponential Spline

If one wishes to parametrically fit a closed body in a smooth fashion, then periodic end conditions must be imposed. The necessary modifications to the preceding analysis are presented herein.

Matrix Structure

The equation for strictly interior points remains unchanged:

$$e_{i-1}\tau''_{i-1} + (d_{i-1}+d_i)\tau''_i + e_i\tau''_{i+1} = b_i \quad (i = 2,\ldots,N-1).$$

We now have the periodic end conditions

$$f_{N+1} = f_1 ,$$
$$\tau''_{N+1} = \tau''_1 .$$

In addition, we imagine an additional point at x_0 ∋

$$h_0 = h_N$$
$$f_0 = f_N$$
$$\tau''_0 = \tau''_N .$$

This effectively identifies the two endpoints. We now write down the equations for i = 1 and N:

$$(d_N+d_1)\tau''_1 + e_1\tau''_2 + e_N\tau''_N = \frac{f_2-f_1}{h_1} - \frac{f_{N+1}-f_N}{h_N}$$

$$e_N\tau''_1 + e_{N-1}\tau''_{N-1} + (d_{N+1}+d_N)\tau''_N = b_N .$$

We therefore redefine

$$b_1 = \frac{f_2-f_1}{h_1} - \frac{f_{N+1}-f_N}{h_N} .$$

Note that the matrix is no longer tridiagonal. Instead it has the structure,

$$\begin{bmatrix} DIAG_1 & E_1 & & & E_N \\ E_1 & & & 0 & \\ & & & & E_{N-1} \\ & 0 & & & \\ E_N & & E_{N-1} & & DIAG_N \end{bmatrix} \begin{bmatrix} \tau''_1 \\ \vdots \\ \tau''_N \end{bmatrix} = \begin{bmatrix} b_1 \\ \vdots \\ b_N \end{bmatrix} .$$

Matrix Solution

The following algorithm [2] may be used to solve for τ''_i (i=1,...,N); let

$$\begin{aligned} PK_1 &= DIAG_1 \\ Q_1 &= -E_1/PK_1 \\ U_1 &= B_1/PK_1 \\ S_1 &= -E_N/PK_1 \end{aligned}$$

then set

$$\left.\begin{aligned} PK_i &= E_{i-1} \cdot Q_{i-1} + DIAG_i \\ Q_i &= -E_i/PK_i \\ U_i &= (B_i - E_{i-1} \cdot U_{i-1}) / PK_i \\ S_i &= -E_{i-1} \cdot S_{i-1} /PK_i \end{aligned}\right\} \quad (i=2,...,N-1)$$

let

$$T_N = 1$$

$$V_N = C$$

then set

$$\left.\begin{aligned} T_i &= Q_i \cdot T_{i+1} + S_i \\ V_i &= Q_i \cdot V_{i+1} + U_i \end{aligned}\right\} \qquad (i=N-1,\ldots,1)$$

and finally

$$\tau''_N = (B_N - E_N \cdot V_1 - E_{N-1} \cdot V_{N-1}) \,/\, (E_N \cdot T_1 + E_{N-1} \cdot T_{N-1} + DIAG_N)$$

$$\tau''_i = Q_i \cdot \tau''_{i+1} + S_i \cdot \tau''_N + U_i \ , \qquad (i=N-1,\ldots,1).$$

Parameter Selection Procedure

We treat the periodic case as follows:

(1) All points $\{(x_i,f_i)\}_{i=1}^{N}$ are treated as interior points.

(2) When $\tau''_1 b_1 < 0$, we modify p_1 and p_N.

The previous scheme is otherwise unaltered.

II.9. Numerical Considerations

In this section we detail the necessary modifications and extensions to the above analysis which practical implementation mandates [10,11,38].

Scaling

The first problem to be treated is that of the inherent number range restriction. In IBM FORTRAN this results in the restriction that the magnitude of exponential arguments be less than 175.366. Since the exponential spline definition requires the computation of quantities such as $\sinh(p_i h_i)$, the data must be scaled so that $\max_{1\le i\le N} \{p_i h_i\} < 175.366$.

However, it should first be shown that a scaling of the abscissae leaves the sign of $\tau_i'' b_i$ $(i=1,\ldots,N+1)$ invariant. This is required since the necessary and sufficient condition for no extraneous inflection points which is the basis for our algorithm is a function of the algebraic signs of these quantities. We proceed as follows:

$$\tau_i'' b_i \equiv \tau''(x_i)\left[\frac{f_{i+1}-f_i}{x_{i+1}-x_i} - \frac{f_i-f_{i-1}}{x_i-x_{i-1}}\right] \; ; \; \because \bar{x} = x/\alpha \Rightarrow$$

$$\tau_i'' b_i = \alpha^{-2}\bar{\tau}''(\bar{x}_i)\left[\frac{f_{i+1}-f_i}{\alpha(\bar{x}_{i+1}-\bar{x}_i)} - \frac{f_i-f_{i-1}}{\alpha(\bar{x}_i-\bar{x}_{i-1})}\right] = \alpha^{-3}\bar{\tau}_i''\bar{b}_i,$$

where $\bar{\tau}(\bar{x}) = \tau(\alpha\bar{x})$

so that

$$\alpha > 0 \Rightarrow$$

$\tau''_i b_i$ and $\tau''_i \bar{b}_i$ have the same sign.

In addition,

$$\tau''_{i-1}\ \tau''_{i+1} = \alpha^{-4} \bar{\tau}''_{i-1}\ \bar{\tau}''_{i+1}$$

so that

$\tau''_{i-1}\tau''_{i+1}$ and $\bar{\tau}''_{i-1}\bar{\tau}''_{i+1}$ have the same sign.

We may now freely scale x by an arbitrary positive factor. We choose to proceed as follows:

Define

$$\mu = \max_{1 \le i \le N} \{p_i h_i\}$$

and

$$\sigma = 150.;$$

then

$$\bar{x}_i = x_i/\alpha \quad \text{with} \quad \alpha \ge \frac{\mu}{\sigma}$$

$$\Rightarrow \quad p_i \bar{h}_i = \frac{1}{\alpha} p_i h_i \le \frac{\mu}{\alpha} \le \frac{\mu}{\mu/\sigma} = \sigma,$$

and we consequently remain within the desired number range. Whenever the p's are updated, a scaling should be done if $\mu > \sigma$. Note that when scaling is done then f'(a) and f'(b) should be multiplied by α.

In practice, an additional source of scaling is the need for a unique interpolant when a change of physical units is made. We now take up this matter.

Invariance Under Linear Transformations

As presently proposed, the exponential spline interpolant is not invariant under change of physical units. To ensure such invariance some sort of normalization must be performed on the input data.

Ordinates

Let $\bar{f}_i = k_1 f_i + k_2$, then $\bar{b}_i = k_1 b_i$ which $\Rightarrow \bar{\tau}_i'' = k_1 \tau_i''$. Therefore $\bar{\tau}(x) = k_1 \tau(x) + k_2$, as it should be. Hence, scaling of the ordinates is unnecessary.

Abscissae

Let $\bar{x} = k_1 x + k_2$. In this case, k_2 is impotent since only differences of abscissae appear in the exponential spline formulation. However, k_1 has a nonlinear effect on the interpolant. Hence, scaling of the abscissae is required. Setting $k_1 = 2/(x_{N+1} - x_1)$ and using this as a scale factor suffices.

Parametric Exponential Spline

In many applications f is not a single-valued function of x for the entire range of the data. Hence, it becomes desirable to fit both x and f versus some parameter, e.g. chordal length or arc length. We prefer the arc length formulation.

We proceed as follows:

(1) Determine the chordal length, s.

(2) Fit x and f versus s, producing exponential splines

x(s) and f(s), respectively.

(3) Determine modified arc lengths from

$$L_{\alpha}^{\beta} = \int_{\alpha}^{\beta} \left\{ [x''(s)]^2 + [f''(s)]^2 \right\}^{1/2} ds \ .$$

This integral can be evaluated numerically, say by a compound Simpson's rule.

(4) If arc lengths have changed appreciably then go to (2); otherwise the current x and f are the sought after parametric fits.

II.10. Examples

We next present a sequence of carefully selected examples. They are chosen to illustrate both the efficiency of the new parameter selection algorithm and the inherent superiority to cubic spline interpolation.

The first test case is taken from Späth's original paper [47]. The cubic spline interpolant, Figure II.10.1a, exhibits extraneous inflection points in the first, third, fourth and eighth intervals. The converged exponential spline interpolant, Figure II.10.1d, is seen to be free of such aberrations. The general behavior of our parameter selection scheme is amply portrayed in this example: The first iteration, Figure II.10.1b, captures the gross features while subsequent iterations, Figures II.10.1c-d, essentially "fine-tune" the first. We note that the scheme proposed by Späth required twelve iterations as opposed to our three iterations with no visible difference in the final interpolants.

The second test case is a unit impulse function. Note the "wiggles" present in the cubic spline interpolant, Figure II.10.2a. This example demonstrates the insensitivity to "outliers" that the exponential spline interpolant, Figure II.10.2b, possesses.

The third test case is a semicircle joined to two straight line segments in such a way as to produce discontinuities in the first derivative. This example begins to

implicate the cubic spline interpolant, Figure II.10.3a, as being deficient as a means of geometric representation. The exponential spline, Figure II.10.3b, on the other hand, performs ideally in this instance. Such a geometry would arise as the computational domain for flow past a cylinder.

The fourth test case is a quarter circle joined by a straight line segment with a discontinuous second derivative at their junction. Once again the cubic spline interpolant, Figure II.10.4a, falls far short of geometric requirements while the exponential spline, Figure II.10.4b, does not falter. If we were to try and calculate supersonic flow past this projectile, the cubic spline inflection point would likely induce a shock wave.

The fifth test case displays the critical sensitivity of the cubic spline interpolant, Figure II.10.5a, to the end conditions imposed. It is an additional advantage of the exponential spline interpolant, Figure II.10.5b, that it automatically compensates for poor end conditions thus restricting them to local influence. If this geometry were axisymmetric the cubic spline would surely replace the blunt body by a pointed one.

Furthermore, we note that preliminary tests have revealed wildly oscillating cubic spline interpolants in the presence of nonuniform knot placement. This defect appears to be absent from the exponential spline interpolant.

An additional comment deserves to be made at this point. That is, in the applications where cubic splines are typically used, a great deal of effort is generally expended in the smoothing ("fudging") of geometric data and in the preoccupation with end conditions. Both of these difficulties are peculiar to the cubic spline. In general, exponential spline interpolation obviates the need for such diversions.

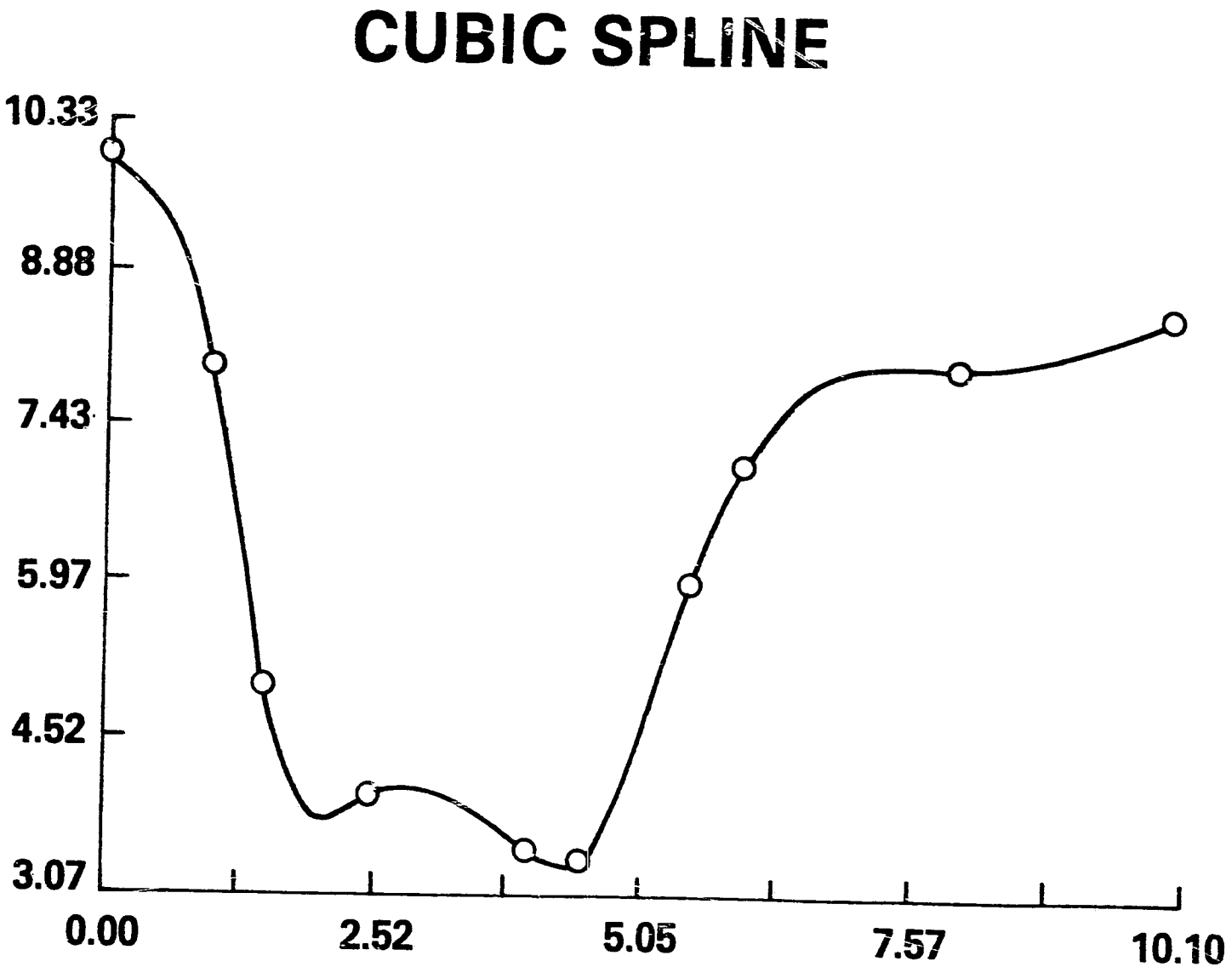


Figure II.10.1a. Späth Test Case (Iteration 0)

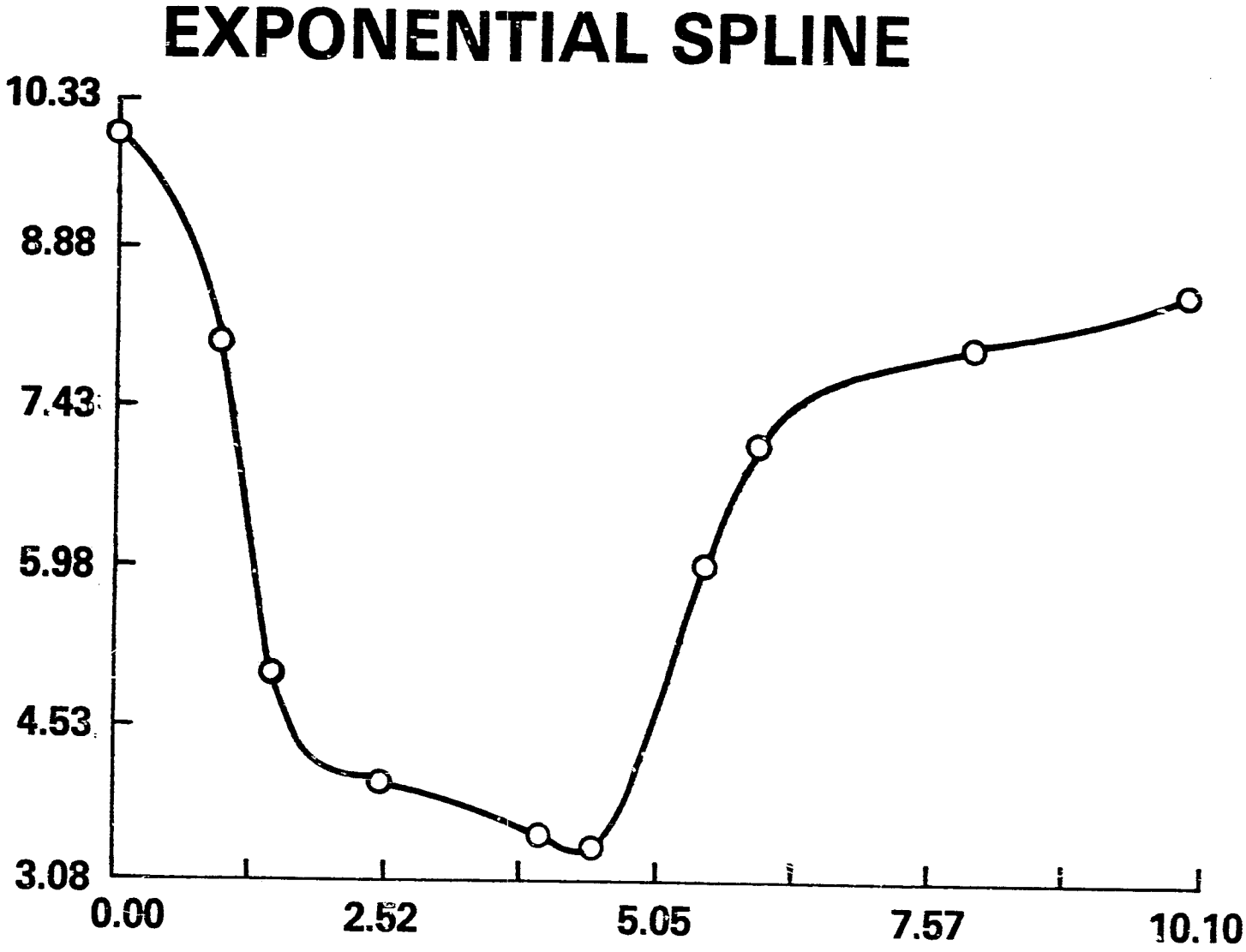


Figure II.10.1b. Späth Test Case (Iteration 1)

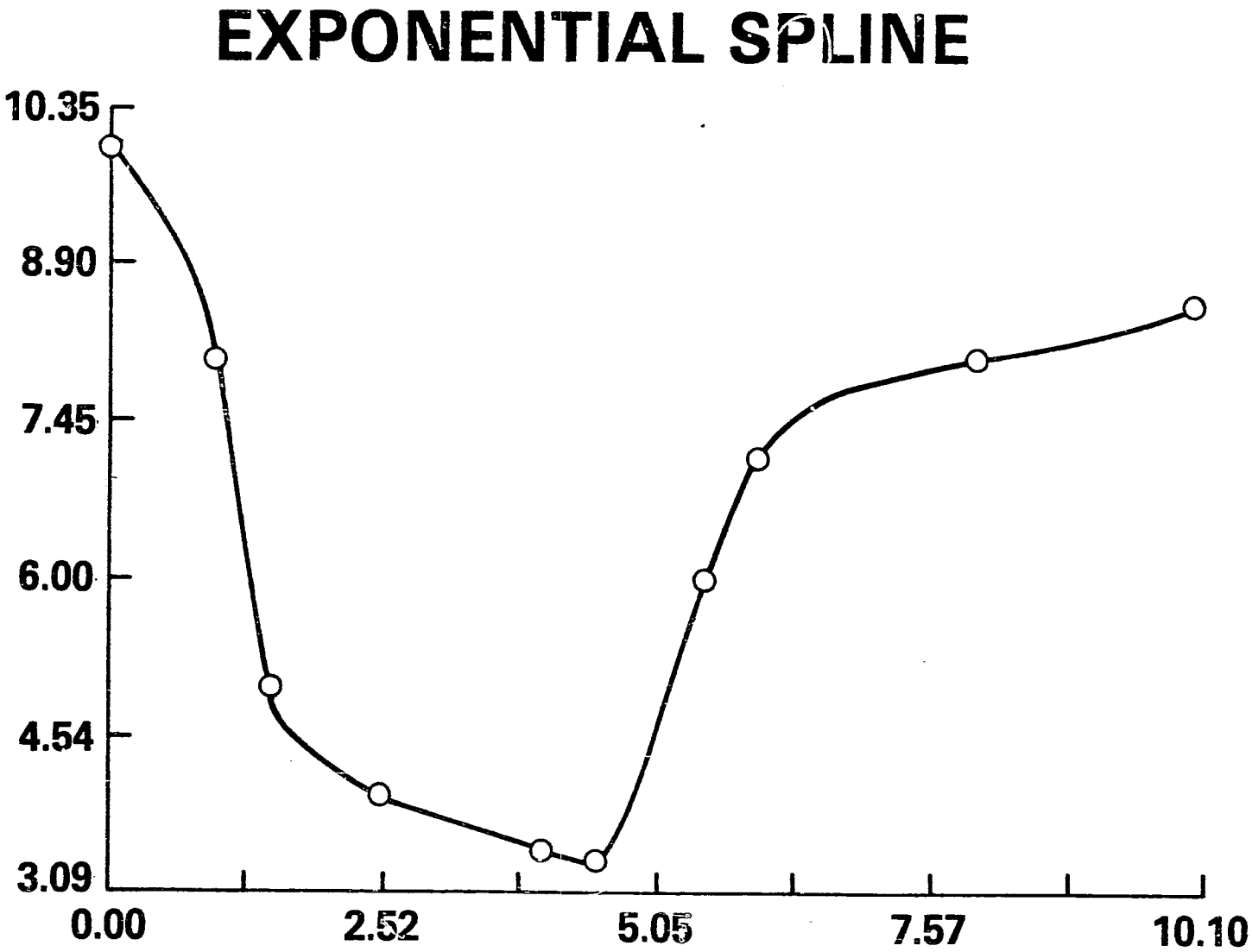


Figure II.10.1c. Späth Test Case (Iteration 2)

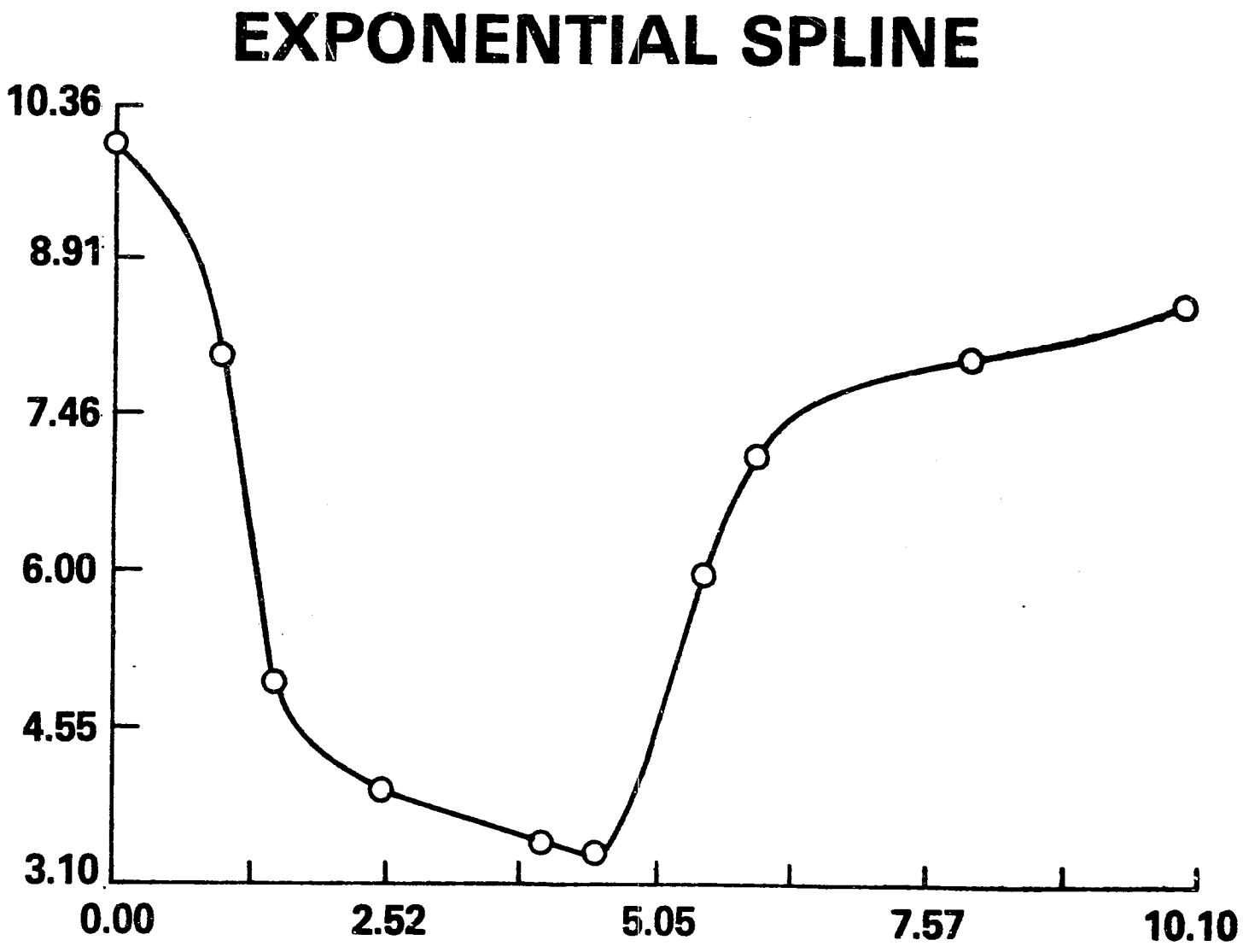


Figure II.10.1d. Späth Test Case (Iteration 3)

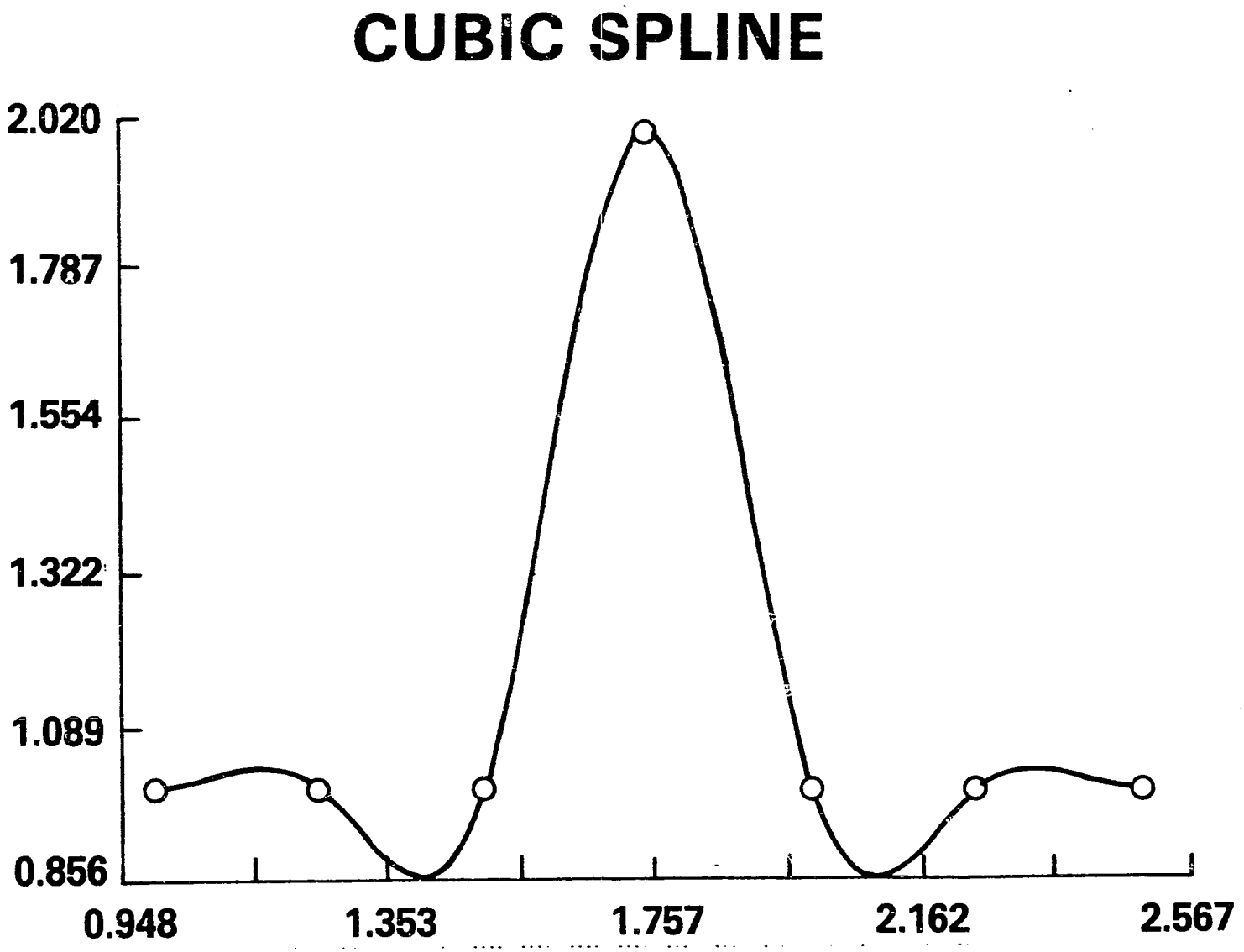


Figure II.10.2a. Outlier Test Case (Cubic Spline)

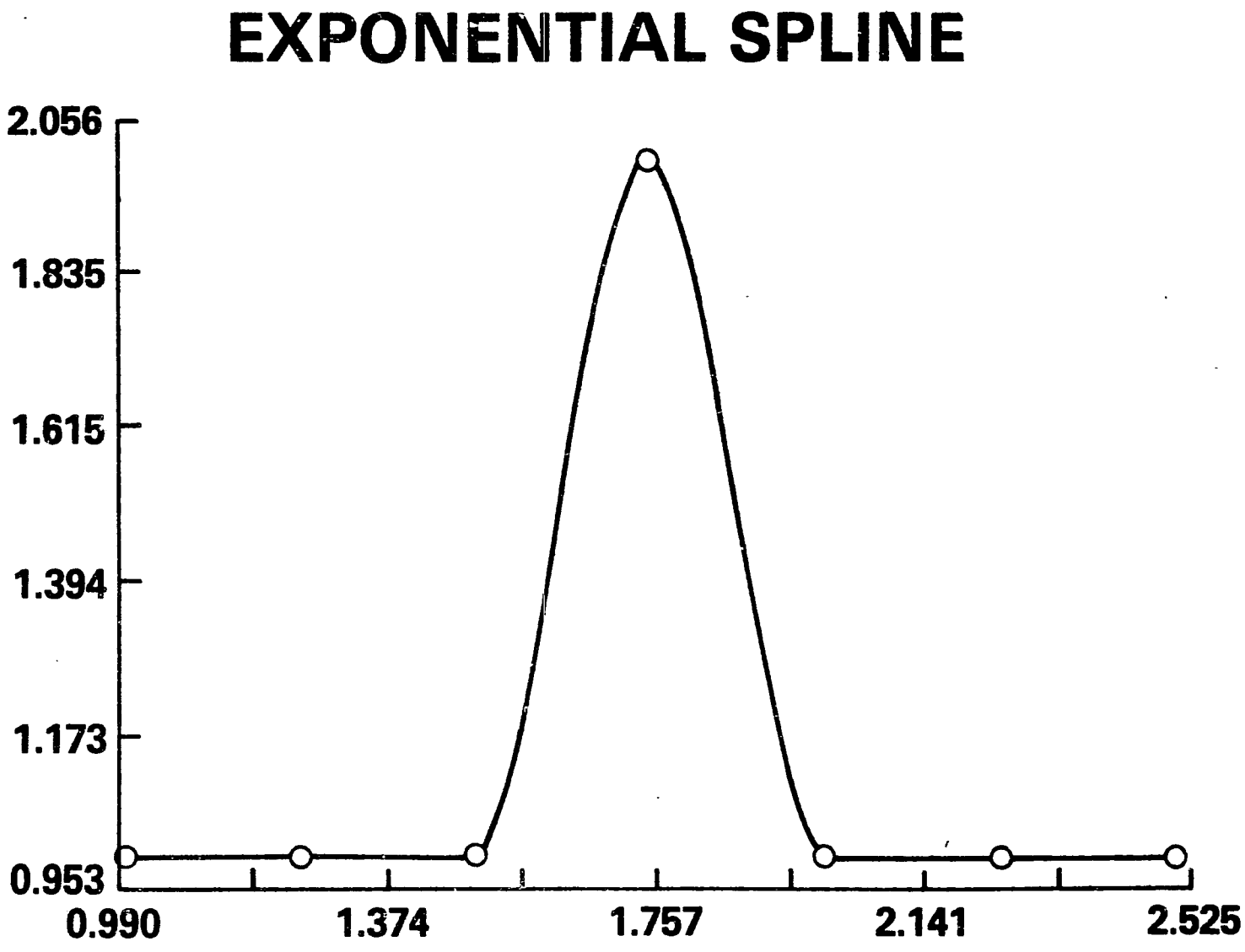


Figure II.10.2b. Outlier Test Case (Exponential Spline)

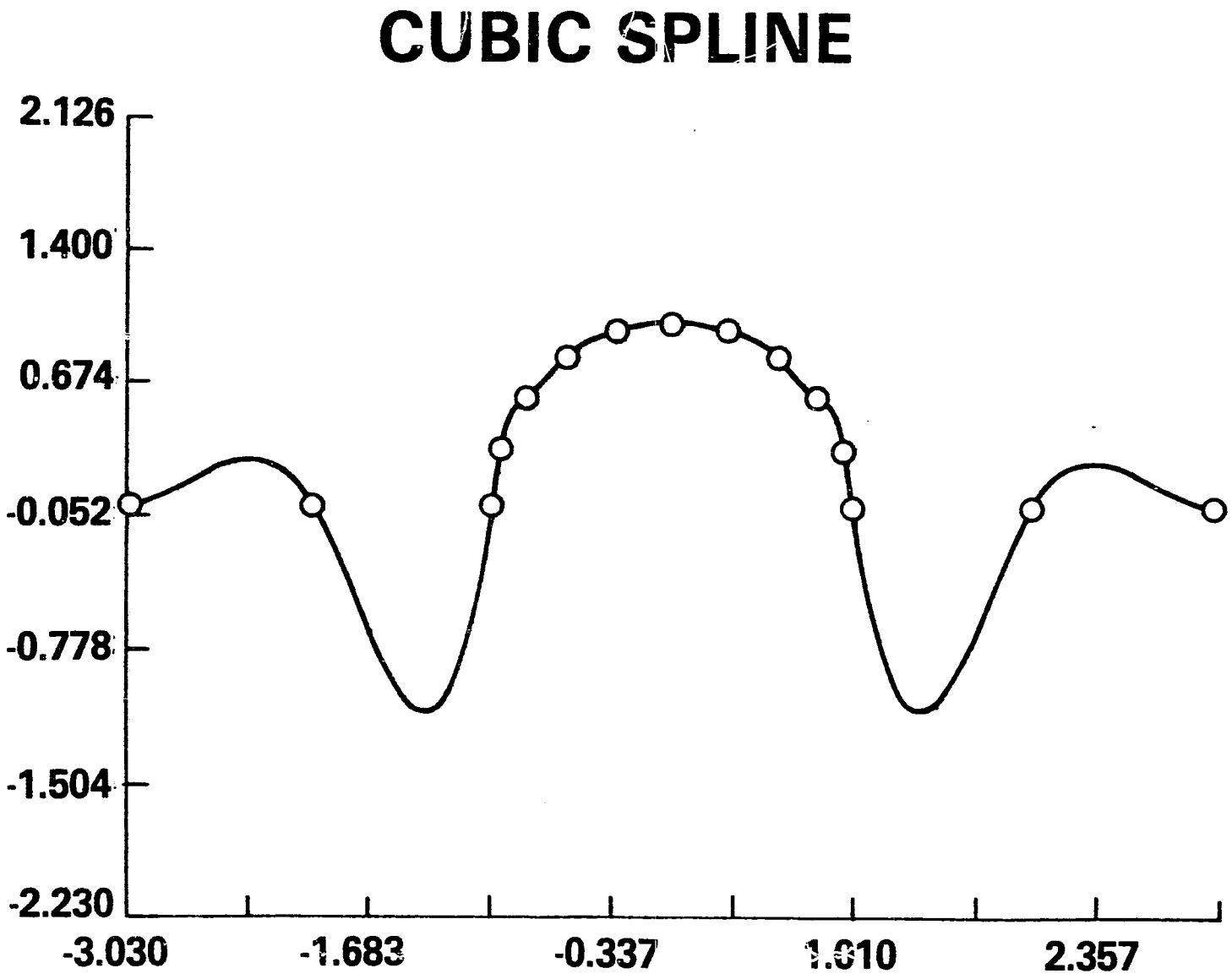


Figure II.10.3a. Slope Discontinuity Test Case (Cubic Spline)

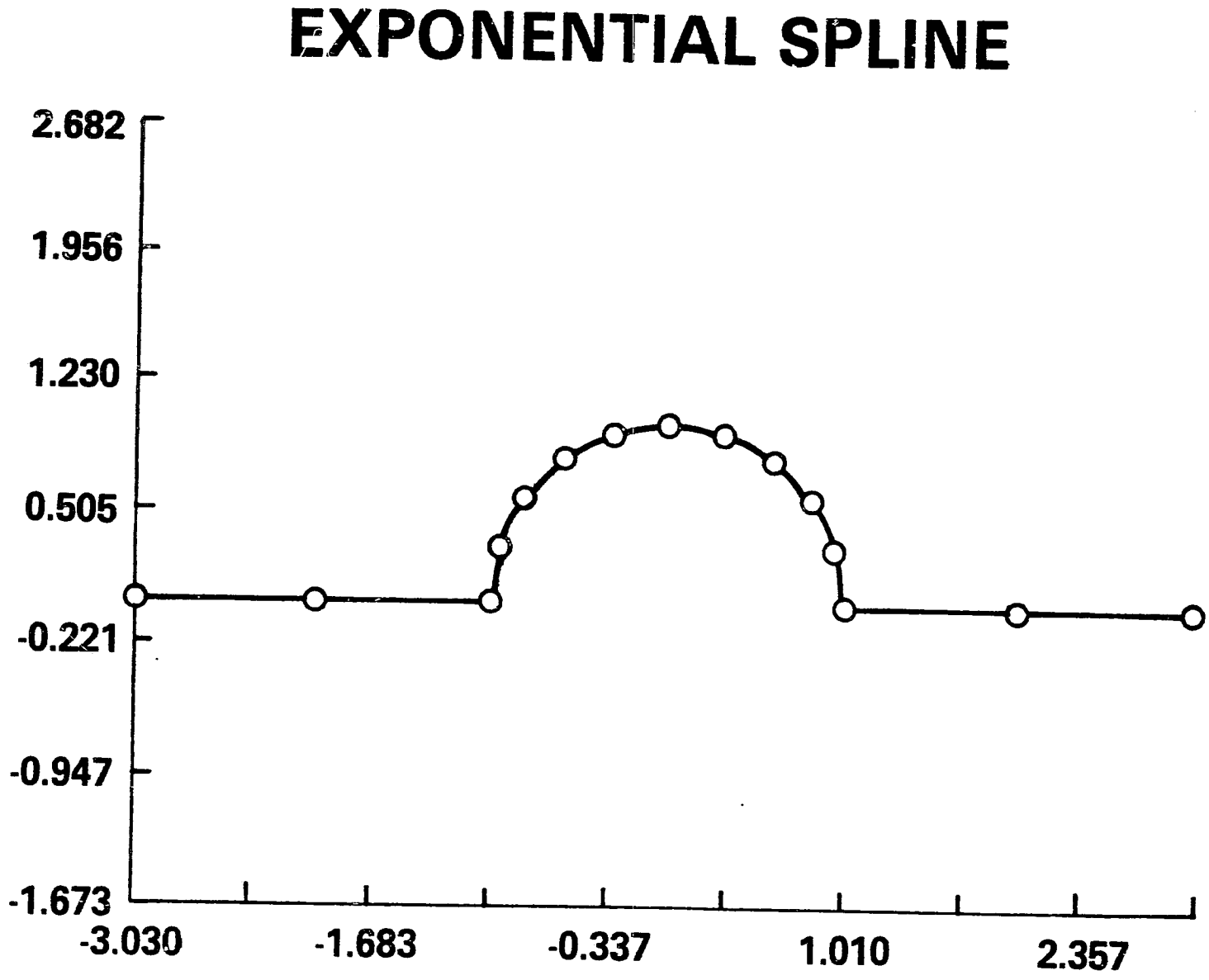


Figure II.10.3b. Slope Discontinuity Test Case (Exponential Spline)

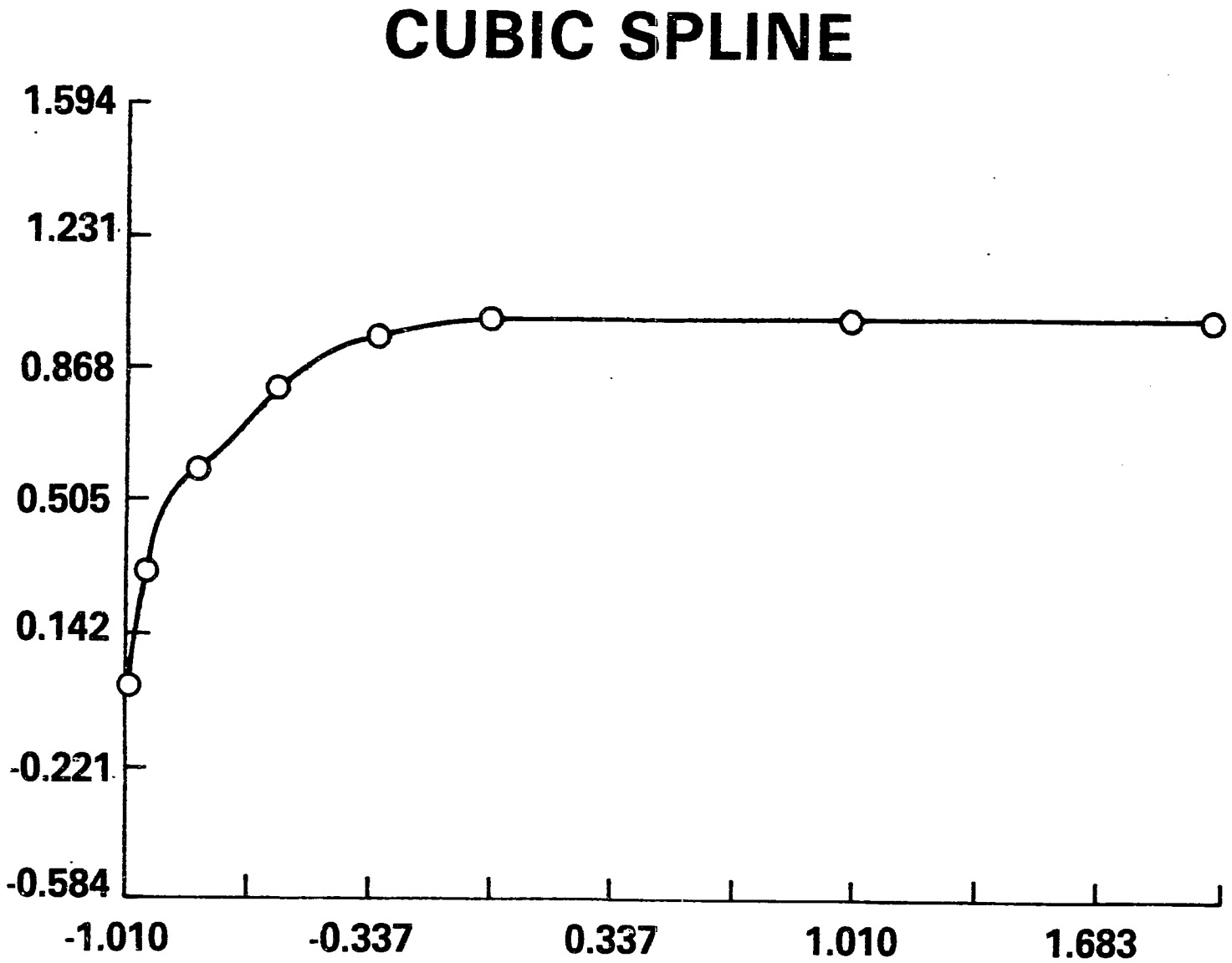


Figure II.10.4a. Curvature Discontinuity Test Case (Cubic Spline)

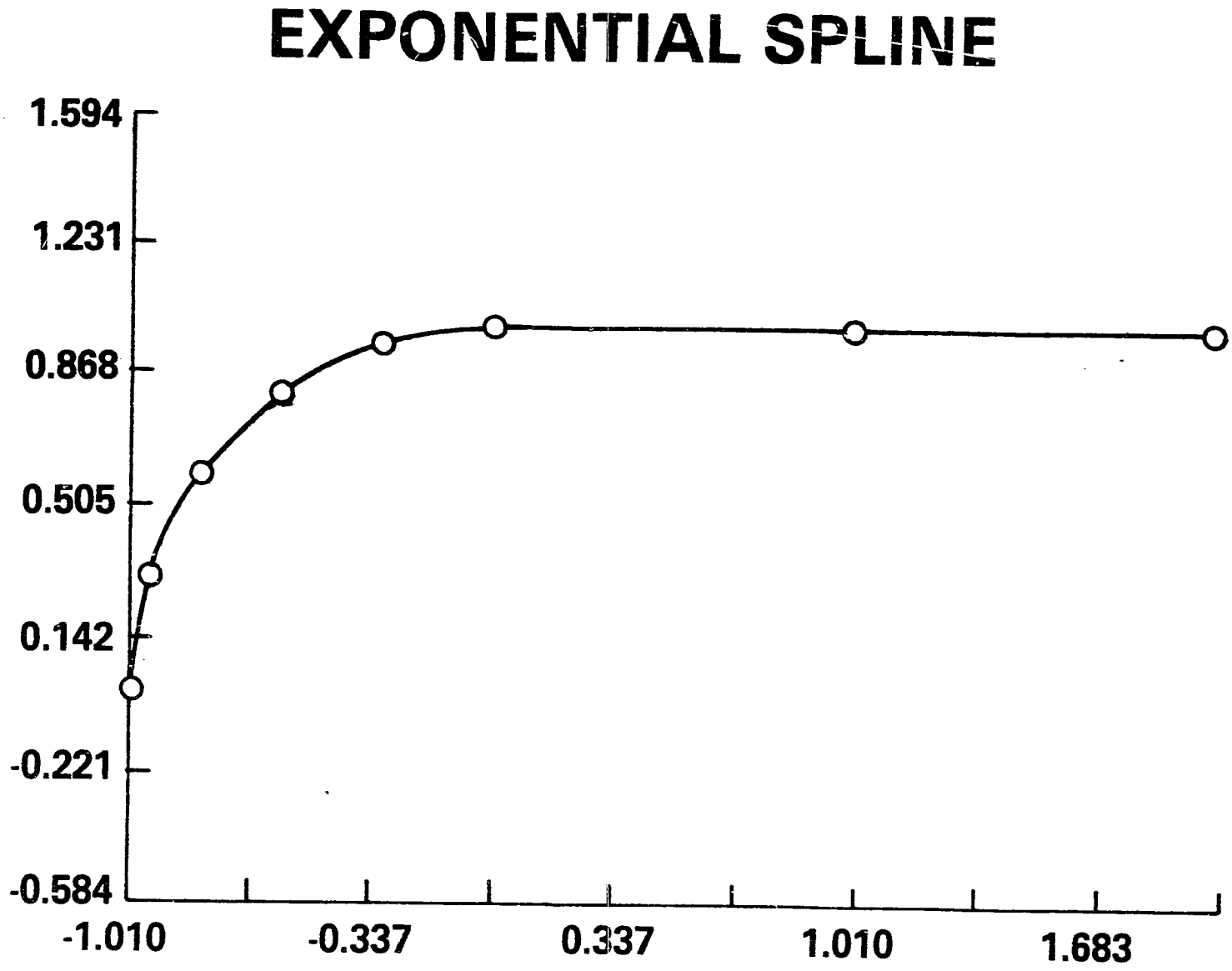


Figure II.10.4b. Curvature Discontinuity Test Case (Exponential Spline)

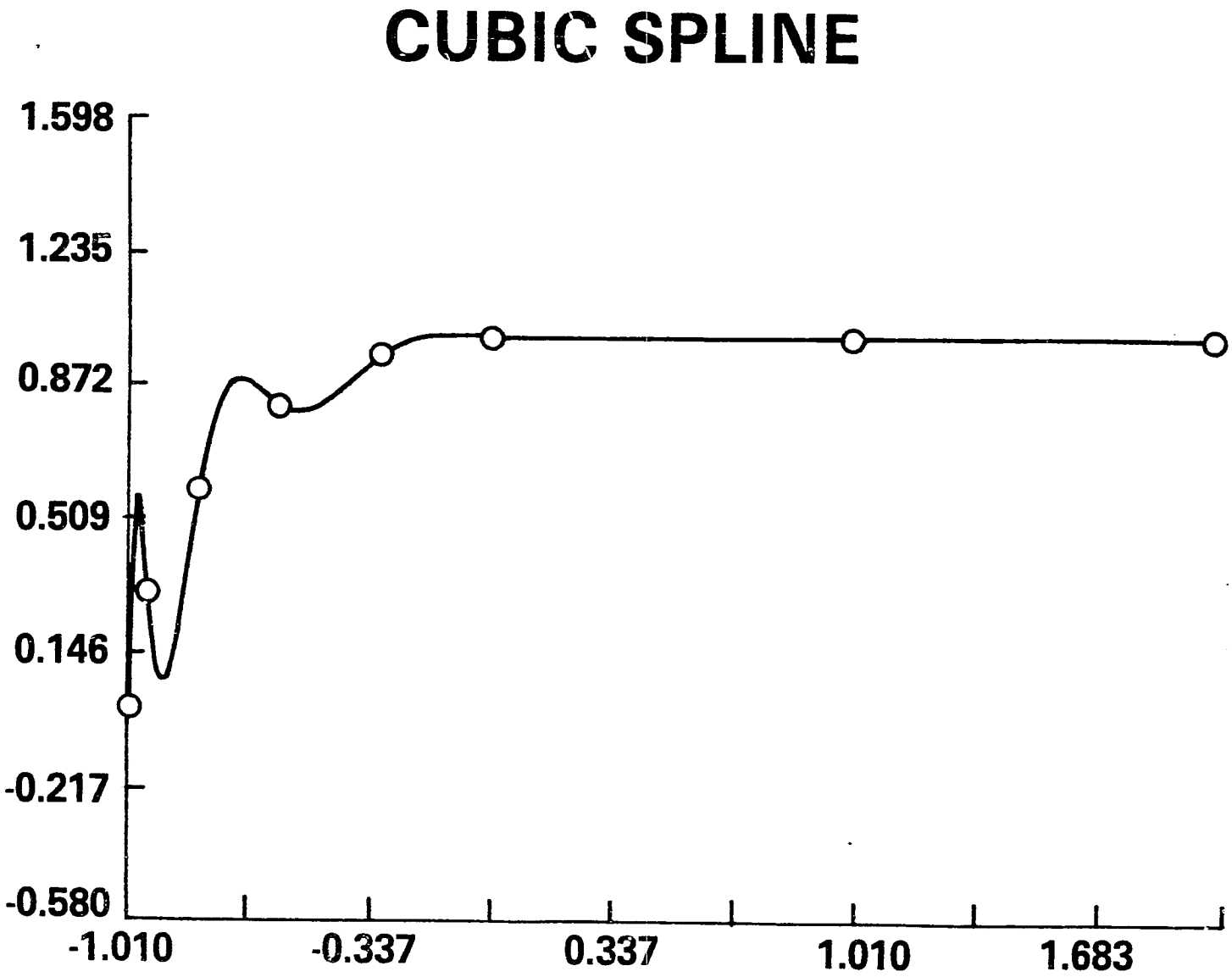


Figure II.10.5a. Vertical Tangent Test Case (Cubic Spline)

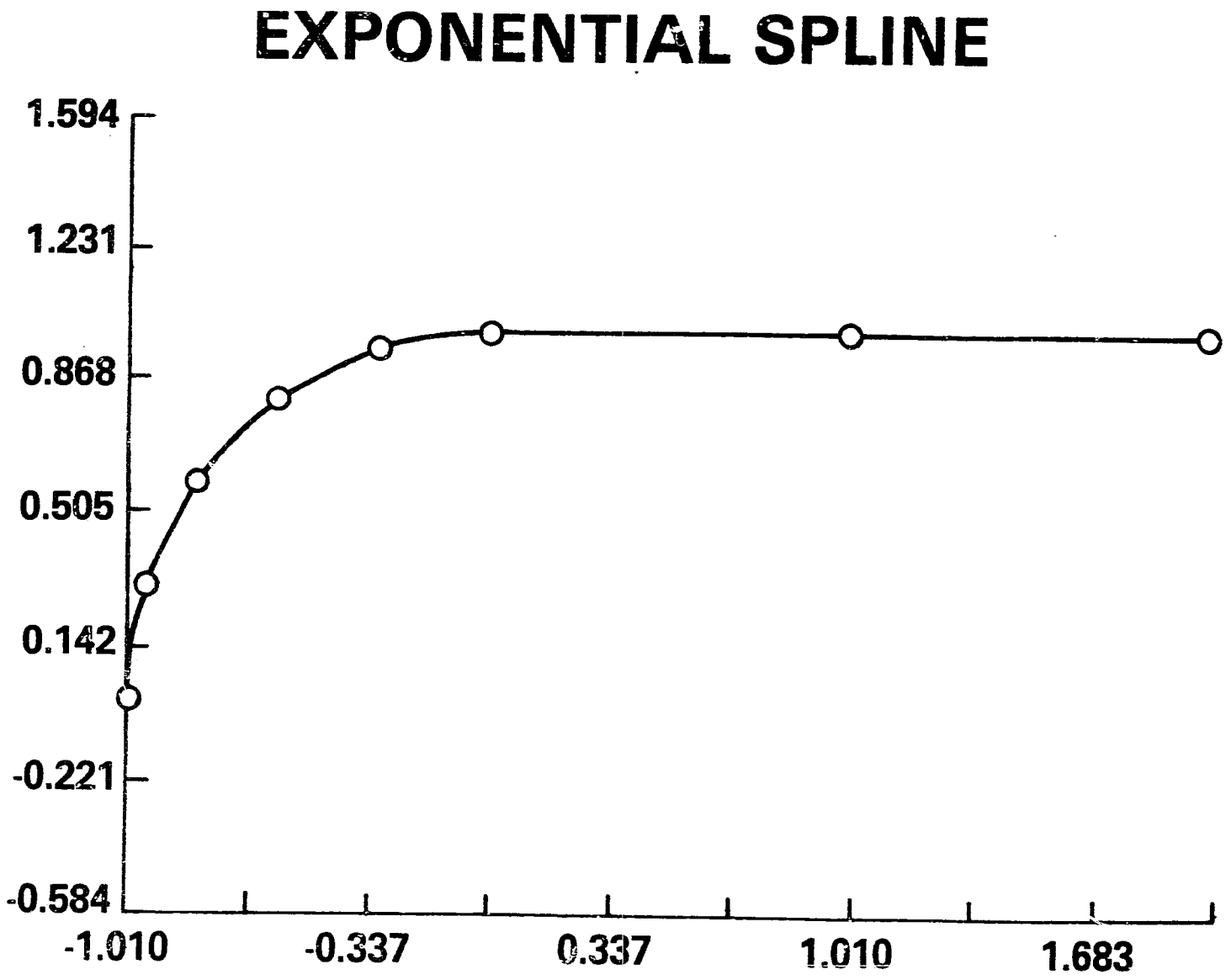


Figure II.10.5b. Vertical Tangent Test Case (Exponential Spline)

Part III: Application of Exponential Splines

III.1. Overview

In this third part we take up the application of our preceding work on exponential splines. We begin by surveying the utility of such splines in some geometric and data fitting problems occurring in computational fluid dynamics. The discussion here is somewhat general with the exception of a detailed boundary layer calculation. The numerical solution of the Laplace, heat, and wave equations is then addressed. Our aim in the final parts of the chapter is to provide a treatment of nonlinear systems of hyperbolic conservation laws and in particular to simulate shock waves without introducing wiggles. We first take up a linear model problem. In this context we analyze the stability and accuracy of a variety of numerical schemes. Nonlinear problems are introduced via the inviscid Burger's equation. At this juncture we generalize the Lax-Wendroff theorem concerning convergence of approximations to weak solutions to the case of our spline scheme. The classical Riemann problem for an infinite shock tube is then approached using this scheme. Finally, the scheme is extended to multi-dimensions and applied to the two-dimensional Euler equations. Specifically, we provide numerical simulation of transonic flow in a channel.

III.2. Geometric Applications in Computational Fluid Dynamics

As if the attendant mathematical difficulties were not inherently severe enough, the calculation of fluid flow fields invariably requires, at one stage or another, the construction of smooth curves from a prescribed set of tabulated data points. Examples that immediately come to mind are the definition of body geometry, the numerical enforcement of boundary conditions and the generation of a computational mesh. Hence, reliable interpolation procedures must be developed. In what follows, we restrict our attention to the two dimensional case for ease of exposition. The techniques developed have no such intrinsic limitation.

The remainder of this section presents a wide variety of applications of exponential splines in computational fluid dynamics. The emphasis throughout is on areas of application where the use of cubic splines would result in a significant degradation if not an utter breakdown of the computation.

The utility of exponential splines as a means of geometric representation has already been intimated. The resulting loss of accuracy in the numerical prescription of initial and/or boundary conditions could distort the solution throughout the entire flow field. In a boundary conforming computational mesh, any abnormalities along the boundary will propagate into the grid. Smith and Wiegel [45]

and Eiseman and Smith [16] have advocated the use of hyperbolic splines in this context. This approach requires that uniform tension be applied, thus resulting in unnecessarily kinky curves.

Along these same lines, conformal mapping techniques often require ex post facto stretchings to concentrate grid points in regions of high gradients. Such C^2 shearings may be conveniently constructed and controlled by employing exponential splines. Figure III.2.1 displays such a stretching function while Figure III.2.2 illustrates its application to a channel-with-bump geometry.

In a slightly more general vein, we recall that strict adherence to convexity constraints is required by many existence [8] and uniqueness [18] results in the theory of plane subsonic fluid flow. It is also well known that in supersonic flows, there is an inherent difference between concave and convex geometries [12]. These considerations imply that convexity must be preserved in the discrete analogues to these problems, e.g., through the use of exponential splines.

We now present the application of our exponential spline code to provide smooth input data for the integration of the boundary layer equations and to prevent failures of the integration scheme due to oscillations in the input. Specifically, we consider the transonic flow (peak Mach number $\tilde{\geq}$ 1.2) over a compressor cascade with 10 degrees of

flow turning as supplied by Korn [26] (see Figure III.2.3). The computed velocity distribution along the upper surface (obtained from a hodograph method) is shown in Figure III.2.4. Fifteen points were selected and fit with both cubic and exponential splines (see Figure III.2.5). The cubic spline oscillates wildly in the vicinities of the supersonic zone and the trailing edge.

The resulting velocity distributions were then input to the STAN5 finite difference boundary layer code [13]. The anomalies present in the cubic spline interpolant cause the boundary layer to separate at the leading edge overspeed thus halting the calculation. On the other hand, the well-behaved nature of the exponential spline interpolant allows the calculation of a fully attached flow.

In order to obtain a more substantial comparison we include 4 additional points on the velocity schedule (see Figures III.2.6 and III.2.7). These are then used as input to STAN5 (see Figures III.2.8 and III.2.9). The output, displayed in Figures III.2.10 and III.2.11, clearly indicates that the cubic spline can produce nonphysical features in the computed flow field.

In contrast, the exponential spline, because of its stricter adherence to the features of the data, is seen to be more reliable. One can well imagine the additional complications which the presence of a shock wave would entail.

Obviously, the use of an extremely fine mesh would rectify the situation. However, a high cost could be incurred in the form of vastly increased computation time. This is especially true when the boundary layer calculation is not being performed a posteriori but rather as part of an inviscid-viscous iteration. Furthermore, the use of the exponential spline would allow the specification of less data than the cubic spline in an inverse design procedure.

We note that the role of exponential splines in the direct numerical solution of singular perturbation problems has recently been investigated by Flaherty and Mathon [17].

In summary, the examples considered indicate that the cubic spline may cause difficulties in a broad spectrum of practical applications. In the past, the lack of a satisfactory scheme for tension parameter selection has hindered the widespread use of the exponential spline as a viable alternative. However, our exponential spline tension parameter selection algorithm has removed this impediment, thus paving the way for it to become an important tool of the computational fluid dynamicist.

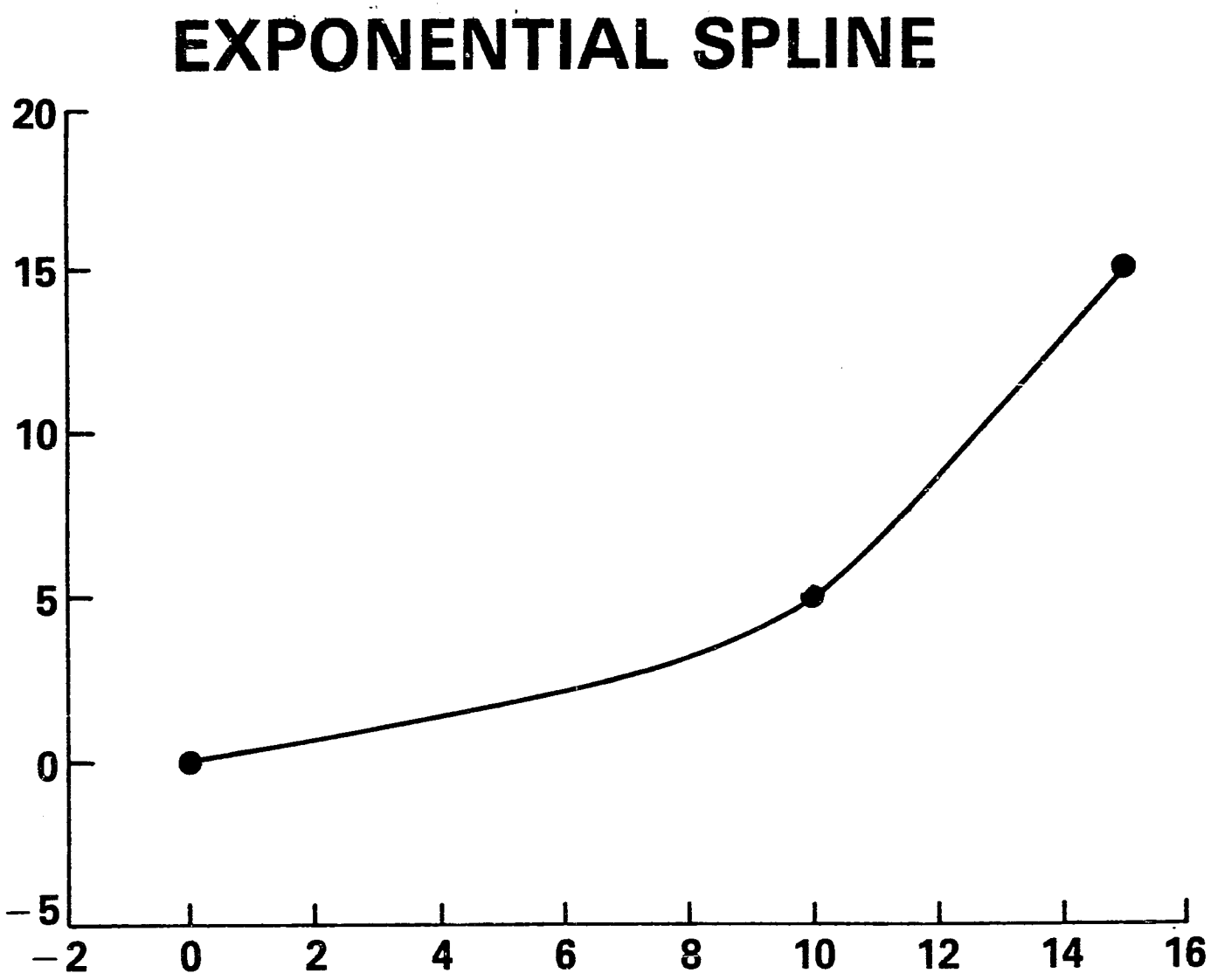


Figure III.2.1. Stretching Function

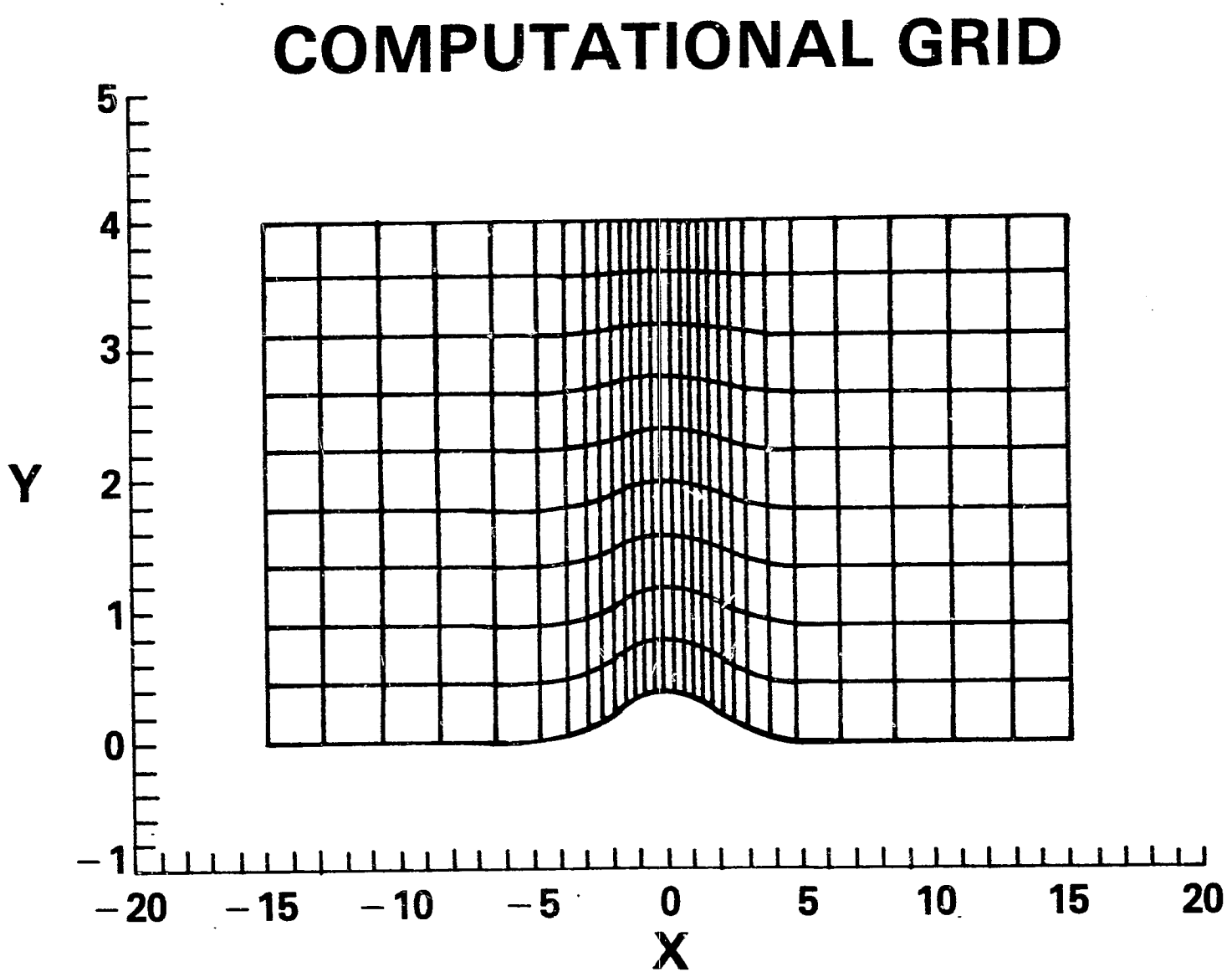


Figure III.2.2. Stretched Mesh

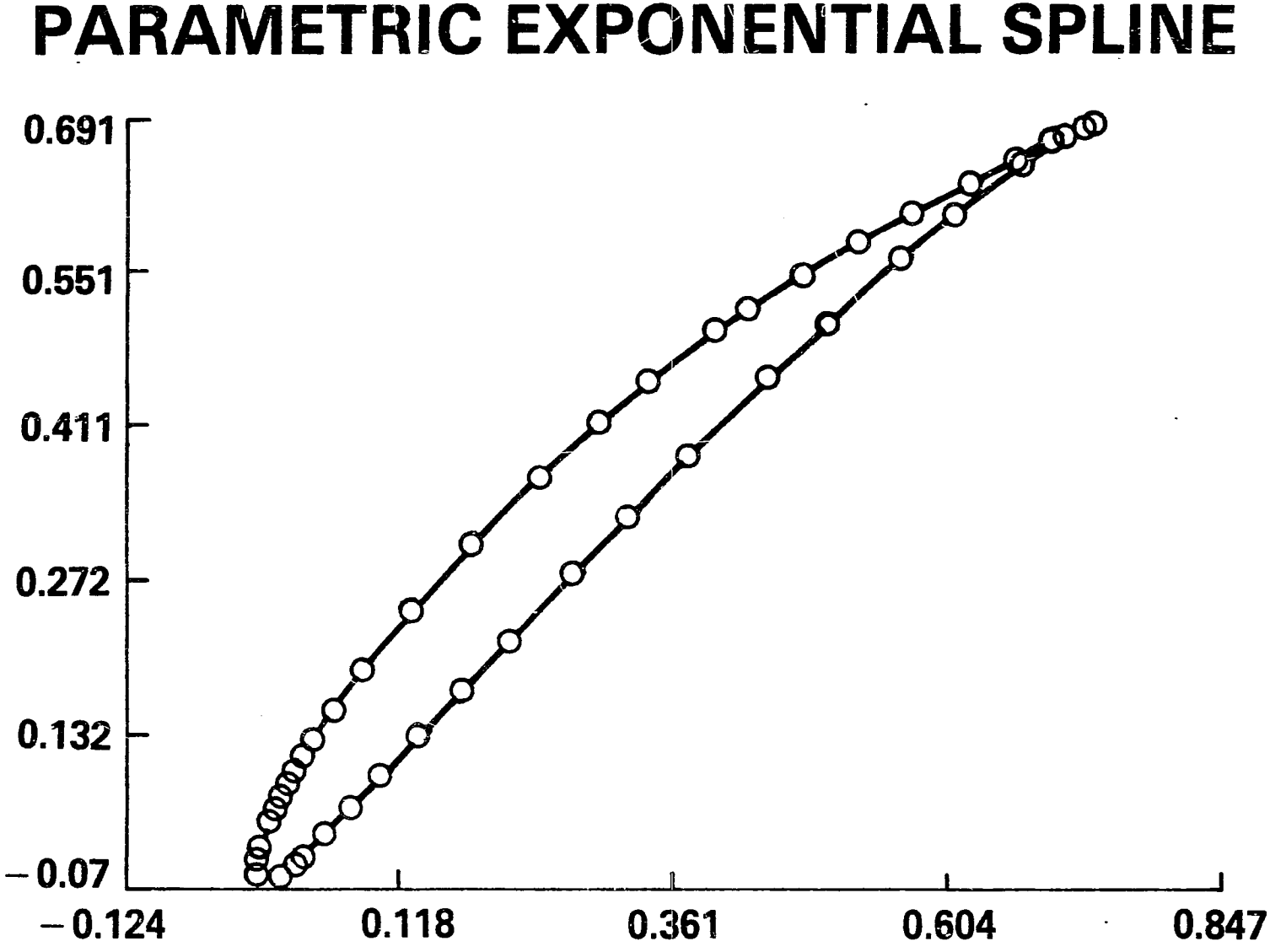


Figure III.2.3. Korn Cascade Airfoil

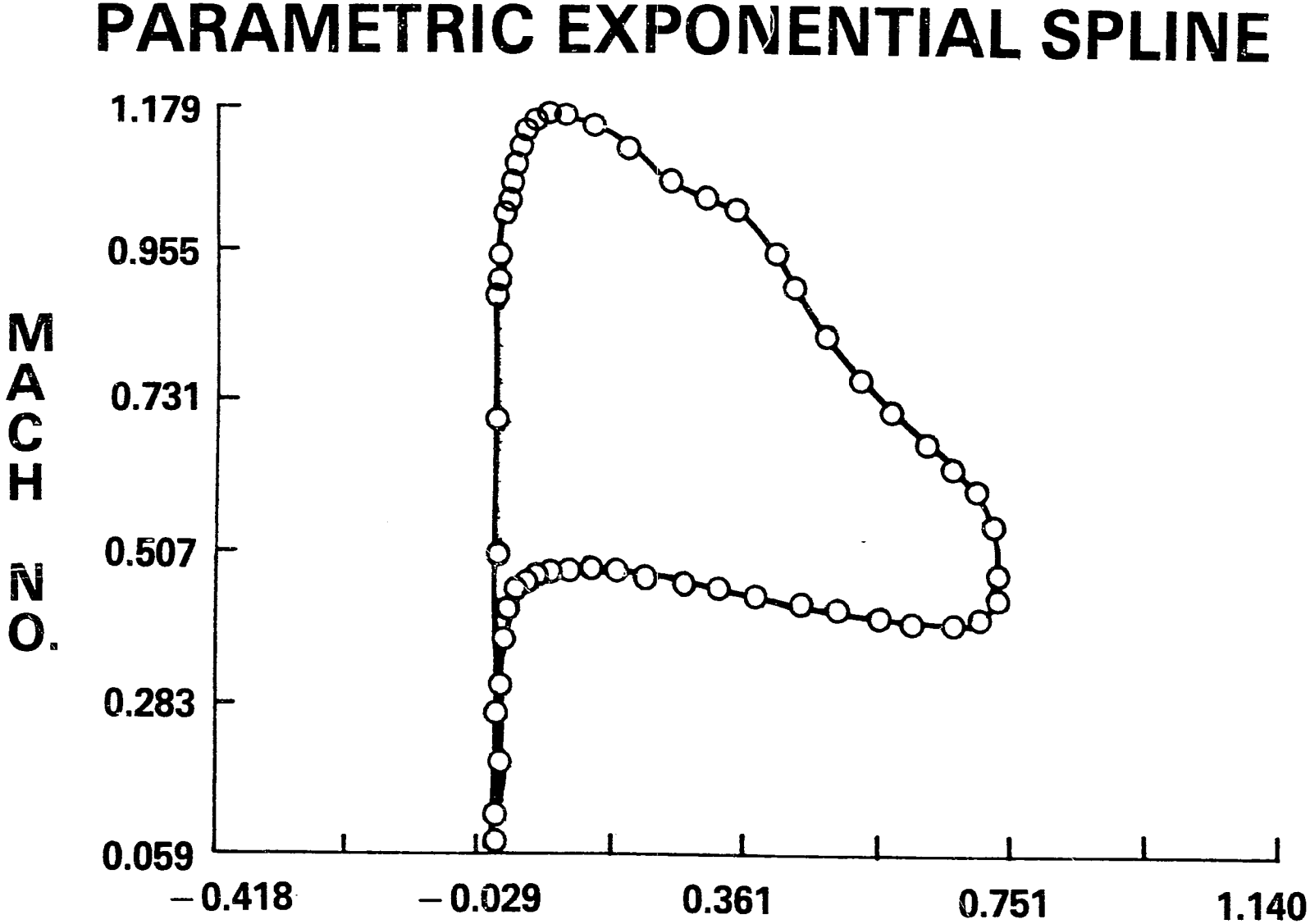


Figure III.2.4. Computed Mach Number Distribution

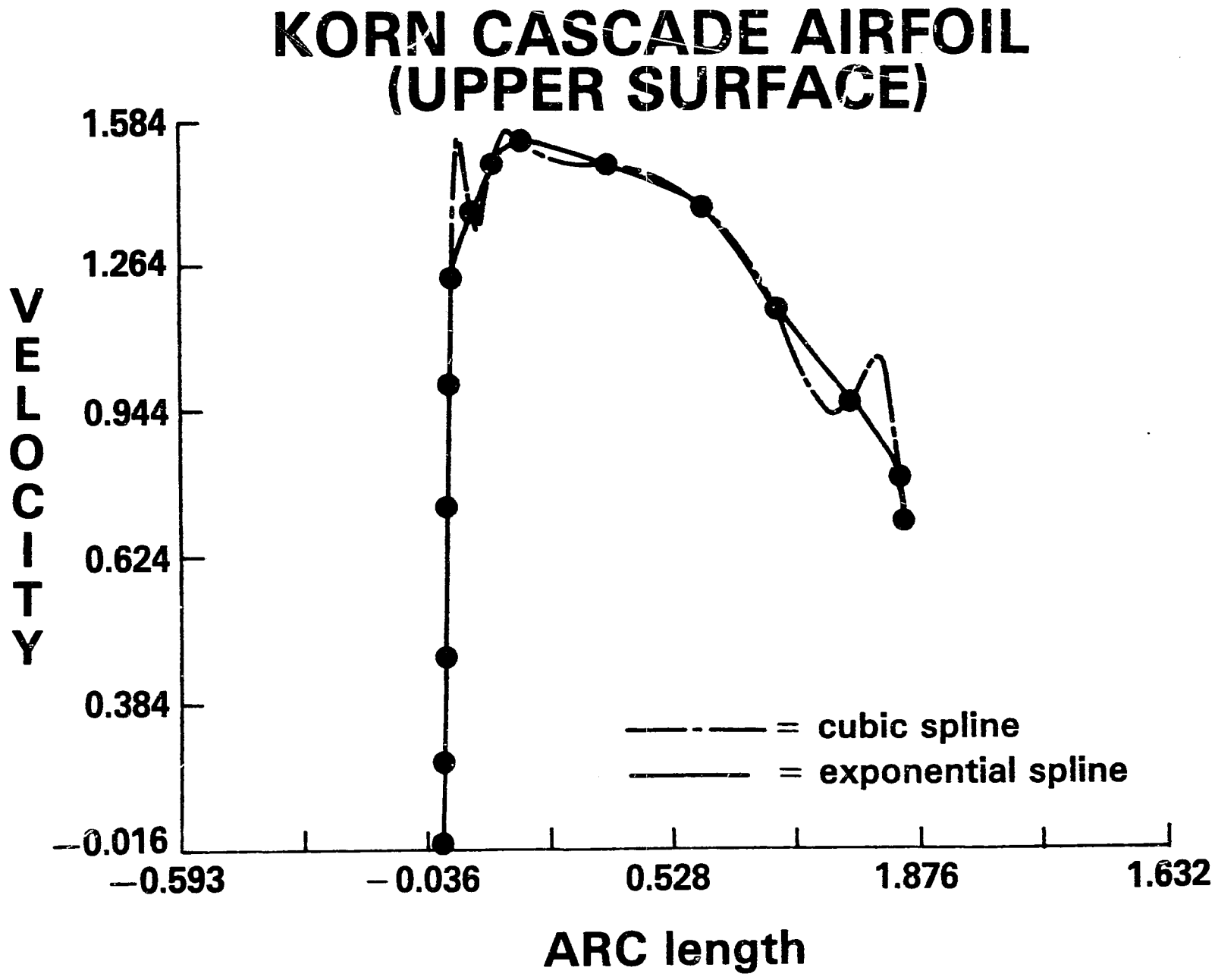


Figure III.2.5. Velocity Schedules (15 points)

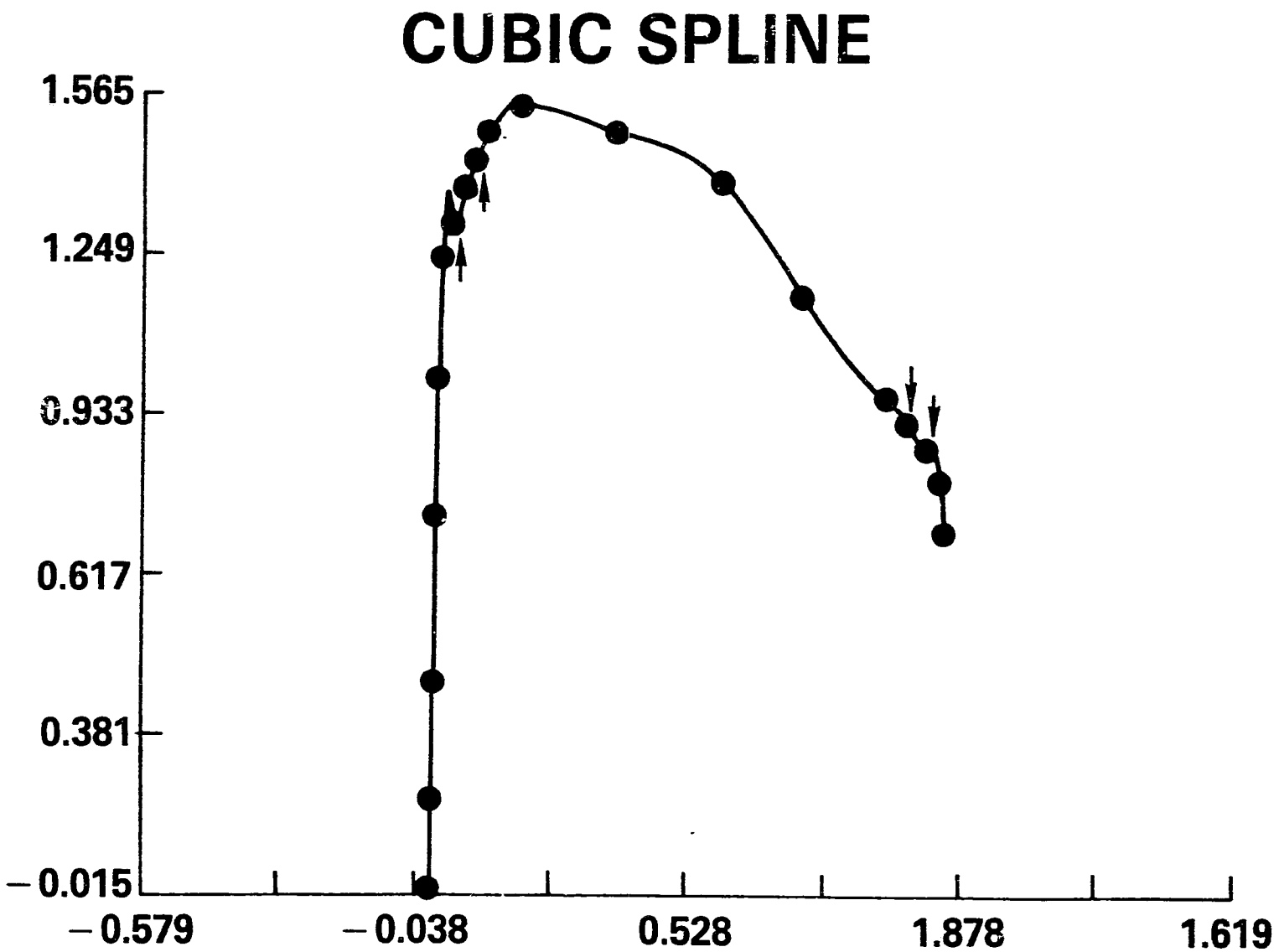


Figure III.2.6. Cubic Spline Velocity Schedule (19 points)

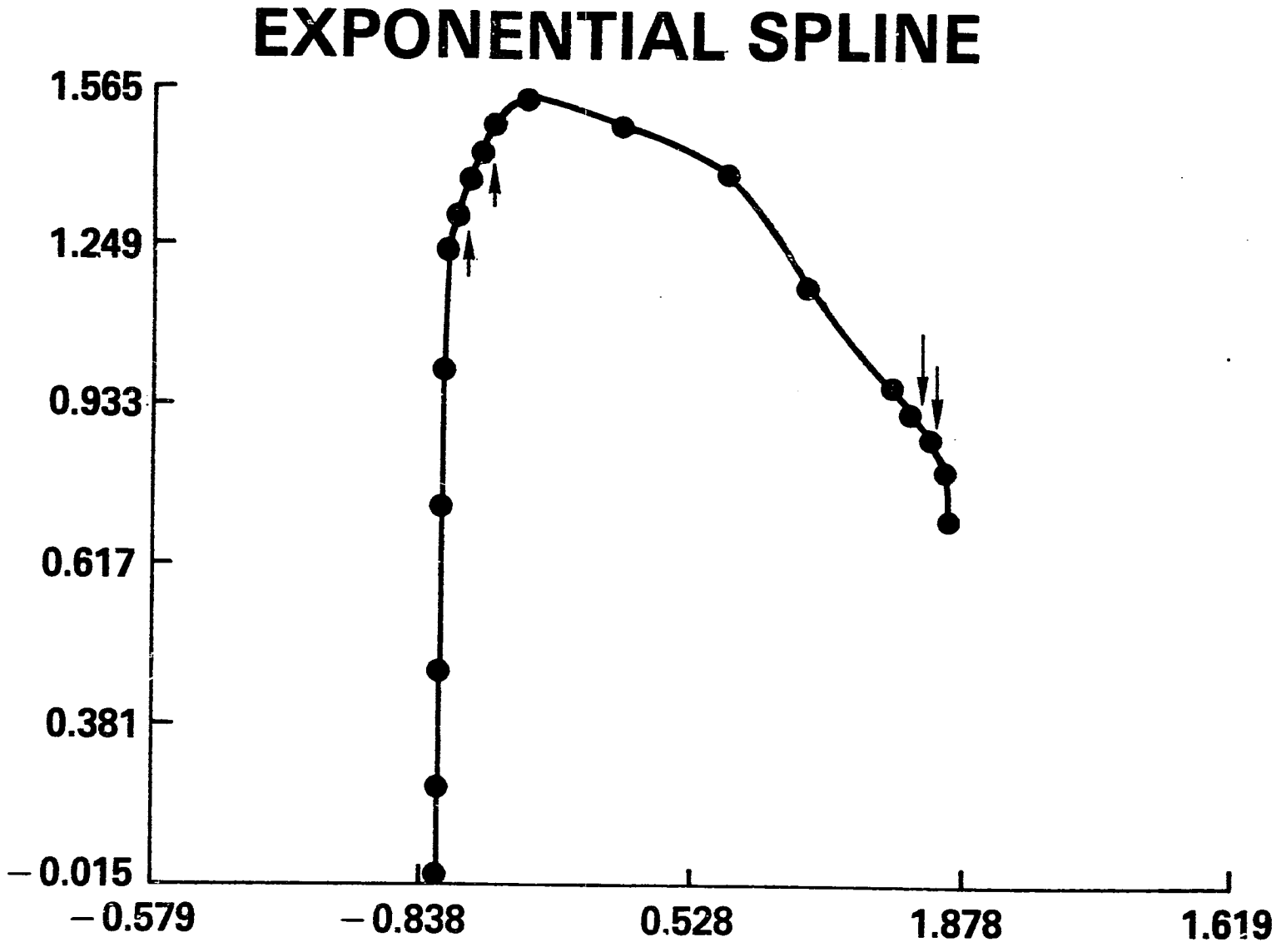


Figure III.2.7. Exponential Spline Velocity Schedule (19 points)

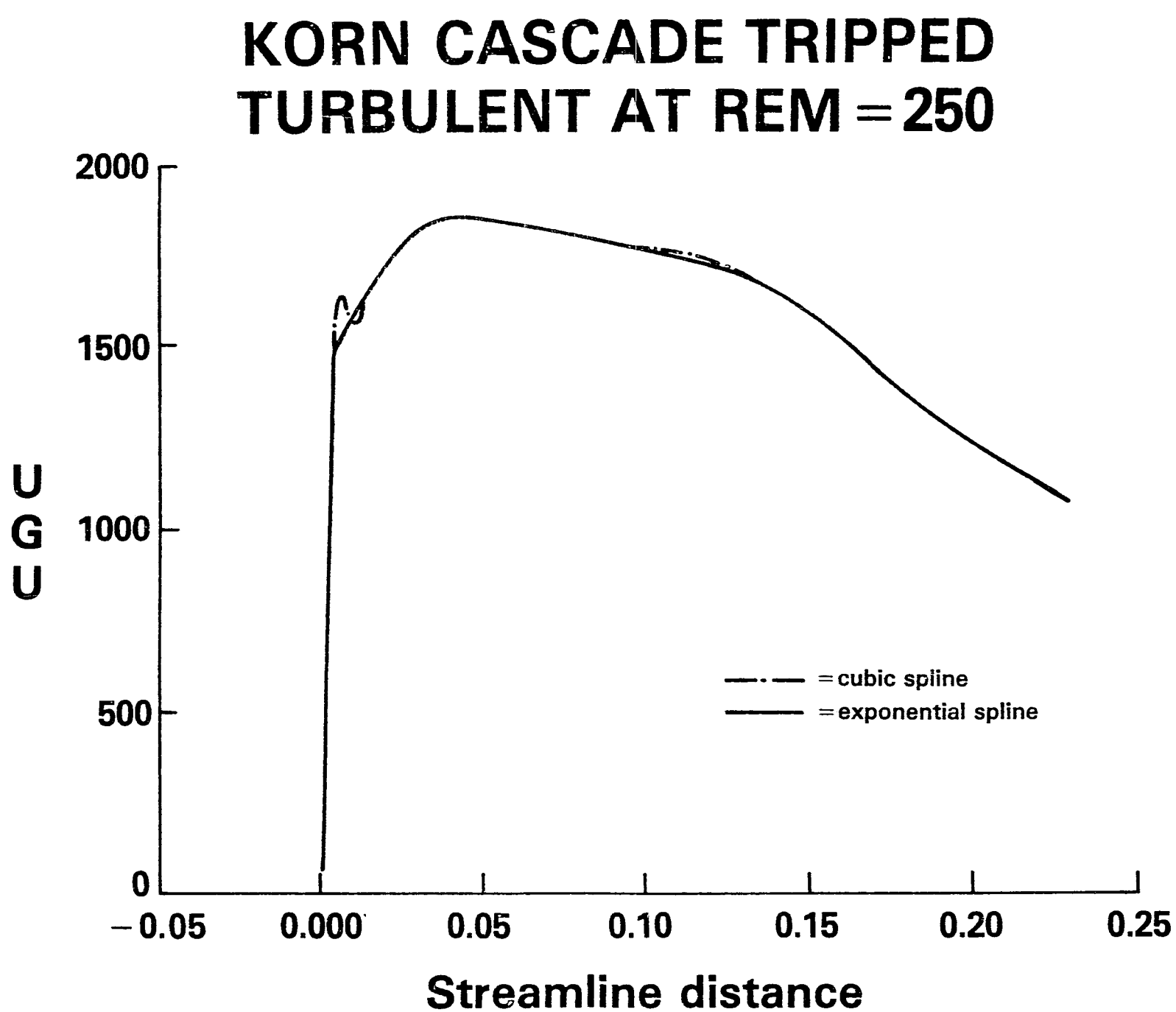


Figure III.2.8. Input Speed Distributions (19 points)

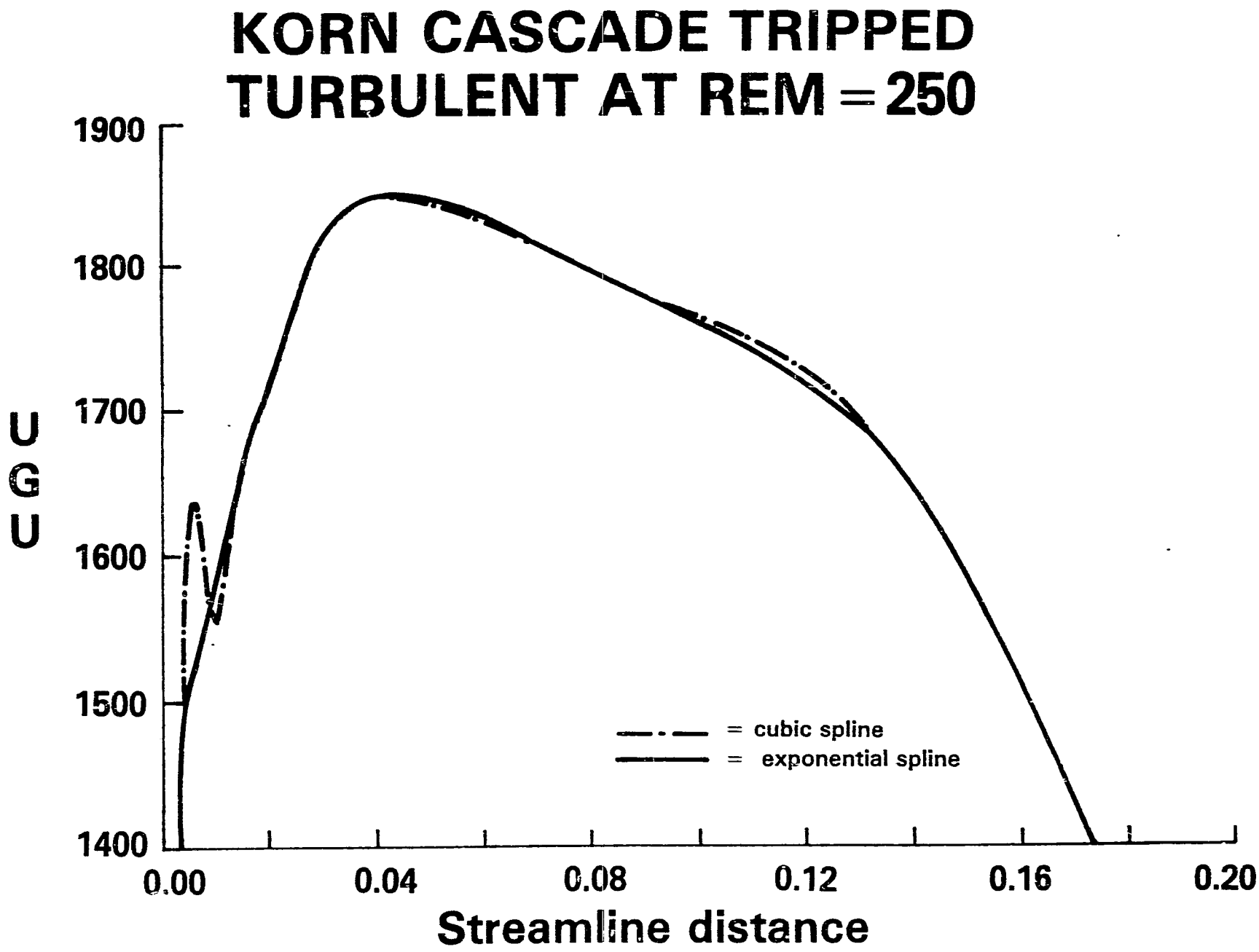


Figure III.2.9. Blowup of Regions of Variation

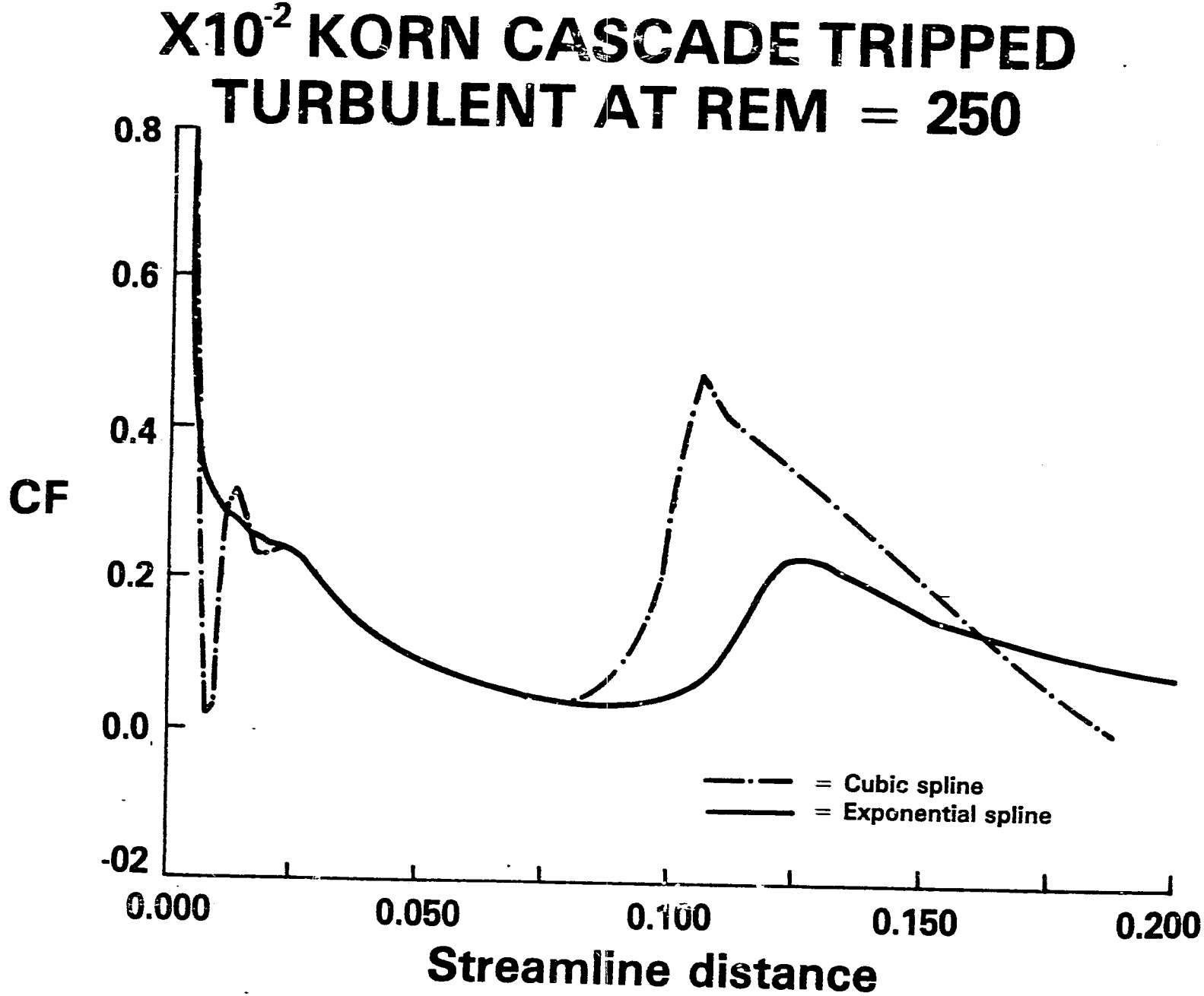


Figure III.2.10. Skin Friction Coefficient

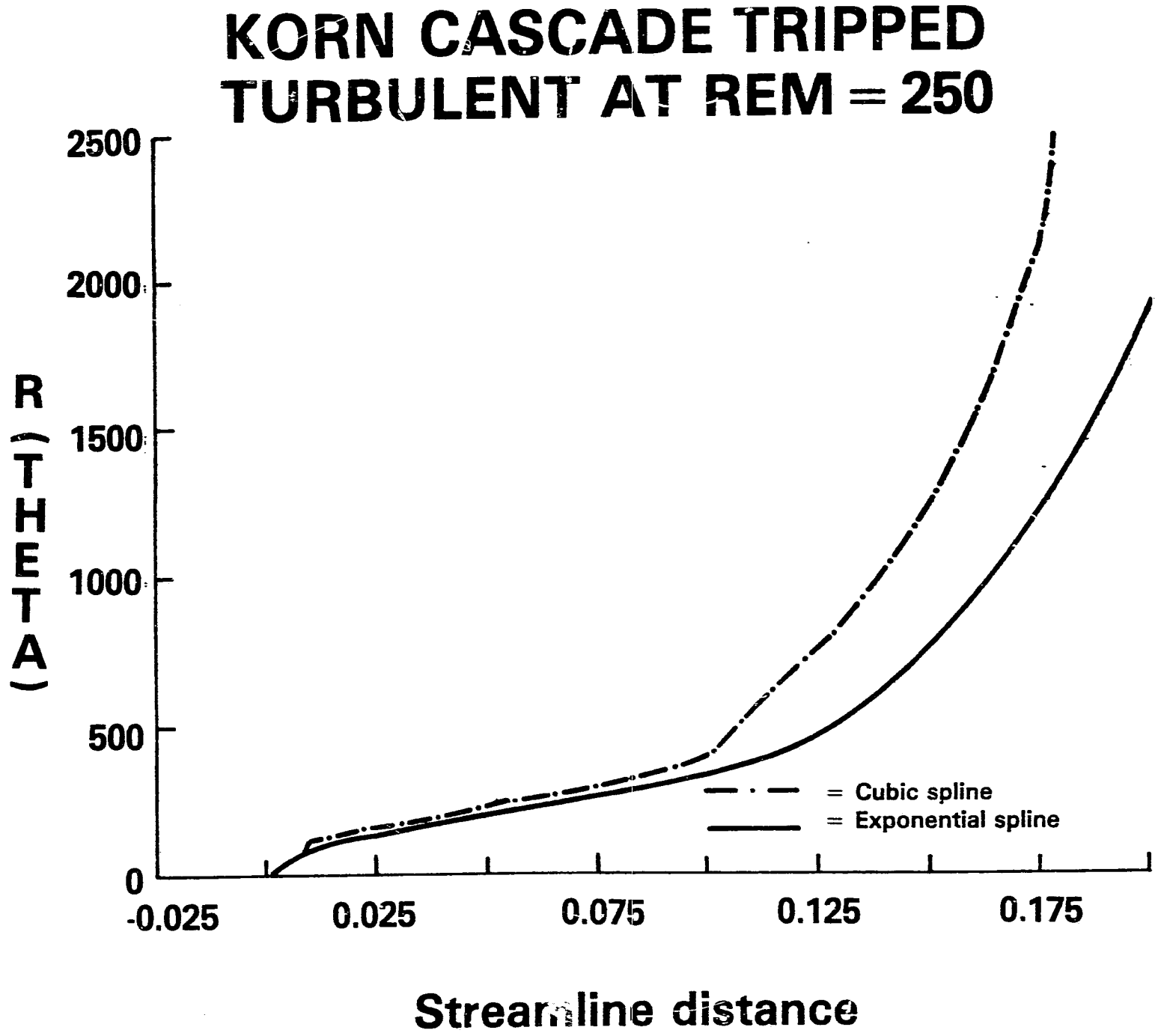


Figure III.2.11. Momentum Thickness Reynolds Number

III.3. Approximation of Classical Partial Differential Equations

In this section we present explicit approximate techniques for the numerical solution of second order linear partial differential equations. Basically, spatial derivatives are replaced by the derivatives at nodes of interpolatory splines. The cases of elliptic, parabolic and hyperbolic equations [18] are considered in turn. Note that these schemes have straightforward implicit counterparts. Our aim here is to lay the ground work for more ambitious applications to nonlinear problems.

Laplace's Equation

To illustrate the use of splines in the solution of elliptic boundary value problems we treat the Laplace equation

$$u_{xx} + u_{yy} = 0$$

in the unit squre $R = [0,1]^2$ with Dirichlet boundary conditions

$$u = f \quad \text{on} \quad \partial R.$$

We perform our calculation on a rectangular mesh of dimensions $(\Delta X, \Delta Y)$. First define

$$F(X,Y) = f(X,0)(1-Y) + f(X,1)Y + f(0,Y)(1-X) + f(1,Y)X$$
$$- f(0,0)(1-X)(1-Y) - f(0,1)(1-X)Y - f(1,0)X(1-Y) - f(1,1)XY,$$

i.e. F is the bilinear blend of the boundary data.

Now use F(X,Y) as Cauchy data for the heat equation

$$u_t = u_{xx} + u_{yy}$$

on $S = [0,1]^2 \times [0,T]$. At each time step we enforce $u = f$ on ∂S. When the solution has reached a steady state we will have our desired solution to $\Delta u = 0$.

The solution of the heat equation now becomes the focus of attention.

Heat Equation

Consider the heat equation (parabolic) in one spatial dimension

$$u_t = u_{xx} \quad \text{on} \quad (-\infty,\infty) \times [0,\infty)$$

together with the initial data

$$u(x,0) = f(x) , \qquad x \in \mathbb{R} .$$

A conventional approach to this problem is to use a forward difference in time coupled with a central difference in space. This results in the explicit scheme

$$u_i^{n+1} = u_i^n + \frac{\Delta t}{\Delta x^2} \left(u_{i+1}^n - 2u_i^n + u_{i-1}^n\right).$$

Letting $\lambda \equiv \frac{\Delta t}{\Delta x^2}$ and performing a von Neumann stability analysis we arrive at the following expression for the amplification factor:

$$u_i^n = e^{ipx} \ , \quad u_i^{n+1} = g\, e^{ipx} \ , \quad \xi = p\Delta x \ \Rightarrow$$

$$g = 1 + 2\lambda\ (\cos\ \xi - 1) \ .$$

Since we want $|g| < 1$ for all Fourier modes we must require that

$$\lambda < \frac{1}{2} \ .$$

Alternatively, we could approximate u_{xx} at the $n^{\underline{th}}$ time level using an exponential spline (with uniform tension).

We then have

$$e(u_{xx})_{i-1} + 2d(u_{xx})_i + e(u_{xx})_{i+1} = \frac{1}{\Delta x}\ (u_{i+1} - 2u_i + u_{i-1})$$

and now

$$u_k^{n+1} = u_k^n + \Delta t(u_{xx})_k^n \ ; \qquad k=i-1,i,i+1$$

$$\Rightarrow e\ u_{i-1}^{n+1} + 2du_i^{n+1} + eu_{i+1}^{n+1} = eu_{i-1}^n + 2du_i^n + eu_{i+1}^n + \frac{\Delta t}{\Delta x}(u_{i+1}^n - 2u_i^n + u_{i-1}^n)$$

$$\Rightarrow g = 1 + \lambda\ \frac{\Delta x\ (\cos\ \xi - 1)}{e\ \cos\ \xi + d}$$

$$\Rightarrow \lambda < \frac{d-e}{\Delta x} \ .$$

For the cubic spline this yields

$$\lambda < \frac{1}{6} \ .$$

However, as the tension increases we have

$$\frac{d-e}{\Delta x} \to 0$$

i.e. an increasingly stringent stability restriction. Hence, large tension is not recommended for this problem. Note that the cubic spline and central difference both provide an $O(\Delta x^2)$ accurate approximation to u_{xx} while the latter has the more favorable stability characteristics.

It is shown in Appendix I that the average of the cubic spline and central difference approximations to u_{xx} is fourth order accurate on a uniform mesh. The remaining questions concern the stability of such a scheme.

Correspondingly, let

$$(u_{xx})_i \sim \frac{1}{2}\left[S_i'' + \frac{u_{i-1} - 2u_i + u_{i+1}}{\Delta x^2}\right]$$

where S(x) denotes the cubic spline. I.e.

$$\frac{1}{6} S_{i-1}'' + \frac{2}{3} S_i'' + \frac{1}{6} S_{i+1}'' = \frac{1}{\Delta x^2}(S_{i-1} - 2S_i + S_{i+1}) .$$

$\therefore$

$$u_k^{n+1} = u_k^n + \frac{\Delta t}{2}\left[S_k'' + \frac{u_{k-1}^n - 2u_k^n + u_{k+1}^n}{\Delta x^2}\right] \; ; \; k=i-1,i,i+1$$

$$\Rightarrow \frac{1}{6} u_{i-1}^{n+1} + \frac{2}{3} u_i^{n+1} + \frac{1}{6} u_{i+1}^{n+1} = \frac{1}{6} u_{i-1}^n + \frac{2}{3} u_i^n + \frac{1}{6} u_{i+1}^n + \frac{\Delta t}{2\Delta x^2}(u_{i-1}^n - 2u_i^n + u_{i+1}^n)$$

$$+ \frac{\Delta t}{2\Delta x^2}\left(\frac{1}{6} u_{i-2}^n + \frac{1}{3} u_{i-1}^n - u_i^n + \frac{1}{3} u_{i+1}^n + \frac{1}{6} u_{i+2}^n\right)$$

$$\Rightarrow g = 1 + \frac{\lambda}{2(\cos\xi + 2)}[8\cos\xi - 9 + \cos 2\xi]$$

$$\Rightarrow \lambda < \frac{1}{4} .$$

Hence we have achieved higher order accuracy together with a reasonable stability restriction.

Wave Equation

Consider the wave equation (hyperbolic) in one spatial dimension

$$u_{tt} = u_{xx} \quad \text{on} \quad (-\infty,\infty) \times [0,\infty)$$

together with the initial conditions

$$u(x,0) = f(x) \ , \quad u_t(x,0) = h(x) \ , \qquad x \in \mathbb{R} \ .$$

One straightforward technique for treating this problem is to use central differences in both time and space. This produces the explicit scheme

$$u_i^{n+1} - 2u_i^n + u_i^{n-1} = \left(\frac{\Delta t}{\Delta x}\right)^2 (u_{i+1}^n - 2u_i^n + u_{i-1}^n)$$

where the calculation initially uses $u_t^0 = h$ to produce the required two previous time levels.

Letting $\lambda \equiv \frac{\Delta t}{\Delta x}$ we once again perform a von Neumann stability analysis.

$$u_i^{n-1} = e^{ipx} \ , \quad u_i^n = g\, e^{ipx} \ , \quad u_i^{n+1} = g^2 e^{ipx} \ , \quad \xi = p\,\Delta x$$

$\Rightarrow \quad g = 1 - \mu \pm \sqrt{\mu(\mu-2)} \quad$ where $\quad \mu \equiv \lambda^2(1 - \cos \xi)$

$\Rightarrow \quad \mu < 2 \Rightarrow \quad \lambda^2 < 1$

$\Rightarrow \quad \lambda < 1 \ .$

If instead we approximate u_{xx} using an exponential spline (with uniform tension) we have

$$\frac{e}{\Delta x}(u_{xx})_{i-1} + \frac{2d}{\Delta x}(u_{xx})_i + \frac{e}{\Delta x}(u_{xx})_{i+1} = \frac{1}{\Delta x^2}(u_{i+1}-2u_i+u_{i-1}).$$

Now

$$u_k^{n+1} - 2u_k^n + u_k^{n-1} = \Delta t^2 (u_{xx})_k \; ; \quad k = i-1,i,i+1$$

$$\Rightarrow \left(\frac{e}{\Delta x} u_{i-1}^{n+1} + \frac{2d}{\Delta x} u_i^{n+1} + \frac{e}{\Delta x} u_{i+1}^{n+1}\right) - 2\left(\frac{e}{\Delta x} u_{i-1}^{n} + \frac{2d}{\Delta x} u_i^{n} + \frac{e}{\Delta x} u_{i+1}^{n}\right)$$

$$+ \left(\frac{e}{\Delta x} u_{i-1}^{n-1} + \frac{2d}{\Delta x} u_i^{n-1} + \frac{e}{\Delta x} u_{i+1}^{n-1}\right) = \left(\frac{\Delta t}{\Delta x}\right)^2 (u_{i+1}^n - 2u_i^n + u_{i-1}^n)$$

$$\Rightarrow (g^2 - 2g + 1)\left[\frac{e \cos \xi + d}{\Delta x}\right] = g\lambda^2 (\cos \xi - 1).$$

Letting $\mu \equiv \dfrac{\lambda^2 (1 - \cos \xi)\, \Delta x}{2 (e \cos \xi + d)}$ we have

$$g^2 - 2g + 1 + 2\mu g = 0 \Rightarrow g = (1-\mu) \pm \sqrt{\mu(\mu-2)}$$

which once again requires

$$\mu < 2 \Rightarrow \lambda^2 < \frac{4(e \cos \xi + d)}{\Delta x (1 - \cos \xi)}$$

$$\Rightarrow \lambda^2 < \frac{2(d-e)}{\Delta x} \Rightarrow \lambda < \sqrt{\frac{2(d-e)}{\Delta x}}\,.$$

For the cubic spline this yields

$$\lambda < \frac{1}{\sqrt{3}}$$

However, as for the heat equation,

$$\frac{d-e}{\Delta x} \to 0 \,,$$

with increasing tension. This stability restriction again discourages the use of large tension for this problem. The cubic spline and central difference are comparable in accuracy.

We next study the use of the average of the cubic spline and central difference approximations which is fourth order accurate on a uniform mesh (see Appendix I). Let

$$(u_{xx})_i \sim \frac{1}{2}\left[S_i'' + \frac{u_{i-1}-2u_i+u_{i+1}}{\Delta x^2}\right]$$

where S(x) denotes the cubic spline. I.e.,

$$\frac{1}{6}S_{i-1}'' + \frac{2}{3}S_i'' + \frac{1}{6}S_{i+1}'' = \frac{1}{\Delta x^2}(S_{i-1} - 2S_i + S_{i+1})$$

$$\therefore \quad u_k^{n+1} - 2u_k^n + u_k^{n-1} = \frac{\Delta t^2}{2}\left[S_i'' + \frac{u_{k+1}^n - 2u_k^n + u_{k-1}^n}{\Delta x^2}\right] \; ; \; k = i-1, i, i+1$$

$$\Rightarrow \left[\frac{1}{6}u_{i-1}^{n+1} + \frac{2}{3}u_i^{n+1} + \frac{1}{6}u_{i+1}^{n+1}\right] - 2\left[\frac{1}{6}u_{i-1}^n + \frac{2}{3}u_i^n + \frac{1}{6}u_{i+1}^n\right]$$

$$+ \left[\frac{1}{6}u_{i-1}^{n-1} + \frac{2}{3}u_i^{n-1} + \frac{1}{6}u_{i+1}^{n-1}\right] = \frac{\Delta t^2}{2\Delta x^2}(u_{i-1}^n - 2u_i^n + u_{i+1}^n)$$

$$+ \frac{\Delta t^2}{2\Delta x^2}\left[\frac{1}{6}u_{i-2}^n + \frac{1}{3}u_{i-1}^n - u_i^n + \frac{1}{3}u_{i+1}^n + \frac{1}{6}u_{i+2}^n\right]$$

$$\Rightarrow (g^2 - 2g + 1)\left[\frac{\cos\xi + 2}{3}\right] = \frac{g\lambda^2}{2}\left[\frac{8}{3}\cos\xi + \frac{1}{3}\cos 2\xi - 3\right]$$

Letting

$$\mu \equiv \frac{\lambda^2}{4(\cos\xi + 2)}\left[10 - 8\cos\xi - 2\cos^2\xi\right]$$

we arrive again at

$$g^2 - 2g + 1 + 2\mu g = 0$$

which we know requires

$$\mu < 2$$

$$\Rightarrow \quad \lambda^2 < \frac{8(\cos \xi + 2)}{10 - 8 \cos \xi - 2 \cos^2 \xi}$$

$$\Rightarrow \quad \lambda^2 < \frac{1}{2} \Rightarrow \quad \lambda < \frac{1}{\sqrt{2}} .$$

Hence this scheme provides higher order accuracy while maintaining stability.

III.4. Model Problem

In subsequent sections we will apply spline approximations to the numerical solution of nonlinear systems of hyperbolic conservation laws. The stability of such a scheme can most easily be studied by considering model equations with constant coefficients.

Therefore, in this section we investigate the prototypical equation

$$u_t + cu_x = 0 \quad \text{on} \quad (-\infty,\infty) \times [0,\infty)$$

with $u(x,0) = f(x)$ $\forall$ $x \in \mathbb{R}$. All calculations are to be performed on a mesh with uniform Δx and Δt.

We begin by considering forward differences in both time and space. This results in

$$u_j^{n+1} = u_j^n - \frac{c\Delta t}{\Delta x} (u_{j+1}^n - u_j^n) .$$

A von Neumann stability analysis reveals an amplification factor

$$g = 1 + \lambda - \lambda e^{i\xi}$$

where $\lambda \equiv \frac{c\,\Delta t}{\Delta x}$. We thus require that $-1 < \lambda < 0$.

Alternatively, use of a backward spatial difference

$$u_j^{n+1} = u_j^n - \lambda(u_j^n - u_{j-1}^n)$$

yields

$$g = 1 - \lambda + \lambda e^{i\xi}.$$

We thus restrict $0 < \lambda < 1$ in this case.

These results are not surprising since they simply restate the C-F-L condition that the numerical domain of dependence should contain the true domain of dependence. However, these one-sided differences are only first order accurate. It would seem that second order accuracy could be achieved by using a central difference since that would fulfill the C-F-L requirement for arbitrary c.

Such a scheme takes the form

$$u_j^{n+1} = u_j^n - \frac{\lambda}{2} (u_{j+1}^n - u_{j-1}^n)$$

with an amplification factor

$$g = 1 - \lambda \sin \xi i \ .$$

This is seen to be unconditionally unstable. Hence care must be taken in choosing a time marching scheme.

The above deficiency can be remedied by approximating

$$(u_t)_j^n \approx u_j^{n+1} - \frac{u_{j+1}^n + u_{j+1}^n}{2}$$

thus producing the Lax scheme

$$u_j^{n+1} = \frac{1}{2} (u_{j+1}^n + u_{j-1}^n) - \frac{\lambda}{2} (u_{j+1}^n - u_{j-1}^n)$$

$$\Rightarrow \quad g = \cos \xi - i\lambda \sin \xi$$

$$\Rightarrow \quad |\lambda| < 1 \ .$$

We can increase spatial accuracy by using a fourth order central difference

$$(u_x)_i \approx \frac{1}{\Delta x}\left[\frac{1}{12}u_{i-2} - \frac{2}{3}u_{i-1} + \frac{2}{3}u_{i+1} - \frac{1}{12}u_{i+2}\right].$$

This yields the scheme

$$u_j^{n+1} = \frac{1}{2}(u_{j+1}^n + u_{j-1}^n) - \lambda\left(\frac{1}{12}u_{i-2}^n - \frac{2}{3}u_{i-1}^n + \frac{2}{3}u_{i+1}^n - \frac{1}{12}u_{i+2}^n\right)$$

$$\Rightarrow \quad g = \cos\xi - \lambda i \sin\xi\left[\frac{4 - \cos\xi}{3}\right]$$

$$\Rightarrow \quad |\lambda| < \frac{3}{4 - \cos\xi}$$

$$\Rightarrow \quad |\lambda| < \frac{3}{5}.$$

When boundaries are present such a scheme would require modifications.

It is shown in Appendix I that a scheme which does not suffer this drawback, yet maintains fourth order accuracy on a uniform mesh, can be constructed using spline approximations to u_x. In this scheme we have

$$\frac{e}{2(d+e)}(u_x)_{j-1} + \frac{d}{d+e}(u_x)_j + \frac{e}{2(d+e)}(u_x)_{j+1} = \frac{u_{j+1}-u_{j-1}}{2\Delta x}.$$

$$u_k^{n+1} - \frac{1}{2}(u_{k+1}^n + u_{k-1}^n) + c\Delta t\ D_x u_k = 0 \ ; \ k=j-1,j,j+1$$

then yields

$$\frac{e}{2(d+e)}\left[u_{j+1}^{n+1} - \frac{1}{2}(u_{j+2}^n + u_j^n)\right] + \frac{d}{d+e}\left[u_j^{n+1} - \frac{1}{2}(u_{j+1}^n + u_{j-1}^n)\right]$$

$$+ \frac{e}{2(d+e)}\left[u_{j-1}^{n+1} - \frac{1}{2}(u_j^n + u_{j-2}^n)\right] + \frac{c\Delta t}{2\Delta x}(u_{j+1}^n - u_{j-1}^n) = 0$$

$$\Rightarrow \quad g = \cos\xi - \lambda \sin\xi\left[\frac{d+e}{e\cos\xi + d}\right]$$

$$\Rightarrow \quad |\lambda| < \frac{d-e}{d+e}.$$

For the cubic spline we have $d = h/3$, $e = h/6$

$$\Rightarrow |\lambda| < 1/3$$

while for the linear spline we recover

$$|\lambda| < \frac{1 - e/d}{1 + e/d} \to 1 \ .$$

Now that we have achieved a high order of spatial accuracy let us turn our attention to obtaining higher order temporal accuracy. This will ensure more accurate unsteady calculations while allowing us to take larger time steps when attempting to march to a steady state.

The time integration schemes considered thus far have been first order accurate. We attempt to extend this accuracy by employing classical Runge-Kutta techniques.

Hence, consider the second order Runge-Kutta scheme

$$u^{(0)} = u^n$$

$$u^{(1)} = u^{(0)} - c\,\Delta t\, D_x\, u^{(0)}$$

$$u^{(2)} = u^{(0)} - c\,\Delta t\, D_x \left[\frac{u^{(0)} + u^{(1)}}{2} \right]$$

$$u^{n+1} = u^{(2)} \ .$$

Here D_x is the two point central difference operator. The first step yields an amplification factor

$$g^{(1)} = 1 - \lambda \sin \xi i = 1 - 2\rho i$$

where $\rho \equiv \frac{\lambda \sin \xi}{2}$.

The second step produces an amplification factor

$$g^{(2)} = 1 - 2\rho i - 2\rho^2 \equiv q \ .$$

Hence,

$$|q|^2 = (1-2\rho^2)^2 + 4\rho^2 = (1-2\mu)^2 + 4\mu$$
$$= 1 + 4\mu^2 > 1$$

where $\mu \equiv \rho^2$. Thus this scheme is unconditionally unstable. Similar results obtain for the fourth order central difference and the spline scheme. We will not dwell on deriving a stable scheme that is second order accurate in time since even higher order accuracy is our ultimate design.

Instead consider the Runge-Kutta scheme of third order

$$u^{(0)} = u^n$$
$$u^{(1)} = u^{(0)} - \frac{c\Delta t}{2} D_x u^{(0)}$$
$$u^{(2)} = u^{(0)} - c\Delta t\, D_x(2u^{(1)} - u^{(0)})$$
$$u^{(3)} = u^{(0)} - \frac{c\Delta t}{6} D_x(u^{(0)} + 4u^{(1)} + u^{(2)})$$
$$u^{n+1} = u^{(3)} .$$

For definiteness let D_x be the two point central difference operator. Letting $\rho \equiv \frac{\lambda \sin \xi}{2}$ this successively yields the amplification factors

$$g^{(1)} = 1 - i\rho$$
$$g^{(2)} = 1 - 2i\rho - 4\rho^2$$
$$g^{(3)} = 1 - 2\rho i - 2\rho^2 + \frac{4}{3} i\rho^3 = (1-2\rho^2) + 2i\rho(\frac{2}{3}\rho^2 - 1) \equiv q$$
$$\Rightarrow \quad |g|^2 = (1-2\rho^2)^2 + 4\rho^2(\frac{2}{3}\rho^2 - 1)^2 = 1 - \frac{4}{3}\mu^2 + \frac{16}{9}\mu^3$$

where $\mu = \rho^2$. We hence restrict $\mu < \frac{3}{4} \Rightarrow |\rho| < \sqrt{\frac{3}{4}}$

$$\Rightarrow \quad |\lambda| < 2\sqrt{\frac{3}{4}} = \sqrt{3}.$$

Hence this scheme is stable as well as accurate. Similar calculations can be done for the other D_x under consideration. We refrain from doing this as the next scheme is the one of primary interest to us.

Consider the Runge-Kutta scheme of fourth order

$$u^{(0)} = u^n$$

$$u^{(1)} = u^{(0)} - \frac{c\Delta t}{2} D_x u^{(0)}$$

$$u^{(2)} = u^{(0)} - \frac{c\Delta t}{2} D_x u^{(1)}$$

$$u^{(3)} = u^{(0)} - c\Delta t\, D_x u^{(2)}$$

$$u^{(4)} = u^{(0)} - \frac{c\Delta t}{6} D_x (u^{(0)} + 2u^{(1)} + 2u^{(2)} + u^{(3)})$$

$$u^{n+1} = u^{(4)}.$$

Initially, let D_x be the two point central difference operator. Then

$$g^{(1)} = 1 - \rho i$$

$$g^{(2)} = 1 - \rho i - \rho^2$$

$$g^{(3)} = 1 - 2\rho i - 2\rho^2 + 2i\rho^3$$

$$g^{(4)} = 1 - 2\rho i - 2\rho^2 + \frac{4}{3}\rho^3 i + \frac{2}{3}\rho^4 \equiv q$$

where $\rho \equiv \frac{\lambda}{2} \sin \xi$

$$\Rightarrow \quad |q|^2 = (1 - 2\rho^2 + \frac{2}{3}\rho^4)^2 + 4\rho^2(\frac{2}{3}\rho^2 - 1)^2$$

$$= (1 - 2\mu + \frac{2}{3}\mu^2)^2 + 4\mu(\frac{2}{3}\mu - 1)^2$$

where $\mu \equiv \rho^2$.

We hence restrict $\mu < 2 \Rightarrow |\rho| < \sqrt{2}$

$\Rightarrow |\lambda| < 2\sqrt{2} \approx 2.828427125.$

Now if we let D_x be the fourth order central difference operator we obtain the same expression for $g^{(4)}$ with ρ redefined as

$$\rho \equiv \frac{\lambda}{2}\left(\frac{4}{3}\sin\xi - \frac{1}{6}\sin 2\xi\right) = \frac{\lambda \sin\xi}{6}(4 - \cos\xi)$$

We again require $|\rho| < \sqrt{2}$ which now yields

$$|\lambda| < 6\left[\max_{\xi\in[0,2\pi]} |\sin\xi|(4 - \cos\xi)\right]^{-1} \cdot\sqrt{2}$$

$$\approx 6\sqrt{2}\ (0.2429150227)$$

$$\approx 2.061202317$$

If we instead use the spline approximation to D_x we once again obtain the same expression for $g^{(4)}$ however with ρ now defined as

$$\rho \equiv \frac{\lambda \sin\xi\,(d+e)}{2(e\cos\xi + d)}$$

$\therefore \quad |\rho| < \sqrt{2} \Rightarrow |\lambda| < 2\sqrt{2} \cdot \sqrt{\frac{2d}{d+e} - 1}$.

Hence, for the cubic spline we have, since $\frac{e}{d} = \frac{1}{2}$,

$$|\lambda| < 2\sqrt{\frac{2}{3}} \approx 1.632993162$$

while for the linear spline we have, since $\frac{e}{d} = 0$,

$$|\lambda| < 2\sqrt{2}\ .$$

Because of the accuracy and stability of this scheme it is used in our subsequent calculations. In addition, no modifications are needed at the boundaries and no loss of accuracy results.

Figures II.4.1a-c show a comparison of the computed solution to the exact solution for our model problem using the exponential spline with fourth order Runge-Kutta scheme. The calculation was done on three successively finer meshes and shows visually the effect of mesh refinement on accuracy.

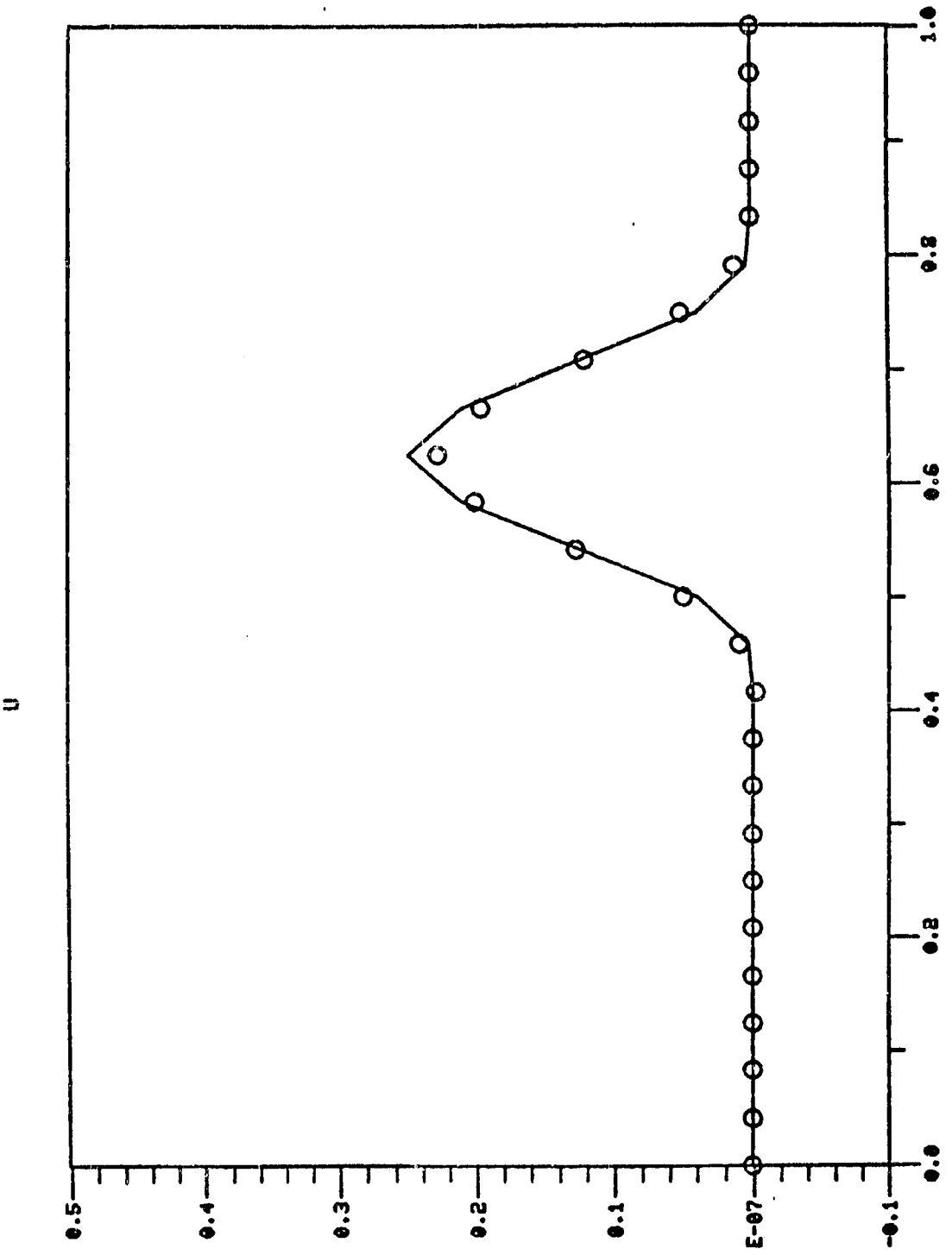


Figure III.4.1a. Model Problem Test Case ($N = 25$, $\Delta x = 1/24$)

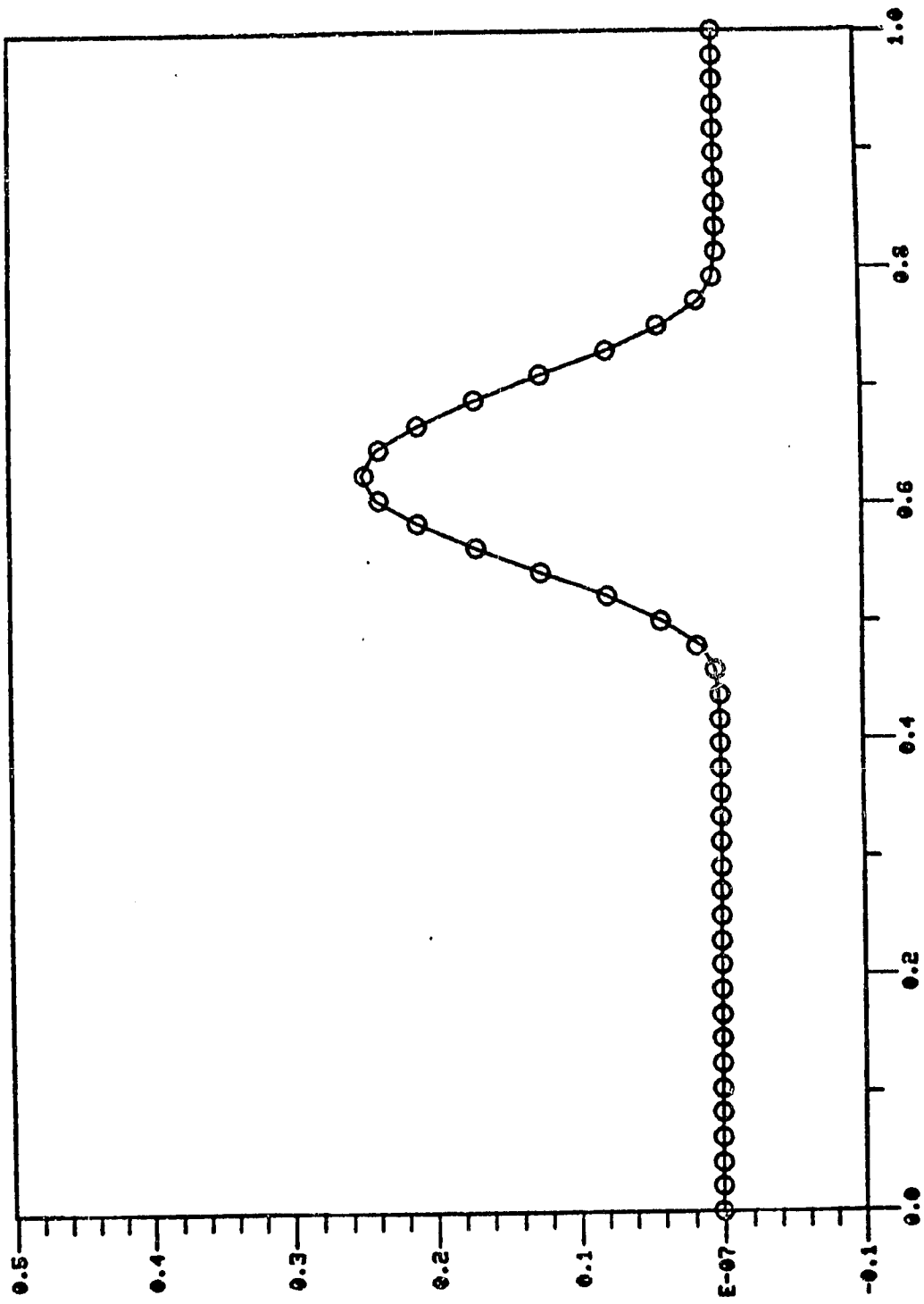


Figure III.4.1b. Model Problem Test Case ($N = 49$, $\Delta x = 1/48$)

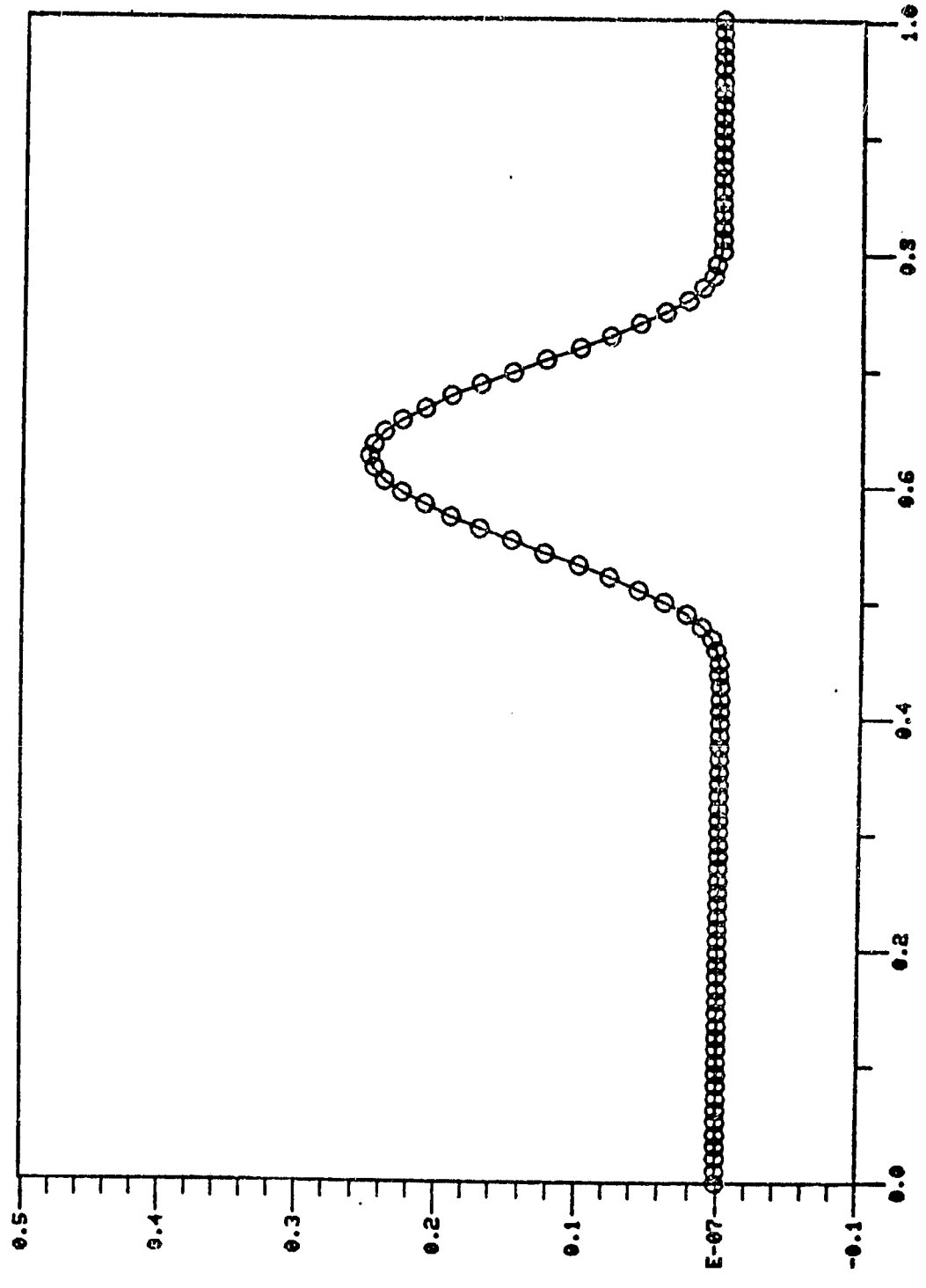


Figure III.4.1c. Model Problem Test Case (N = 97, $\Delta x = 1/96$)

III.5. Inviscid Burger's Equation

In this section we begin our treatment of nonlinear hyperbolic conservation laws using splines. The subject of discussion is a nonlinear analogue of our model problem, namely the inviscid Burger's equation [39]

$$u_t + uu_x = 0 \quad \text{on} \quad (x,t) \in (-\infty,\infty) \times [0,\infty) \ .$$

This equation which is in quasilinear form can be put into the conservation, or divergence, form

$$u_t + (\frac{u^2}{2})_x = 0 \ .$$

Before proceeding further with this equation, we digress for a general discussion of the single conservation law [29]

$$u_t = f_x(u)$$

or

$$u_t = a\, u_x \qquad \text{w/} \quad a \equiv df/du$$

subject to the initial condition

$$u(x,0) = \phi(x) \ .$$

Since f(u) is in general nonlinear we cannot guarantee the existence of a smooth solution for all time. Instead we seek weak solutions satisfying

$$\int_0^\infty \int_{-\infty}^{\infty} (w_t u - w_x f)\, dx\, dt + \int_{-\infty}^{\infty} w(x,0)\, \phi(x)\, dx = 0 \ ,$$

for all smooth test functions, w, which vanish for $|x|+t$ large enough.

One must consider whether a numerical solution technique has the capability of capturing such a weak solution. We now present the following convergence result which is a generalization of a theorem of Lax and Wendroff [28]. First recall that the exponential spline, $\tau(x)$, can be defined in terms of its first derivatives at the knots. These values are determined by the tridiagonal system of linear algebraic equations

$$A_i\tau'_{i-1} + B_i\tau'_i + C_i\tau'_{i+1} = \alpha_i \cdot \frac{f_{i+1}-f_i}{h_i} + \beta_i \cdot \frac{f_i-f_{i-1}}{h_{i-1}}$$

where

$$A_i = \frac{(e_{i-1})(d_i - e_i)}{(d_{i-1}+e_{i-1})[(d_{i-1}-e_{i-1})+(d_i-e_i)]}$$

$$B_i = \left[\frac{(d_{i-1})(d_i-e_i)}{(d_{i-1}+e_{i-1})} + \frac{(d_i)(d_{i-1}-e_{i-1})}{(d_i + e_i)}\right] \Big/ [(d_{i-1}-e_{i-1})+(d_i-e_i)]$$

$$C_i = \frac{(e_i)(d_{i-1}-e_{i-1})}{(d_i+e_i)[(d_{i-1}-e_{i-1})+(d_i-e_i)]}$$

$$\alpha_i = \frac{d_{i-1}-e_{i-1}}{d_{i-1}-e_{i-1}+d_i-e_i} = \frac{1}{1 + \frac{d_i - e_i}{d_{i-1}-e_{i-1}}}$$

$$\beta_i = \frac{d_i-e_i}{d_{i-1}-e_{i-1}+d_i-e_i} .$$

What is of key importance to us here is that

$$A_i + B_i + C_i = \alpha_i + \beta_i = 1 .$$

Theorem. If we approximate $u_t = f_x$ by

$$\frac{\Delta v}{\Delta t} = \tau_x$$

(where $\tau(x)$ is the exponential spline interpolant) and as Δt and $h = \max_i h_i$ tend to zero v converges boundedly almost everywhere to some function $u(x,t)$ then $u(x,t)$ is a weak solution.

Proof: We have by assumption

$$\frac{v_i^{n+1}-v_i^n}{\Delta t} = (\tau_x)_i^n \; ; \qquad v_i^0 = \phi_i$$

and hence that

$$A_i^n\left[\frac{v_{i-1}^{n+1}-v_{i-1}^n}{\Delta t}\right] + B_i^n\left[\frac{v_i^{n+1}-v_i^n}{\Delta t}\right] + C_i^n\left[\frac{v_{i+1}^{n+1}-v_{i+1}^n}{\Delta t}\right]$$
$$= \alpha_i^n \cdot \frac{f_{i+1}^n - f_i^n}{h_i} + \beta_i^n \cdot \frac{f_i^n - f_{i-1}^n}{h_{i-1}} .$$

Multiplying both sides by $W_i^n \Delta t\, h_i$ and summing over all grid points yields

$$\sum_{n=0}^{\infty}\sum_{i=-\infty}^{\infty} \frac{W_i^n}{\Delta t}\,[A_i^n(v_{i-1}^{n+1}-v_{i-1}^n) + B_i^n(v_i^{n+1}-v_i^n) + C_i^n(v_{i+1}^{n+1}-v_{i+1}^n)]\Delta t\, h_i$$
$$= \sum_{n=0}^{\infty}\sum_{i=-\infty}^{\infty} W_i^n[\alpha_i^n \frac{f_{i+1}^n - f_i^n}{h_i} + \beta_i^n \frac{f_i^n - f_{i-1}^n}{h_{i-1}}]\Delta t\; h_i$$

or LHS = RHS for convenience.

LHS =

(let $\Delta g_i^j = g_i^{j+1} - g_i^j$)

$$\sum_{n=0}^{\infty}\sum_{i=-\infty}^{\infty} h_i[W_i^n A_i^n\, \Delta v_{i-1}^n + W_i^n B_i^n \Delta v_i^n + W_i^n C_i^n\, \Delta v_{i+1}^n].$$

Lemma (summation by parts)

$$\sum_{n=0}^{\infty} u^n \cdot \Delta v^n = - u^0 v^0 - \sum_{n=0}^{\infty} \Delta u^n \cdot v^{n+1} .$$

$$\Rightarrow \text{LHS} = - \sum_{n=0}^{\infty} \sum_{i=-\infty}^{\infty} \Delta t \, h_i \left[\frac{\Delta (W_i^n A_i^n) v_{i-1}^{n+1}}{\Delta t} + \frac{\Delta (W_i^n B_i^n) v_i^{n+1}}{\Delta t} + \frac{\Delta (W_i^n C_i^n) v_{i+1}^{n+1}}{\Delta t} \right]$$

$$- \sum_{i=-\infty}^{\infty} h_i \left(W_i^0 A_i^0 v_{i-1}^0 + W_i^0 B_i^0 v_i^0 + W_i^0 C_i^0 v_{i+1}^0 \right) .$$

$$\therefore \text{LHS} \to - \int_0^{\infty} \int_{-\infty}^{\infty} W_t u \, dx \, dt - \int_{-\infty}^{\infty} W(x,0) \, \phi(x) \, dx .$$

Now

$$\text{RHS} =$$

$$\sum_{n=0}^{\infty} \sum_{i=-\infty}^{\infty} \Delta t \, h_i \, W_i^n \alpha_i^n \frac{f_{i+1}^n}{h_i} - \Delta t \, h_i \, W_i^n \alpha_i^n \frac{f_i^n}{h_i} + \Delta t \, h_i W_i^n \beta_i^n \frac{f_i^n}{h_{i-1}}$$

$$- \Delta t \, h_i \, W_i^n \beta_i^n \frac{f_{i-1}^n}{h_{i-1}}$$

$$= \sum_{n=0}^{\infty} \sum_{i=-\infty}^{\infty} \Delta t \, h_{i-1} W_{i-1}^n \alpha_{i-1}^n \frac{f_i^n}{h_{i-1}} - \Delta t \, h_i W_i^n \alpha_i^n \frac{f_i^n}{h_i}$$

$$+ \Delta t \, h_i \, W_i^n \beta_i^n \frac{f_i^n}{h_{i-1}} - \Delta t \, h_{i+1} \, W_{i+1}^n \beta_{i+1}^n \frac{f_i^n}{h_i}$$

$$= \sum_{n=0}^{\infty} \sum_{i=-\infty}^{\infty} \Delta t \, f_i^n \left\{ W_{i-1}^n \alpha_{i-1}^n - W_i^n \alpha_i^n + \frac{h_i}{h_{i-1}} W_i^n \beta_i^n - \frac{h_{i+1}}{h_i} W_{i+1}^n \beta_{i+1}^n \right\}$$

$$= \sum_{n=0}^{\infty} \sum_{i=-\infty}^{\infty} \Delta t\ f_i^n \Big\{ W_{i-1}^n \alpha_{i-1}^n - W_i^n \alpha_i^n + \frac{h_i}{h_{i-1}} W_i^n (1-\alpha_i^n) -$$

$$- \frac{h_{i+1}}{h_i} W_{i+1}^n (1-\alpha_{i+1}^n) \Big\}$$

$$= \sum_{n=0}^{\infty} \sum_{i=-\infty}^{\infty} \Delta t\ f_i^n \Big\{ W_{i-1}^n \alpha_{i-1}^n - W_i^n \alpha_i^n + \left[\frac{h_i}{h_{i-1}} - \frac{h_{i+1}}{h_i} \right] W_i^n (1-\alpha_i^n)$$

$$+ \frac{h_{i+1}}{h_i} [W_i^n (1-\alpha_i^n) - W_{i+1}^n (1-\alpha_{i+1}^n)] \Big\}$$

$$= \sum_{n=0}^{\infty} \sum_{i=-\infty}^{\infty} \Delta t\ f_i^n \Big\{ W_{i-1}^n \alpha_{i-1}^n - W_i^n \alpha_i^n - \frac{h_{i+1}}{h_i} W_i^n \alpha_i^n + \frac{h_{i+1}}{h_i} W_{i+1}^n \alpha_{i+1}^n$$

$$+ \frac{h_{i+1}}{h_i} [W_i^n - W_{i+1}^n] + \left[\frac{h_i}{h_{i-1}} - \frac{h_{i+1}}{h_i} \right] W_i^n (1-\alpha_i^n) \Big\}$$

Now let

$$\frac{h_{i+1}}{h_i} \to 1 + \delta_i$$

then

$$\left[\frac{h_i}{h_{i-1}} - \frac{h_{i+1}}{h_i} \right] = (1-\delta_{i-1}) - (1-\delta_i) = \delta_i - \delta_{i-1}$$

$$\Rightarrow \text{RHS} \to \int_0^{\infty} \int_{-\infty}^{\infty} [(\Delta x) \cdot (\alpha W)_{xx} + \delta \cdot (\alpha W)_x - (1+\delta)(W_x) + \delta_x \cdot W(1-\alpha)]\ dx\ dt$$

so that to recover a weak solution we require

(i) $(\alpha W)_{xx}$ bounded; (ii) $\delta[(\alpha-1)W]_x + \delta_x[W(1-\alpha)] = 0$.

The restrictions $p_i \to p$, $h_i \to h$ $\forall i$ together with the assumed smoothness of $W(x,t)$ allow us to conclude

$$\text{RHS} \to -\int_0^\infty \int_{-\infty}^{\infty} W_x f \, dx \, dt \, .$$

Hence

$$\int_0^\infty \int_{-\infty}^{\infty} (W_t u - W_x f) \, dx \, dt + \int_{-\infty}^{\infty} W(x,0) \, \phi(x) \, dx = 0$$

Q.E.D.

Note that the requirement of an asymptotically uniform mesh appears unavoidable. Meanwhile, the assumption of asymptotically uniform tension may be relaxed to p → a step function. Furthermore, p(x) a step function is sufficiently general to avoid having to employ uniform tension while allowing for the fitting of local phenomena such as shocks and other discontinuities. Hence, it is not overly restrictive.

We now return to our discussion of Burger's equation. This equation admits both compression waves and expansion waves as solutions [29]. We consider two examples.

Compression Wave

$$\phi(x) = \begin{cases} 1 \, , & x \leq 0 \\ 1-x, & 0 \leq x \leq 1 \\ 0 \, , & 1 \leq x \end{cases}$$

This wave front steepens until at $t = 1$ a discontinuity develops. I.e., for $t < 1$,

$$u(x,t) = \begin{cases} 1 \, , & x \leq t \\ \frac{x-1}{t-1}, & t \leq x \leq 1 \\ 0 \, , & 1 \leq x \end{cases}$$

while for $t \geq 1$,

$$u(x,t) = \begin{cases} 1 , & x < \frac{1+t}{2} \\ 0 , & \frac{1+t}{2} < x \end{cases} .$$

Expansion Wave

$$\phi(x) = \begin{cases} 0 , & x < 0 \\ 1 , & 0 < x \end{cases}$$

This problem has the two solutions

$$\bar{u}(x,t) = \begin{cases} 0 , & x < t/2 \\ 1 , & t/2 < x \end{cases}$$

and

$$u(x,t) = \begin{cases} 0 , & x < 0 \\ \frac{x}{t} , & 0 \leq x \leq t \\ 1 , & t < x \end{cases} .$$

The physically relevant solution is the one satisfying the so called entropy condition (inequality) which requires that the characteristics issuing from either side of the discontinuity curve in the direction of increasing t should intersect the line of discontinuity. This excludes the discontinuous solution $\bar{u}(x,t)$.

The task before us is now clear. We must devise a numerical scheme that will capture discontinuities as well as select the physically relevant solution from the, generally infinite, collection of weak solutions to an initial value problem.

This is accomplished by adding a suitable artificial viscosity term [24] to the exponential spline - Runge-Kutta scheme as applied to the model problem in the previous

section. This effectively shifts the derivative calculation upstream thus modeling the correct domain of dependence while prohibiting expansion shocks.

The switching function will be dependent on $|u_{xx}|$ since we know this to be large for the high tension values anticipated in the neighborhood of a shock. Specifically

$$D_x[f(u_i)] = (\tau_x)_i - \varepsilon_{i+1/2}(\tau_x)_{i+1/2} + \varepsilon_{i-1/2}(\tau_x)_{i-1/2}$$

where

$$\varepsilon = \min\ (h^2 \cdot |f_{xx}|,\ 1/2).$$

Hence, when high tension is present, the point of evaluation of f_x is shifted to the mid-point of the appropriate adjoining interval since

$$D_x[f(u_i)] \approx (\tau_x)_i - h[(\varepsilon\tau_x)_x]_i = [\tau_x - h(\varepsilon\tau_x)_x]_i,$$

which, with $\varepsilon = 1/2$, becomes

$$[\tau_x - \frac{h}{2}\,\tau_{xx}]_i\ .$$

If $u < 0$ then the approximation is

$$D_x[f(u_i)] = (\tau_x)_i + \varepsilon_{i+1/2}(\tau_x)_{i+1/2} - \varepsilon_{i-1/2}(\tau_x)_{i-1/2}$$

$$\approx (\tau_x)_i + h[(\varepsilon\tau_x)_x]_i = [\tau_x + h(\varepsilon\tau_x)_x]_i\ .$$

Note that these artificial viscosity terms are themselves in divergence form thus preserving the overall conservation form necessary for convergence to a weak solution in

the limit of vanishing mesh width.

Note also that the above choice for ε produces an $O(h^3)$ artificial viscosity term away from the shock. This is consistent with the spatial discretization error on a nonuniform mesh. For a uniform mesh we may alter this term as follows

$$\varepsilon = \min(h^3 \cdot |f_{xx}|, 1/2) \ .$$

The corresponding artificial viscosity term is now $O(h^4)$.

We now calculate solutions to our two previous examples. In Figures III.5.3a-f the expansion wave is seen to be accurately portrayed with any inadequacies being the result of the curve fit for the discontinuous initial data. The compression wave calculation displayed in Figures III.5.4a-k likewise reflects a high degree of accuracy. Most notable are the predicted location of the shock for large t, the sharp shock resolution, and the conspicuous lack of overshoots and undershoots at the shock.

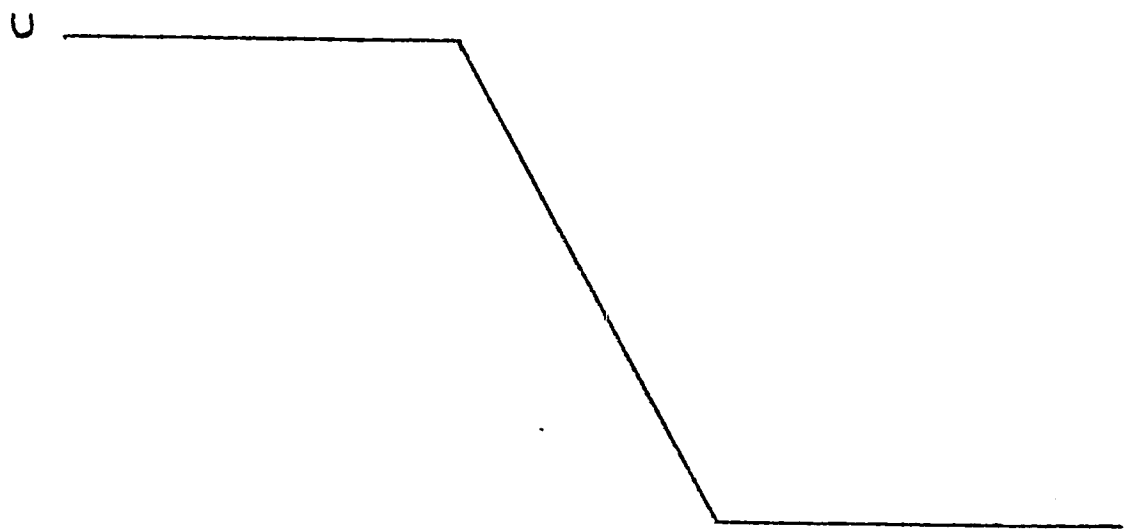


Figure III.5.1. Cauchy Data for Compression Wave

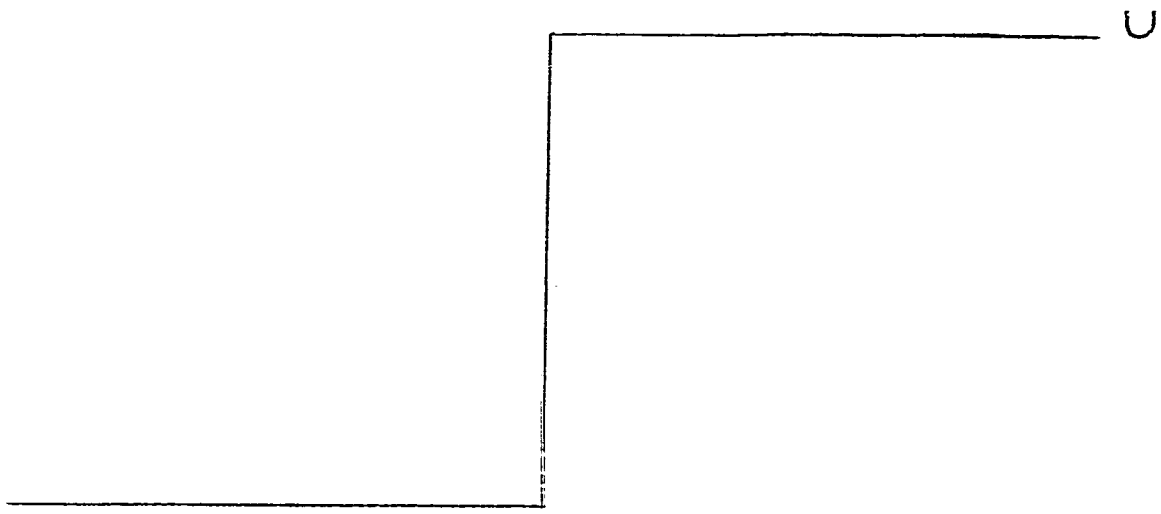


Figure III.5.2. Cauchy Data For Expansion Wave

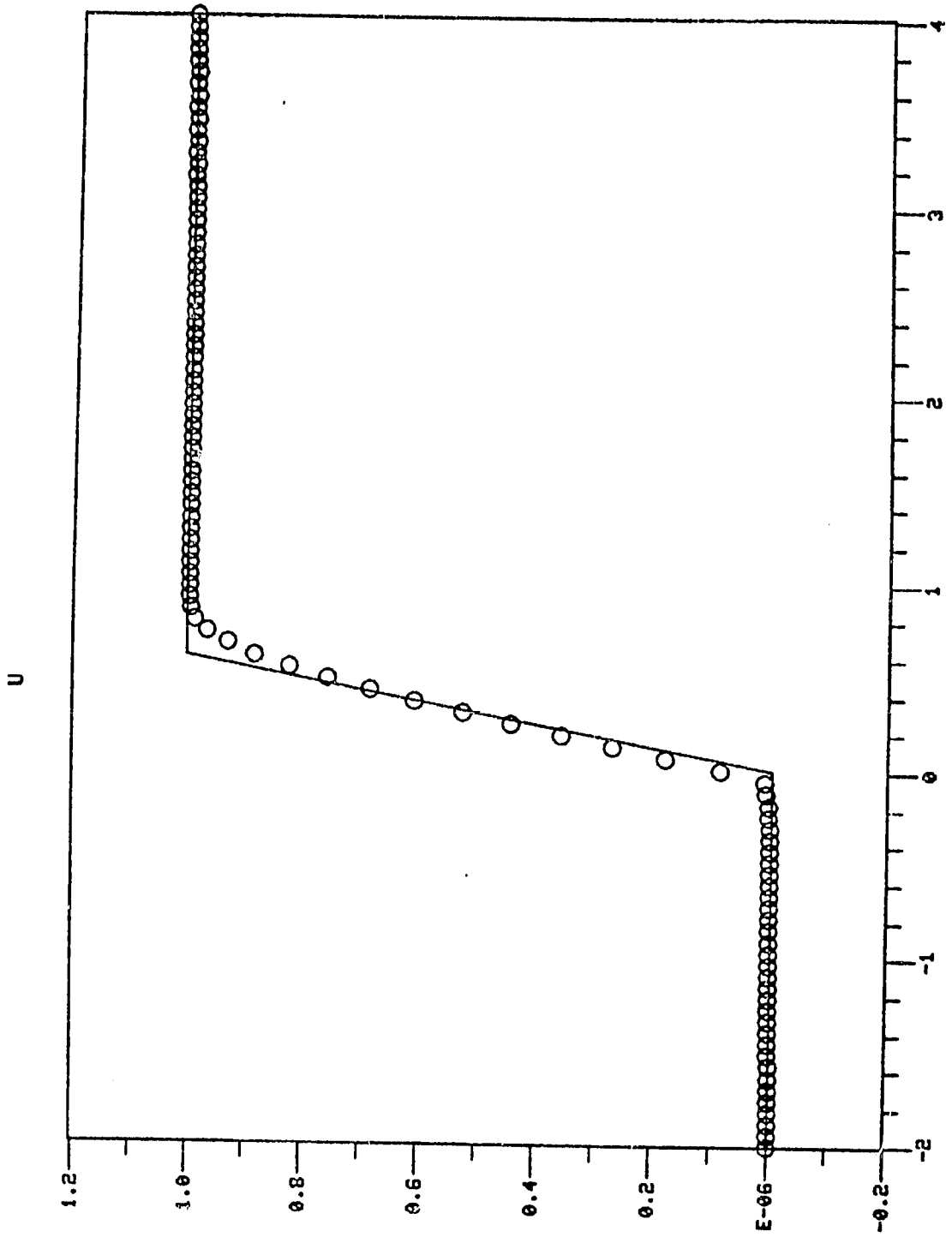


Figure III.5.3a. Expansion Wave (N = 100, t = 0.6, NSTEP = 10)

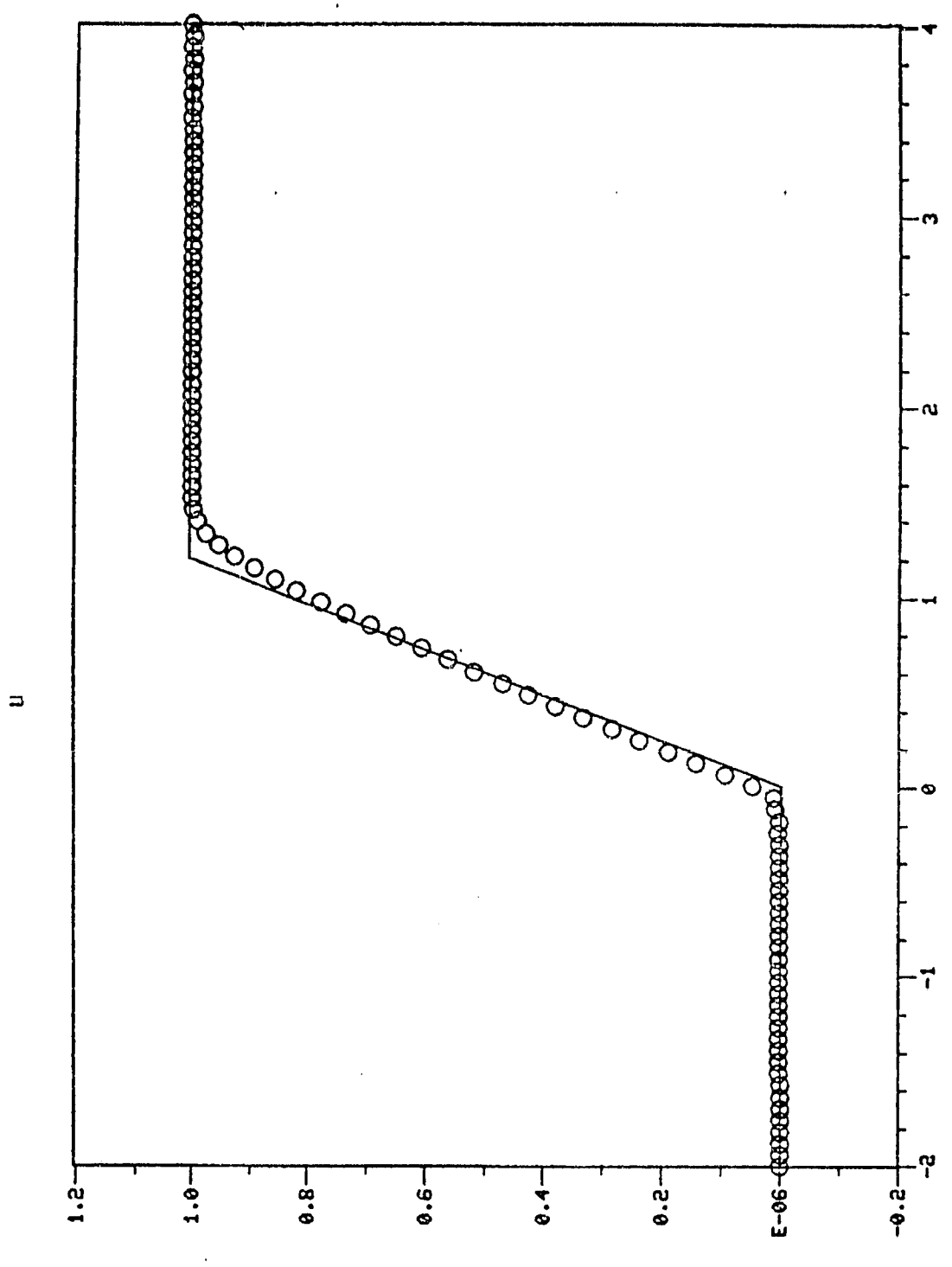


Figure III.5.3b. Expansion Wave (N = 100, t = 1.2, NSTEP = 20)

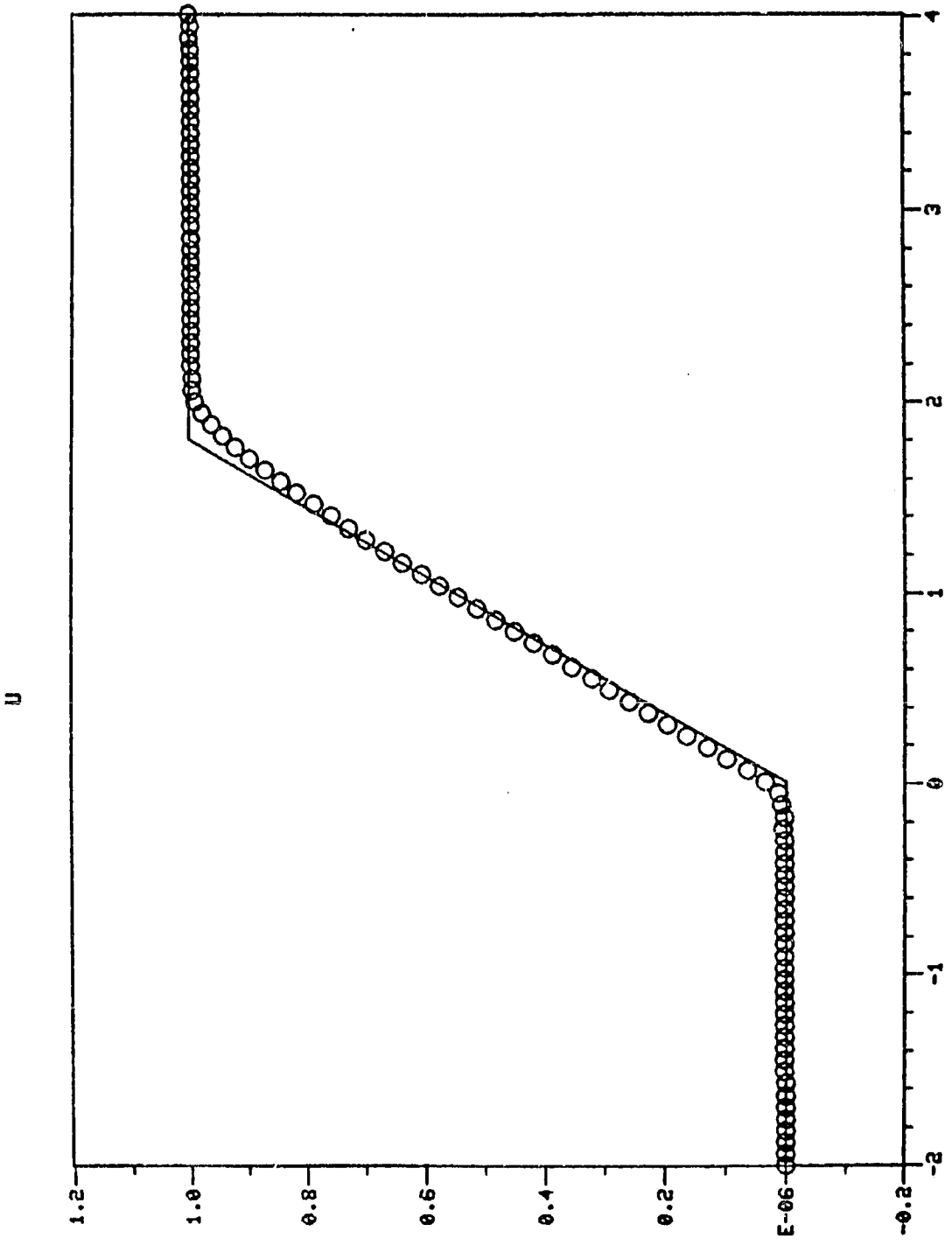


Figure III.5.3c. Expansion Wave (N = 100, t = 1.8, NSTEP = 30)

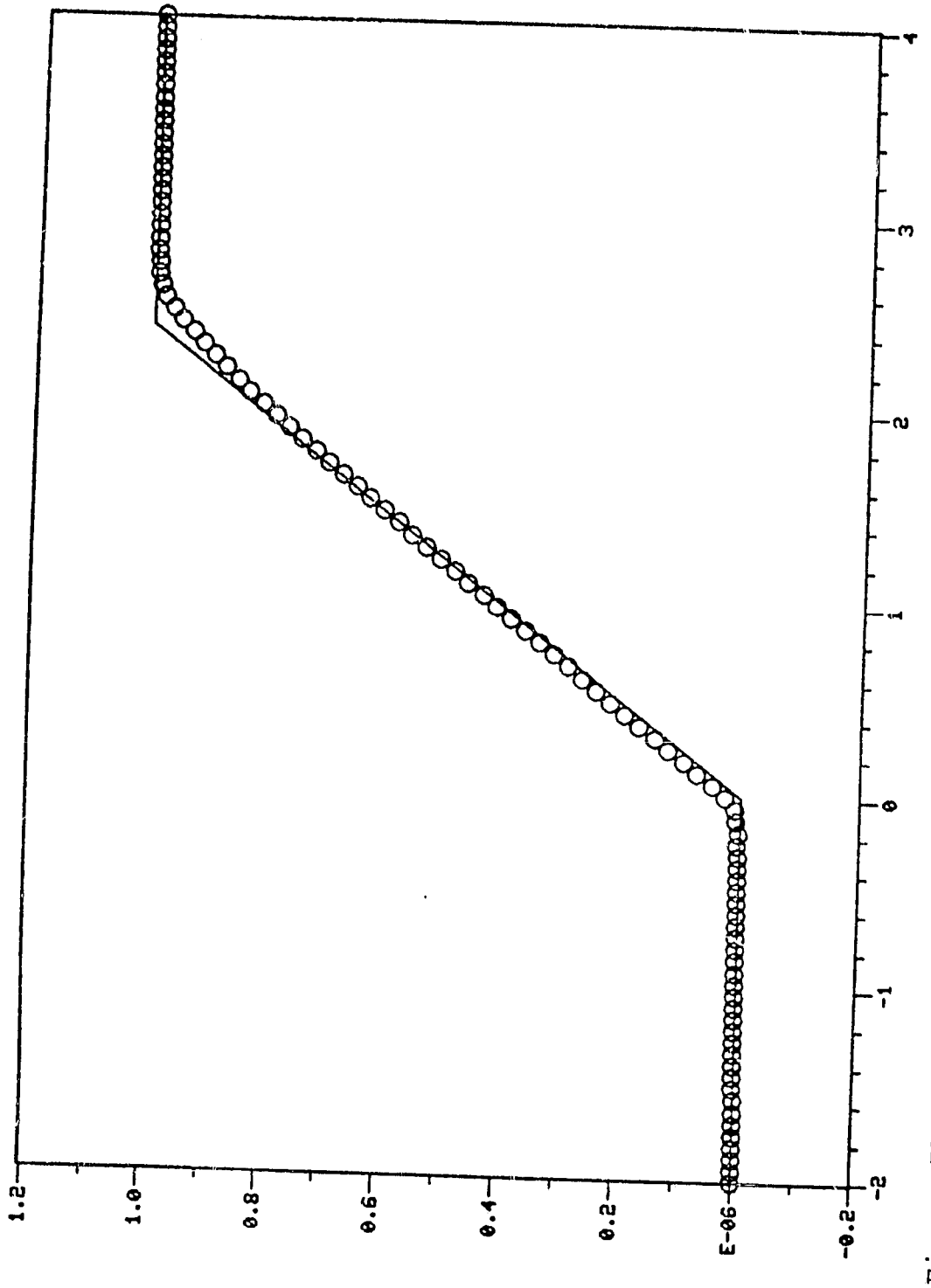


Figure III.5.3d. Expansion Wave (N = 100, t = 2.4, NSTEP = 40)

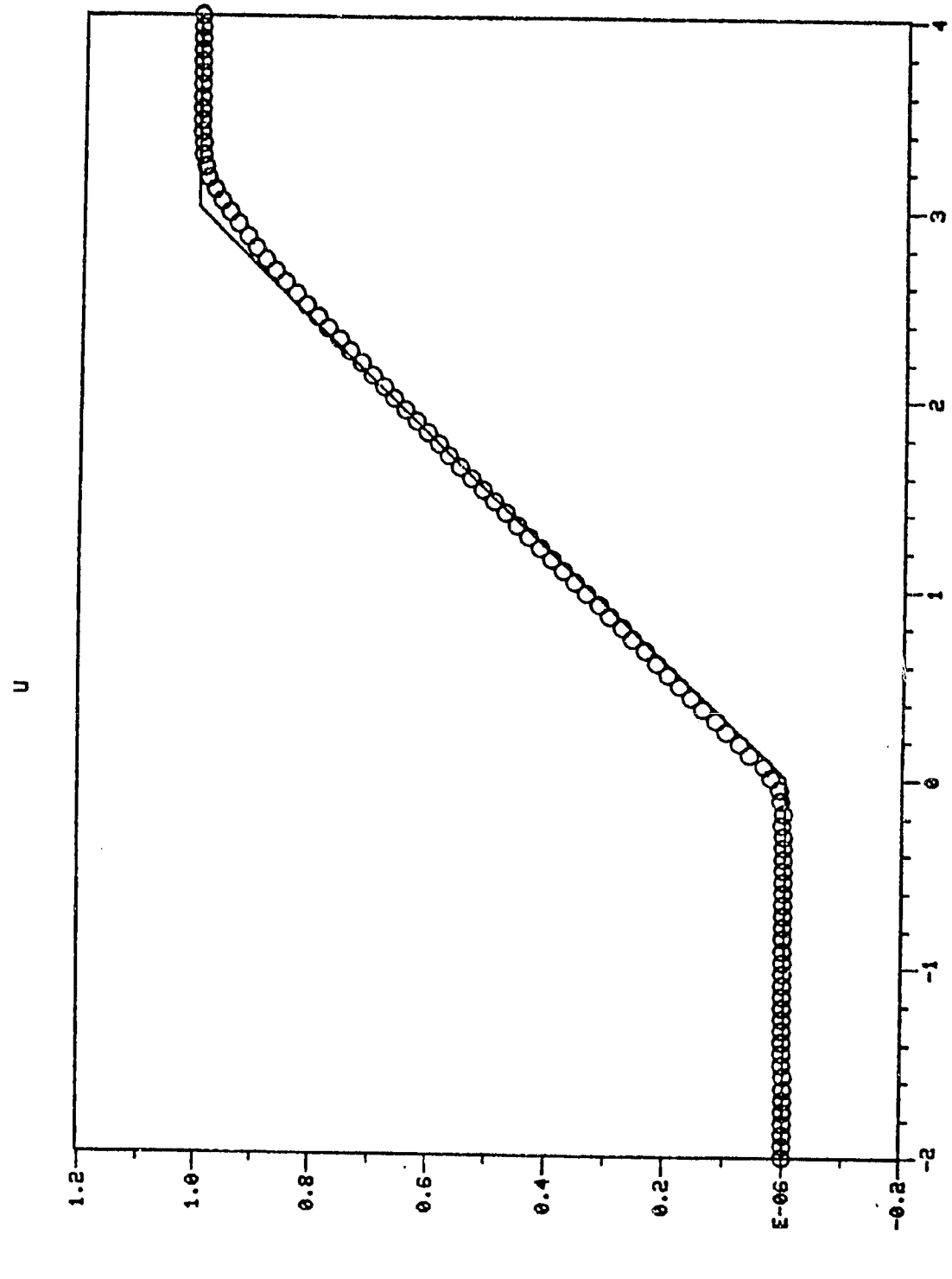


Figure III.5.3e. Expansion Wave (N = 100, t = 3. , NSTEP = 50)

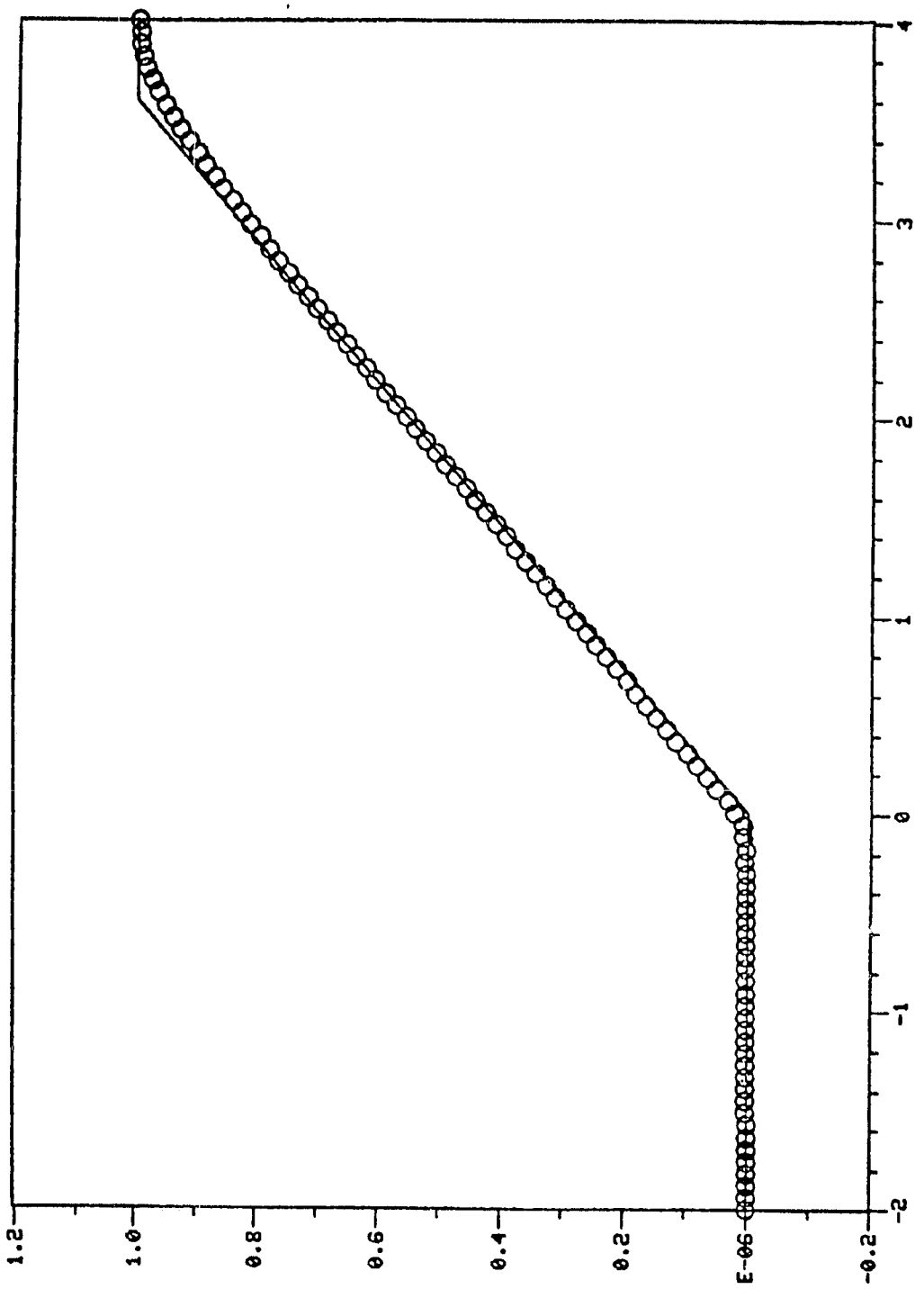


Figure III.5.3f. Expansion Wave (N = 100, t = 3.6, NSTEP = 60)

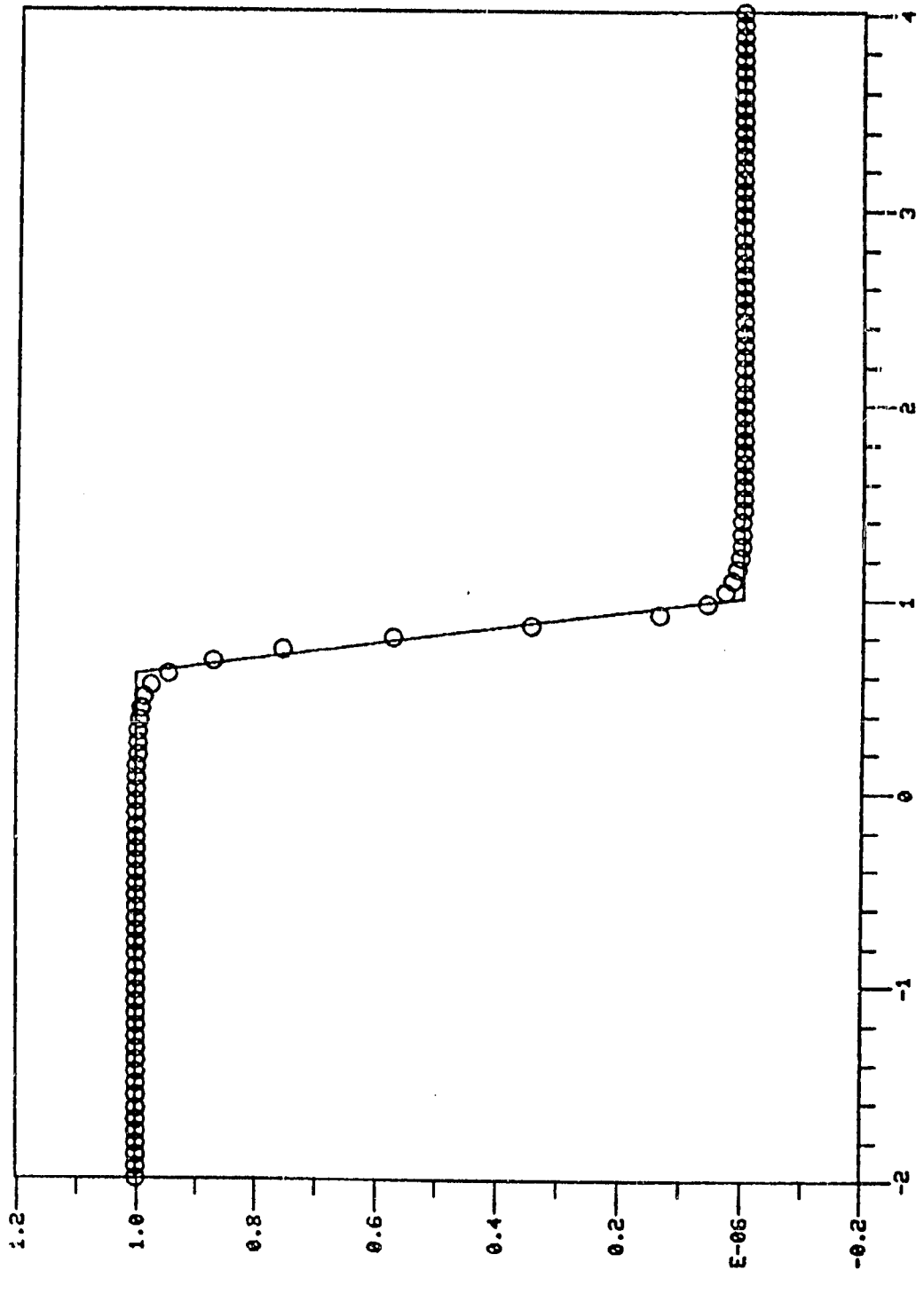


Figure III.5.4a. Compression Wave (N = 100, t = 0.6, NSTEP = 10)

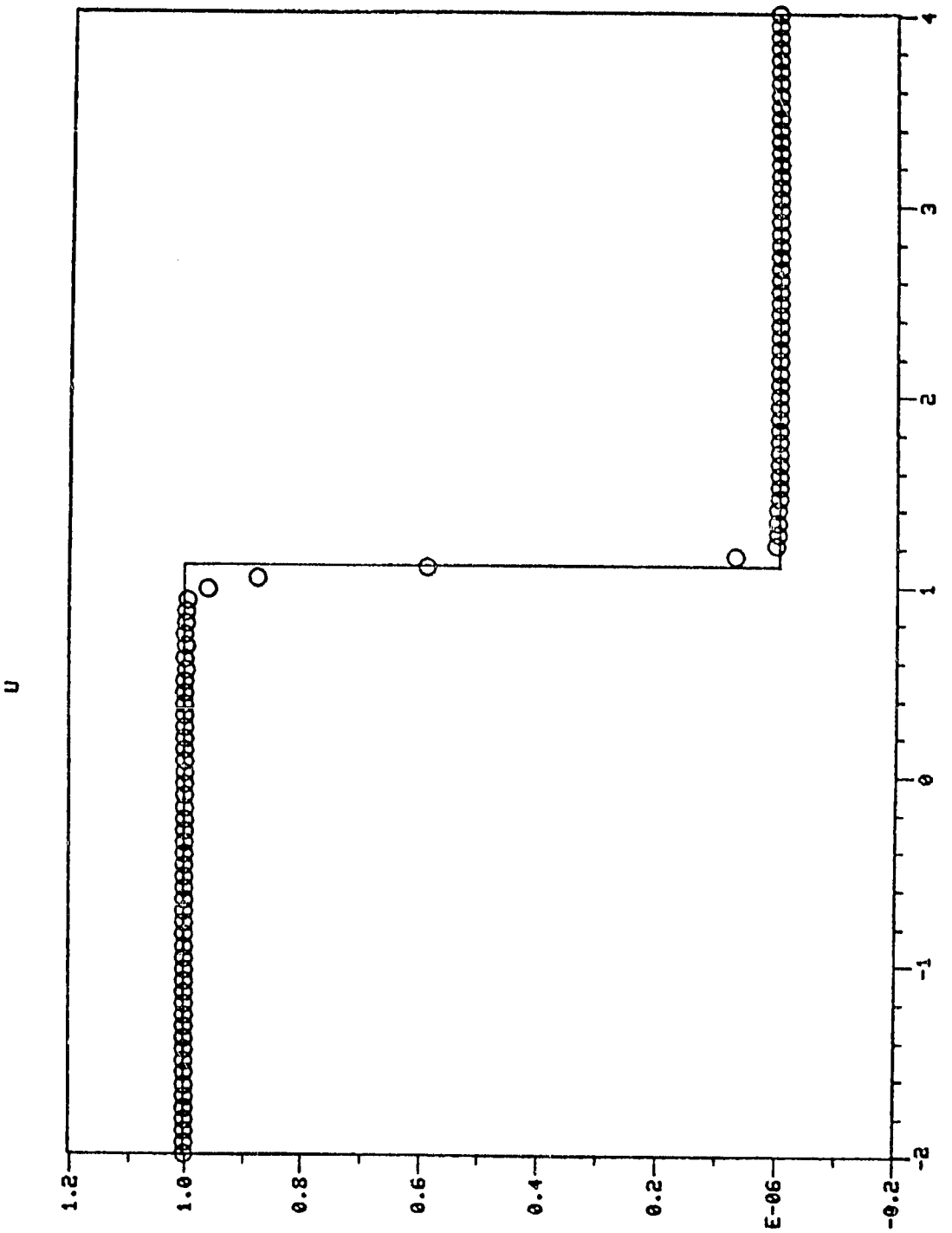


Figure III.5.4b. Compression Wave (N = 100, t = 1.2, NSTEP = 20)

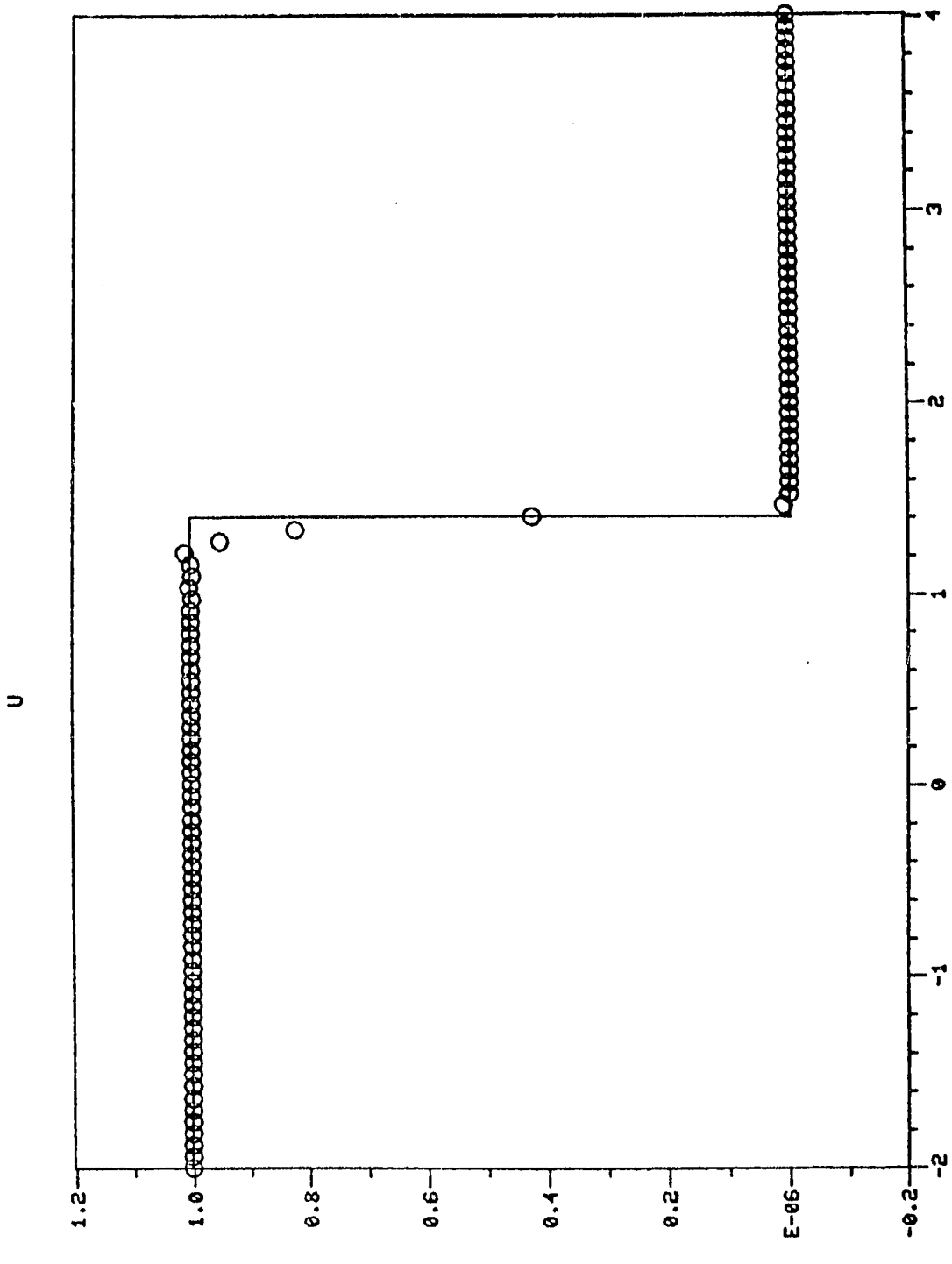


Figure III.5.4c. Compression Wave (N = 100, t = 1.8, NSTEP = 30)

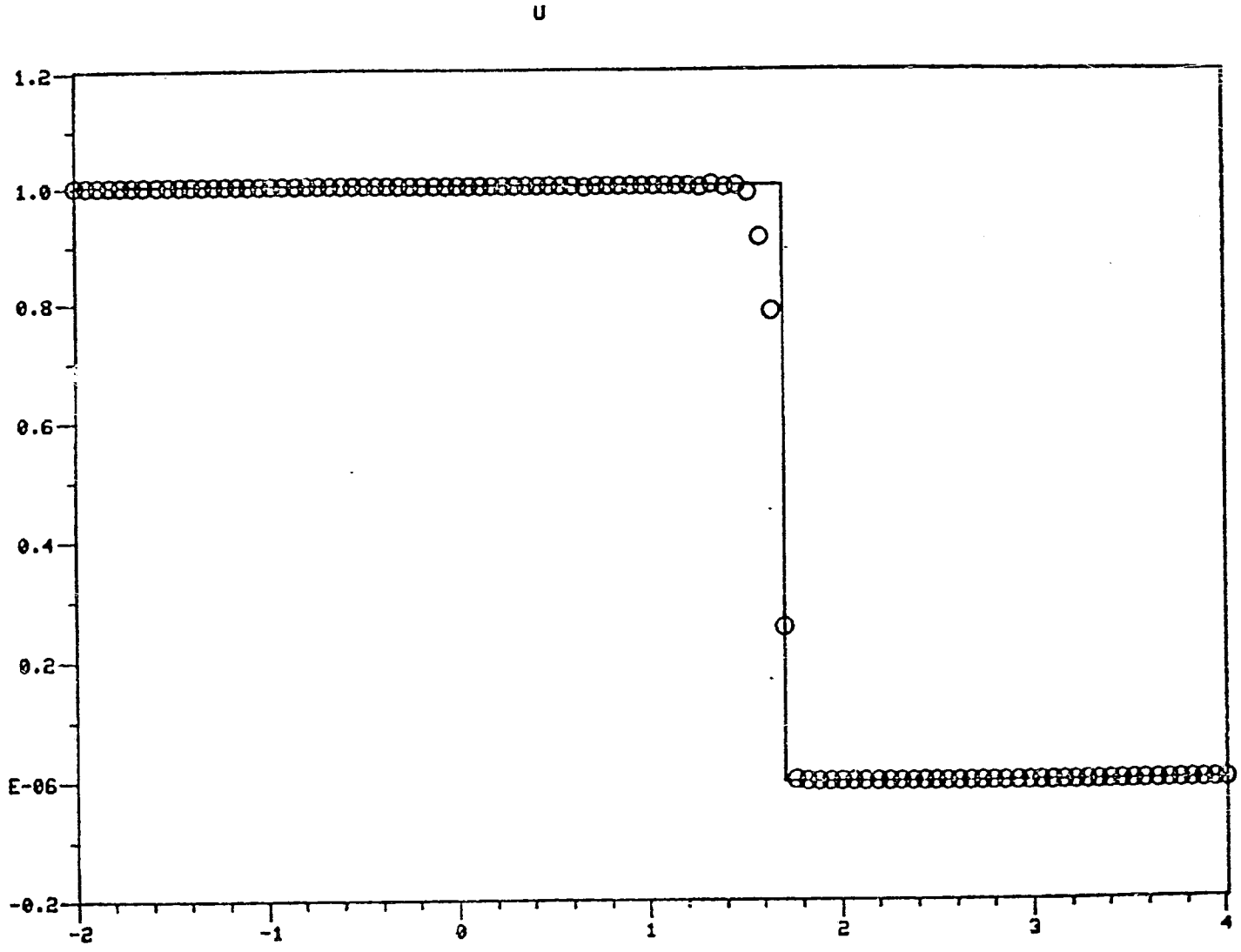


Figure III.5.4d. Compression Wave (N = 100, t = 2.4, NSTEP = 40)

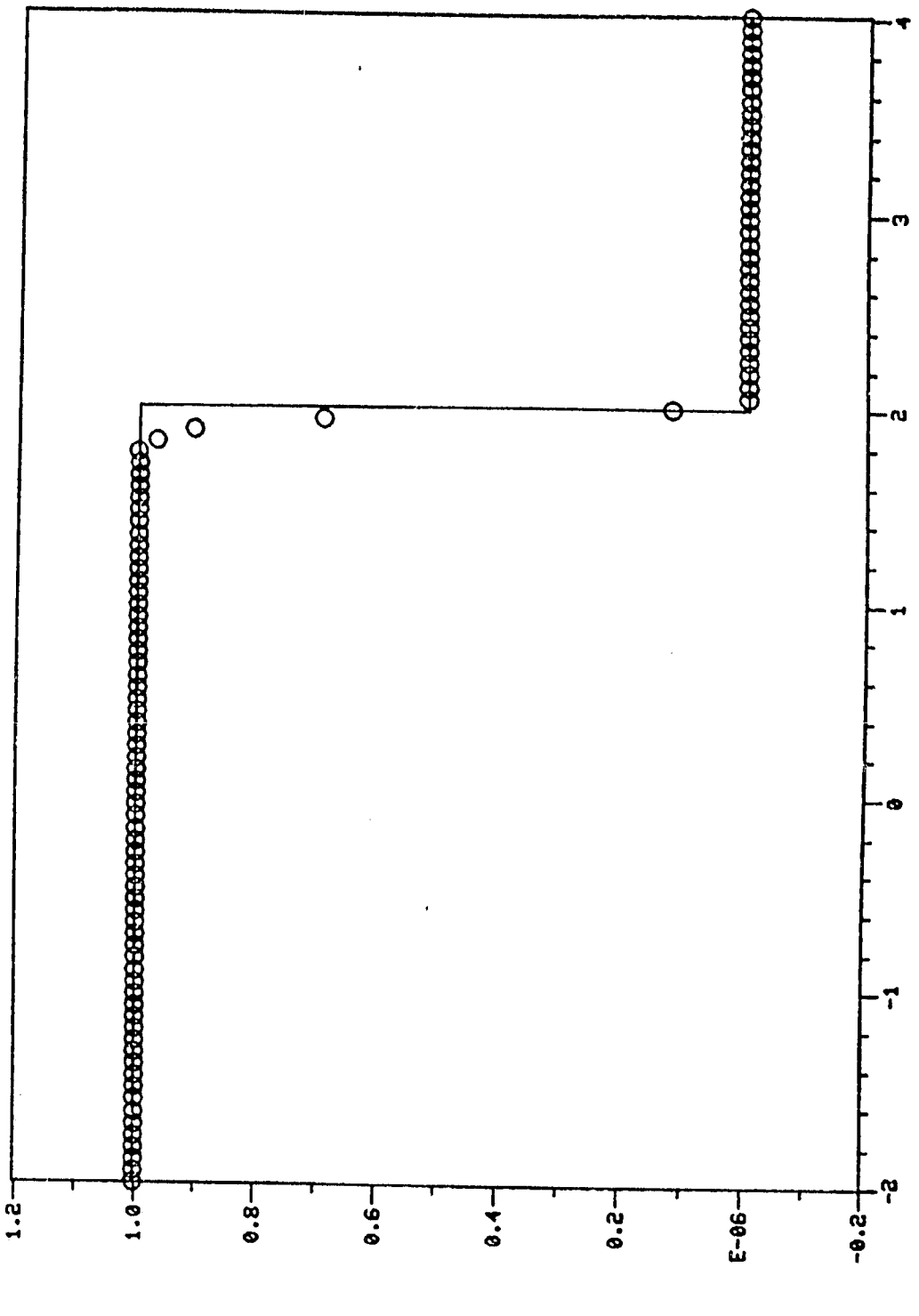


Figure III.5.4e. Compression Wave (N = 100, t = 3. , NSTEP = 50)

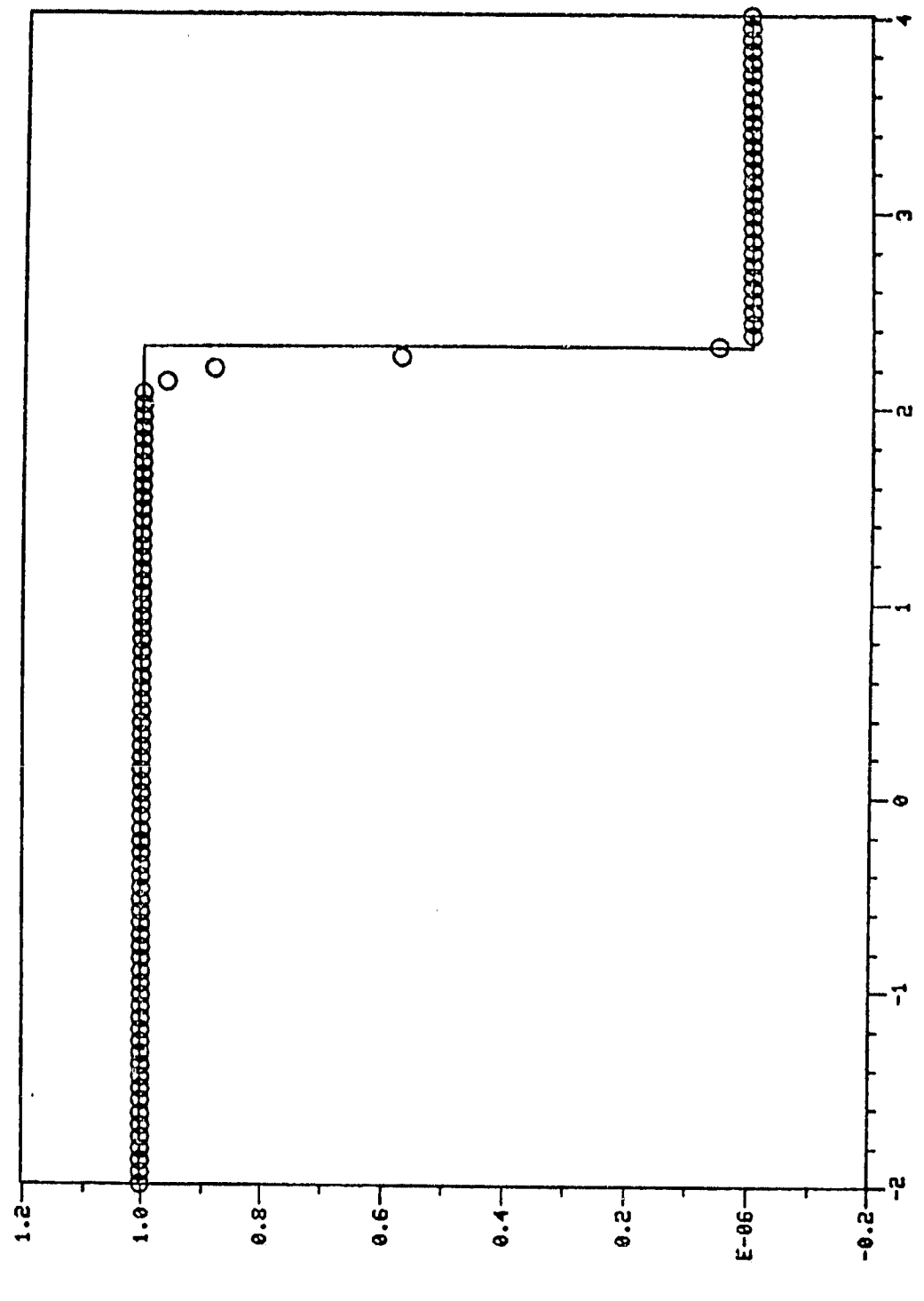


Figure III.5.4f. Compression Wave (N = 100, t = 3.6, NSTEP = 60)

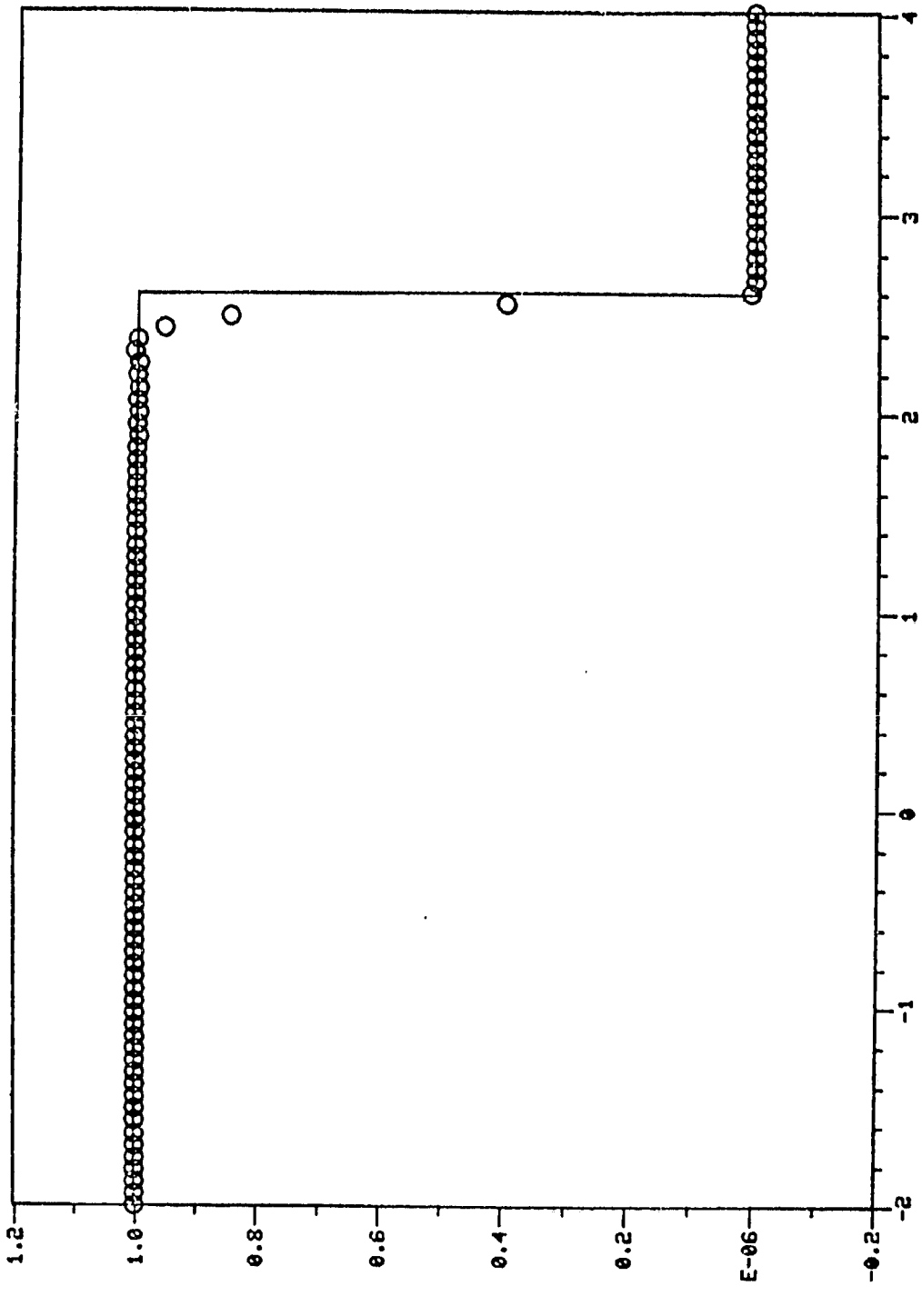


Figure III.5.4g. Compression Wave (N = 100, t = 4.2, NSTEP = 70)

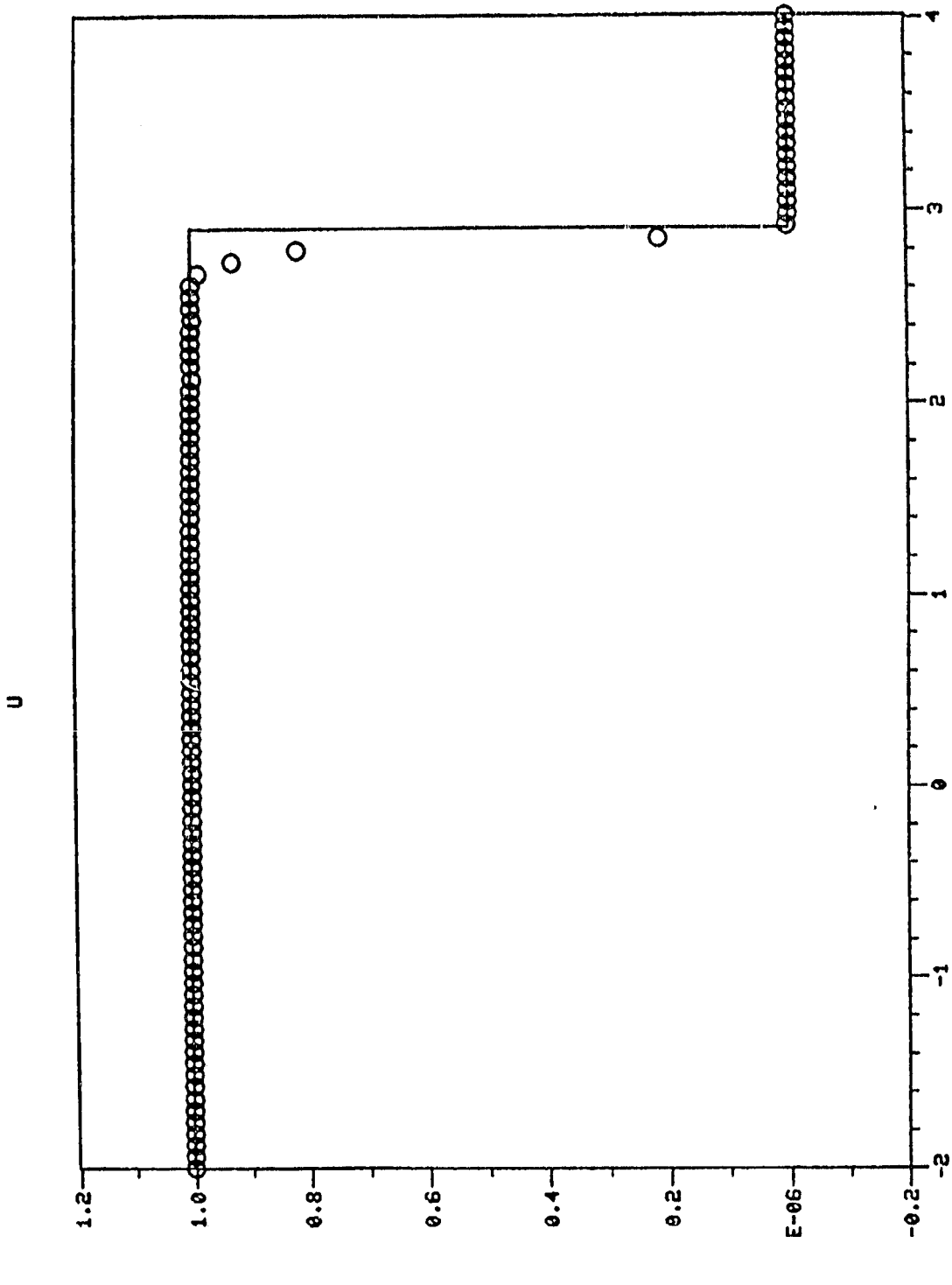


Figure III.5.4h. Compression Wave (N = 100, t = 4.8, NSTEP = 80)

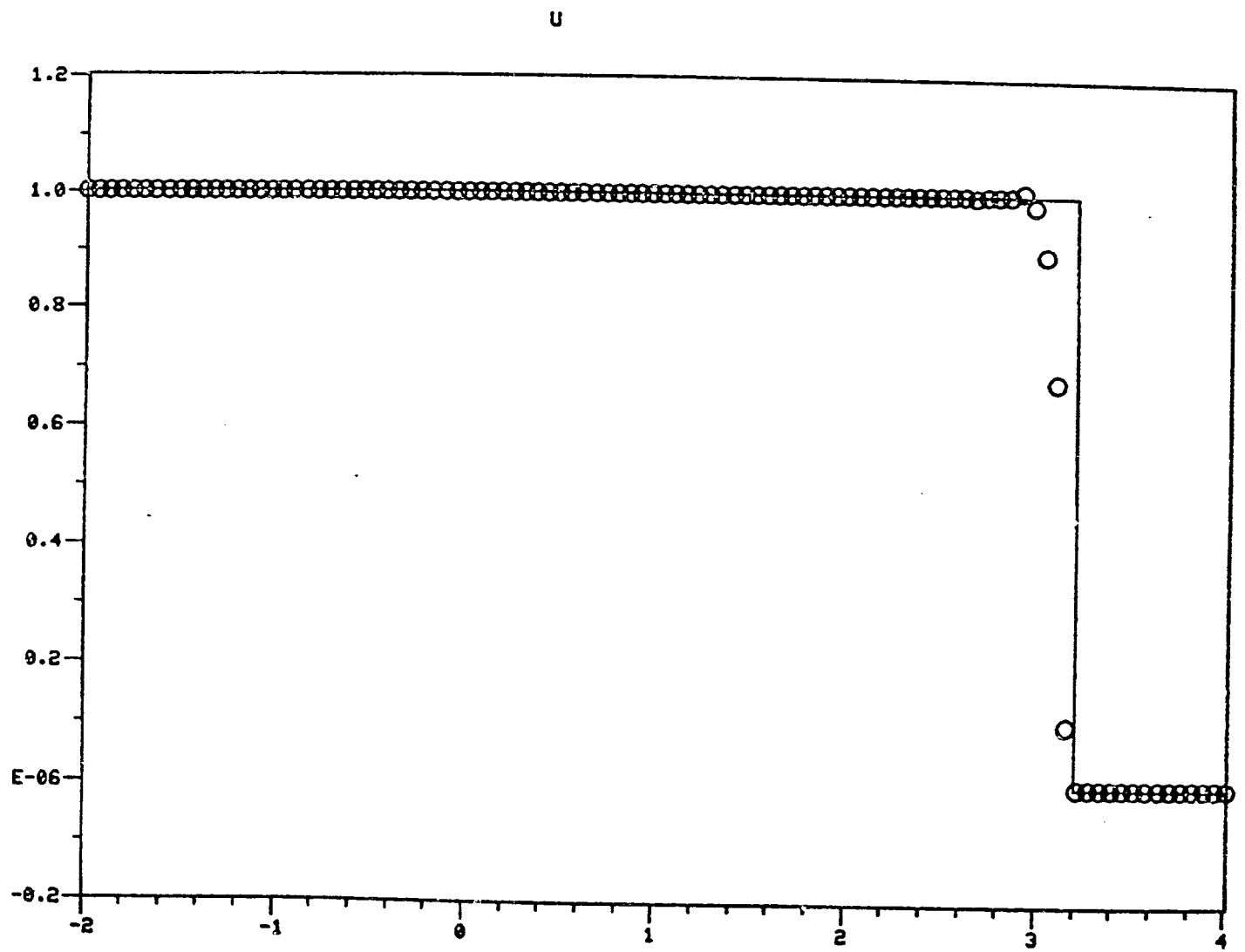


Figure III.5.4i. Compression Wave (N = 100, t = 5.4, NSTEP = 90)

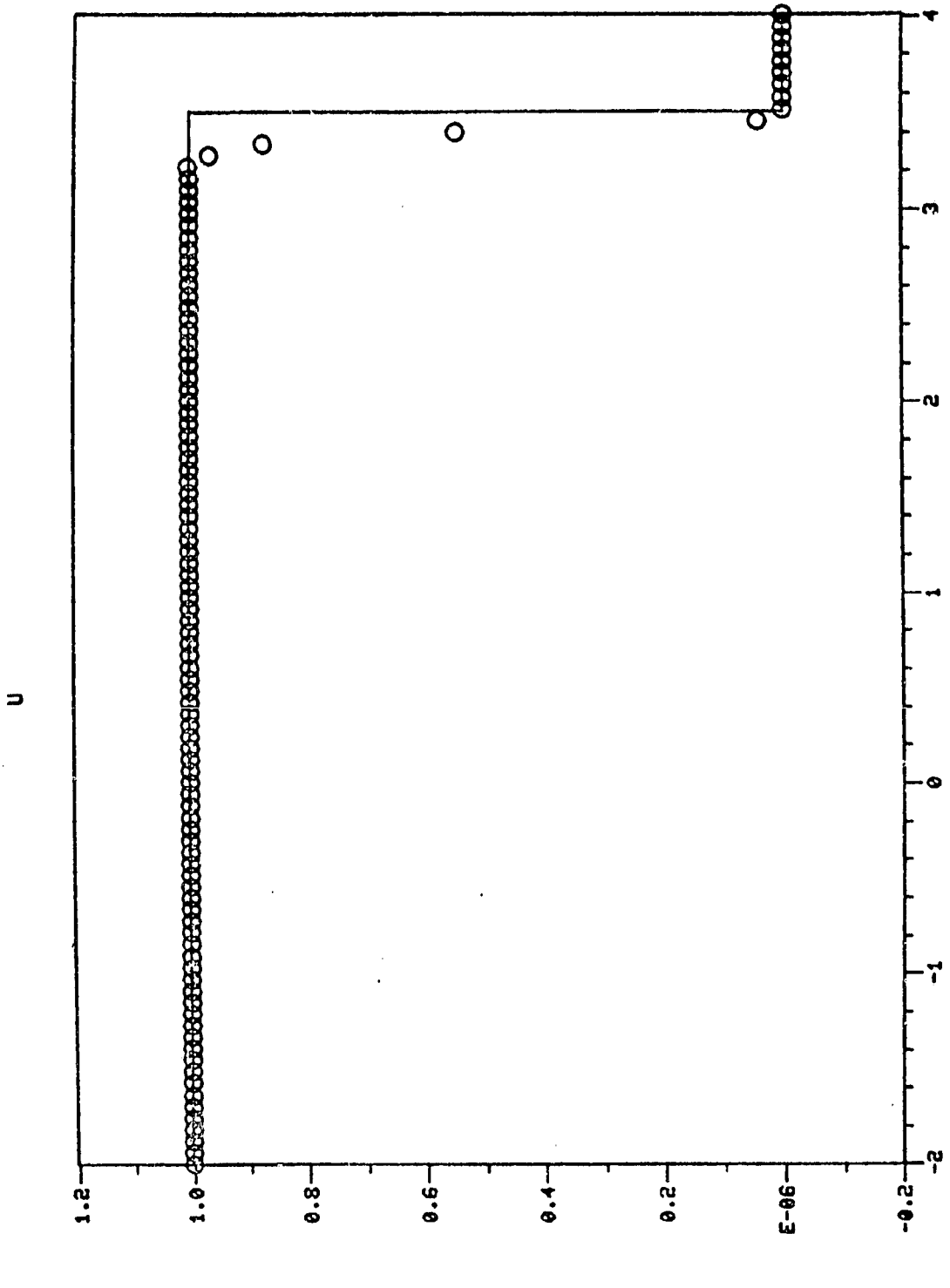


Figure III.5.4j. Compression Wave (N = 100, t = 6. , NSTEP = 100)

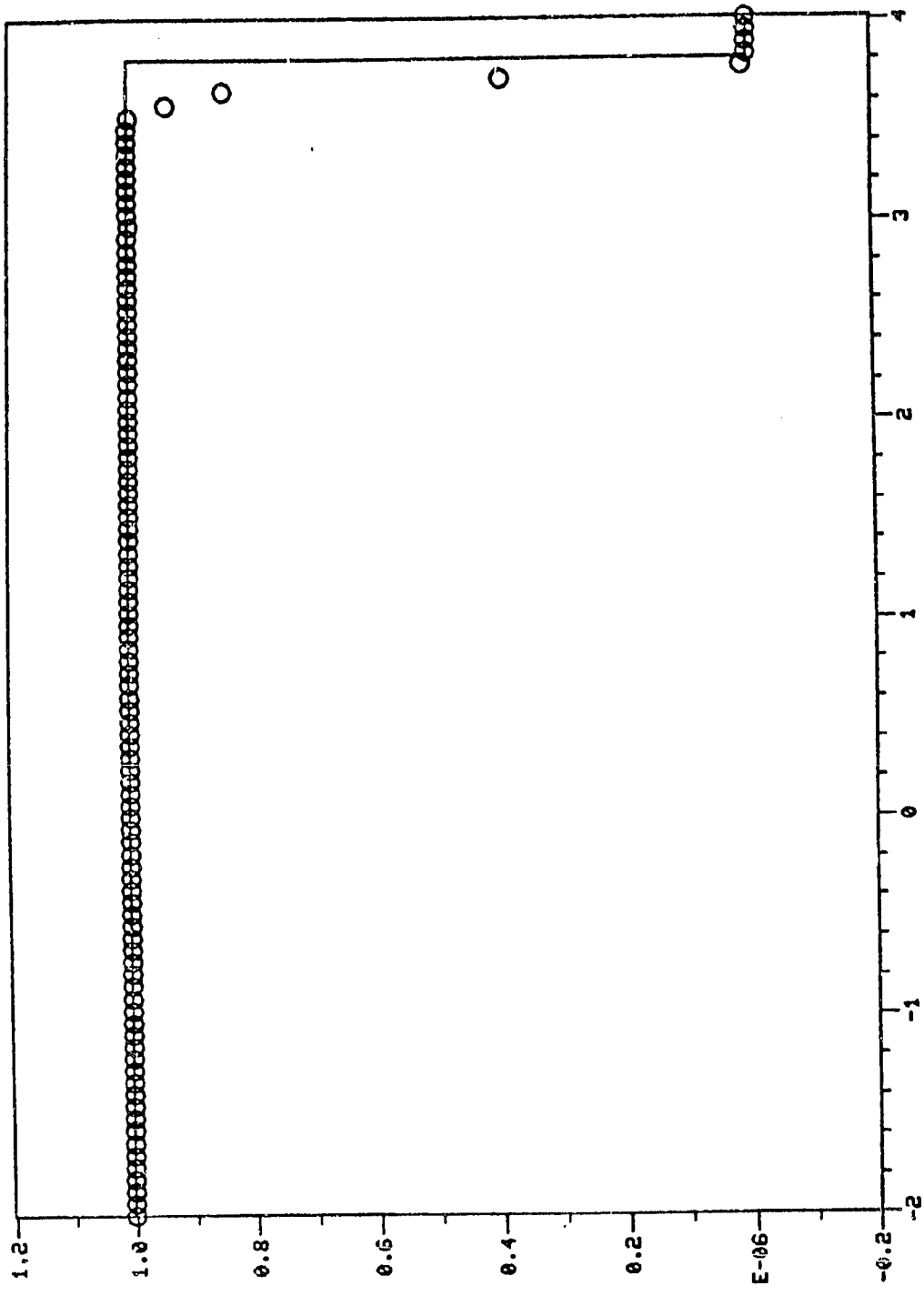


Figure III.5.4k. Compression Wave (N = 100, t = 6.6, NSTEP = 110)

III.6. Riemann Problem

We now take up a much more formidable problem from the physical, mathematical and computational points of view. We refer to the classical shock tube problem of Riemann. At t = 0 we have a tube of infinite extent containing a gas in two distinct states separated by a diaphragm. The diaphragm is then instantaneously removed the result being a shock wave moving in one direction followed by a contact discontinuity, these in addition to a rarefaction wave travelling in the opposite direction. This situation is illustrated diagrammatically in Figure III.6.1.

The explicit solution to this problem can be provided, e.g. see Liepmann and Roshko [31]. Specifically all of the state variables change on passing from one side of the centered expansion wave to the other (i.e. $p_4 \neq p_3$, $\rho_4 \neq \rho_3$, $u_4 \neq u_3$). This transition occurs in a continuous fashion although derivatives of fluid flow quantities may have a jump. At the contact discontinuity, which is the locus of the fluid initially at x = 0, the density is discontinuous (i.e. $p_2 = p_3$, $\rho_2 \neq \rho_3$, $u_2 = u_3$). All flow quantities generally experience a jump at the shock (i.e., $p_2 \neq p_1$, $\rho_2 \neq \rho_1$, $u_2 \neq u_1$). Here p is pressure, ρ is density, and u is velocity.

The important parameters can be obtained as follows. The shock strength, p_2/p_1, is implicitly related to the diaphragm pressure ratio, p_4/p_1, by the basic shock tube equation

$$\frac{p_4}{p_1} = \frac{p_2}{p_1}\left[1 - \frac{(\gamma-1)(a_1/a_4)(p_2/p_1-1)}{\sqrt{4\gamma^2+2\gamma(\gamma+1)(p_2/p_1-1)}}\right]^{-2\gamma/(\gamma-1)}$$

where $a_i = (\gamma p_i/\rho_i)^{1/2}$. Thus we can use Newton's method with an initial guess of $(1+p_4/p_1)/2$ to iteratively calculate p_2/p_1 and hence p_2. We already know that $p_3 = p_2$. We thus have p_1, p_2, p_3, p_4. We can then determine ρ_2 from the Rankine-Hugoniot relation

$$\frac{\rho_2}{\rho_1} = \frac{1 + \frac{\gamma+1}{\gamma-1}\cdot\frac{p_2}{p_1}}{\frac{\gamma+1}{\gamma-1} + \frac{p_2}{p_1}}.$$

Since we assume a perfect gas we have the isentropic relation

$$\frac{\rho_3}{\rho_4} = \left[\frac{p_3}{p_4}\right]^{1/\gamma}$$

for the determination of ρ_3. Now we also have $\rho_1, \rho_2, \rho_3, \rho_4$. The shock speed, c_s, is determined by

$$c_s = a_1\left[\frac{\gamma-1}{2\gamma} + \frac{\gamma-1}{2\gamma}\cdot\frac{p_2}{p_1}\right]^{1/2}.$$

The head of the rarefaction wave travels at the local speed of sound, a_4, through the undisturbed fluid. The tail of the rarefaction wave travels at the speed

$$c_3 = a_4 - \frac{\gamma+1}{2}|u_p|$$

where $|u_p| = u_2 = u_3$ is the speed of propagation of the contact discontinuity which may be calculated from either

$$u_2 = a_1\left(\frac{p_2}{p_1} - 1\right)\left[\frac{2/\gamma}{(\gamma+1)p_2/p_1+(\gamma-1)}\right]^{1/2}$$

or

$$u_3 = \frac{2a_4}{\gamma-1}\left[1 - \left(\frac{p_3}{p_4}\right)^{(\gamma-1)/2\gamma}\right].$$

These quantities furnish a complete solution to the problem with the exception of the variations within the expansion wave. Thus we have at our disposal the means for an exact comparison on a problem of substance.

The one dimensional Euler equations may be written in the conservation law form [46]

$$\rho_t + (\rho u)_x = 0$$

$$(m)_t + \left(\frac{m^2}{\rho} + p\right)_x = 0$$

$$(e)_t + \left(\frac{m}{\rho}(e+p)\right)_x = 0$$

where $m = \rho u$ is momentum and $e = \rho\varepsilon + \frac{1}{2}\rho u^2$ is the total energy per unit volume with ε the internal energy per unit mass. For a polytropic gas $\varepsilon = p/(\gamma-1)\rho$. Hence we have

$$p = (\gamma-1)\rho\varepsilon = (\gamma-1)\left(e - \frac{1}{2}\rho u^2\right) = (\gamma-1)\left(e - \frac{1}{2}\frac{m^2}{\rho}\right) .$$

We may write the above in the vector form

$$\vec{U}_t + \vec{F}_x(\vec{U}) = 0 .$$

with the state vector

$$\vec{U} \equiv \begin{bmatrix} \rho \\ m \\ e \end{bmatrix}$$

and the flux vector

$$\vec{F}(\vec{U}) \equiv \begin{bmatrix} m \\ \frac{m^2}{\rho} + p \\ \frac{m}{\rho}(e+p) \end{bmatrix} .$$

This may also be written in the quasilinear form

$$\vec{U}_t + A(\vec{U})\vec{U}_x = 0$$

where

$$A(\vec{U}) \equiv \frac{\partial \vec{F}}{\partial \vec{U}} = \begin{bmatrix} 0 & 1 & 0 \\ (\gamma-3)u^2/2 & (3-\gamma)u & (\gamma-1) \\ (\gamma-1)u^3 - \gamma eu/\rho & \frac{\gamma e}{\rho} - 3(\gamma-1)\frac{u^2}{2} & \gamma u \end{bmatrix}$$

with eigenvalues $\lambda_1 = u$, $\lambda_2 = u+c$, $\lambda_3 = u-c$. Here c is the local speed of sound and is determined from

$$c^2 = \frac{\gamma p}{\rho} = \frac{\gamma}{\rho}(\gamma-1)\left[e - \frac{1}{2}\frac{m^2}{\rho}\right] .$$

At this point we note that many subsequent relations are merely one dimensional specializations of two dimensional results derived in the next section. Hence, we omit detailed derivations in this instance so as to avoid redundancy.

In what follows we would like to produce a $Q \ni$

$$Q^{-1} A\, Q = \Lambda \quad \text{(diagonal)}.$$

However, computing the eigenvalues and eigenvectors of A is quite formidable due to the complexity of the Jacobian matrix. Therefore, we take the following circuitous route [53].

The one dimensional Euler equations may be written as

$$\vec{\tilde{U}}_t + \tilde{A}\,\vec{\tilde{U}}_x = 0$$

where

$$\vec{\tilde{U}} = [\rho \quad u \quad p]^T$$

and

$$\tilde{A} = \begin{bmatrix} u & \rho & 0 \\ 0 & u & 1/\rho \\ 0 & \rho c^2 & u \end{bmatrix} .$$

Here $\tilde{A} = M^{-1} A M$ where

$$M = \begin{bmatrix} 1 & 0 & 0 \\ u & \rho & 0 \\ u^2/2 & \rho u & 1/(\gamma-1) \end{bmatrix} \Rightarrow$$

$$M^{-1} = \begin{bmatrix} 1 & 0 & 0 \\ -u/\rho & 1/\rho & 0 \\ (\gamma-1)u^2/2 & (1-\gamma)u & (\gamma-1) \end{bmatrix} .$$

$\tilde{A}$ is more easily diagonalized as

$$\tilde{Q}^{-1}\tilde{A}\,\tilde{Q} = \Lambda \equiv \text{diag}\,[u \quad u+c \quad u-c]$$

with

$$\tilde{Q} = \begin{bmatrix} 1 & \rho/(\sqrt{2}\,c) & \rho/(\sqrt{2}\,c) \\ 0 & 1/\sqrt{2} & -1/\sqrt{2} \\ 0 & \rho c/\sqrt{2} & \rho c/\sqrt{2} \end{bmatrix} \Rightarrow \tilde{Q}^{-1} = \begin{bmatrix} 1 & 0 & -1/c^2 \\ 0 & 1/\sqrt{2} & 1/(\sqrt{2}\,\rho c) \\ 0 & -1/\sqrt{2} & 1/(\sqrt{2}\,\rho c) \end{bmatrix} .$$

Hence $Q = M\tilde{Q} \Rightarrow Q^{-1} = \tilde{Q}^{-1}M^{-1}$.

Now consider our original equations

$$\vec{U}_t + A\vec{U}_x = 0 \Rightarrow$$
$$\vec{U}_t + Q\Lambda Q^{-1}\vec{U}_x = 0 .$$

Now freeze $\vec{U}$, i.e. perform a local linearization of the system of equations. Also let $\vec{\bar{U}} = Q^{-1}\vec{U}$. Then we have

$$\vec{\bar{U}}_t + \Lambda\vec{\bar{U}}_x = 0$$

which is nothing more than our model problem in triplicate. However, since for subsonic flow we have $u + c$ and $u - c$ of opposite sign we need to shift spatial derivatives in conflicting directions.

We could circumvent this difficulty if we could split our flux vector, $\vec{F}$, into pieces with univalent eigenvalues. This is precisely the motivation for the Flux Vector Splitting technique of Steger and Warming [50] which we now adapt to our purposes.

Observe that $\vec{F}(\vec{U})$ is a homogeneous function of degree one, i.e. $\vec{F}(\alpha\vec{U}) = \alpha\vec{F}(\vec{U})$. Euler's theorem on homogeneous functions then implies that $\vec{F} = A\vec{U}$. Thus

$$\vec{F} = A\vec{U} = Q\Lambda Q^{-1}\vec{U} = Q(\Lambda^+ + \Lambda^-)Q^{-1}\vec{U}$$

$$= Q\Lambda^+Q^{-1}\vec{U} + Q\Lambda^- Q^{-1}\vec{U} = \vec{F}^+ + \vec{F}^-$$

where Λ^+ has nonnegative elements and Λ^- has nonpositive elements. We prefer the following split:

$$\lambda_1^+ = \frac{u + |u|}{2} , \qquad \lambda_1^- = \frac{u - |u|}{2} ,$$

$$\lambda_2^+ = \lambda_1^+ + c \quad , \qquad \lambda_2^- = \lambda_1^- \quad ,$$

$$\lambda_3^+ = \lambda_1^+ \quad , \qquad \lambda_3^- = \lambda_1^- - c \quad .$$

Note that this splitting yields homogeneous $\vec{F}^+$ and $\vec{F}^-$.

In general, if

$$\hat{\Lambda} \equiv \text{diag}\,[\hat{\lambda}_1 \quad \hat{\lambda}_2 \quad \hat{\lambda}_3]$$

then

$$\hat{F} \equiv \frac{\rho}{2\gamma} \begin{bmatrix} 2(\gamma-1)\hat{\lambda}_1+\hat{\lambda}_2+\hat{\lambda}_3 \\ 2(\gamma-1)\hat{\lambda}_1 u+\hat{\lambda}_2(u+c)+\hat{\lambda}_3(u-c) \\ (\gamma-1)\hat{\lambda}_1 u^2+\frac{\hat{\lambda}_2}{2}(u+c)^2+\frac{\hat{\lambda}_3}{2}(u-c)^2+\hat{w} \end{bmatrix} = Q\hat{\Lambda}Q^{-1}\vec{U}$$

where $$\hat{w} = \frac{(3-\gamma)(\hat{\lambda}_2+\hat{\lambda}_3)c^2}{2(\gamma-1)} .$$

We now generalize our scheme to this setting. Once again a fourth order Runge-Kutta integration is used in time while $\vec{F}_x(\vec{U})$ is approximated by $D_x\vec{F} = D_x^-\vec{F}^+ + D_x^+\vec{F}^-$ where, letting F^+ and F^- be the generic components of $\vec{F}^+$ and $\vec{F}^-$ respectively,

$$D_x^- F^+(\vec{U}_i) = (\tau_x)_i - \varepsilon^+_{i+1/2}(\tau_x)_{i+1/2} + \varepsilon^+_{i-1/2}(\tau_x)_{i-1/2}$$

$$D_x^+ F^-(\vec{U}_i) = (\tau_x)_i + \varepsilon^-_{i+1/2}(\tau_x)_{i+1/2} - \varepsilon^-_{i-1/2}(\tau_x)_{i-1/2} .$$

Here

$$\varepsilon^+ = \min\,(h^2 \cdot |F^+_{xx}|,\ 1/2)$$

$$\varepsilon^- = \min\,(h^2 \cdot |F^-_{xx}|,\ 1/2) .$$

We now present some computational results obtained using this technique. The items of interest are the prediction of correct quantity levels, the location of the shock, contact discontinuity and rarefaction wave, the degree of resolution of the discontinuities and the suppression of oscillations, overshoots and undershoots in regions of rapid variation.

The first example is taken from Sod's survey paper [46]. The pressure ratio has a value of 10 with the diaphragm location at $x = 0.5$. The calculation was performed on both a coarse, Figs. III.6.2a-e, and a fine mesh, Figures III.6.3a-e.

The second example is from Steger and Warming's Flux Vector Splitting report [50]. The pressure ratio has a value of 5 with the diaphragm location at $x = 3$. This calculation was also performed on both a coarse, Figures III.6.4a-e, and a fine mesh, Figures III.6.5a-e.

It is clear that the method performs very well by all the above criteria. Of particular note are the invariance of shock smearing (in units of mesh widths) to mesh size and the resolution of the contact discontinuity for large t. These results are competitive with those displayed in the referenced works.

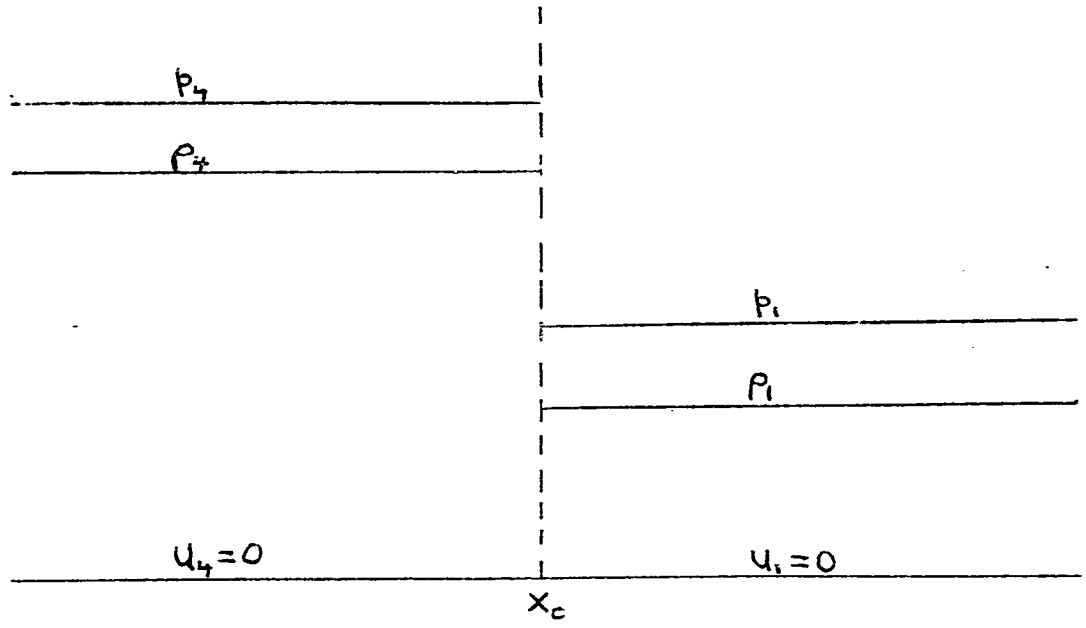


Figure III.6.1a. Shock Tube at t = 0

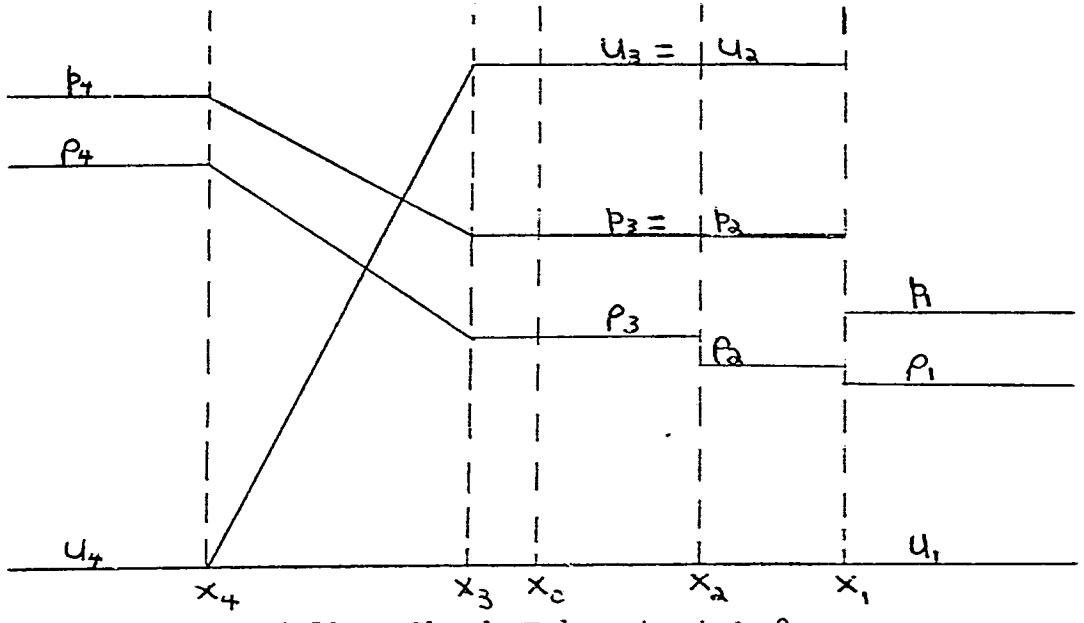


Figure III.6.1b. Shock Tube at t > 0

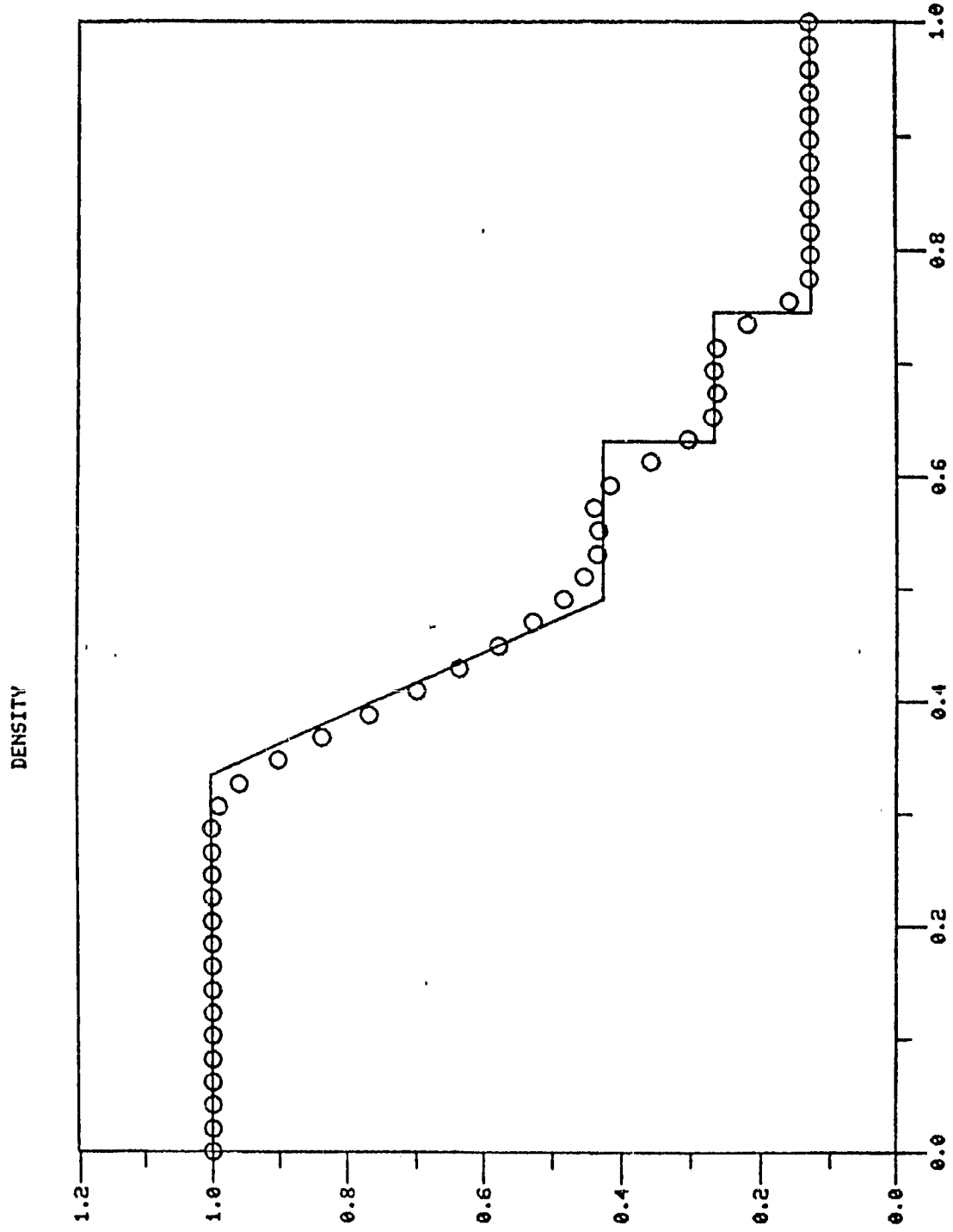


Figure III.6.2a. Sod Test Case (N = 50, t = 0.14, NSTEP = 35)

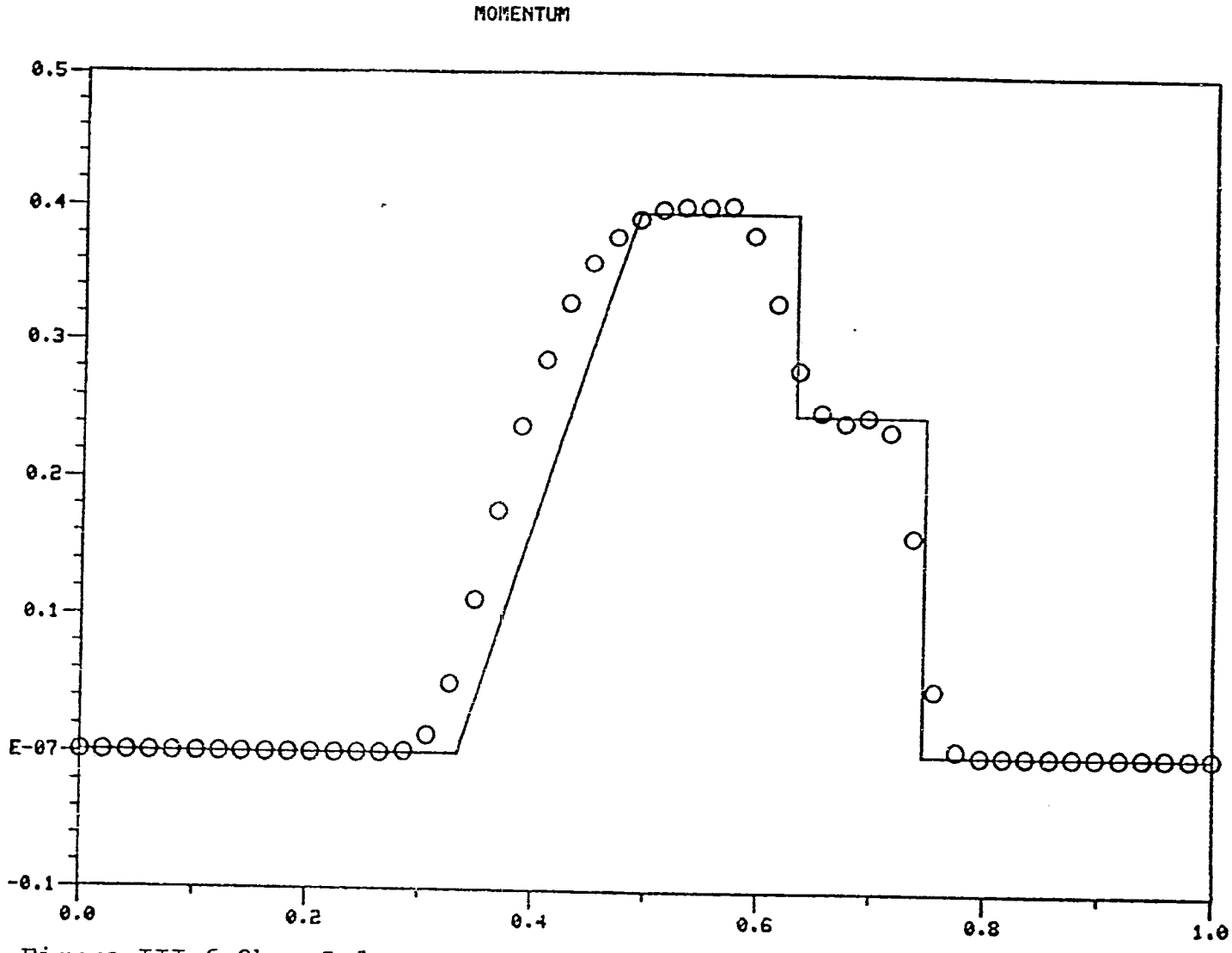


Figure III.6.2b. Sod Test Case (N = 50, t = 0.14, NSTEP = 35)

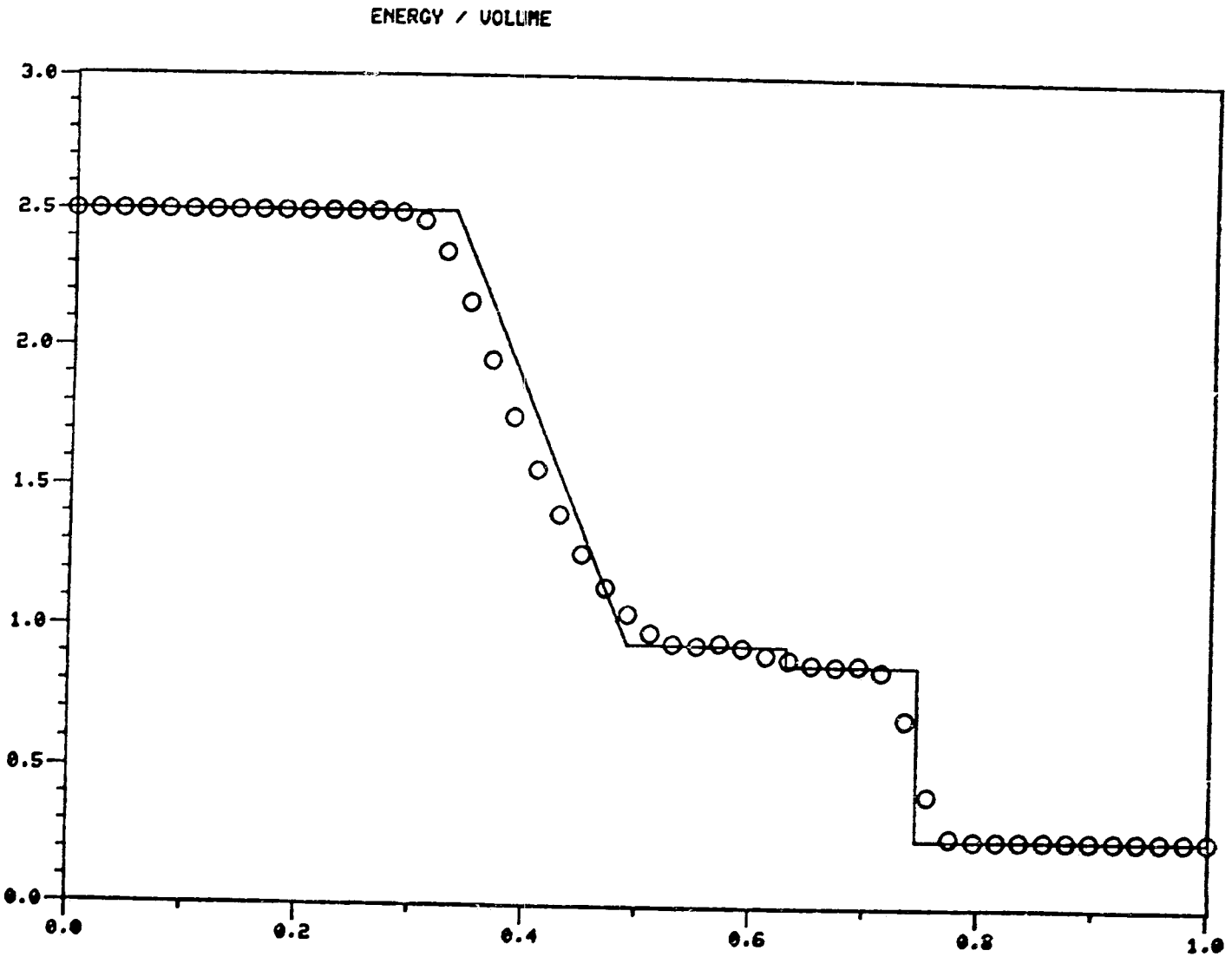


Figure III.6.2c. Sod Test Case (N = 50, t = 0.14, NSTEP = 35)

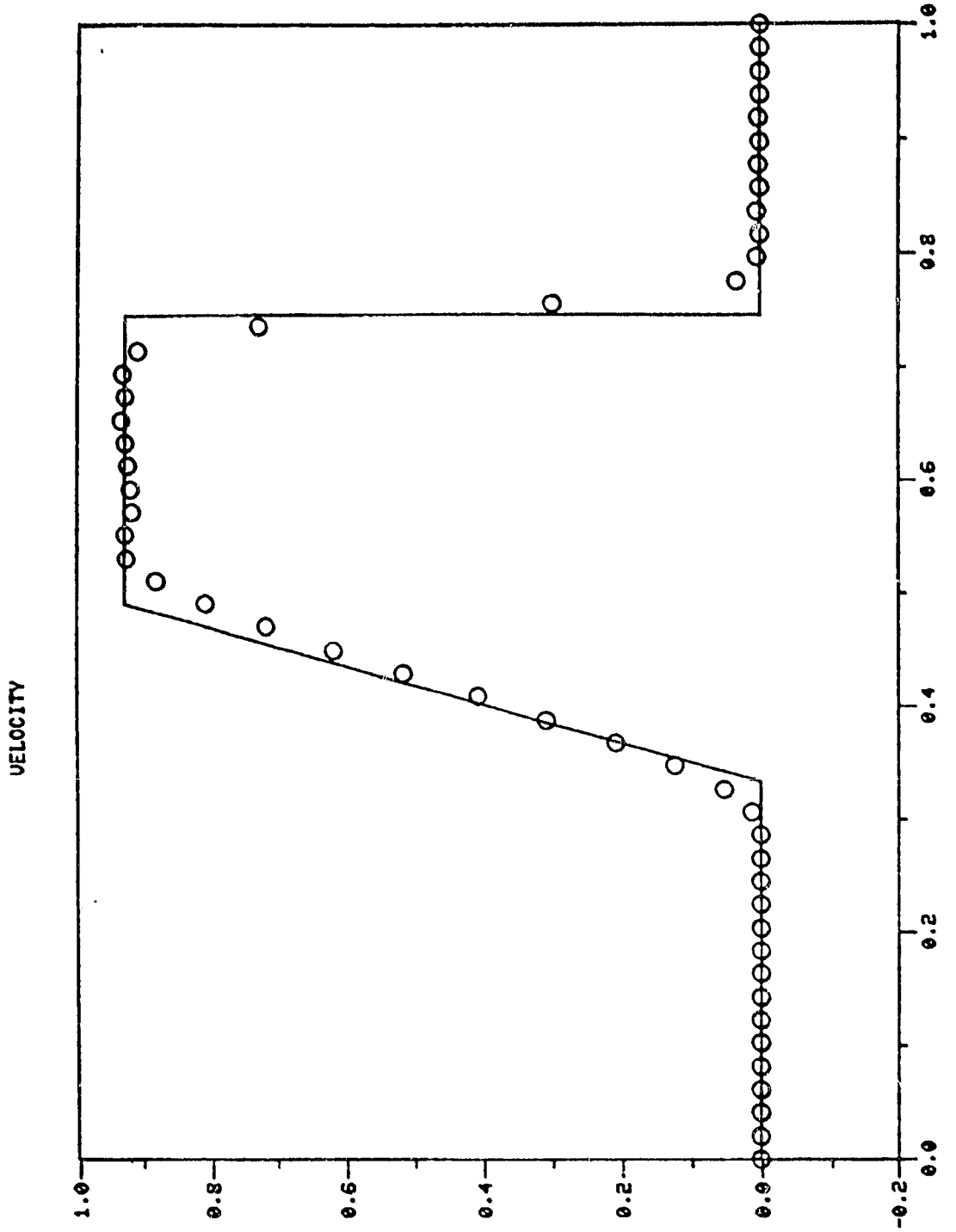


Figure III.6.2d. Sod Test Case (N = 50, t = 0.14, NSTEP = 35)

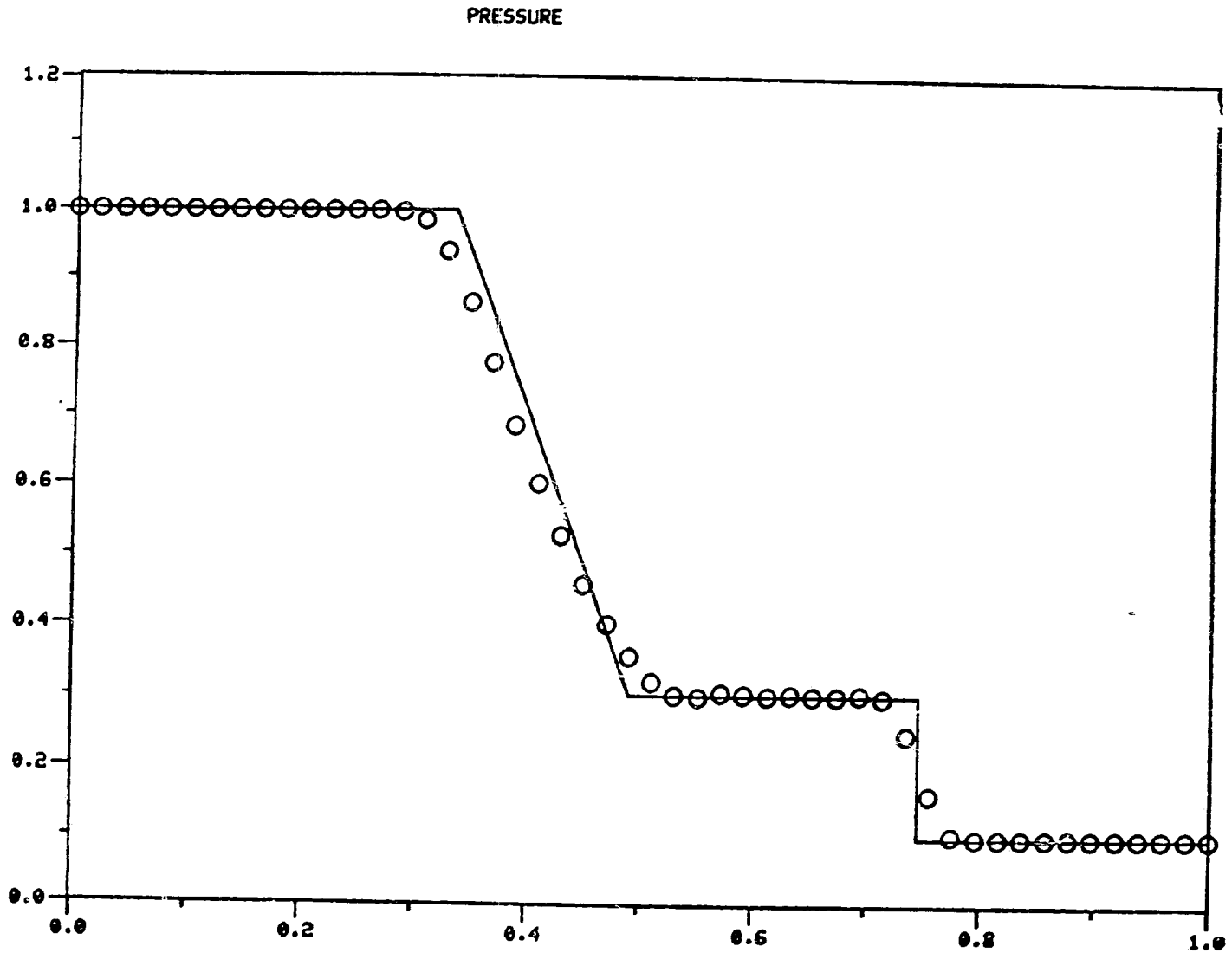


Figure III.6.2e. Sod Test Case (N = 50, t = 0.14, NSTEP = 35)

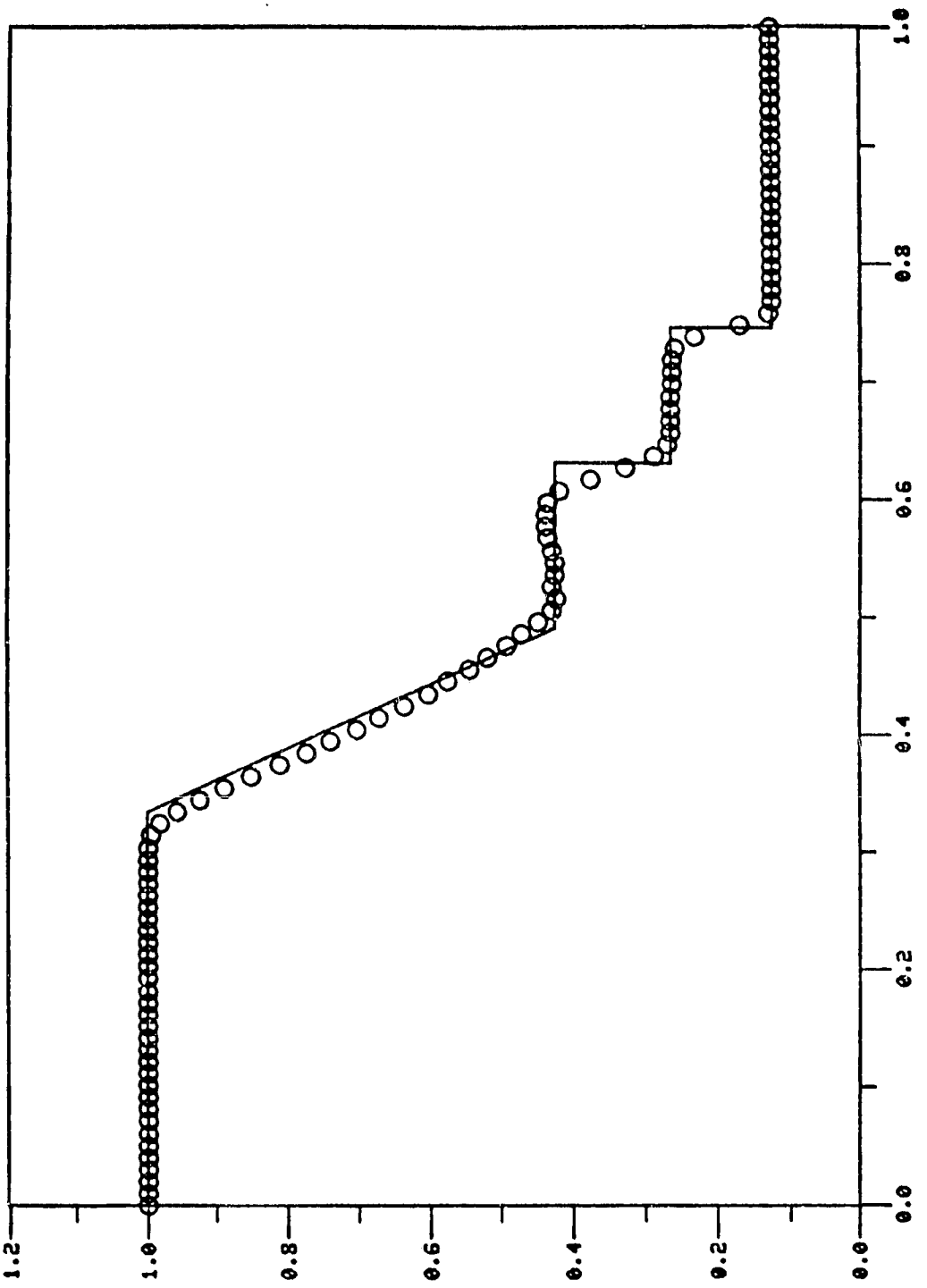


Figure III.6.3a. Sod Test Case (N = 100, t = 0.14, NSTEP = 70)

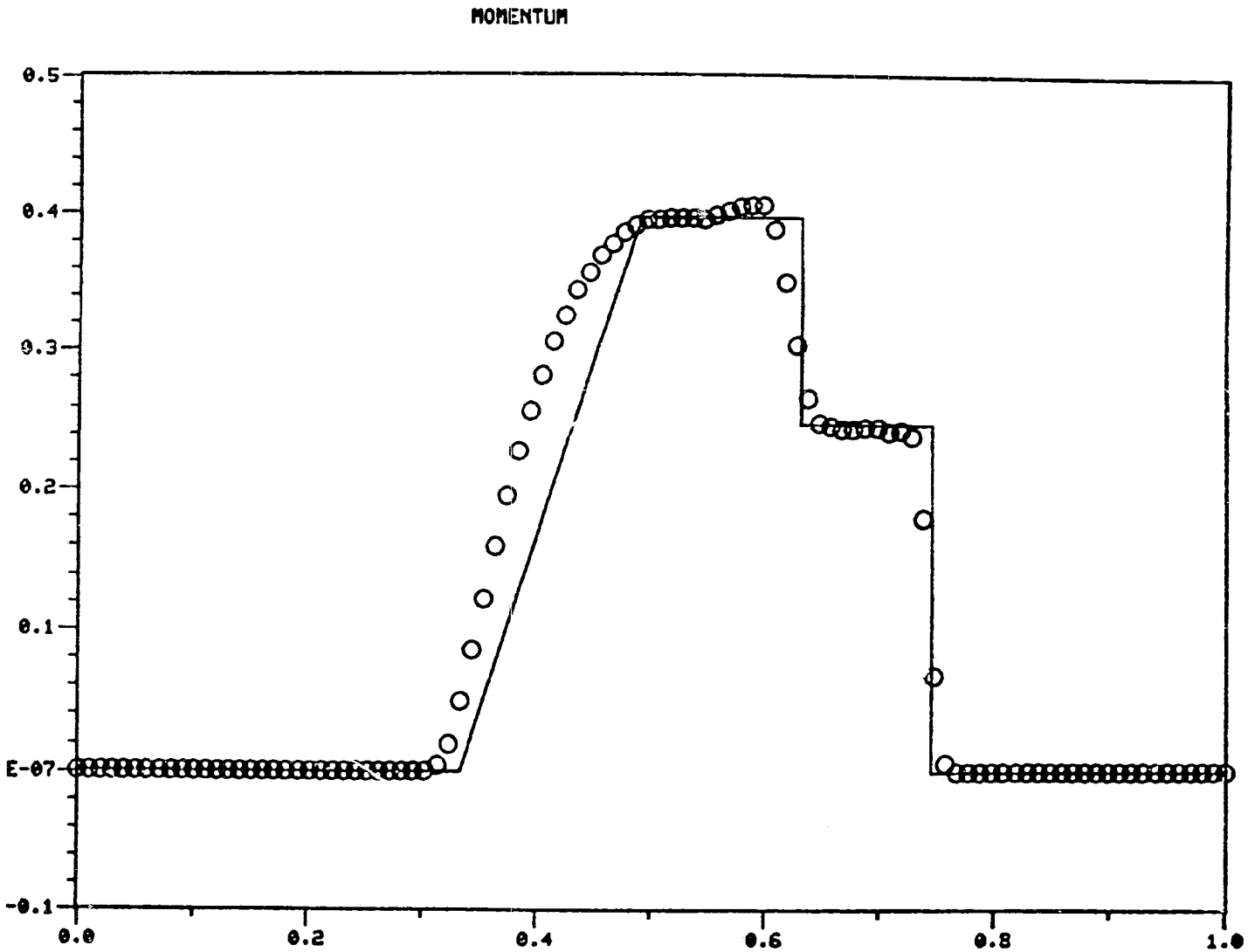


Figure III.6.3b. Sod Test Case (N = 100, t = 0.14, NSTEP = 70)

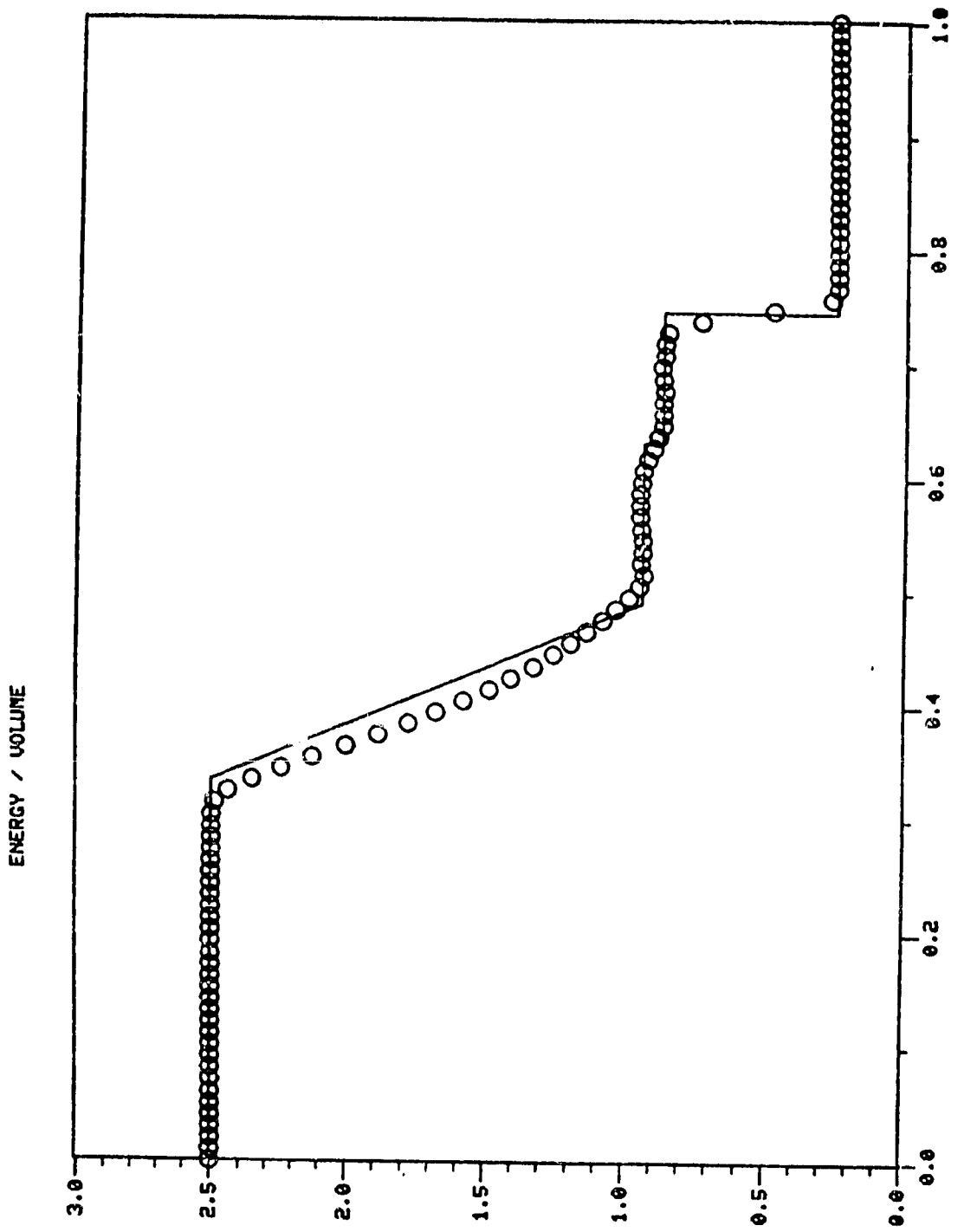


Figure III.6.3c. Sod Test Case (N = 100, t = 0.14, NSTEP = 70)

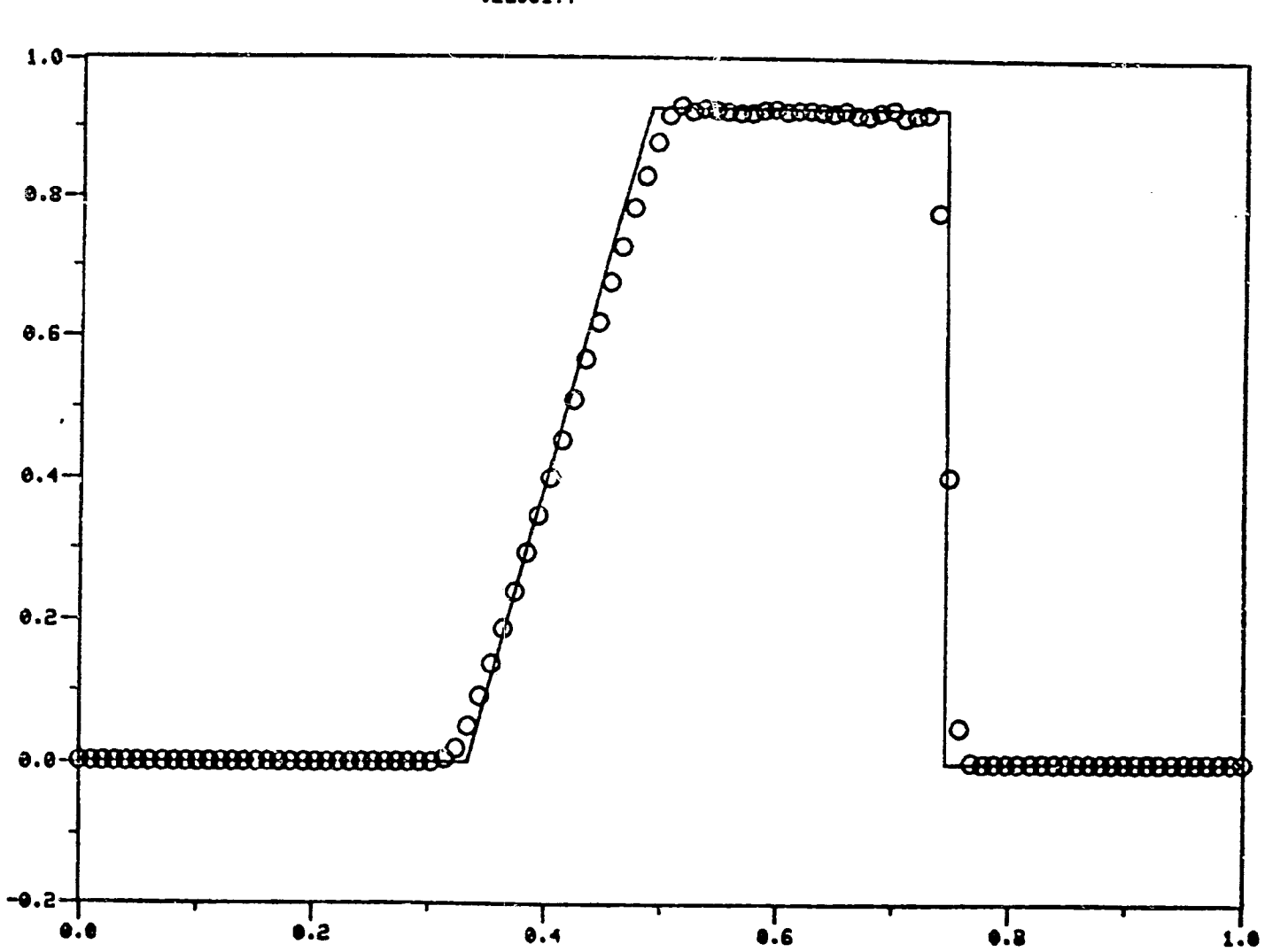


Figure III.6.3d. Sod Test Case (N = 100, t = 0.14, NSTEP = 70)

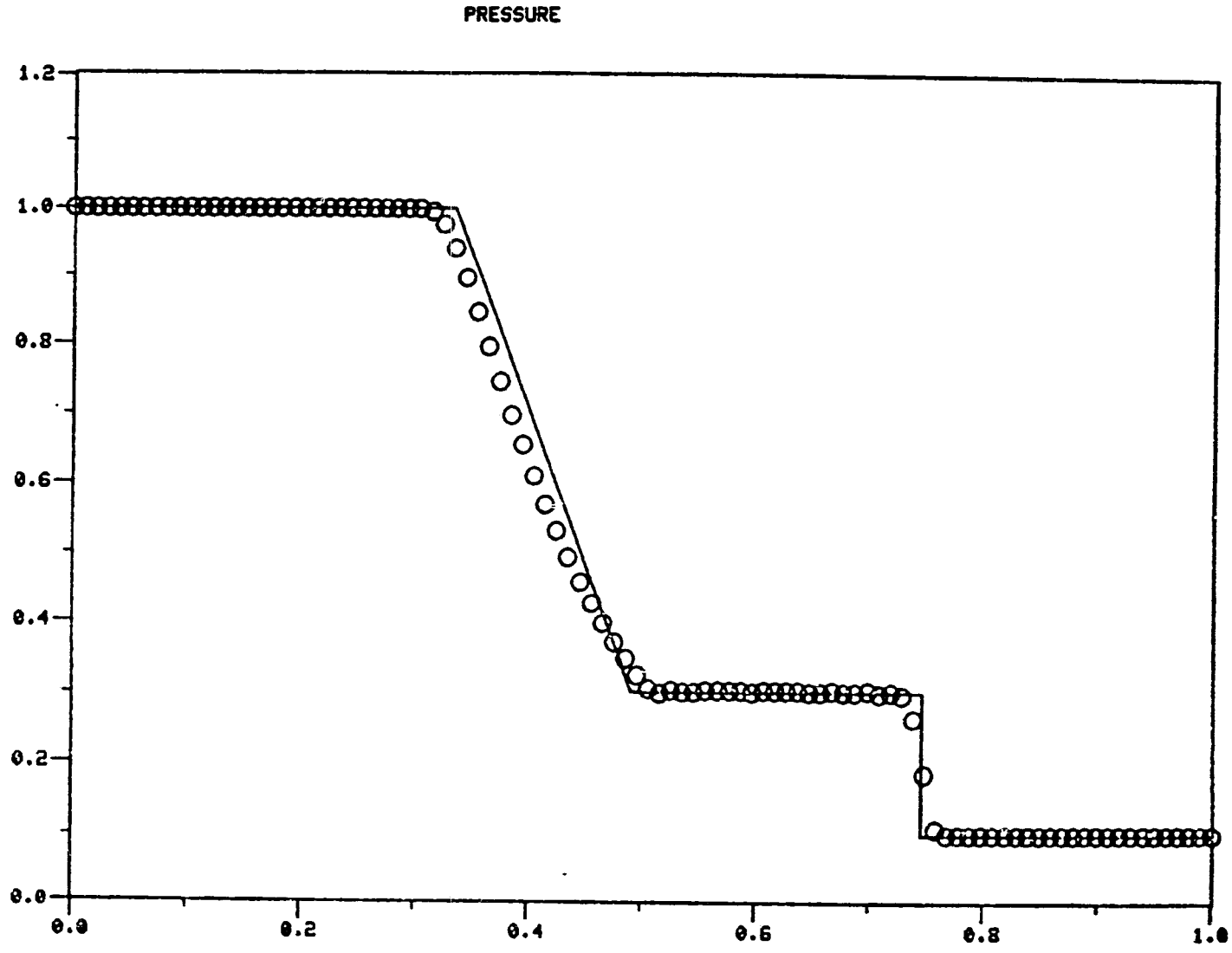


Figure III.6.3e. Sod Test Case (N = 100, t = 0.14, NSTEP = 70)

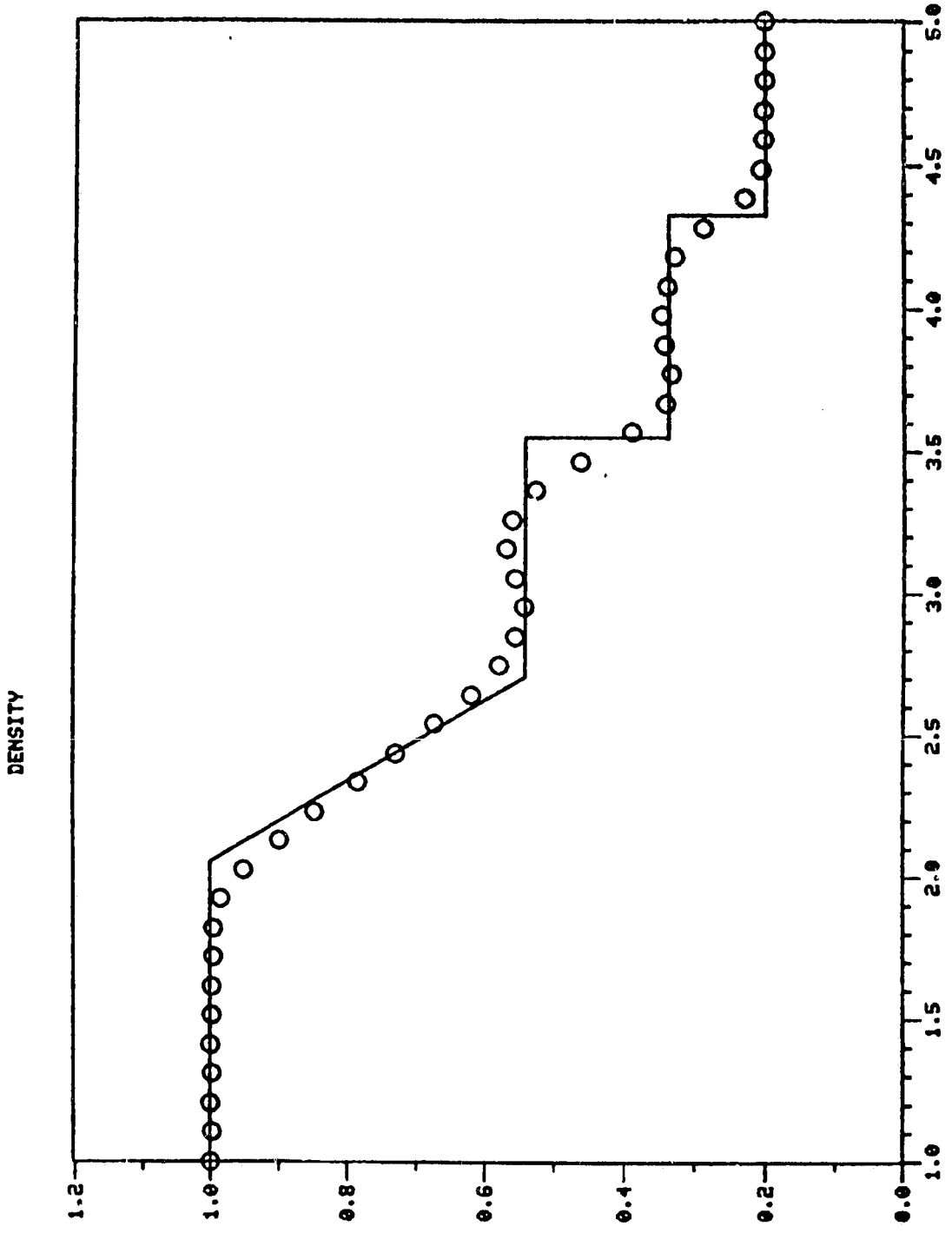


Figure III.6.4a. Steger & Warming Test Case (N=40, t=0.8, NSTEP = 20)

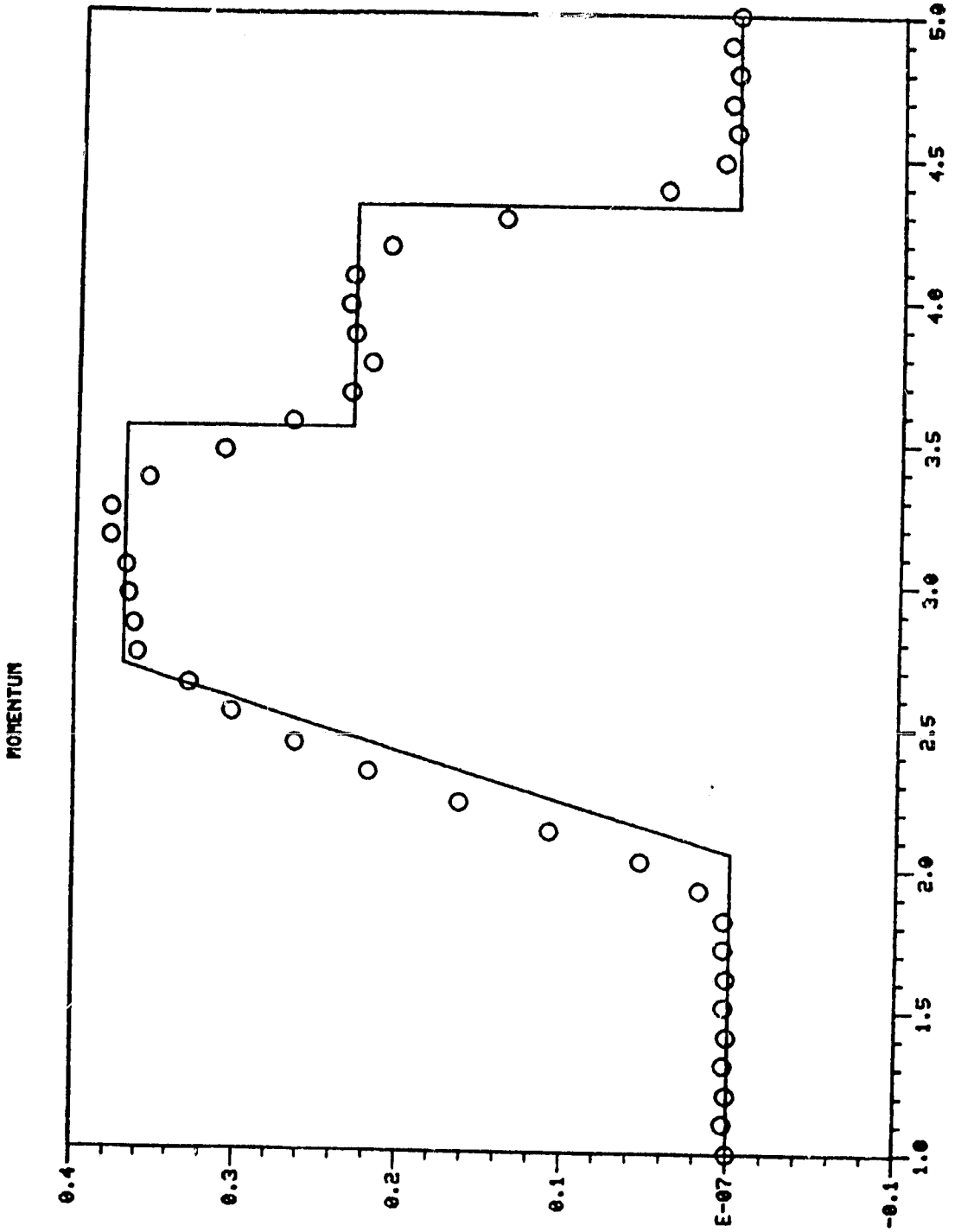


Figure III.6.4b. Steger & Warming Test Case (N = 40, t = 0.8, NSTEP = 20)

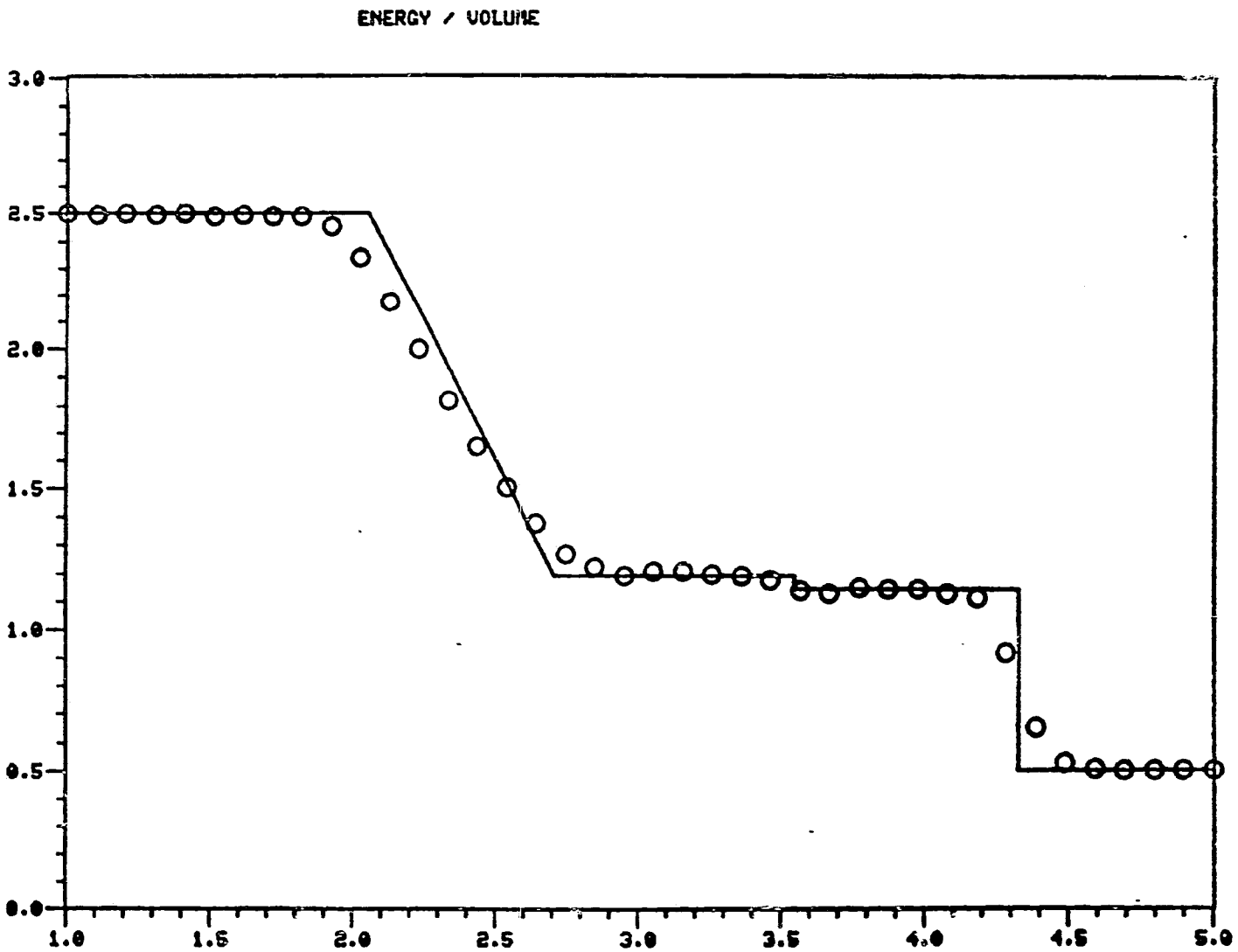


Figure III.6.4c. Steger & Warming Test Case (N = 40, t = 0.8, NSTEP = 20)

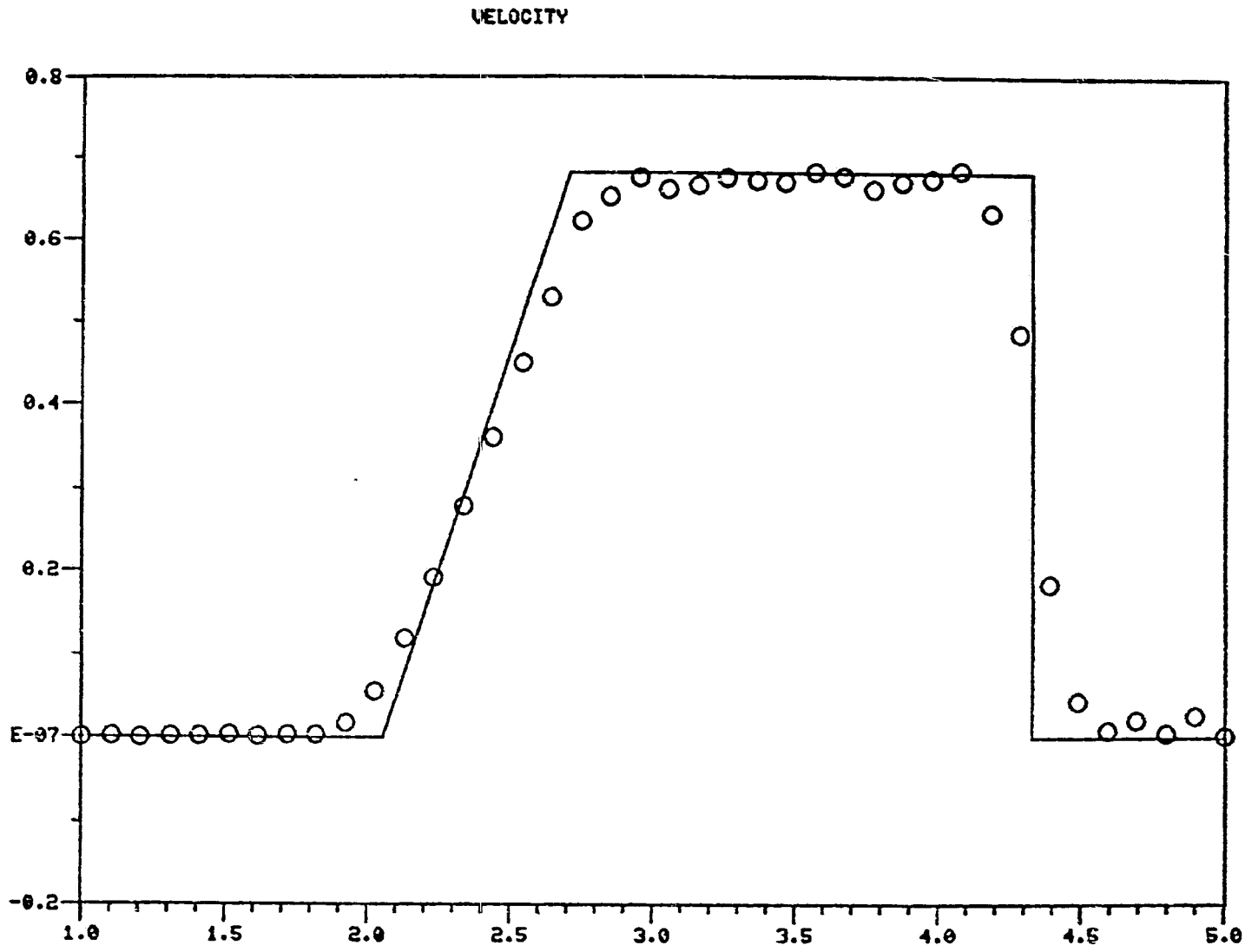


Figure III.6.4d. Steger & Warming Test Case (N = 40, t = 0.8, NSTEP = 20)

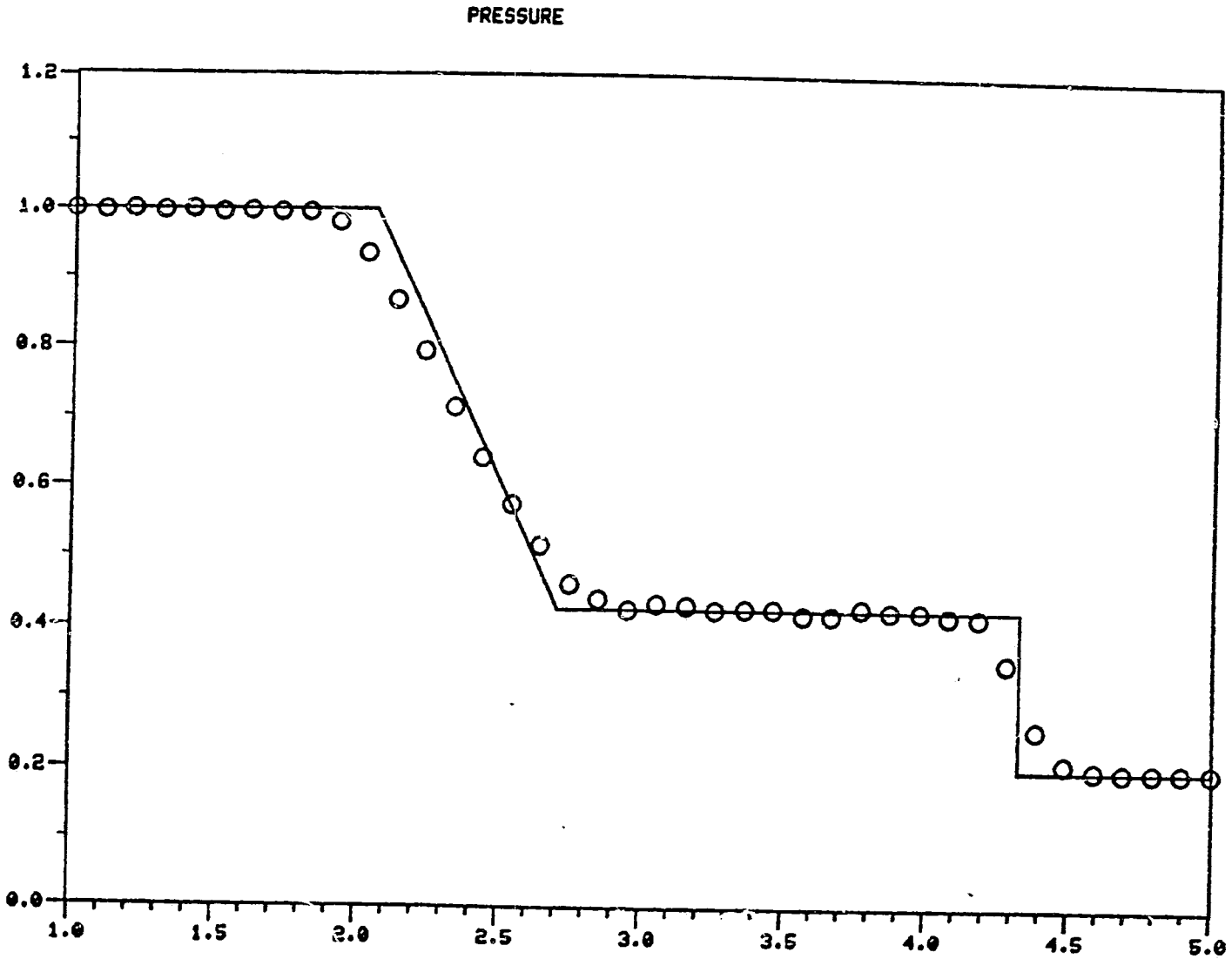


Figure III.6.4e. Steger & Warming Test Case (N = 40, t = 0.8, NSTEP = 20)

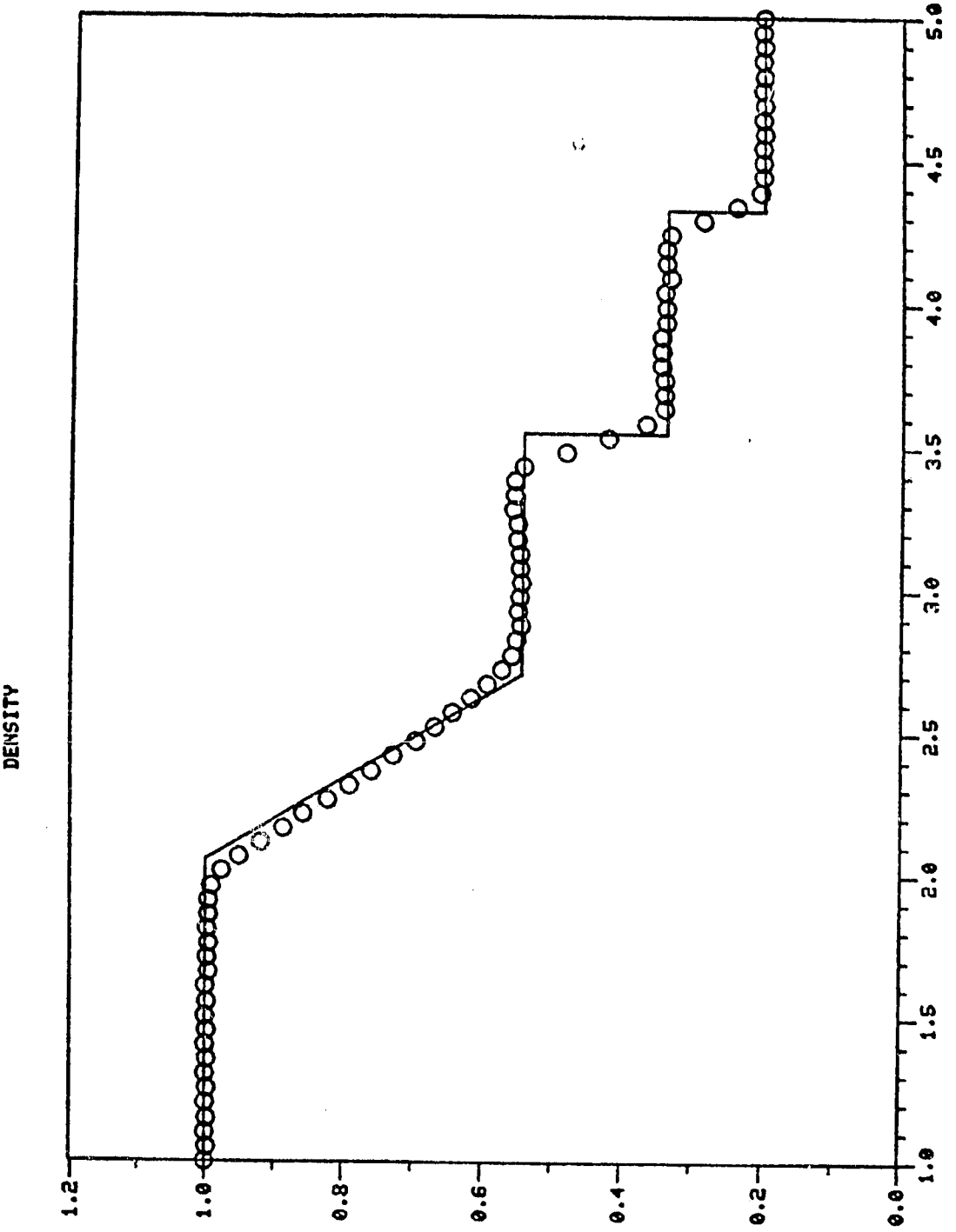


Figure III.6.5a. Steger & Warming Test Case (N = 80, t = 0.8, NSTEP = 40)

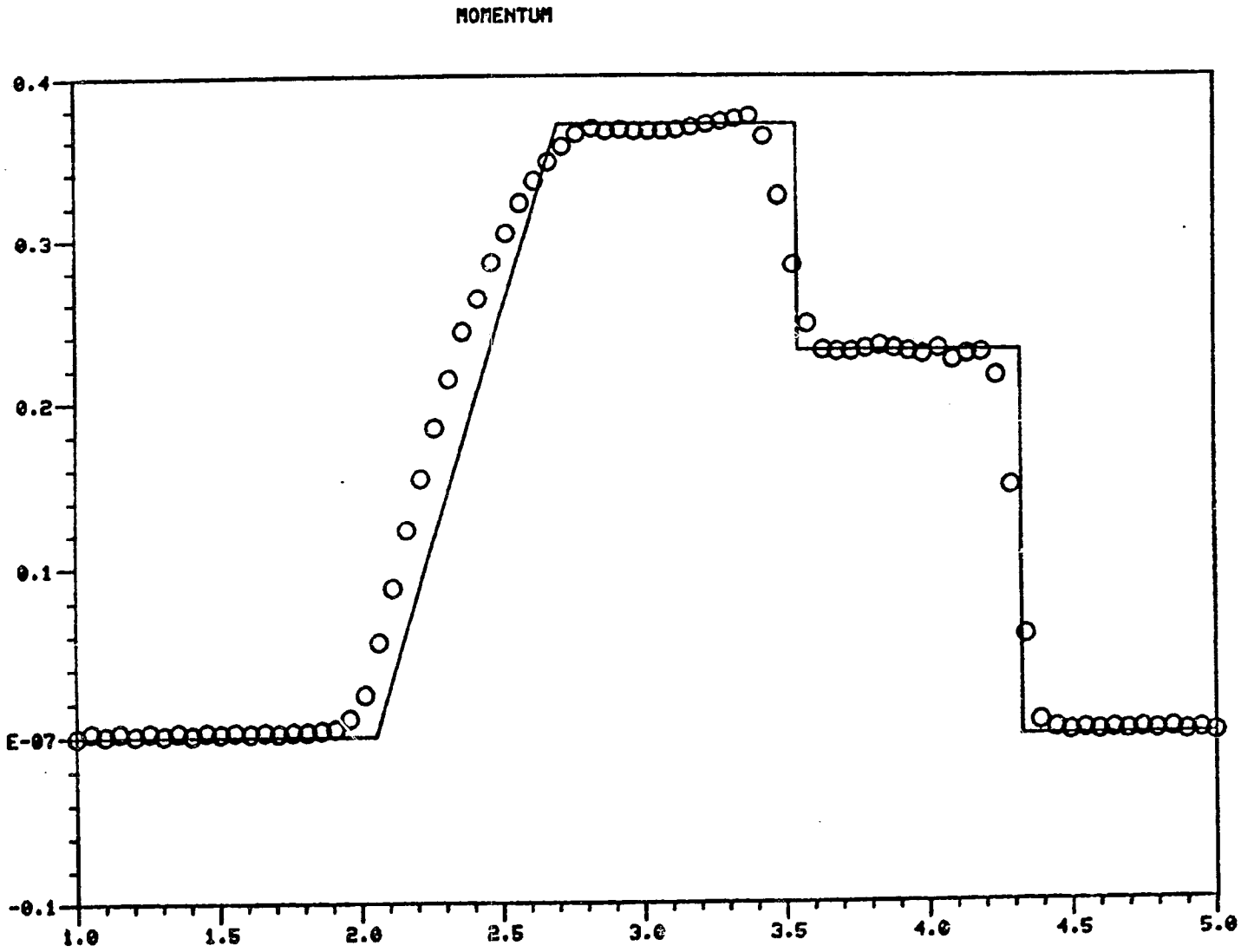


Figure III.6.5b. Steger & Warming Test Case (N = 80, t = 0.8, NSTEP = 40)

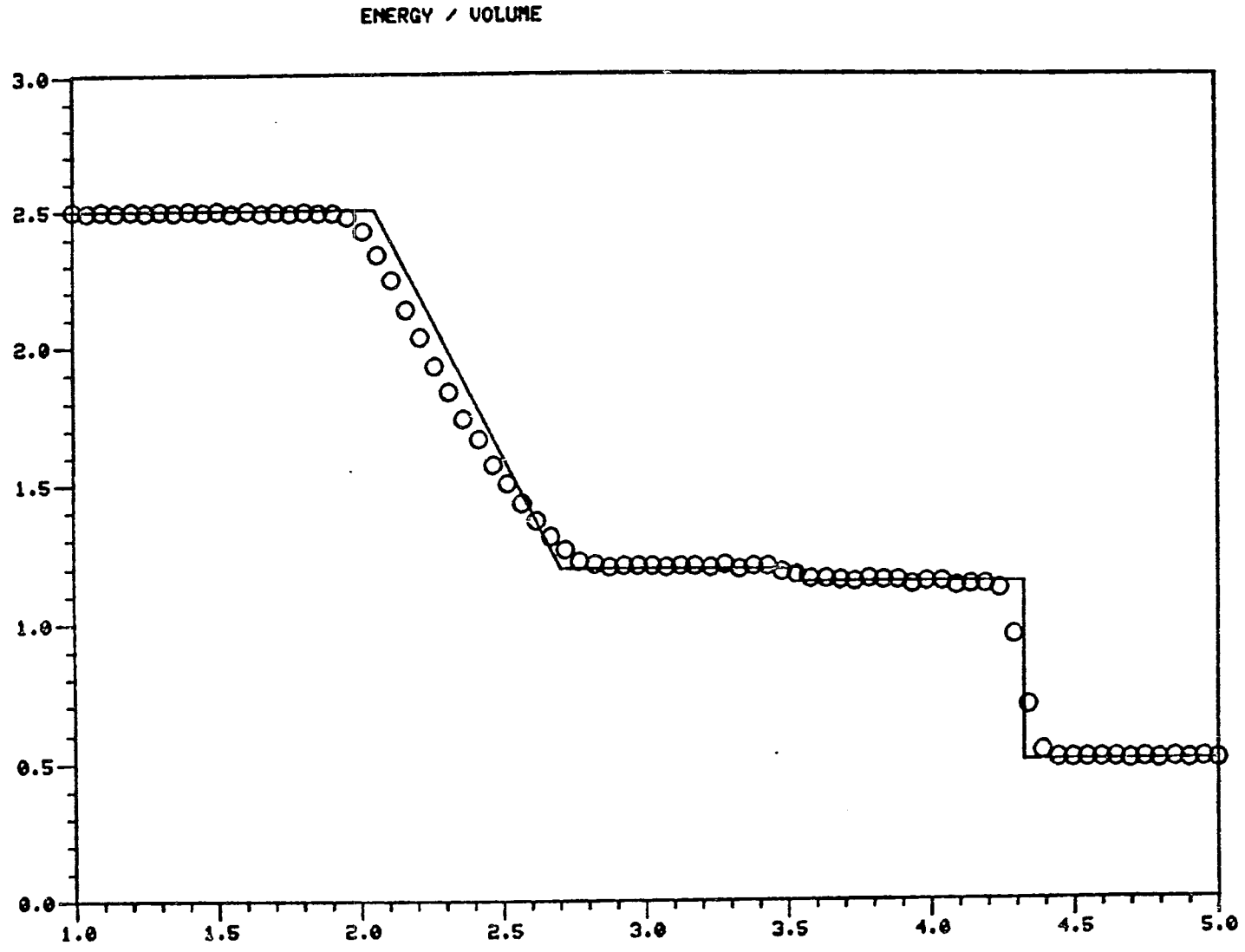


Figure III.6.5c. Steger & Warming Test Case (N = 80, t = 0.8, NSTEP = 40)

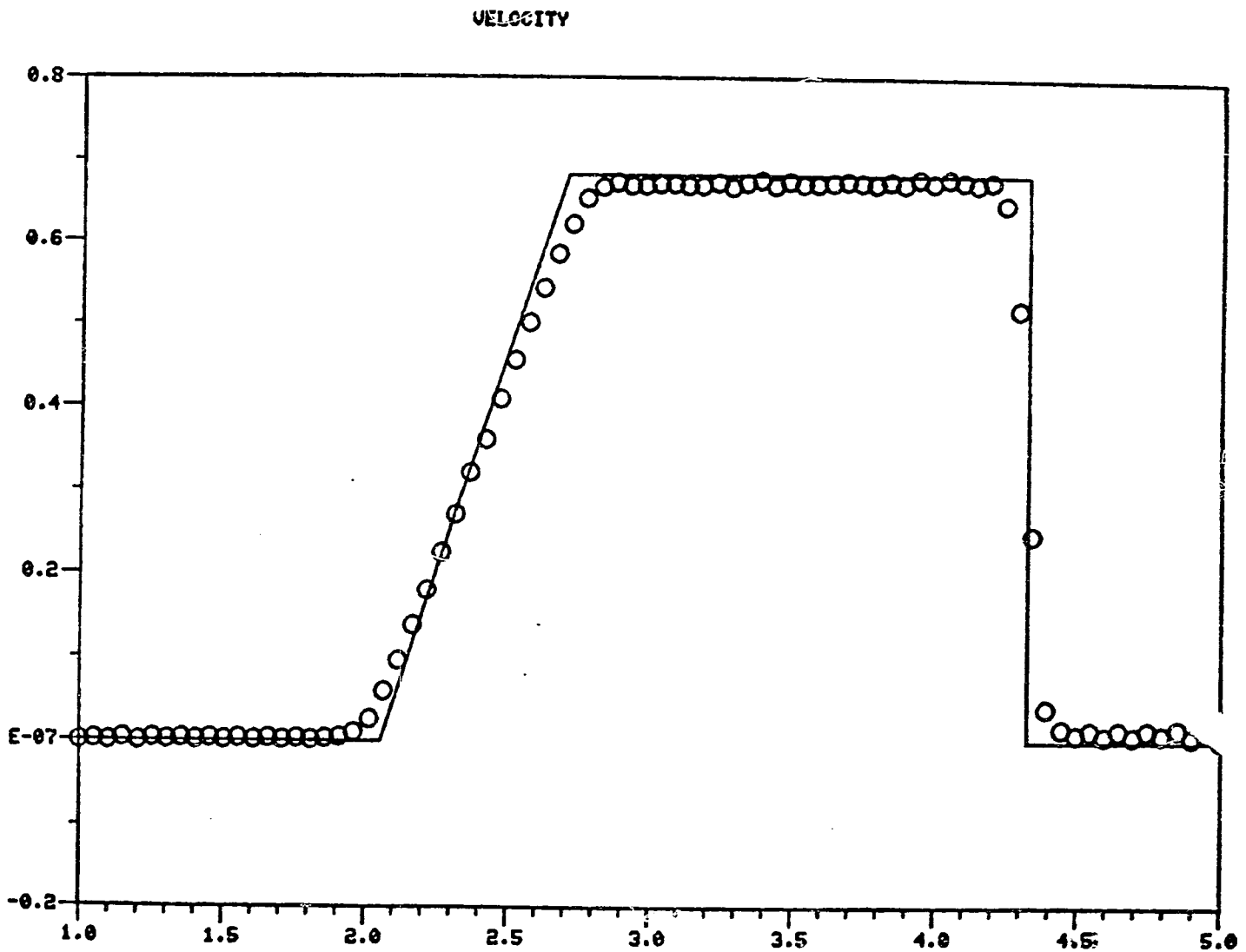


Figure III.6.5a. Steger & Warming Test Case (N = 80, t = 0.8, NSTEP = 40)

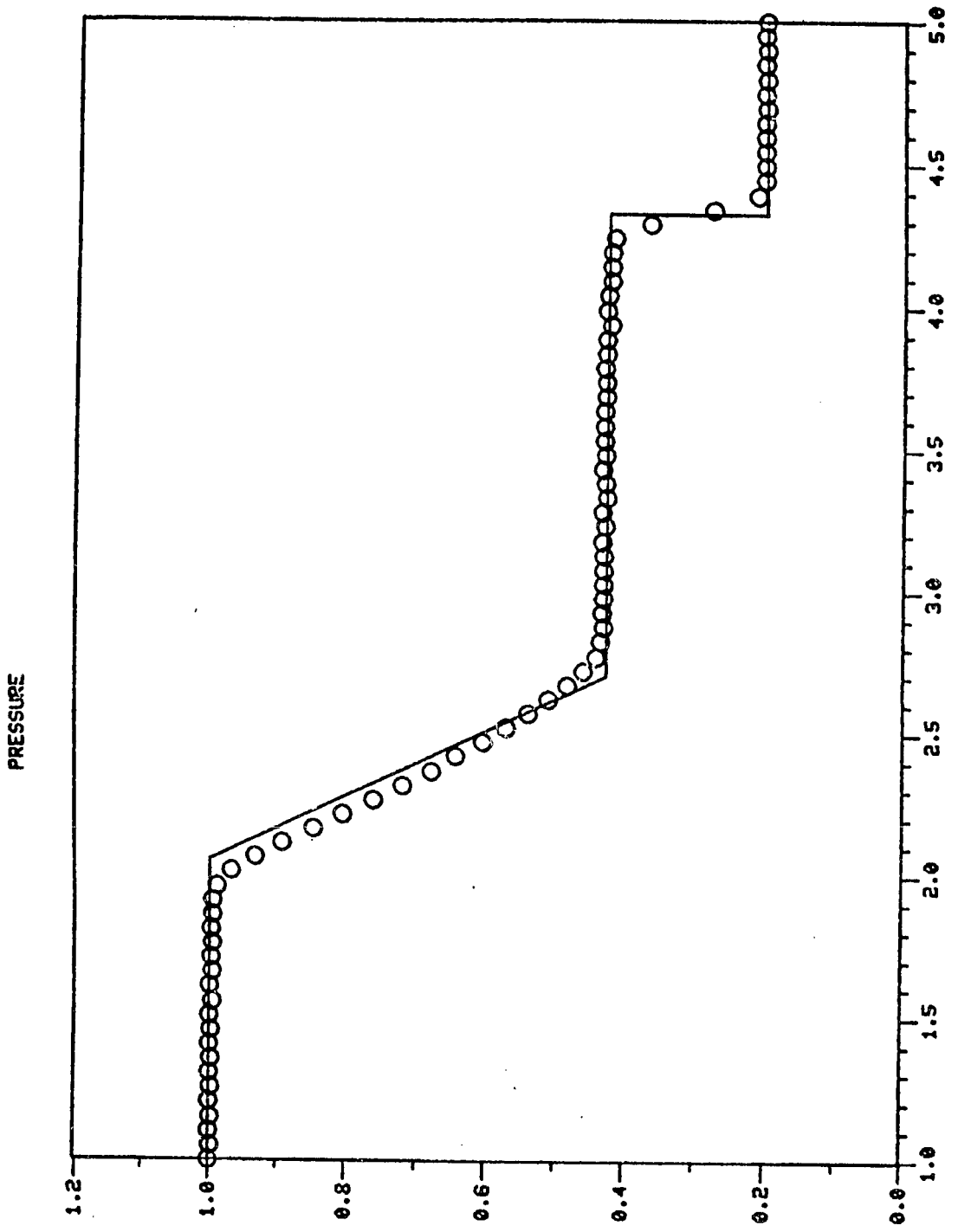


Figure III.6.5e. Steger & Warming Test Case (N = 80, t = 0.8, NSTEP = 40)

III.7. Transonic Channel Problem

This section takes up the generalization of our approach to multidimensional problems. The problem that we consider, transonic flow in a channel, presents the additional complication of requiring the treatment of boundaries [19,22,27,52].

We first discuss the two dimensional Euler equations. This is followed by the extension of Flux Vector Splitting to two dimensions in an arbitrary coordinate system. As promised earlier we take this occasion to supply full details. With this background material available to us, we next describe the problem of transonic flow over a bump in a channel. We then detail our numerical approach, the discussion of the consistent enforcement of boundary conditions occupying a central position. We conclude with the results of some numerical calculations.

2D Euler Equations

The two-dimensional Euler equations may be written as the system of hyperbolic conservation laws

$$\frac{\partial U}{\partial t} + \frac{\partial F}{\partial x} + \frac{\partial G}{\partial y} = 0$$

where

$$U \equiv \begin{bmatrix} \rho \\ m \\ n \\ e \end{bmatrix}, \quad F \equiv \begin{bmatrix} \rho u \\ \rho u^2+p \\ \rho uv \\ u(e+p) \end{bmatrix}, \quad G \equiv \begin{bmatrix} \rho v \\ \rho uv \\ \rho v^2+p \\ v(e+p) \end{bmatrix}$$

and $m = \rho u$, $n = \rho v$.

Here ρ is the density, u and v are the Cartesian velocity components in the x and y directions respectively, and e is the total energy per unit volume. Note that e is related to ε, the internal energy per unit mass, by

$$e = \rho\varepsilon + \frac{\rho}{2}(u^2 + v^2) = \rho\varepsilon + (m^2 + n^2)/2\rho.$$

In general, the equation of state is

$$p = p(\rho, \varepsilon)$$

which for a perfect gas becomes

$$p = (\gamma - 1)\rho\varepsilon$$

$$\Rightarrow \quad p = (\gamma-1)[e - (m^2 + n^2)/2\rho] \Rightarrow e = \frac{p}{\gamma-1} + \frac{m^2+n^2}{2\rho}$$

where γ is the ratio of specific heats.

The above system of equations may also be written in the quasilinear form

$$\frac{\partial U}{\partial t} + A\frac{\partial U}{\partial x} + B\frac{\partial U}{\partial y} = 0$$

where $A \equiv \frac{\partial F}{\partial U}$, $B \equiv \frac{\partial G}{\partial U}$, are the Jacobian matrices. Let $P = k_1 A + k_2 B$; $k_1, k_2 \in \mathbb{R}$. The system is hyperbolic at the point (x,y,t,u) if $\exists\, Q \ni Q^{-1} P\, Q = \Lambda$ (diagonal) with λ_i real $\forall i$ and the norms of Q and Q^{-1} uniformly bounded. We next explicitly calculate the elements of A and B.

$$F = \begin{bmatrix} m \\ \frac{m^2}{\rho} + (\gamma-1)\left[e-(m^2+n^2)/2\rho\right] \\ \frac{mn}{\rho} \\ \frac{m}{\rho}\left(e + (\gamma-1)\left[e-(m^2+n^2)/2\rho\right]\right) \end{bmatrix} \Rightarrow \frac{\partial F}{\partial U}$$

$$= -\begin{bmatrix} 0 & -1 & 0 & 0 \\ \frac{3-\gamma}{2}u^2 + \frac{1-\gamma}{2}v^2 & (\gamma-3)u & (\gamma-1)v & 1-\gamma \\ uv & -v & -u & 0 \\ \frac{\gamma eu}{\rho} + (1-\gamma)u(u^2+v^2) & -\frac{\gamma e}{\rho} + \frac{\gamma-1}{2}(3u^2+v^2) & (\gamma-1)uv & -\gamma u \end{bmatrix} \equiv A.$$

$$G = \begin{bmatrix} n \\ \frac{mn}{\rho} \\ \frac{n^2}{\rho} + (\gamma-1)\left[e-(m^2+n^2)/2\rho\right] \\ \frac{n}{\rho}\left(e+(\gamma-1)\left[e-(m^2+n^2)/2\rho\right]\right) \end{bmatrix} \Rightarrow \frac{\partial G}{\partial U} =$$

$$= -\begin{bmatrix} 0 & 0 & -1 & 0 \\ uv & -v & -u & 0 \\ \frac{3-\gamma}{2}v^2 + \frac{1-\gamma}{2}u^2 & (\gamma-1)u & (\gamma-3)v & 1-\gamma \\ \frac{\gamma ev}{\rho} + (1-\gamma)v(u^2+v^2) & (\gamma-1)uv & -\frac{\gamma e}{\rho} + \frac{\gamma-1}{2}(3v^2+u^2) & -\gamma v \end{bmatrix}$$

$\equiv B.$

Because of the complexity of these Jacobian matrices, the calculation of Λ, Q, Q^{-1} is quite cumbersome. Hence we recast the equations in the nonconservative or primitive variable form

$$\frac{\partial \tilde{U}}{\partial t} + \tilde{A}\,\frac{\partial \tilde{U}}{\partial x} + \tilde{B}\,\frac{\partial \tilde{U}}{\partial y} = 0$$

where

$$\tilde{U} \equiv \begin{bmatrix} \rho \\ u \\ v \\ p \end{bmatrix}, \qquad \tilde{A} \equiv \begin{bmatrix} u & \rho & 0 & 0 \\ 0 & u & 0 & 1/\rho \\ 0 & 0 & u & 0 \\ 0 & \rho c^2 & 0 & u \end{bmatrix},$$

$$\tilde{B} \equiv \begin{bmatrix} v & 0 & \rho & 0 \\ 0 & v & 0 & 0 \\ 0 & 0 & v & 1/\rho \\ 0 & 0 & \rho c^2 & v \end{bmatrix} \quad \text{where } c^2 \equiv \gamma p/\rho = \frac{\gamma(\gamma-1)}{\rho}\left[e - \frac{(m^2+n^2)}{2\rho}\right]$$

is the local speed of sound.

We obtain this result as follows:

$$\frac{\partial U}{\partial t} + A\,\frac{\partial U}{\partial x} + B\,\frac{\partial U}{\partial y} = 0$$

$$\Rightarrow \frac{\partial U}{\partial \tilde{U}} \cdot \frac{\partial \tilde{U}}{\partial U} \cdot \frac{\partial U}{\partial t} + A \cdot \frac{\partial U}{\partial \tilde{U}} \cdot \frac{\partial \tilde{U}}{\partial U} \cdot \frac{\partial U}{\partial x} + B \cdot \frac{\partial U}{\partial \tilde{U}} \cdot \frac{\partial \tilde{U}}{\partial U} \cdot \frac{\partial U}{\partial y} = 0$$

$$\Rightarrow \frac{\partial U}{\partial \tilde{U}} \cdot \frac{\partial \tilde{U}}{\partial t} + A \cdot \frac{\partial U}{\partial \tilde{U}} \cdot \frac{\partial \tilde{U}}{\partial x} + B \cdot \frac{\partial U}{\partial \tilde{U}} \cdot \frac{\partial \tilde{U}}{\partial y} = 0$$

$$\Rightarrow \frac{\partial \tilde{U}}{\partial t} + \left[\frac{\partial \tilde{U}}{\partial U} \cdot A \cdot \frac{\partial U}{\partial \tilde{U}}\right] \cdot \frac{\partial \tilde{U}}{\partial x} + \left[\frac{\partial \tilde{U}}{\partial U} \cdot B \cdot \frac{\partial U}{\partial \tilde{U}}\right] \cdot \frac{\partial \tilde{U}}{\partial y} = 0$$

$$\Rightarrow \frac{\partial \tilde{U}}{\partial t} + \tilde{A}\,\frac{\partial \tilde{U}}{\partial x} + \tilde{B}\,\frac{\partial \tilde{U}}{\partial y} = 0$$

with $\tilde{A} = M^{-1} A M$, $\tilde{B} = M^{-1} B M$ where $M = \frac{\partial U}{\partial \tilde{U}} \Rightarrow M^{-1} = \frac{\partial \tilde{U}}{\partial U}$.

Now, $U = [\rho, m, n, e]^T$ and $\tilde{U} = [\rho, u, v, p]^T \Rightarrow$

$$\frac{\partial U}{\partial \tilde{U}} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ u & \rho & 0 & 0 \\ v & \rho & 0 & 0 \\ \frac{u^2+v^2}{2} & \rho u & \rho v & \frac{1}{\gamma-1} \end{bmatrix}$$

and

$$\frac{\partial \tilde{U}}{\partial U} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ -\frac{m}{\rho^2} & \frac{1}{\rho} & 0 & 0 \\ -\frac{n}{\rho^2} & 0 & \frac{1}{\rho} & 0 \\ \frac{(\gamma-1)}{2} \cdot \frac{m^2+n^2}{\rho^2} & (1-\gamma)\frac{m}{\rho} & (1-\gamma)\frac{n}{\rho} & \gamma-1 \end{bmatrix}$$

so that

$$M = \begin{bmatrix} 1 & 0 & 0 & 0 \\ u & \rho & 0 & 0 \\ v & \rho & 0 & 0 \\ (u^2+v^2)/2 & \rho u & \rho v & 1/(\gamma-1) \end{bmatrix}$$

and

$$M^{-1} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ -u/\rho & 1/\rho & 0 & 0 \\ -v/\rho & 0 & 1/\rho & 0 \\ (\gamma-1)(u^2+v^2)/2 & (1-\gamma)u & (1-\gamma)v & (\gamma-1) \end{bmatrix}.$$

We thus have

$$\tilde{A} = M^{-1} A M , \qquad \tilde{B} = M^{-1} B M$$

so that if we have

$$\tilde{P} \equiv k_1 \tilde{A} + k_2 \tilde{B}$$

and

$$\tilde{Q}^{-1} \tilde{P} \tilde{Q} = \tilde{\Lambda}$$

then

$$\tilde{P} = k_1 M^{-1} A M + k_2 M^{-1} B M = M^{-1} (k_1 A + k_2 B) M = M^{-1} P M .$$

This first of all implies that $\tilde{\Lambda} = \Lambda$ since $\tilde{P}$ and P are now seen to be similar. In addition, we now have

$$\tilde{Q}^{-1} M^{-1} P M \tilde{Q} = \Lambda \Rightarrow$$

$$Q = M \tilde{Q} \quad \text{and} \quad Q^{-1} = \tilde{Q}^{-1} M^{-1} .$$

It remains to diagonalize $\tilde{P} = k_1 \tilde{A} + k_2 \tilde{B}$.

$$\tilde{P} = \begin{bmatrix} k_1 u + k_2 v & k_1 \rho & k_2 \rho & 0 \\ 0 & k_1 u + k_2 v & 0 & k_1/\rho \\ 0 & 0 & k_1 u + k_2 v & k_2/\rho \\ 0 & k_1 \rho c^2 & k_2 \rho c^2 & k_1 u + k_2 v \end{bmatrix}$$

so that

$$\det (\tilde{P} - \lambda I) = 0 \Rightarrow$$

$$\lambda_1 = \lambda_2 = k_1 u + k_2 v; \ \lambda_3 = \lambda_1 + c (k_1^2 + k_2^2)^{1/2}; \ \lambda_4 = \lambda_1 - c (k_1^2 + k_2^2)^{1/2} .$$

The corresponding eigenvectors are obtained as follows.

(i) $\lambda = k_1 u + k_2 v \Rightarrow$

$$\begin{bmatrix} 0 & k_1\rho & k_2\rho & 0 \\ 0 & 0 & 0 & k_1/\rho \\ 0 & 0 & 0 & k_2/\rho \\ 0 & k_1\rho c^2 & k_2\rho c^2 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \vec{0} \Rightarrow$$

$$x_4 = 0 , \quad x_3 = -\frac{k_1}{k_2} x_2 .$$

$\therefore$ let

$$\tilde{Q}_1 = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \qquad \tilde{Q}_2 = \begin{bmatrix} 0 \\ \tilde{k}_2 \\ -\tilde{k}_1 \\ 0 \end{bmatrix}$$

where $\tilde{k}_1 = k_1/(k_1^2+k_2^2)^{1/2}$, $\tilde{k}_2 = k_2/(k_1^2+k_2^2)^{1/2}$.

(ii) $\lambda = k_1 u + k_2 v + c(k_1^2+k_2^2)^{1/2} \Rightarrow$

$$\begin{bmatrix} -c(k_1^2+k_2^2)^{1/2} & k_1\rho & k_2\rho & 0 \\ 0 & -c(k_1^2+k_2^2)^{1/2} & 0 & k_1/\rho \\ 0 & 0 & -c(k_1^2+k_2^2)^{1/2} & k_2/\rho \\ 0 & k_1\rho c^2 & k_2\rho c^2 & -c(k_1^2+k_2^2)^{1/2} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix}$$

$= \vec{0} \Rightarrow x_2 = \frac{\tilde{k}_1}{\rho c} x_4 .$

$\therefore$ let

$$\tilde{Q}_3 = \begin{bmatrix} \rho/(\sqrt{2}\ c) \\ \tilde{k}_1/\sqrt{2} \\ \tilde{k}_2/\sqrt{2} \\ \rho c/\sqrt{2} \end{bmatrix}$$

(iii) $\lambda = k_1 u + k_2 v - c(k_1^2+k_2^2)^{1/2} \Rightarrow$

$$\begin{bmatrix} c(k_1^2+k_2^2)^{1/2} & k_1\rho & k_2\rho & 0 \\ 0 & c(k_1^2+k_2^2)^{1/2} & 0 & k_1/\rho \\ 0 & 0 & c(k_1^2+k_2^2)^{1/2} & k_2/\rho \\ 0 & k_1\rho c^2 & k_2\rho c^2 & c(k_1^2+k_2^2)^{1/2} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix}$$

$$= \vec{0} \Rightarrow \quad x_2 = -\frac{\tilde{k}_1}{\rho c} x_4 \ .$$

$\therefore$ let

$$\tilde{Q}_4 = \begin{bmatrix} \rho/(\sqrt{2}\ c) \\ -\tilde{k}_1/\sqrt{2} \\ -\tilde{k}_2/\sqrt{2} \\ \rho c/\sqrt{2} \end{bmatrix}.$$

Hence

$$\tilde{Q} = \begin{bmatrix} 1 & 0 & \rho/(\sqrt{2}\ c) & \rho/(\sqrt{2}\ c) \\ 0 & \tilde{k}_2 & \tilde{k}_1/\sqrt{2} & -\tilde{k}_1/\sqrt{2} \\ 0 & -\tilde{k}_1 & \tilde{k}_2/\sqrt{2} & -\tilde{k}_2/\sqrt{2} \\ 0 & 0 & \rho c/\sqrt{2} & \rho c/\sqrt{2} \end{bmatrix}$$

and

$$\tilde{Q}^{-1} = \begin{bmatrix} 1 & 0 & 0 & -1/c^2 \\ 0 & \tilde{k}_2 & -\tilde{k}_1 & 0 \\ 0 & \tilde{k}_1/\sqrt{2} & \tilde{k}_2/\sqrt{2} & 1/(\sqrt{2}\ \rho c) \\ 0 & -\tilde{k}_1/\sqrt{2} & -\tilde{k}_2/\sqrt{2} & 1/(\sqrt{2}\ \rho c) \end{bmatrix}.$$

Note that

$$\tilde{Q}^{-1}\tilde{A}\,\tilde{Q} = \begin{bmatrix} u & 0 & 0 & 0 \\ 0 & u & c\tilde{k}_2/\sqrt{2} & c\tilde{k}_2/\sqrt{2} \\ 0 & c\tilde{k}_2/\sqrt{2} & u+c\tilde{k}_1 & 0 \\ 0 & c\tilde{k}_2/\sqrt{2} & 0 & u-c\tilde{k}_1 \end{bmatrix}$$

and

$$\tilde{Q}^{-1}\tilde{B}\,\tilde{Q} = \begin{bmatrix} v & 0 & 0 & 0 \\ 0 & v & -c\tilde{k}_1/\sqrt{2} & -c\tilde{k}_1/\sqrt{2} \\ 0 & -c\tilde{k}_1/\sqrt{2} & v+c\tilde{k}_2 & 0 \\ 0 & -c\tilde{k}_1/\sqrt{2} & 0 & v-c\tilde{k}_2 \end{bmatrix}$$

so that this particular choice of $\tilde{Q}$ not only diagonalizes $\tilde{P}$ but also simultaneously symmetrizes $\tilde{A}$ and $\tilde{B}$.

2D Flux Vector Splitting

$$\frac{\partial U}{\partial t} + \frac{\partial F}{\partial x} + \frac{\partial G}{\partial y} = 0$$

$$\frac{\partial U}{\partial t} + A \cdot \frac{\partial U}{\partial x} + B \cdot \frac{\partial G}{\partial y} = 0 \; ; \quad A \equiv \frac{\partial F}{\partial U} \; , \quad B \equiv \frac{\partial G}{\partial U} \; .$$

The crucial observation is to note that both F(U) and G(U) are homogeneous functions of degree one. I.e.

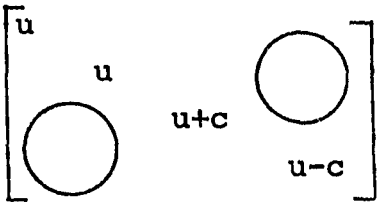
$$F(\alpha U) = \alpha F(U) \; ; \qquad G(\alpha U) = \alpha G(U) \; .$$

Euler's theorem on homogeneous functions then allows us to conclude that

$$F = AU \; , \qquad G = BU \; .$$

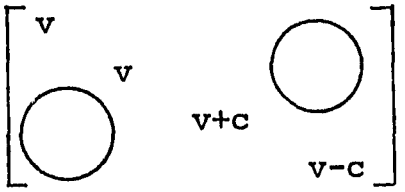
We have noted before that ($Q = Q(k_1, k_2)$ etc.)

$$Q^{-1}(1,0)\ A\ Q(1,0) = \Lambda(1,0) \equiv$$



and

$$Q^{-1}(0,1)\ B\ Q(0,1) = \Lambda(0,1) \equiv$$



$$A = Q(1,0)\ \Lambda(1,0)\ Q^{-1}(1,0)$$

and

$$B = Q(0,1)\ \Lambda(0,1)\ Q^{-1}(0,1)$$

hence

$$F = Q(1,0)\ \Lambda(1,0)\ Q^{-1}(1,0)\ U$$

and

$$G = Q(0,1)\ \Lambda(0,1)\ Q^{-1}(0,1)\ U\ .$$

Now decompose

$$\Lambda(1,0) = \Lambda^{+}(1,0) + \Lambda^{-}(1,0)$$
$$\Lambda(0,1) = \Lambda^{+}(0,1) + \Lambda^{-}(0,1)$$

where Λ^{+} and Λ^{-} have positive and negative eigenvalues, respectively. This now leads to the following splitting of the flux vectors F and G.

$$\begin{cases} F = Q(1,0)\,[\Lambda^{+}(1,0) + \Lambda^{-}(1,0)]\ Q^{-1}(1,0)\ U \\ G = Q(0,1)\,[\Lambda^{+}(0,1) + \Lambda^{-}(0,1)]\ Q^{-1}(0,1)\ U \end{cases}$$

$$\begin{cases} F = [Q(1,0)\ \Lambda^{+}(1,0)\ Q^{-1}(1,0) + Q(1,0)\ \Lambda^{-}(1,0)\ Q^{-1}(1,0)]\ U \\ \qquad\qquad \equiv (A^{+} + A^{-})\ U \\ G = [Q(0,1)\ \Lambda^{+}(0,1)\ Q^{-1}(0,1) + Q(0,1)\ \Lambda^{-}(0,1)\ Q^{-1}(0,1)]\ U \\ \qquad\qquad \equiv (B^{+} + B^{-})\ U \end{cases}$$

$$\begin{cases} F = A^+U + A^-U \equiv F^+ + F^- \\ G = B^+U + B^-U \equiv G^+ + G^-. \end{cases}$$

Note that, in general,

$$A^+ \neq \frac{\partial F^+}{\partial U}\ , \quad A^- \neq \frac{\partial F^-}{\partial U}\ , \quad B^+ \neq \frac{\partial G^+}{\partial U}\ , \quad B^- = \frac{\partial G^-}{\partial U}\ ,$$

as has been observed by Steger and Warming [50]. However, we do have the following result which has eluded the attention of earlier authors.

Theorem. If the Λ splitting is chosen so as to produce a homogeneous F^+ then $A^+ = \frac{\partial F^+}{\partial U}$.

Proof: We have $F^+ \equiv A^+(U)U$. Define $A_+(U) \equiv \frac{\partial F^+}{\partial U}$. Since F^+ is homogeneous we have $F^+ = A_+(U)U$ and hence $A^+(U)U = A_+(U)U \ \forall\ U \in \mathbb{R}^3$. Thus $\Sigma(U)\cdot U = 0 \ \forall\ U$ where $\Sigma \equiv A^+ - A_+$. Let $P^{-1}\Sigma P = J$ be the Jordan canonical form and hence $\Sigma = P J P^{-1}$. This results in $J\bar{U} = 0 \ \forall\ \bar{U} \in \mathbb{R}^3$ where $\bar{U} \equiv P^{-1}U$. Therefore, all the eigenvalues of Σ are zero and J has no ones along its super-diagonal. Hence $\Sigma = 0$, i.e. $A^+ = A_+$. Q.E.D.

Note that similar results obtain for A^-, B^+, and B^-. With these considerations aside, we solve the problem

$$U_t + F^+_x + F^-_x + G^+_y + G^-_y = 0$$

by upwinding F^+ and G^+ while downwinding F^- and G^-.

As we have seen above

$$\begin{aligned} F^+ &= Q(1,0)\ \Lambda^+(1,0)\ Q^{-1}(1,0)\ U \\ F^- &= Q(1,0)\ \Lambda^-(1,0)\ Q^{-1}(1,0)\ U \\ G^+ &= Q(0,1)\ \Lambda^+(0,1)\ Q^{-1}(0,1)\ U \\ G^- &= Q(0,1)\ \Lambda^-(0,1)\ Q^{-1}(0,1)\ U\ . \end{aligned}$$

In order to facilitate the computation of these flux vector splittings we present the general form for $\Phi \equiv Q(k_1,k_2)\hat{\Lambda}(k_1,k_2)Q^{-1}(k_1,k_2)U$. Recalling that $Q = M\tilde{Q}$ and $Q^{-1} = \tilde{Q}^{-1}M^{-1}$ and upon performing the required multiplications we arrive at

$$\Phi = \frac{\rho}{2\gamma} \cdot \begin{bmatrix} 2(\gamma-1)\hat{\lambda}_1+\hat{\lambda}_3+\hat{\lambda}_4 \\ 2(\gamma-1)\hat{\lambda}_1 u+\hat{\lambda}_3(u+c\tilde{k}_1)+\hat{\lambda}_4(u-c\tilde{k}_1) \\ 2(\gamma-1)\hat{\lambda}_1 v+\hat{\lambda}_3(v+c\tilde{k}_2)+\hat{\lambda}_4(v-c\tilde{k}_2) \\ (\gamma-1)\hat{\lambda}_1(u^2+v^2)+\frac{\hat{\lambda}_3}{2}[(u+c\tilde{k}_1)^2+(v+c\tilde{k}_2)^2] \\ +\frac{\hat{\lambda}_4}{2}[(u-c\tilde{k}_1)^2+(v-c\tilde{k}_2)^2]+\hat{w} \end{bmatrix}$$

where

$$\hat{w} \equiv \frac{(3-\gamma)(\hat{\lambda}_3+\hat{\lambda}_4)c^2}{2(\gamma-1)}$$

and, once again, $\tilde{k}_1 = k_1/(k_1^2+k_2^2)^{1/2}$, $\tilde{k}_2 = k_2/(k_1^2+k_2^2)^{1/2}$. We use the homogeneous splitting

$$\lambda_1^+(1,0)=\lambda_2^+(1,0)=\frac{u+|u|}{2}\ ,\ \lambda_1^-(1,0)=\lambda_2^-(1,0)=\frac{u-|u|}{2}$$
$$\lambda_1^+(0,1)=\lambda_2^+(0,1)=\frac{v+|v|}{2}\ ,\ \lambda_1^-(0,1)=\lambda_2^-(0,1)=\frac{v-|v|}{2}$$
$$\lambda_3^+(1,0)=\lambda_1^+(1,0)+c\ ,\ \lambda_3^-(1,0)=\lambda_1^-(1,0)$$
$$\lambda_3^+(0,1)=\lambda_1^+(0,1)+c\ ,\ \lambda_3^-(0,1)=\lambda_1^-(0,1)$$
$$\lambda_4^+(1,0)=\lambda_1^+(1,0)\ ,\ \lambda_4^-(1,0)=\lambda_1^-(1,0)-c$$
$$\lambda_4^+(0,1)=\lambda_1^+(0,1)\ ,\ \lambda_4^-(0,1)=\lambda_1^-(0,1)-c$$

These results can be generalized to an arbitrary coordinate system as follows. Consider the transformation of coordinates $(x,y) \to (X,Y)$ and its effect on

$$\frac{\partial U}{\partial t} + \frac{\partial F}{\partial x} + \frac{\partial G}{\partial y} = 0 \ .$$

$$\Rightarrow \quad (JU)_t + (y_Y F - x_Y G)_X + (x_X G - y_X F)_Y = 0$$

(where $J \equiv x_X y_Y - x_Y y_X$) which by the homogeneity of F and G

$$\Rightarrow (JU)_t + \left(\frac{y_Y}{J} F(JU) - \frac{x_Y}{J} G(JU)\right)_X + \left(\frac{x_X}{J} G(JU) - \frac{y_X}{J} F(JU)\right)_Y = 0.$$

Letting $\hat{U} \equiv JU$ we arrive at

$$\frac{\partial \hat{U}}{\partial t} + \frac{\partial}{\partial X}\left[\frac{y_Y}{J} \cdot F(\hat{U}) - \frac{x_Y}{J} \cdot G(\hat{U})\right] + \frac{\partial}{\partial Y}\left[\frac{x_X}{J} \cdot G(\hat{U}) - \frac{y_X}{J} \cdot F(\hat{U})\right]_Y = 0$$

while letting

$$\hat{F}(\cdot) \equiv \frac{y_Y}{J} \cdot F(\cdot) - \frac{x_Y}{J} \cdot G(\cdot)$$

$$\hat{G}(\cdot) \equiv -\frac{y_X}{J} \cdot F(\cdot) + \frac{x_X}{J} \cdot G(\cdot)$$

produces

$$\frac{\partial \hat{U}}{\partial t} + \frac{\partial}{\partial X} \hat{F}(\hat{U}) + \frac{\partial}{\partial Y} \hat{G}(\hat{U}) \ .$$

Hence, conservation form is preserved under such a transformation as is homogeneity of the flux vectors. We are thus led to splitting $\hat{F}(\hat{U})$ and $\hat{G}(\hat{U})$. The following observation simplifies matters considerably.

In order to split, e.g.,

$$\hat{F}(\hat{U}) = \frac{y_Y}{J} \cdot F(\hat{U}) - \frac{x_Y}{J} \cdot G(\hat{U}) = y_Y \cdot F(U) - x_Y \cdot G(U) = \bar{F}(U)$$

we simply split

$$\bar{F}(U) = y_Y \cdot F(U) - x_Y \cdot G(U) = k_1 F + k_2 G$$

with $k_1 = y_Y$ and $k_2 = -x_Y$. This is conveniently done using

$$\bar{F}^+ = Q(k_1,k_2)\ \Lambda^+(k_1,k_2)\ Q^{-1}(k_1,k_2)\ U$$

$$\bar{F}^- = Q(k_1,k_2)\ \Lambda^-(k_1,k_2)\ Q^{-1}(k_1,k_2)\ U$$

which are given in general form by Φ. Note that

$$\Lambda = \begin{bmatrix} k_1u+k_2v & & & \\ & k_1u+k_2v & & \\ & & k_1u+k_2v+c(k_1^2+k_2^2)^{1/2} & \\ & & & k_1u+k_2v-c(k_1^2+k_2^2)^{1/2} \end{bmatrix}.$$

We use the splitting

$$\lambda_1^+ = \lambda_2^+ = (k_1u+k_2v+|k_1u+k_2v|)/2,\ \lambda_3^+ = \lambda_1^+ + c,\ \lambda_4^+ = \lambda_1^+,$$

$$\lambda_1^- = \lambda_2^- = (k_1u+k_2v-|k_1u+k_2v|)/2,\ \lambda_3^- = \lambda_1^-,\ \lambda_4^- = \lambda_1^- - c.$$

The same split is used for $\hat{G}(\hat{U}) = \bar{G}(U)$ with $k_1 = -y_X$ and $k_2 = x_X$.

Flow Over a Bump

Consider a channel whose upper wall is straight and whose lower wall is likewise straight and parallel with the exception of a protrusion or "bump". We consider the calculation of inviscid rotational flow in this geometry. In our calculations this bump will be a B-spline, $S(x)$.

We assume that the flow is uniform and subsonic at upstream infinity. With a normalization this becomes $\rho_\infty = 1$, $u_\infty = 1$, $v_\infty = 0$. At downstream infinity, which is also assumed subsonic, we specify $p = p_\infty$. The flow tangency condition is invoked along the channel walls.

We will attempt to march from an initial flow field to a steady state continually enforcing the above boundary conditions. The required initial conditions are generated as follows.

$$\rho_0 = \rho_\infty \equiv 1$$

$$e_0 = e_\infty \equiv \frac{1}{\gamma(\gamma-1)M_\infty^2} + \frac{1}{2}$$

To obtain an initial velocity field we generate u and v along the walls such that we have flow tangency there. The resulting distributions are then linearly blended.

$$u(x,y_{max}) = 1 \ , \quad v(x,y_{max}) = 0.$$

Along $y = 0$, $u^2+v^2 = 1$ together with $v = u \cdot S'(x) \Rightarrow$

$$u(x,0) = \frac{1}{\sqrt{1+[S'(x)]^2}} \ , \quad v(x,0) = \frac{S'(x)}{\sqrt{1+[S'(x)]^2}} \ ;$$

$$u(x,y) = \frac{y}{y_{max}} \cdot u(x,y_{max}) + (1 - \frac{y}{y_{max}}) \cdot u(x,0) \ ;$$

$$v(x,y) = \frac{y}{y_{max}} \cdot v(x,y_{max}) + (1 - \frac{y}{y_{max}}) \cdot v(x,0) \ .$$

For calculational purposes, we will need a mapping of the above geometry to a rectangle. To obtain a boundary conforming grid we induce a shearing of the y-coordinate. In order to cluster grid points over the bump we invoke

a stretching of the x-coordinate.

We are thus led to consider the transformation

$$t = T, \quad x = \xi(X), \quad y = Y + S(x(X))\cdot\left[1 - \frac{Y}{y_{max}}\right] \Rightarrow$$

$$x_X = \xi'(X), \qquad x_Y = 0$$

$$y_X = S'(x)\cdot x_X\cdot\left[1 - \frac{Y}{y_{max}}\right], \qquad y_Y = 1 - \frac{S(x)}{y_{max}}.$$

This sends the conservation law

$$\vec{U}_t + \vec{f}_x + \vec{g}_y = 0$$

to

$$(J\vec{U})_t + (y_Y\vec{f} - x_Y\vec{g})_X + (x_X\vec{g} - y_X\vec{f})_Y = 0$$

where $J \equiv x_X y_Y - x_Y y_X$. Letting $\vec{r} \equiv J\vec{U}$, this becomes

$$\vec{r}_t + \left[\frac{y_X}{J}\vec{f}(\vec{r}) - \frac{x_X}{J}\vec{g}(\vec{r})\right]_X + \left[-\frac{y_X}{J}\vec{f}(\vec{r}) + \frac{x_X}{J}\vec{g}(\vec{r})\right]_Y = 0$$

by the assumed homogeneity of $\vec{f}$ and $\vec{g}$. Figure III.7.1 displays the resulting mesh.

Numerical Solution of the Channel Problem

We now generalize our Runge-Kutta spline scheme to two dimensions. It will be clear that the corresponding generalizations to even higher dimensions are straightforward. The enforcement of boundary conditions is not considered in this section but is deferred to the following section.

We saw in the previous section that our problem reduces to solving

$$\vec{r}_t + \vec{F}_X(\vec{r}) + \vec{G}_Y(\vec{r}) = 0$$

on a rectangular domain. Here $\vec{r} \equiv J[\rho \quad m \quad n \quad e]^T$ and

$$\vec{F} \equiv [\frac{y_Y}{J} \cdot \vec{f} - \frac{x_Y}{J} \cdot \vec{g}]$$

$$\vec{G} \equiv [-\frac{y_X}{J} \cdot \vec{f} + \frac{x_X}{J} \cdot \vec{g}]$$

where $\vec{f}$ and $\vec{g}$ are the flux vectors in the 2D Euler equations. Note that we have preserved conservation form!

A quick calculation shows that in this context the fourth order Runge-Kutta scheme becomes

$$\vec{U}_0 \equiv \vec{U}^{(n)}$$

$$\vec{U}_1 = \vec{U}_0 - \frac{\Delta t}{2J} [D_X\vec{\Phi}(\vec{U}_0) + D_Y\vec{\Gamma}(\vec{U}_0)]$$

$$\vec{U}_2 = \vec{U}_0 - \frac{\Delta t}{2J} [D_X\vec{\Phi}(\vec{U}_1) + D_Y\vec{\Gamma}(\vec{U}_1)]$$

$$\vec{U}_3 = \vec{U}_0 - \frac{\Delta t}{J} [D_X\vec{\Phi}(\vec{U}_2) + D_Y\vec{\Gamma}(\vec{U}_2)]$$

$$\vec{U}_4 = \vec{U}_0 - \frac{\Delta t}{6J} [D_X(\vec{\Phi}(\vec{U}_0) + 2\vec{\Phi}(\vec{U}_1) + 2\vec{\Phi}(\vec{U}_2) + \vec{\Phi}(\vec{U}_3))$$

$$+ D_Y(\vec{\Gamma}(\vec{U}_0) + 2\vec{\Gamma}(\vec{U}_1) + 2\vec{\Gamma}(\vec{U}_2) + \vec{\Gamma}(\vec{U}_3))]$$

$$\equiv \vec{U}^{(n+1)}$$

where

$$J\vec{F}(\vec{U}) = y_Y\vec{f}(\vec{U}) - x_Y\vec{g}(\vec{U}) \equiv \vec{\Phi}(\vec{U})$$

$$J\vec{G}(\vec{U}) = -y_X\vec{f}(\vec{U}) + x_Y\vec{g}(\vec{U}) \equiv \vec{\Gamma}(\vec{U}) \quad .$$

Now, $D_X\vec{\Phi}$ and $D_Y\vec{\Gamma}$ are defined as in the one dimensional case. First $\vec{\Phi}$ and $\vec{\Gamma}$ are split as previously described producing

$$D_X^{\rightarrow}\Phi = D_X^-\Phi^+ + D_X^+\Phi^-$$
$$D_Y^{\rightarrow}\Gamma = D_Y^-\Gamma^+ + D_Y^+\Gamma^- \ .$$

Letting Φ^+, Φ^-, Γ^+, Γ^- denote the generic components of $\overset{\rightarrow}{\Phi}^+, \overset{\rightarrow}{\Phi}^-, \overset{\rightarrow}{\Gamma}^+, \overset{\rightarrow}{\Gamma}^-$ respectively we define

$$D_X^-\Phi^+(\vec{U}_i) = (\tau_X)_i - \varepsilon^+_{i+1/2}(\tau_X)_{i+1/2} + \varepsilon^+_{i-1/2}(\tau_X)_{i-1/2}$$
$$D_X^+\Phi^-(U_i) = (\tau_X)_i + \varepsilon^-_{i+1/2}(\tau_X)_{i+1/2} - \varepsilon^-_{i-1/2}(\tau_X)_{i-1/2}$$

where

$$\varepsilon^+ = \min(\Delta X^2 \cdot |\Phi^+_{XX}|, 1/2)$$

and

$$\varepsilon^- = \min(\Delta X^2 \cdot |\Phi^-_{XX}|, 1/2) .$$

Correspondingly

$$D_Y^-\Gamma^+(\vec{U}_i) = (\tau_Y)_i - \varepsilon^+_{i+1/2}(\tau_Y)_{i+1/2} + \varepsilon^+_{i-1/2}(\tau_Y)_{i-1/2}$$
$$D_Y^+\Gamma^-(\vec{U}_i) = (\tau_Y)_i + \varepsilon^-_{i+1/2}(\tau_Y)_{i+1/2} - \varepsilon^-_{i-1/2}(\tau_Y)_{i-1/2}$$

where $\varepsilon^+ = \min(\Delta Y^2 \cdot |\Gamma^+_{YY}|, 1/2)$

and $\varepsilon^- = \min(\Delta Y^2 \cdot |\Gamma^-_{YY}|, 1/2)$.

Boundary Condition Treatment

For the discussion of boundary conditions alone, we consider the two-dimensional (2D) Euler equations in nonconservation form. All required notation is defined in the previous section on the 2D Euler equations. Hence, in terms of the primitive variables (ρ,u,v,p), we have

$$\tilde{U}_t + \tilde{A}\tilde{U}_x + \tilde{B}\tilde{U}_y = 0 \ .$$

Letting $k_1 = 1$, $k_2 = 0$ in the definition of $\tilde{Q}$ yields

$$S \equiv \begin{bmatrix} 1 & 0 & \frac{\rho}{c\sqrt{2}} & \frac{\rho}{c\sqrt{2}} \\ 0 & 0 & \frac{1}{\sqrt{2}} & \frac{-1}{\sqrt{2}} \\ 0 & -1 & 0 & 0 \\ 0 & 0 & \frac{\rho c}{\sqrt{2}} & \frac{\rho c}{\sqrt{2}} \end{bmatrix}$$

and hence

$$S^{-1} = \begin{bmatrix} 1 & 0 & 0 & \frac{-1}{c^2} \\ 0 & 0 & -1 & 0 \\ 0 & \frac{1}{\sqrt{2}} & 0 & \frac{1}{\rho c\sqrt{2}} \\ 0 & \frac{-1}{\sqrt{2}} & 0 & \frac{1}{\rho c\sqrt{2}} \end{bmatrix}.$$

Note that

$$S^{-1}\tilde{A}\,S = \begin{bmatrix} u & 0 & 0 & 0 \\ 0 & u & 0 & 0 \\ 0 & 0 & u+c & 0 \\ 0 & 0 & 0 & u-c \end{bmatrix}, \quad S^{-1}\tilde{B}\,S = \begin{bmatrix} v & 0 & 0 & 0 \\ 0 & v & \frac{-c}{\sqrt{2}} & \frac{-c}{\sqrt{2}} \\ 0 & \frac{-c}{\sqrt{2}} & v & 0 \\ 0 & \frac{-c}{\sqrt{2}} & 0 & v \end{bmatrix}.$$

Letting $k_1 = 0$, $k_2 = 1$ in the definition of $\tilde{Q}$ yields

$$T \equiv \begin{bmatrix} 1 & 0 & \frac{\rho}{c\sqrt{2}} & \frac{\rho}{c\sqrt{2}} \\ 0 & 1 & 0 & 0 \\ 0 & 0 & \frac{1}{\sqrt{2}} & \frac{-1}{\sqrt{2}} \\ 0 & 0 & \frac{\rho c}{\sqrt{2}} & \frac{\rho c}{\sqrt{2}} \end{bmatrix}$$

and hence

$$T^{-1} = \begin{bmatrix} 1 & 0 & 0 & \frac{-1}{c^2} \\ 0 & 1 & 0 & 0 \\ 0 & 0 & \frac{1}{\sqrt{2}} & \frac{1}{\rho c\sqrt{2}} \\ 0 & 0 & \frac{-1}{\sqrt{2}} & \frac{1}{\rho c\sqrt{2}} \end{bmatrix} .$$

Note that

$$T^{-1}\tilde{A}\,T = \begin{bmatrix} u & 0 & 0 & 0 \\ 0 & u & \frac{c}{\sqrt{2}} & \frac{c}{\sqrt{2}} \\ 0 & \frac{c}{\sqrt{2}} & u & 0 \\ 0 & \frac{c}{\sqrt{2}} & 0 & u \end{bmatrix} , \quad T^{-1}\tilde{B}\,T = \begin{bmatrix} v & 0 & 0 & 0 \\ 0 & v & 0 & 0 \\ 0 & 0 & v+c & 0 \\ 0 & 0 & 0 & v-c \end{bmatrix} .$$

Henceforth, we identify a frozen quantity by a bar.

Now, along the channel inlet and outlet the number and type of boundary conditions are determined by the equation

$$\tilde{U}_t + \tilde{A}\,\tilde{U}_x = 0 .$$

We now perform a local linearization via frozen values of the elements of $\tilde{A}$. Recall that

$$S^{-1}\tilde{A}\,S = \Lambda_u \equiv \text{diag } [u,u,u+c,u-c]$$

$$\therefore \quad \tilde{A} = S\,\Lambda_u\,S^{-1}$$

$$\Rightarrow \quad \tilde{U}_t + S\,\Lambda_u S^{-1}\cdot\tilde{U}_x = 0$$

$$\Rightarrow \quad S^{-1}\tilde{U}_t + \Lambda_u S^{-1}\tilde{U}_x = 0 .$$

Making the substitution

$$\tilde{W} = S^{-1}\tilde{U}$$

we have, as a consequence of the linearization,

$$\tilde{W}_t + \Lambda_u \tilde{W}_x = 0$$

which is a system of uncoupled scalar equations. Since the inlet is assumed subsonic we have three positive eigenvalues and one negative eigenvalue. We hence specify three conditions at the inlet being careful not to determine the characteristic variable, $\tilde{W}_4$, associated with u-c. At the outlet we have the opposite situation and hence specify one condition being careful not to determine $\tilde{W}_1$, $\tilde{W}_2$ or $\tilde{W}_3$. We thus have

$$\tilde{W} = \begin{bmatrix} \tilde{W}_1 \\ \tilde{W}_2 \\ \tilde{W}_3 \\ \tilde{W}_4 \end{bmatrix} = \begin{bmatrix} \rho - \frac{p}{\bar{c}^2} \\ -v \\ \frac{1}{\sqrt{2}}\left[u + \frac{p}{\bar{\rho}\bar{c}}\right] \\ \frac{1}{\sqrt{2}}\left[-u + \frac{p}{\bar{\rho}\bar{c}}\right] \end{bmatrix} \quad (= S^{-1}\tilde{U}) \ .$$

Along the upper channel wall the relevant equation is

$$\tilde{U}_t + \tilde{B}\,\tilde{U}_Y = 0 \ .$$

Recall that

$$T^{-1}\tilde{B}\,T = \Lambda_v \equiv \text{diag}\,[v, v, v+c, v-c]$$

$\therefore$

$$\tilde{B} = T \Lambda_v T^{-1}$$

$$\Rightarrow \quad \tilde{U}_t + T \Lambda_v T^{-1} \cdot \tilde{U}_Y = 0$$

$$\Rightarrow \quad T^{-1} \tilde{U}_t + \Lambda_v T^{-1} \tilde{U}_Y = 0 \ .$$

Making the substitution

$$\tilde{V} = T^{-1} \tilde{U}$$

together with freezing the elements of $\tilde{B}$ at their current values yields

$$\tilde{V}_t + \Lambda_v \tilde{V}_Y = 0 \ ,$$

which is once again an uncoupled system of scalar equations. Since the upper wall is straight the flow tangency condition requires that $v = 0$ along it. We hence have one positive eigenvalue and one negative eigenvalue. Therefore, along the upper channel wall we stipulate one condition being careful not to fully specify $\tilde{V}_1$, $\tilde{V}_2$ or $\tilde{V}_3$.
We thus have

$$\tilde{V} = \begin{bmatrix} \tilde{V}_1 \\ \tilde{V}_2 \\ \tilde{V}_3 \\ \tilde{V}_4 \end{bmatrix} = \begin{bmatrix} \rho - \frac{p}{c^2} \\ u \\ \frac{1}{\sqrt{2}} \left[v + \frac{p}{\rho c} \right] \\ \frac{1}{\sqrt{2}} \left[-v + \frac{p}{\rho c} \right] \end{bmatrix} \quad (= T^{-1} \tilde{U}) \ .$$

Since the lower channel wall is not flat we must perform a local rotation (and translation) of coordinate system.
For this discussion, define

$$\alpha = \tan^{-1}\left(\frac{dy}{dx}\right)$$

where we use the principal value ($-\pi/2 \leq \alpha \leq \pi/2$) and dy/dx is the slope at the generic point (x_0, y_0) along the lower wall. We have

$$\tilde{U}_t + \tilde{A}\,\tilde{U}_X + \tilde{B}\,\tilde{U}_Y = 0.$$

Invoke the transformation

$$\bar{X} = (X-X_0)\cos\alpha + (Y-Y_0)\sin\alpha$$

$$\bar{Y} = -(X-X_0)\sin\alpha + (Y-Y_0)\cos\alpha.$$

The above equation becomes

$$\tilde{U}_t + \bar{A}\,\tilde{U}_{\bar{X}} + \bar{B}\,\tilde{U}_{\bar{Y}} = 0$$

with

$$\bar{A} \equiv \cos\alpha \cdot \tilde{A} + \sin\alpha \cdot \tilde{B}$$

$$\bar{B} \equiv -\sin\alpha \cdot \tilde{A} + \cos\alpha \cdot \tilde{B}\ .$$

The relevant equation for the boundary conditions along the lower channel wall is therefore

$$\tilde{U}_t + \bar{B}\,\tilde{U}_{\bar{Y}} = 0\ .$$

Hence, we are interested in diagonalizing $\bar{B}$. Using the formulae of Beam and Warming with $k_1 = -\sin\alpha$ and $k_2 = \cos\alpha$ yields ($\tilde{k}_1 = k_1$, $\tilde{k}_2 = k_2$)

$$R^{-1}\bar{B}\,R = \Lambda \equiv \text{diag}\,[\lambda_1, \lambda_2, \lambda_3, \lambda_4]$$

with $\lambda_1 = \lambda_2 = -\sin\alpha \cdot u + \cos\alpha \cdot v$, $\lambda_3 = \lambda_1 + c$, $\lambda_4 = \lambda_1 - c$ (note that $\lambda_1 = \lambda_2$ = normal component of the flow) and

$$R = \begin{bmatrix} 1 & 0 & \frac{\rho}{c\sqrt{2}} & \frac{\rho}{c\sqrt{2}} \\ 0 & \cos\alpha & \frac{-\sin\alpha}{\sqrt{2}} & \frac{\sin\alpha}{\sqrt{2}} \\ 0 & \sin\alpha & \frac{\cos\alpha}{\sqrt{2}} & \frac{-\cos\alpha}{\sqrt{2}} \\ 0 & 0 & \frac{\rho c}{\sqrt{2}} & \frac{\rho c}{\sqrt{2}} \end{bmatrix};$$

$$R^{-1} = \begin{bmatrix} 1 & 0 & 0 & \frac{-1}{c^2} \\ 0 & \cos\alpha & \sin\alpha & 0 \\ 0 & \frac{-\sin\alpha}{\sqrt{2}} & \frac{\cos\alpha}{\sqrt{2}} & \frac{1}{\rho c\sqrt{2}} \\ 0 & \frac{\sin\alpha}{\sqrt{2}} & \frac{-\cos\alpha}{\sqrt{2}} & \frac{1}{\rho c\sqrt{2}} \end{bmatrix}.$$

$\therefore$

$$\bar{B} = R \Lambda R^{-1}$$

$$\Rightarrow \quad \tilde{U}_t + R \Lambda R^{-1} \tilde{U}_{\bar{Y}} = 0$$

$$\Rightarrow \quad R^{-1}\tilde{U}_t + \Lambda R^{-1}\tilde{U}_{\bar{Y}} = 0 .$$

Now make the substitution

$$\tilde{Z} = R^{-1} \tilde{U}$$

while freezing the elements of $\bar{B}$ which results in the uncoupled

scalar equations

$$\tilde{z}_t + \Lambda \tilde{z}_{\bar{Y}} = 0 .$$

We thus have $\lambda_3 > 0$, $\lambda_4 < 0$ which implies that we should specify one condition along the lower channel wall taking care not to determine $\tilde{z}_1$, $\tilde{z}_2$ or $\tilde{z}_4$. We obtain

$$\tilde{z} = \begin{bmatrix} \tilde{z}_1 \\ \tilde{z}_2 \\ \tilde{z}_3 \\ \tilde{z}_4 \end{bmatrix} = \begin{bmatrix} \rho - \frac{p}{\bar{c}^2} \\ \cos\alpha \cdot u + \sin\alpha \cdot v \\ \frac{-\sin\alpha}{\sqrt{2}} \cdot u + \frac{\cos\alpha}{\sqrt{2}} \cdot v + \frac{p}{\bar{\rho}\,\bar{c}\,\sqrt{2}} \\ \frac{\sin\alpha}{\sqrt{2}} \cdot u - \frac{\cos\alpha}{\sqrt{2}} \cdot v + \frac{p}{\bar{\rho}\,\bar{c}\,\sqrt{2}} \end{bmatrix} (= R^{-1}\tilde{U}).$$

We now treat the boundary conditions along the inlet, outlet, upper wall, and lower wall. In what follows, once $\tilde{U}^{(n+1)} = [\rho^{(n+1)} \; u^{(n+1)} \; v^{(n+1)} \; p^{(n+1)}]^T$ is determined we find $U^{(n+1)} = [\rho^{(n+1)} \; m^{(n+1)} \; n^{(n+1)} \; e^{(n+1)}]^T$ by using the appropriate relationships.

Inlet

We specify $\rho^{(n+1)} = \rho_\infty$, $u^{(n+1)} = u_\infty$, $v^{(n+1)} = v_\infty$ while discarding the equations for $\tilde{W}_1$, $\tilde{W}_2$, $\tilde{W}_3$. We allow $\tilde{W}_4$ to be determined by the differential equation. This produces

$$p^{(n+1)} = \bar{p} + \bar{\rho}\,\bar{c}\,(u_\infty - \bar{u}) .$$

Outlet

We specify $p^{(n+1)} = p_\infty$ and allow $\tilde{W}_1$, $\tilde{W}_2$, $\tilde{W}_3$ to be determined by the differential equation. This produces

$$\rho^{(n+1)} = \bar{\rho} + (p_\infty - \bar{p})/\bar{c}^2$$

$$u^{(n+1)} = \bar{u} + (\bar{p} - p_\infty)/\bar{\rho}\,\bar{c}$$

$$v^{(n+1)} = \bar{v} \quad .$$

Upper Wall

We specify $v = 0$ in lieu of the relation for $\tilde{V}_4$. $\tilde{V}_1$, $\tilde{V}_2$, $\tilde{V}_3$ are then determined by the differential equation. This produces

$$\rho^{(n+1)} = \bar{\rho} + \bar{\rho}\bar{v}/\bar{c}$$

$$u^{(n+1)} = \bar{u}$$

$$p^{(n+1)} = \bar{p} + \bar{\rho}\,\bar{c}\,\bar{v} \quad .$$

Lower Wall

We wish to have the normal component of the velocity vanish. Hence we require

$$-\sin\alpha \cdot u + \cos\alpha \cdot v = 0$$

$$\left(\text{i.e. } \frac{v}{u} = \tan\alpha = \frac{dy}{dx}\right) \quad .$$

We replace the relation for $\tilde{Z}_3$ with this requirement and then allow $\tilde{Z}_1$, $\tilde{Z}_2$, $\tilde{Z}_4$ to be determined by the differential equation. This produces

$$\rho^{(n+1)} = \bar{\rho} - \frac{\bar{\rho}}{\bar{c}} (- \sin \alpha \cdot \bar{u} + \cos \alpha \cdot \bar{v})$$

$$u^{(n+1)} = \cos^2\alpha \cdot \bar{u} + \sin \alpha \cdot \cos \alpha \cdot \bar{v}$$

$$v^{(n+1)} = \sin \alpha \cdot \cos \alpha \cdot \bar{u} + \sin^2\alpha \cdot \bar{v}$$

$$p^{(n+1)} = \bar{p} - \bar{\rho}\,\bar{c} (- \sin \alpha \cdot \bar{u} + \cos \alpha \cdot \bar{v}) .$$

Numerical Results

The preceding analysis was implemented and tested on a subsonic problem and a transonic problem. The results of the calculations are displayed in Figures III.7.2-III.7.3. The subsonic results reveal the anticipated symmetry. The transonic results produce the expected asymmetry together with a pressure loss upon passing through the shock along the lower wall. The successful treatment of upstream and downstream boundary conditions is evident in both cases.

However, it should be noted that these two dimensional calculations require long execution times. The source of this difficulty is the multitude of required exponential function evaluations. Future work will focus on replacing the exponential basis functions by rational functions with the aim of rectifying this situation.

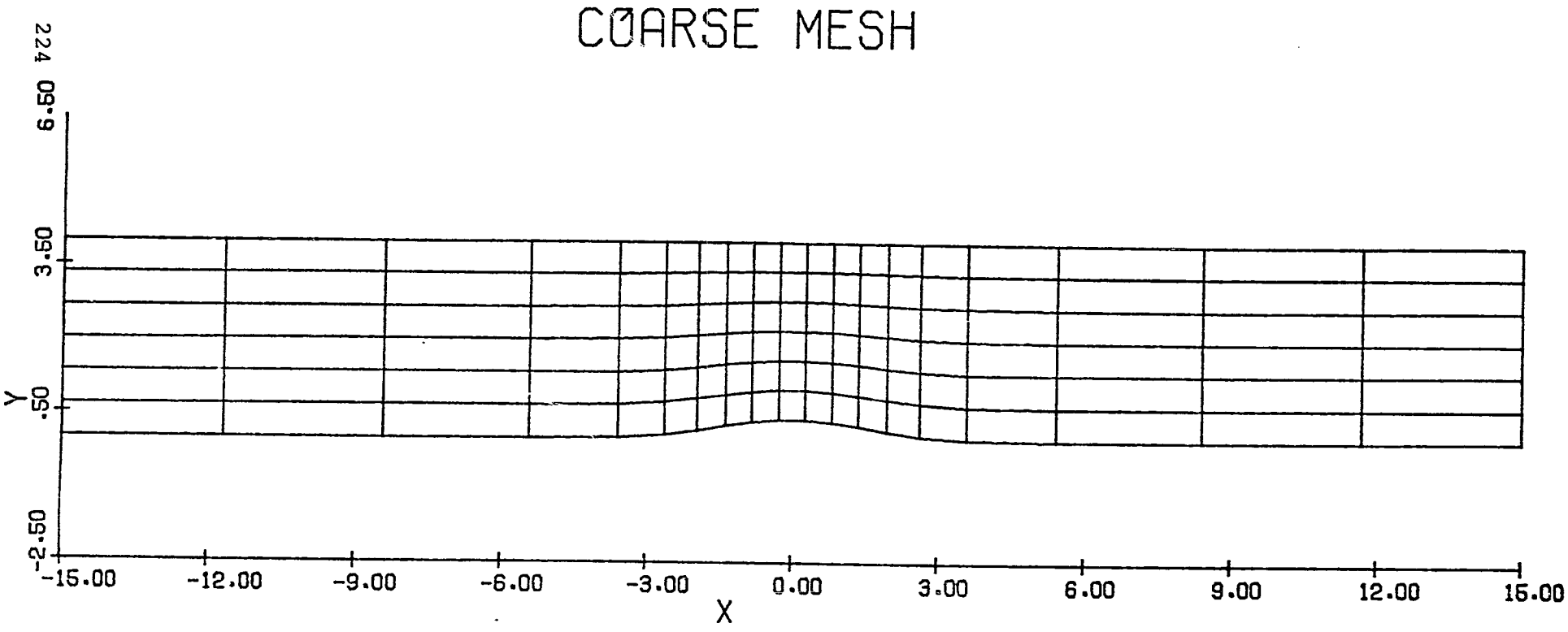


Figure III.7.1a. Computational Mesh--Coarse (20 × 7)

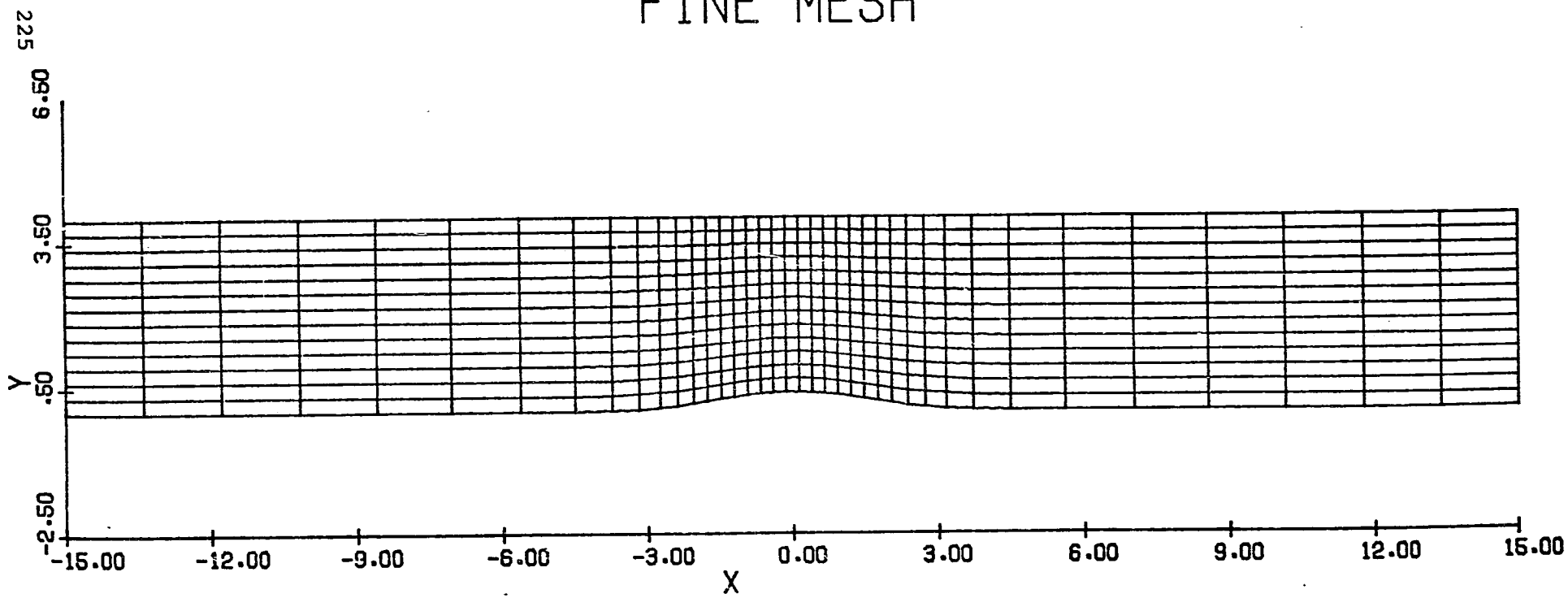


Figure III.7.1b. Computational Mesh--Fine (40 × 14)

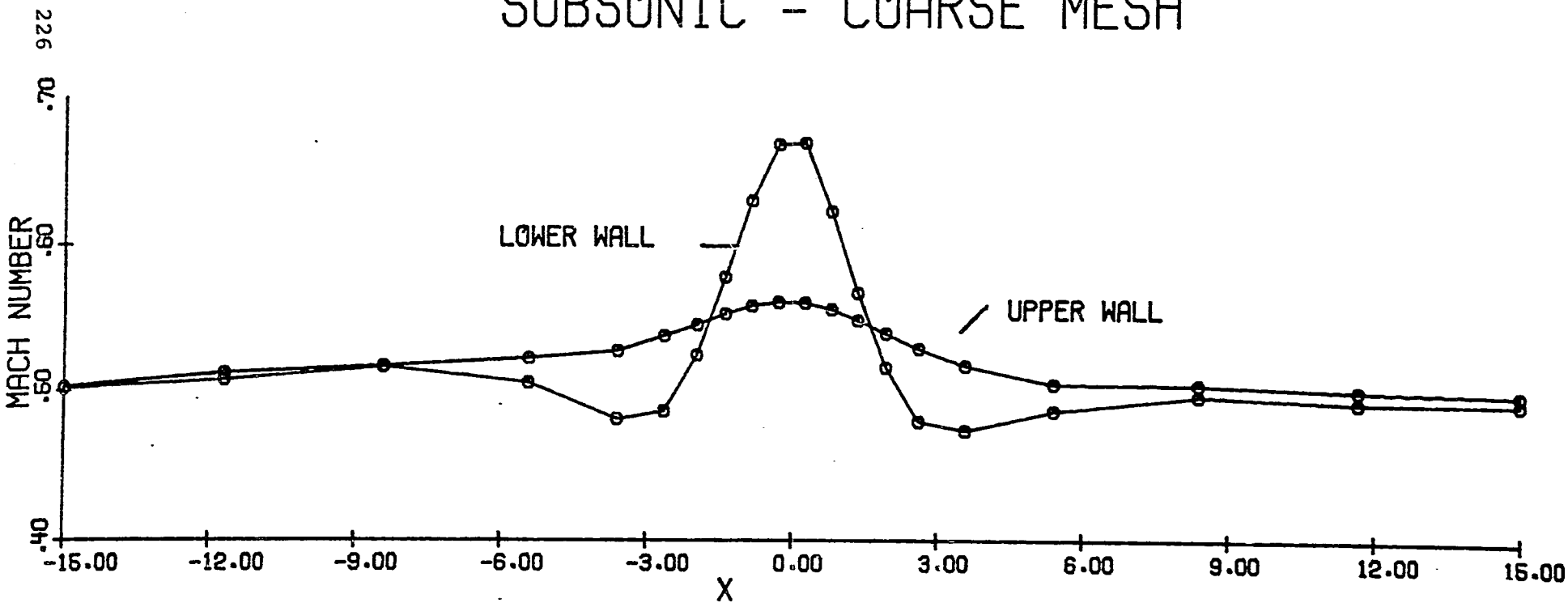


Figure III.7.2a. Computational Results – Coarse Mesh – Subsonic (NSTEP = 500)

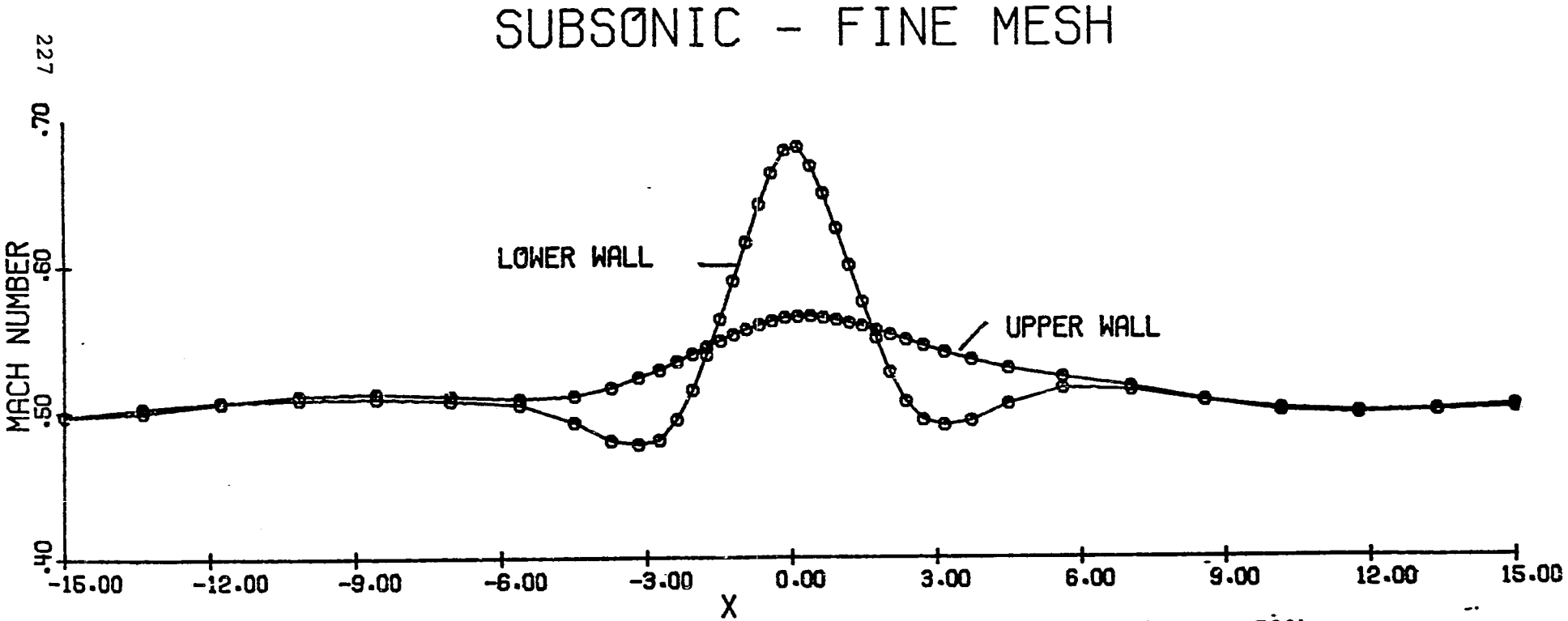


Figure III.7.2b. Computational Results - Fine Mesh - Subsonic (NSTEP = 700)

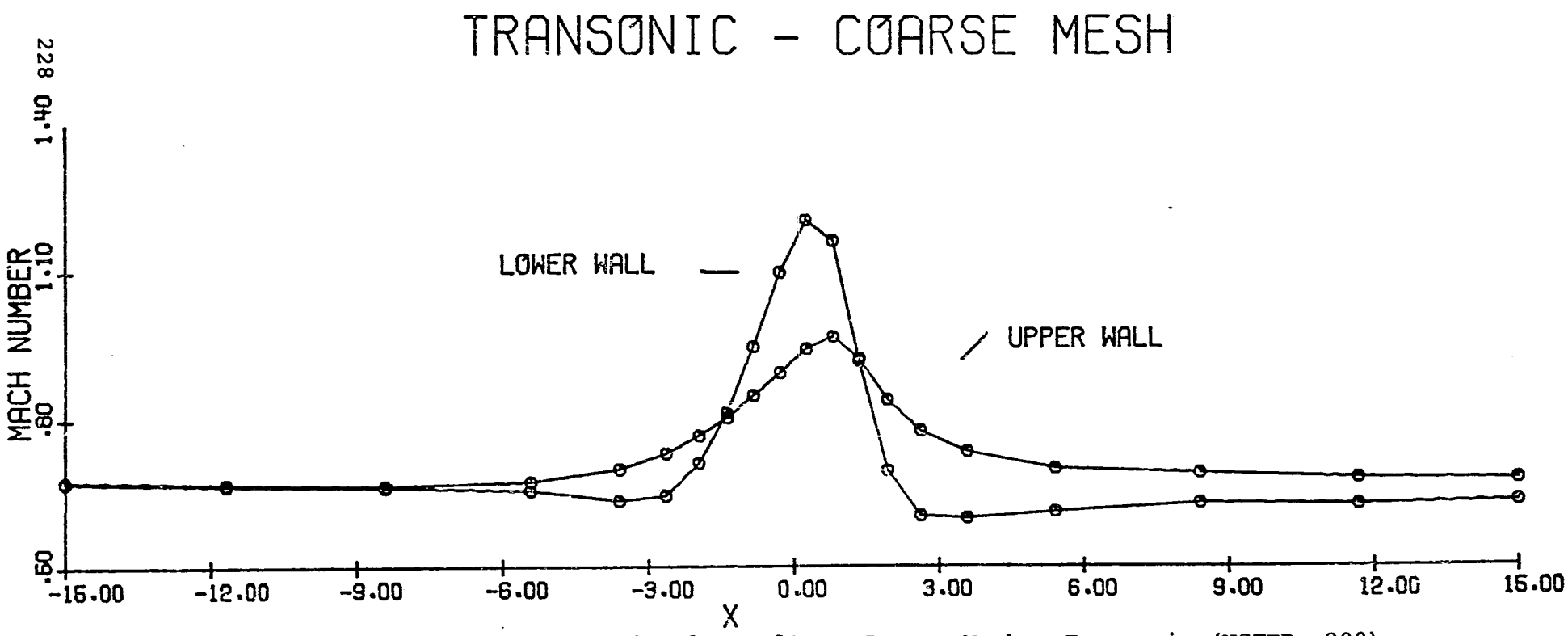


Figure III.7.3a. Computational Results - Coarse Mesh - Transonic (NSTEP = 800)

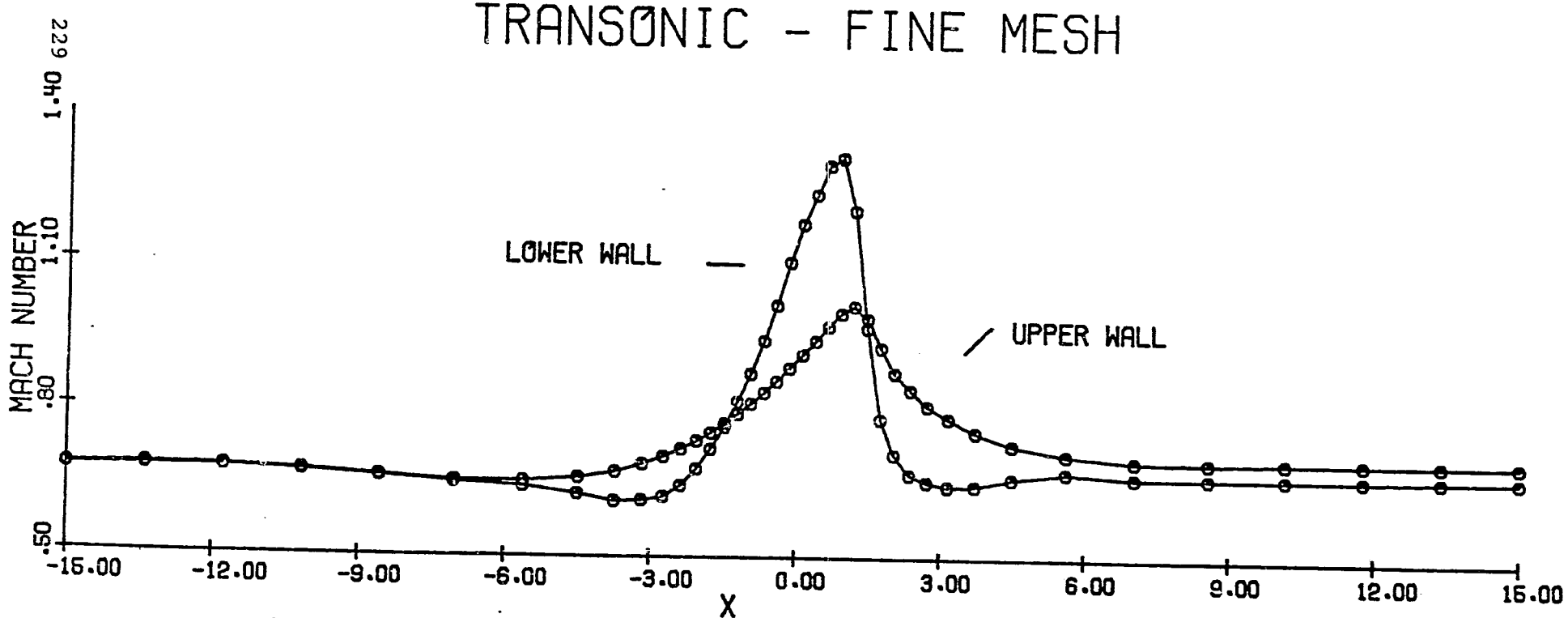


Figure III.7.3b. Computational Results - Fine Mesh - Transonic (NSTEP = 1400)

Bibliography

1. Abramowitz, M., and Stegun, I., Handbook of Mathematical Functions, National Bureau of standards, 1964.
2. Ahlberg, J. H., Nilson, E. N., and Walsh, J. L., The Theory of Splines and Their Applications, Academic Press, 1967.
3. Ahlberg, J. H., Nilson, E. N., and Walsh, J. L., "Fundamental Properties of Generalized Splines," Proc. Nat. Acad. Sci., Vol. 52, No. 6, 1964, pp. 1412-1419.
4. Ahlberg, J. H., Nilson, E. N., and Walsh, J. L., "Convergence Properties of Generalized Splines," Proc. Nat. Acad. Sci., Vol. 54, No. 2, 1965, pp. 344-350.
5. Amos, D. E., "Computation with Splines and B-Splines," Sandia, SAND78-1968, Mar. 1979.
6. Baum, A. M., "An Algebraic Approach to Simple Hyperbolic Splines on the Real Line," J. Approx. Theory, Vol. 17, 1976, pp. 189-199.
7. Birkhoff, G., and de Boor, C., "Error Bounds for Spline Interpolation," J. Math. Mech., Vol. 13, No. 5, 1964, pp. 827-835.
8. Birkhoff, G., and Zarantonello, E. H., Jets, Wakes, and Cavities, Academic, 1957, pp. 139 and 153.
9. deBoor, C., A Practical Guide to Splines, Springer-Verlag, 1978.

10. Cline, A. K., "Scalar- and Planar-Valued Curve Fitting Using Splines Under Tension," Comm. ACM, Vol. 17, No. 4, 1974, pp. 218-220.

11. Cline, A. K., "Six Subprograms for Curve Fitting Using Splines Under Tension," Comm. ACM, Vol. 17, No. 4, 1974, pp. 220-223.

12. Courant, R., and Friedrichs, K. O., Supersonic Flow and Shock Waves, Springer-Verlag, 1976, pp. 273-282.

13. Crawford, M. E., and Kays, W. M., "STAN5--A Program for Numerical Computation of Two Dimensional Internal and External Boundary Layer Flows," NASA CR-2742, 1976.

14. Curtis, A. R., and Powell, M. J. D., "Using Cubic Splines to Approximate Functions of One Variable to Prescribed Accuracy," Harwell AERE-R5602, 1967.

15. Dube, R. P., "Univariate Blending Functions and Alternatives," Comp. Graphics and Image Proc., Vol. 6, 1977, pp. 394-408.

16. Eiseman, P. R., and Smith, R. E., "Mesh Generation Using Algebraic Techniques," Proc. NASA/ICASE Workshop on Numerical Grid Generation Techniques for Partial Differential Equations, Oct., 1980, pp. 73-120.

17. Flaherty, J. E., and Mathon, W., "Collocation with Polynomial and Tension Splines for Singularly-Perturbed Boundary Value Problems," SIAM J. Sci. Stat. Comput., Vol. 1, No. 2, Jun, 1980, pp. 260-284.

18. Garabedian, P. R., Partial Differential Equations, Wiley, 1964, p. 599.

19. Gottlieb, D., Gunzburger, M., and Turkel, E., "On Numerical Boundary Treatment for Hyperbolic Systems," ICASE Rept. No. 78-13, Sep. 7, 1978.

20. Greville, T. N. E., "Spline Functions, Interpolation and Numerical Quadrature," in Mathematical Methods for Digital Computers: Vol. II, A. Ralston and H. S. Wilf (eds.), Wiley, 1967, pp. 156-168.

21. Greville, T. N. E., "Interpolation by Generalized Splines," MRC TSR #476, May, 1964.

22. Gustafsson, B., and Oliger, J., "Stable Boundary Discretizations of $u_t = A D_0 u$," Uppsala University, Dept. of Comp. Sci. Rept. No. 87, Sep. 1980.

23. Hill, D. W., "Estimation of Probability Functions Using Splines," Ph.D. Thesis, University of New Mexico, 1973.

24. Jameson, A., "Transonic Flow Computations," in Numerical Methods in Fluid Dynamics, M. J. Wirz and J. J. Smoldern (eds.), Hemisphere, 1978, pp. 1-87.

25. Kershaw, D., "A Note on the Convergence of Interpolatory Cubic Splines," SIAM J. Numer. Anal., Vol. 8, No. 1, 1971, pp. 67-74.

26. Korn, D. G., "Numerical Design of Transonic Cascades," Courant Institute of Mathematical Sciences, ERDA Mathematics and Computing Laboratory Rept. No. COO-3077-72, Jan., 1975.

27. Kreiss, H.-O., Oliger, J., Methods for the Approximate Solution of Time Dependent Problems, GARP Publ. Series No. 10, Feb., 1973, pp. 64-70.

28. Lax, P. D., and Wendroff, B., "Systems of Conservation Laws," Comm. Pure and Appl. Math., Vol. 13, 1960, pp. 217-237.

29. Lax, P. D., Hyperbolic Systems of Conservation Laws and the Mathematical Theory of Shock Waves, CBMS Vol. 11, SIAM, 1973.

30. Lawson, C. L., "C^1-Compatible Interpolation over a Triangle," JPL Tech. Memo. 33-770, May 1, 1976.

31. Liepmann, H. W., and Roshko, A., Elements of Gas Dynamics, Wiley, 1957, pp. 62-83.

32. Nielsen, G. M., "Some Piecewise Polynomial Alternatives to Splines Under Tension," in Computer Aided Geometric Design, R. E. Barnhill and R. F. Riesenfeld (eds.), Academic, 1974, pp. 209-235.

33. Prenter, P. M., "Piecewise L-Splines," Numer. Math., Vol. 18, 1971, pp. 243-253.

34. Prenter, P. M., Splines and Variational Methods, Wiley, 1975.

35. Pruess, S., "Properties of Splines in Tension," J. Approx. Theory, Vol. 17, 1976, pp. 86-96.

36. Pruess, S., "An Algorithm for Computing Smoothing Splines in Tension," Computing, Vol. 19, 1978, pp. 365-373.

37. Pruess, S., "Alternatives to the Exponential Spline in Tension," Math. Comp., Vol. 33, No. 148, 1979, pp. 1273-1281.

38. Rentrop, P., "An Algorithm for the Computation of the Exponential Spline," Numer. Math., Vol. 35, 1980, pp. 81-93.

39. Rubin, S. G., and Graves, R. A., "A Cubic Spline Approximation for Problems in Fluid Mechanics," NASA TR R-436, Oct., 1975.

40. Schultz, M. H., and Varga, R. S., "L-Splines," Numer. Math., Vol. 10, 1967, pp. 345-369.

41. Schumaker, L., Spline Functions: Basic Theory, Wiley, 1981.

42. Schweikert, D. G., "The Spline in Tension (Hyperbolic Spline) and the Reduction of Extraneous Inflection Points," Ph.D. Thesis, Brown University, 1966.

43. Schweikert, D. G., "An Interpolation Curve Using a Spline in Tension," J. Math. and Phys., Vol. 45, 1966, pp. 312-317.

44. Seidman, T. I., and Korsan, R. J., "Endpoint Formulas for Interpolatory Cubic Splines," Math. Comp., Vol. 26, No. 120, 1972, pp. 897-900.

45. Smith, R. E., and Wiegel, B. L., "Analytic and Approximate Boundary Fitted Coordinate Systems for Fluid Flow Simulation," AIAA Paper 80-0192, 1980.

46. Sod, G. A., "A Survey of Several Finite Difference Methods for Systems of Nonlinear Hyperbolic Conservation Laws," J. Comp. Phys., Vol. 27, 1978, pp. 1-31.

47. Späth, H., "Exponential Spline Interpolation," Computing, Vol. 4, 1969, pp. 225-233.

48. Späth, H., "Two-Dimensional Exponential Splines," Computing, Vol. 7, 1971, pp. 364-369.

49. Späth, H., Spline Algorithms for Curves and Surfaces, Utilitas Mathematica, 1974.

50. Steger, J. L., and Warming, R. F., "Flux Vector Splitting of the Inviscid Gasdynamic Equations with Application to Finite Difference Methods," NASA Tech. Memo. 78605, Jul., 1979.

51. Swartz, B. K., and Varga, R. S., "Error Bounds for Spline and L-Spline Interpolation," J. Approx. Theory, Vol. 6, 1972, pp. 6-49.

52. Turkel, E., "Numerical Methods for Large-Scale, Time-Dependent Partial Differential Equations," ICASE Rept. No. 79-20, Aug. 20, 1979.

53. Warming, R. F., and Beam, R. M., "On the Construction and Application of Implicit Factored Schemes for Conservation Laws," in Computational Fluid Dynamics, SIAM-AMS Proceedings, Vol. 11, 1978, pp. 85-124.

Appendix I: Formal Expansion of Exponential Spline Derivatives

In this appendix we formally derive some expansions for the derivatives of the exponential spline at a node in terms of the derivatives at the node of the function being approximated. The primary tool is the calculus of finite difference operators [14,39]. Consequently, the results so obtained are not rigorous thus requiring supplementary proofs. However, this technique is useful for suggesting candidates for such relationships. These results are referred to in Sections III.3 and III.4.

First Derivative

As we know, the exponential spline satisfies

$$\left[\frac{e_{i-1}}{d_{i-1}^2-e_{i-1}^2}\right]\tau'_{i-1} + \left[\frac{d_{i-1}}{d_{i-1}^2-e_{i-1}^2} + \frac{d_i}{d_i^2-e_i^2}\right]\tau'_i + \left[\frac{e_i}{d_i^2-e_i^2}\right]\tau'_{i+1}$$

$$= \frac{1}{d_{i-1}-e_{i-1}} \quad \frac{f_i-f_{i-1}}{h_{i-1}} + \frac{1}{d_i-e_i} \cdot \frac{f_{i+1}-f_i}{h_i}$$

at interior nodes. This may be rewritten as

$$\left\{\left[\frac{e_{i-1}}{d_{i-1}^2-e_{i-1}^2}\right]E^- + \left[\frac{d_{i-1}}{d_{i-1}^2-e_{i-1}^2} + \frac{d_i}{d_i^2-e_i^2}\right]I + \left[\frac{e_i}{d_i^2-e_i^2}\right]E^+\right\}\tau'_i$$

$$= \left\{\frac{1}{(d_{i-1}-e_{i-1})h_{i-1}}(I-E^-) + \frac{1}{(d_i-e_i)h_i}(E^+-I)\right\}f_i$$

where I is the identity operator, $If_i = f(x_i)$, E^+ is the forward shift operator, $E^+f_i = f(x_{i+1})$, and E^- is the backward shift operator, $E^-f_i = f(x_{i-1})$.

Proceeding symbolically, we have

$$\tau_i' = \left\{\left[\frac{e_{i-1}}{d_{i-1}^2-e_{i-1}^2}\right]E^- + \left[\frac{d_{i-1}}{d_{i-1}^2-e_{i-1}^2} + \frac{d_i}{d_i^2-e_i^2}\right]I + \left[\frac{e_i}{d_i^2-e_i^2}\right]E^+\right\}^{-1}$$

$$\cdot\left\{\left[\frac{-1}{(d_{i-1}-e_{i-1})h_{i-1}}\right]E^- + \left[\frac{1}{(d_{i-1}-e_{i-1})h_{i-1}} - \frac{1}{(d_i-e_i)h_i}\right]I\right.$$

$$\left. + \left[\frac{1}{(d_i-e_i)h_i}\right]E^+\right\}f_i \ .$$

Here

$$E^+ = e^{h_i D} = I + (h_i D) + \frac{(h_i D)^2}{2!} + \ldots \infty$$

$$E^- = e^{-h_{i-1} D} = I - (h_{i-1} D) + \frac{(h_{i-1} D)^2}{2!} + \ldots \infty$$

Before proceeding further we require some power series expansions for the spline coefficients:

(i) $e = \frac{h}{6}\left[1 - \frac{7}{60}p^2h^2 + \frac{31}{2,520}p^4h^4 + O(p^6h^6)\right]$

(ii) $d = \frac{h}{3}\left[1 - \frac{1}{15}p^2h^2 + \frac{2}{315}p^4h^4 + O(p^6h^6)\right]$

(iii) $\frac{1}{d-e} = \frac{1}{h}\left[6 + \frac{1}{10}p^2h^2 - \frac{1}{1,400}p^4h^4 + O(p^6h^6)\right]$

(iv) $\frac{1}{d+e} = \frac{1}{h}\left[2 + \frac{1}{6}p^2h^2 - \frac{1}{360}p^4h^4 + O(p^6h^6)\right]$

$$\text{(v)}\quad \frac{1}{h(d-e)} = \frac{1}{h^2}\left[6 + \frac{1}{10}\,p^2h^2 - \frac{1}{1,400}\,p^4h^4 + O(p^6h^6)\right]$$

$$\text{(vi)}\quad \frac{1}{d^2-e^2} = \frac{1}{h^2}\left[12 + \frac{6}{5}\,p^2h^2 - \frac{1}{700}\,p^4h^4 + O(p^6h^6)\right]$$

$$\text{(vii)}\quad \frac{e}{d^2-e^2} = \frac{1}{h}\left[2 - \frac{1}{30}\,p^2h^2 + \frac{13}{12,600}\,p^4h^4 + O(p^6h^6)\right]$$

$$\text{(viii)}\quad \frac{d}{d^2-e^2} = \frac{1}{h}\left[4 + \frac{2}{15}\,p^2h^2 - \frac{11}{6,300}\,p^4h^4 + O(p^6h^6)\right] .$$

We may now proceed by inserting the appropriate expansions into the relation between τ_i' and f_i. This yields

$$\tau_i' = \left\{a_0 I + a_1 D + a_2 D^2 + a_3 D^3 + a_4 D^4 + a_5 D^5 + \ldots\right\}^{-1}$$

$$\cdot\left\{c_1 D + c_2 D^2 + c_3 D^3 + c_4 D^4 + c_5 D^5 + \ldots\right\} f_i$$

where D is the differentiation operator, $Df_i = f'(x_i)$.
In the above

$$a_0 = 6\left(\frac{1}{h_{i-1}} + \frac{1}{h_i}\right) + \frac{1}{10}\left(p_{i-1}^2 h_{i-1} + p_i^2 h_i\right) - \frac{1}{1,400}\left(p_{i-1}^4 h_{i-1}^3 + p_i^4 h_i^3\right) + \ldots \infty$$

$$a_1 = \frac{1}{30}\left(p_{i-1}^2 h_{i-1}^2 - p_i^2 h_i^2\right) - \frac{13}{12,600}\left(p_{i-1}^4 h_{i-1}^4 - p_i^4 h_i^4\right) + \ldots \infty$$

$$a_2 = \left(h_{i-1} + h_i\right) - \frac{1}{60}\left(p_{i-1}^2 h_{i-1}^3 + p_i^2 h_i^3\right) + \frac{13}{25,200}\left(p_{i-1}^4 h_{i-1}^5 + p_i^4 h_i^5\right) + \ldots \infty$$

$$a_3 = -\frac{1}{3}\left(h_{i-1}^2 - h_i^2\right) + \frac{1}{180}\left(p_{i-1}^2 h_{i-1}^4 - p_i^2 h_i^4\right) - \frac{13}{75,600}\left(p_{i-1}^4 h_{i-1}^6 - p_i^4 h_i^6\right) + \ldots \infty$$

$$a_4 = \frac{1}{12}\left(h_{i-1}^3 + h_i^3\right) - \frac{1}{720}\left(p_{i-1}^2 h_{i-1}^5 + p_i^2 h_i^5\right) + \frac{13}{302,400}\left(p_{i-1}^4 h_{i-1}^7 + p_i^4 h_i^7\right) + \ldots \infty$$

$$a_5 = -\frac{1}{60}\,(h_{i-1}^4 - h_i^4) + \frac{1}{3{,}600}\,(p_{i-1}^2 h_{i-1}^6 - p_i^2 h_i^6)$$

$$- \frac{13}{1{,}512{,}000}\,(p_{i-1}^4 h_{i-1}^8 - p_i^4 h_i^8) + \ldots \infty$$

$$c_1 = a_0$$

$$c_2 = -\frac{1}{20}\,(p_{i-1}^2 h_{i-1}^2 - p_i^2 h_i^2) + \frac{1}{2{,}800}\,(p_{i-1}^4 h_{i-1}^4 - p_i^4 h_i^4) + \ldots \infty$$

$$c_3 = (h_{i-1} + h_i) + \frac{1}{60}\,(p_{i-1}^2 h_{i-1}^3 + p_i^2 h_i^3) - \frac{1}{8{,}400}\,(p_{i-1}^4 h_{i-1}^5 + p_i^4 h_i^5)$$

$$+ \ldots \infty$$

$$c_4 = -\frac{1}{4}\,(h_{i-1}^2 - h_i^2) - \frac{1}{240}\,(p_{i-1}^2 h_{i-1}^4 - p_i^2 h_i^4)$$

$$+ \frac{1}{33{,}600}\,(p_{i-1}^4 h_{i-1}^6 - p_i^4 h_i^6) + \ldots \infty$$

$$c_5 = \frac{1}{20}\,(h_{i-1}^3 + h_i^3) + \frac{1}{1{,}200}\,(p_{i-1}^2 h_{i-1}^5 + p_i^2 h_i^5)$$

$$- \frac{1}{168{,}000}\,(p_{i-1}^4 h_{i-1}^7 + p_i^4 h_i^7) + \ldots\infty.$$

Now

$$\left\{a_0 I + a_1 D + a_2 D^2 + a_3 D^3 + a_4 D^4 + a_5 D^5 + \ldots\right\}^{-1}$$

$$= b_0 I + b_1 D + b_2 D^2 + b_3 D^3 + b_4 D^4 + b_5 D^5 + \ldots\infty$$

$$\Rightarrow \quad b_0 = \frac{1}{a_0}$$

$$b_1 = -\frac{a_1}{a_0}$$

$$b_2 = \frac{a_1^2}{a_0^3} - \frac{a_2}{a_0^2}$$

$$b_3 = -\frac{a_1^3}{a_0^4} + \frac{2a_1 a_2}{a_0^3} - \frac{a_3}{a_0^2}$$

$$b_4 = \frac{a_1^4}{a_0^5} - \frac{3a_1^2 a_2}{a_0^4} + \frac{a_2^2 + a_1 a_3}{a_0^3} - \frac{a_4}{a_0}$$

$$b_5 = - b_4 \frac{a_1}{a_0} - b_3 \frac{a_2}{a_0} - b_2 \frac{a_3}{a_0} - b_1 \frac{a_4}{a_0} - b_0 \frac{a_5}{a_0} .$$

Hence,

$$\tau_1' = \Big\{ (b_0c_1)D + (b_1c_1+b_0c_2)D^2 + (b_2c_1+b_1c_2+b_0c_3)D^3$$
$$+ (b_3c_1+b_2c_2+b_1c_3+b_0c_4)D^4$$
$$+ (b_4c_1+ b_3c_2+ b_2c_3+ b_1c_4+ b_0c_5)D^5 + \ldots \Big\} f_i .$$

Note that $b_0c_1 = 1$.

When $p_ih_i = ph \quad \forall i$ we have

$$b_0c_2 + b_1c_1 = 0$$
$$b_0c_3 + b_1c_2 + b_2c_1 = O(h^4)$$
$$b_3c_1 + b_2c_2 + b_1c_3 + b_0c_4 = 0$$

while all succeeding terms are $O(h^4)$. That is for uniform tension and mesh width the spline first derivative is a fourth order accurate approximation to the first derivative of the approximated function (at the nodes).

Second Derivative

As we know, the exponential spline satisfies

$$e_{i-1}\tau''_{i-1} + (d_{i-1}+d_i)\tau''_i + e_i\tau''_{i+1} = \frac{f_{i+1}-f_i}{h_i} - \frac{f_i-f_{i-1}}{h_{i-1}}$$

at interior nodes.

With the necessary operators being defined as in

the previous section, this may be rewritten as

$$\left\{e_{i-1}E^{-} + (d_{i-1}+d_i)I + e_iE^{+}\right\}\tau_i'' = \left\{\frac{1}{h_i}E^{+} - \left[\frac{1}{h_i} + \frac{1}{h_{i-1}}\right]I + \frac{1}{h_{i-1}}E^{-}\right\}f_i.$$

Proceeding symbolically, we have

$$\tau_i'' = \left\{e_{i-1}E^{-} + (d_{i-1}+d_i)I + e_iE^{+}\right\}^{-1} \cdot \left\{\frac{1}{h_i}E^{+} - \left[\frac{1}{h_i} + \frac{1}{h_{i-1}}\right]I + \frac{1}{h_{i-1}}E^{-}\right\}f_i.$$

Utilizing our by now standard expansions for the spline coefficients yields

$$\tau_i'' = \left\{a_0I + a_1D + a_2D^2 + a_3D^3 + a_4D^4 + a_5D^5 + \ldots \infty\right\}^{-1} \cdot \left\{c_2D^2 + c_3D^3 + c_4D^4 + c_5D^5 + \ldots \infty\right\} f_i.$$

In the above,

$$a_0 = \frac{1}{2}(h_{i-1}+h_i) - \frac{1}{24}(p_{i-1}^2h_{i-1}^3+p_i^2h_i^3) + \frac{1}{240}(p_{i-1}^4h_{i-1}^5+p_i^4h_i^5) + \ldots \infty$$

$$a_1 = \frac{1}{6}(h_i^2-h_{i-1}^2) - \frac{7}{360}(p_i^2h_i^4-p_{i-1}^2h_{i-1}^4) + \frac{31}{15,120}(p_i^4h_i^6-p_{i-1}^4h_{i-1}^6) + \ldots \infty$$

$$a_2 = \frac{1}{12}(h_{i-1}^3+h_i^3) - \frac{7}{720}(p_{i-1}^2h_{i-1}^5+p_i^2h_i^5) + \frac{31}{30,240}(p_{i-1}^4h_{i-1}^7+p_i^4h_i^7) + \ldots\infty$$

$$a_3 = \frac{1}{36}(h_i^4-h_{i-1}^4) - \frac{7}{2,160}(p_i^2h_i^6-p_{i-1}^2h_{i-1}^6) + \frac{31}{90,720}(p_i^4h_i^8-p_{i-1}^4h_{i-1}^8) + \ldots\infty$$

$$a_4 = \frac{1}{144}(h_{i-1}^5+h_i^5) - \frac{7}{8,640}(p_{i-1}^2h_{i-1}^7+p_i^2h_i^7) + \frac{31}{362,880}(p_{i-1}^4h_{i-1}^9+p_i^4h_i^9) + \ldots \infty$$

$$a_5 = \frac{1}{720}\,(h_i^6 - h_{i-1}^6) - \frac{7}{43{,}200}\,(p_i^2 h_i^8 - p_{i-1}^2 h_{i-1}^8)$$

$$+ \frac{31}{1{,}814{,}400}\,(p_i^4 h_i^{10} - p_{i-1}^4 h_{i-1}^{10}) + \ldots \infty$$

$$c_2 = \frac{1}{2}\,(h_i + h_{i-1}) \,, \qquad c_3 = \frac{1}{6}\,(h_i^2 - h_{i-1}^2) \,,$$

$$c_4 = \frac{1}{24}\,(h_i^3 + h_{i-1}^3) \,, \qquad c_5 = \frac{1}{120}(h_i^4 - h_{i-1}^4) \,.$$

With the b's as defined in the previous section we have

$$\tau_i'' = \left\{b_0 + b_1 D + b_2 D^2 + b_3 D^3 + b_4 D^4 + b_5 D^5 + \ldots \infty\right\}$$

$$\cdot\left\{c_2 D^2 + c_3 D^3 + c_4 D^4 + c_5 D^5 + \ldots \infty\right\} \cdot f_i$$

$$= [(b_0 c_2) D^2 + (b_1 c_2 + b_0 c_3) D^3 + (b_2 c_2 + b_1 c_3 + b_0 c_4) D^4 + \ldots \infty] f_i .$$

For $p_i h_i = ph \quad \forall i$ we have

$$\tau_i'' = (f_{xx})_i + [\frac{p^2}{12}\,(f_{xx})_i - \frac{1}{12}\,(f_{xxxx})_i] h^2 + O(h^4) \,.$$

We see that the uniformity conditions do not yield the increase in accuracy encountered in our analysis of first derivative approximation.

In spite of this, the above expansion can be exploited in the following fashion. An expansion of f reveals

$$\frac{f_{i+1} - 2f_i + f_{i-1}}{h^2} = (f_{xx})_i + (f_{xxxx})_i \frac{h^2}{12} + O(h^4) \,.$$

Combining this with our previous expansion produces

$$\tau_i'' + \frac{f_{i+1} - 2f_i + f_{i-1}}{h^2} = 2(f_{xx})_i + \frac{(ph)^2}{12}\,(f_{xx})_i + O(h^4)$$

$$= \frac{24 + (ph)^2}{12}\,(f_{xx})_i + O(h^4)$$

$\therefore$

$$\frac{12}{24+(ph)^2}\left[\tau''_i + \frac{f_{i+1}-2f_i+f_{i-1}}{h^2}\right] = (f_{xx})_i + O(h^4)$$

i.e. a fourth order accurate approximation. Note that for the cubic spline ($p = 0$) this reduces to the arithmetic mean.

Recasting the spline equations as

$$\frac{e}{h}\tau''_{i-1} + \frac{2d}{h}\tau''_i + \frac{e}{h}\tau''_{i+1} = \frac{f_{i+1}-2f_i+f_{i-1}}{h^2}$$

we can rewrite the above as

$$(f_{xx})_i = \frac{12}{24+(ph)^2}\left[\frac{e}{h}\tau''_{i-1} + \left(1+\frac{2d}{h}\right)\tau''_i + \frac{e}{h}\tau''_{i+1}\right] + O(h^4) .$$

Hence

$$\frac{12+(ph)^2}{12}(f_{xx})_i = \frac{12+(ph)^2}{24+(ph)^2}\left[\frac{e}{h}\tau''_{i-1} + \left(1+\frac{2d}{h}\right)\tau''_i + \frac{e}{h}\tau''_{i+1}\right] + O(h^4)$$

$$\Rightarrow \frac{12+(ph)^2}{12}(f_{xx})_i - \tau''_i = \frac{12+(ph)^2}{24+(ph)^2}\left[\frac{e}{h}\tau''_{i-1} + \left(1+\frac{2d}{h} - \frac{24+(ph)^2}{12+(ph)^2}\right)\tau''_i + \frac{e}{h}\tau''_{i+1}\right] + O(h^4) .$$

However, our original expansion for τ''_i $\Rightarrow$

$$(f_{xxxx})_i = \frac{12}{h^2}\left[\frac{12+(ph)^2}{12}(f_{xx})_i - \tau''_i\right] + O(h^4) .$$

This gives us the added bonus of a second order accurate approximation to the fourth derivative

$$(f_{xxxx})_i = \frac{12}{h^2}\left\{\frac{12+(ph)^2}{24+(ph)^2}\left[\frac{e}{h}\tau''_{i-1} + \left(1+\frac{2d}{h} - \frac{24+(ph)^2}{12+(ph)^2}\right)\tau''_i + \frac{e}{h}\tau''_{i+1}\right]\right\} + O(h^2) .$$

Note that in the case of the cubic spline ($p = 0$, $e = h/6$, $d = h/3$) this reduces to

$$(f_{xxxx})_i = \frac{\tau''_{i-1} - 2\tau''_i + \tau''_{i+1}}{h^2} + O(h^2)$$

i.e. a central differencing of $\tau''(x)$ at $x = x_i$.

The uniform mesh expansions lead to expressions for third derivative approximations.

$$\left.\begin{aligned} \tau'''(x_i^+) &= \frac{p}{s}\,[\tau''_{i+1} - \tau''_i \cdot c] \\ \tau'''(x_i^-) &= \frac{p}{s}\,[-\tau''_{i-1} + \tau''_i \cdot c] \end{aligned}\right\} \Rightarrow$$

$$\text{(i)}\quad \frac{1}{2}\,[\tau'''(x_i^+) + \tau'''(x_i^-)] = \frac{p}{2s}\,[\tau''_{i+1} - \tau''_{i-1}]$$

$$= f'''_i + \frac{h^2}{12}\,(f_i^{(v)} - p^2 f'''_i) + O(h^4)$$

$$\text{(ii)}\quad [\tau'''(x_i^+) - \tau'''(x_i)] = \frac{p}{s}\,[\tau''_{i+1} - 2c\cdot\tau''_i + \tau''_{i-1}]$$

$$= h[(f_i^{(iv)} - p^2 f''_i) + O(h^2)]$$

$$\Rightarrow \frac{\tau'''(x_i^+) - \tau'''(x_i^-)}{h} = f_i^{(iv)} - p^2 f''_i + O(h^2)$$

$$= f_i^{(iv)} - p^2 \tau''_i + O(h^2)$$

$$\Rightarrow f_i^{(iv)} = p^2\tau''_i + \frac{\tau'''_i(x_i^+) - \tau'''(x_i^-)}{h} + O(h^2)$$

$$= p^2\tau''_i + \frac{p}{s}\,[\tau''_{i+1} - 2c\cdot\tau''_i + \tau''_{i-1}] + O(h^2)$$

$$= \frac{p}{s}\,[\tau''_{i+1} + (ps - 2c)\tau''_i + \tau''_{i-1}] + O(h^2)$$

So that we have yet another second order accurate approximation to the fourth derivative.

Appendix II: Spline Routines

```
C ***
C ***
C *** DRIVER FOR PARAMETRIC EXPONENTIAL SPLINE PACKAGE:
C *** B. J. MCCARTIN  10/80
C ***
C ***
      DIMENSION XP(100),YP(100),S(100),BX(100),BY(100),TAUDPX(100),
     1TAUDPY(100),PX(100),PY(100),HX(100),HY(100)
      CALL ERRSET(207,256,-1,1,0)
      CALL ERRSET(208,256,-1,1,0)
      CALL TKINIT
      READ(15,10) NP1
   10 FORMAT(I3)
      WRITE(16,20) NP1
   20 FORMAT(1X,'NP1 = ',I3)
      WRITE(16,30)
   30 FORMAT('         XP              YP')
      DO 60 I=1,NP1
      READ(15,40) XP(I),YP(I)
   40 FORMAT(2E15.7)
      WRITE(16,50) XP(I),YP(I)
   50 FORMAT(1X,2E15.7)
   60 CONTINUE
      READ(15,70) XENDL,YENDL,XENDR,YENDR,EPS
   70 FORMAT(5E15.7)
      WRITE(16,80) XENDL,YENDL,XENDR,YENDR,EPS
   80 FORMAT(1X,'XENDL = ',E15.7,' YENDL = ',E15.7,' XENDR = ',E15.7,
     1' YENDR = ',E15.7,' EPS = ',E15.7)
      CALL PAREXP(XP,YP,NP1,XENDL,YENDL,XENDR,YENDR,EPS,S,BX,BY,
     1TAUDPX,TAUDPY,PX,PY,HX,HY)
      RETURN
      END
C ***
C ***
C *** PAREXP: B. J. MCCARTIN  10/80
C ***
C ***
C *** FUNCTION: CONSTRUCTS THE PARAMETRIC EXPONENTIAL SPLINE
C ***           INTERPOLANT TO A SET OF POINTS IN THE PLANE
C ***           USING SPECIFIED DERIVATIVE END CONDITIONS
C ***           NOTE: ARC LENGTH IS USED AS THE PARAMETER
C ***
C *** REFERENCE: B. J. MCCARTIN, NUMERICAL COMPUTATION OF
C ***             EXPONENTIAL SPLINES, COURANT MATHEMATICS
C ***             AND COMPUTING LABORATORY, OCTOBER 1980
C ***
C *** DESCRIPTION OF CALLING ARGUMENTS:
C ***      XP     = ABSCISSAE
C ***      YP     = ORDINATES
C ***      NP1    = NUMBER OF ORDERED TRIPLETS OF DATA
C ***      XENDL  = LEFT HAND DERIVATIVE END CONDITION FOR X(S)
C ***      YENDL  = LEFT HAND DERIVATIVE END CONDITION FOR Y(S)
C ***      XENDR  = RIGHT HAND DERIVATIVE END CONDITION FOR X(S)
C ***      YENDR  = RIGHT HAND DERIVATIVE END CONDITION FOR Y(S)
C ***      EPS    = TOLERANCE FOR STOPPING CRITERIA IN
```

```
C ***                ARC LENGTH ITERATION
C ***       S      = ARC LENGTH
C ***       BX     = RIGHT HAND SIDE OF SPLINE EQUATIONS FOR X(S)
C ***       BY     = RIGHT HAND SIDE OF SPLINE EQUATIONS FOR Y(S)
C ***       TAUDPX = SOLUTION OF SPLINE EQUATIONS FOR X(S)
C ***       TAUDPY = SOLUTION OF SPLINE EQUATIONS FOR Y(S)
C ***       PX     = EXPONENTIAL SPLINE TENSION PARAMETERS FOR X(S)
C ***       PY     = EXPONENTIAL SPLINE TENSION PARAMETERS FOR Y(S)
C ***       HX     = DELTA S FOR X(S)
C ***       HY     = DELTA S FOR Y(S)
C ***
      SUBROUTINE PAREXP(XP,YP,NP1,XEND1,YEND1,XEND2,YEND2,EPS,S,BX,BY,
     1TAUDPX,TAUDPY,PX,PY,HX,HY)
      DIMENSION XP(1),YP(1),S(1),BX(1),BY(1),TAUDPX(1),TAUDPY(1),
     1PX(1),PY(1),HX(1),HY(1),SN(100)
C ***
C *** COMPUTE CHORD LENGTH
C ***
      S(1)=0.
      DO 10 I=2,NP1
   10 S(I)=S(I-1)+SQRT((XP(I)-XP(I-1))**2+(YP(I)-YP(I-1))**2)
   15 CONTINUE
C ***
C *** FIT X VS. ARC LENGTH
C ***
      CALL EXPSPL(S,XP,XEND1,XEND2,NP1,BX,TAUDPX,PX,HX,ALPHAX)
C ***
C *** FIT Y VS. ARC LENGTH
C ***
      CALL EXPSPL(S,YP,YEND1,YEND2,NP1,BY,TAUDPY,PY,HY,ALPHAY)
C ***
C *** PLOT Y VS. X
C ***
      CALL PLTFIT(S,XP,YP,NP1,BX,BY,TAUDPX,TAUDPY,PX,PY,HX,HY,
     1ALPHAX,ALPHAY)
C ***
C *** RECALCULATE ARC LENGTH
C ***
      CALL ARCLEN(S,NP1,BX,BY,TAUDPX,TAUDPY,PX,PY,HX,HY,ALPHAX,ALPHAY,
     1SN,XP,YP)
C ***
C *** DETERMINE SIGNFICANCE OF RELATIVE CHANGE IN ARC LENGTH
C ***
      DO 30 I=2,NP1
      IF(ABS((SN(I)-S(I))/(S(I)-S(I-1))).GE.EPS) GO TO 31
   30 CONTINUE
      GO TO 35
   31 DO 32 I=2,NP1
   32 S(I)=SN(I)
      GO TO 15
   35 CONTINUE
      RETURN
      END
C ***
C ***

C *** ARCLEN: B. J. MCCARTIN  10/80
C ***
C ***
C *** FUNCTION: APPROXIMATE ARC LENGTH INTEGRAL USING
C ***           COMPOUND SIMPSON'S RULE
C ***
```

```
C ***  DESCRIPTION OF CALLING ARGUMENTS:
C ***       S       = INPUT ARC LENGTH
C ***       NP1     = NUMBER OF ORDERED TRIPLETS OF DATA
C ***       BX      = RIGHT HAND SIDE OF SPLINE EQUATIONS FOR X(S)
C ***       BY      = RIGHT HAND SIDE OF SPLINE EQUATIONS FOR Y(S)
C ***       TAUDPX  = SOLUTION OF SPLINE EQUATIONS FOR X(S)
C ***       TAUDPY  = SOLUTION OF SPLINE EQUATIONS FOR Y(S)
C ***       PX      = EXPONENTIAL SPLINE TENSION PARAMETERS FOR X(S)
C ***       PY      = EXPONENTIAL SPLINE TENSION PARAMETERS FOR Y(S)
C ***       HX      = DELTA S X(S)
C ***       HY      = DELTA S FOR Y(S)
C ***       ALPHAX  = SCALING PARAMETER FOR X(S)
C ***       ALPHAY  = SCALING PARAMETER FOR Y(S)
C ***       SN      = OUTPUT ARC LENGTH
C ***       XP      = ABSCISSAE
C ***       YP      = ORDINATES
C ***
      SUBROUTINE ARCLEN(S,NP1,BX,BY,TAUDPX,TAUDPY,PX,PY,HX,HY,
     1ALPHAX,ALPHAY,SN,XP,YP)
      DIMENSION S(1),BX(1),BY(1),TAUDPX(1),TAUDPY(1),PX(1),PY(1),
     1HX(1),HY(1),SN(1),F(11),XP(1),YP(1)
C ***
C *** CALCULATE ARC LENGTH
C ***
      SN(1)=0.
      DO 10 K=2,NP1
      DS=S(K)-S(K-1)
      H=DS/10.
      DO 5 L=1,11
      S1=(L-1)*H + S(K-1)
      CALL ESEVAL(S,XP,NP1,BX,TAUDPX,PX,HX,S1,X1,DERIVX,ALPHAX)
      CALL ESEVAL(S,YP,NP1,BY,TAUDPY,PY,HY,S1,Y1,DERIVY,ALPHAY)
    5 F(L)=SQRT(DERIVX**2+DERIVY**2)
   10 SN(K)=SN(K-1)+(F(1)+4.*(F(2)+F(4)+F(6)+F(8)+F(10))
     1+2.*(F(3)+F(5)+F(7)+F(9))+F(11))*H/3.
      RETURN
      END
C ***
C ***
C *** PLTFIT: B. J. MCCARTIN  10/80
C ***
C ***
C *** FUNCTION: PLOTS THE PARAMETRIC EXPONENTIAL SPLINE
C ***
C *** DESCRIPTION OF CALLING ARGUMENTS:
C ***       S       = ARC LENGTH
C ***       XP      = ABSCISSAE
C ***       YP      = ORDINATES
C ***       NP1     = NUMBER OF ORDERED TRIPLETS OF DATA
C ***       BX      = RIGHT HAND SIDE OF SPLINE EQUATIONS FOR X(S)
C ***       BY      = RIGHT HAND SIDE OF SPLINE EQUATIONS FOR Y(S)
C ***       TAUDPX  = SOLUTION OF SPLINE EQUATIONS FOR X(S)
C ***       TAUDPY  = SOLUTION OF SPLINE EQUATIONS FOR Y(S)
C ***       PX      = EXPONENTIAL SPLINE TENSION PARAMETERS FOR X(S)
C ***       PY      = EXPONENTIAL SPLINE TENSION PARAMETERS FOR Y(S)
C ***       HX      = DELTA S FOR X(S)
C ***       HY      = DELTA S FOR Y(S)
C ***       ALPHAX  = SCALING PARAMETER FOR X(S)
C ***       ALPHAY  = SCALING PARAMETER FOR Y(S)
C ***
```

```
      SUBROUTINE PLTFIT(S,XP,YP,NP1,BX,BY,TAUDPX,TAUDPY,PX,PY,HX,HY,
     1ALPHAX,ALPHAY)
      DIMENSION S(1),XP(1),YP(1),BX(1),BY(1),TAUDPX(1),TAUDPY(1),
     1PX(1),PY(1),HX(1),HY(1),XPLT(11000),YPLT(11000),
     2IPSTRT(2),IPNUM(2),IPSYM(2),IPINCR(2),TITLE(60),
     3XLBL(5),YLBL(5),IPLINE(2),Z(2)
      DATA XLBL/5*'     '/
      DATA YLBL/5*'     '/
      DATA BLANK/'     '/
      DATA TITLE/'*   ','PARA','METR','IC E','XPON','ENTI',
     1'AL S','PLIN','E   ',48*'     '/
      Z(1)=BLANK
      Z(2)=BLANK
      NZ=2
      IPSTRT(1)=1
      IPSTRT(2)=NP1+1
      IPNUM(1)=NP1
      IPNUM(2)=(NP1-1)*100+1
      IPLINE(1)=-1
      IPLINE(2)=4
      IPSYM(1)=1
      IPSYM(2)=0
      IPINCR(1)=1
      IPINCR(2)=1
      DO 10 I=1,NP1
      XPLT(I)=XP(I)
      YPLT(I)=YP(I)
   10 CONTINUE
      M=NP1+1
      NP=NP1-1
      DO 20 K=1,NP
      DS=S(K+1)-S(K)
      H=DS/100.
      DO 25 L=1,100
      S1=S(K)+(L-1)*H
      CALL ESEVAL(S,XP,NP1,BX,TAUDPX,PX,HX,S1,XPLT(M),DERIVX,ALPHAX)
      CALL ESEVAL(S,YP,NP1,BY,TAUDPY,PY,HY,S1,YPLT(M),DERIVY,ALPHAY)
   25 M=M+1
   20 CONTINUE
      XPLT(M)=XP(NP1)
      YPLT(M)=YP(NP1)
      DO 30 I=1,M
   30 CONTINUE
      CALL PLTEK(XPLT,YPLT,Z,NZ,IPSTRT,IPNUM,IPLINE,
     1IPSYM,IPINCR,TITLE,XLBL,YLBL,DUM)
      RETURN
      END
C ***
C ***
C *** EXPSPL: B. J. MCCARTIN  10/80
C ***
C ***
C *** FUNCTION: CONSTRUCTS THE INTERPOLATORY EXPONENTIAL SPLINE
C ***           TO A SET OF POINTS IN THE PLANE WITH MONOTONICALLY
C ***           INCREASING ABSCISSAE USING SPECIFIED DERIVATIVE
C ***           END CONDITIONS
C ***
C *** REFERENCE: B. J. MCCARTIN, NUMERICAL COMPUTATION OF
C ***             EXPONENTIAL SPLINES, COURANT MATHEMATICS
C ***             AND COMPUTING LABORATORY, OCTOBER 1980
C ***
```

```
C *** DESCRIPTION OF CALLING ARGUMENTS:
C ***      X       = ABSCISSAE
C ***      F       = ORDINATES
C ***      FPA     = LEFT HAND DERIVATIVE END CONDITION
C ***      FPB     = RIGHT HAND DERIVATIVE END CONDITION
C ***      NP1     = NUMBER OF ORDERED PAIRS OF DATA
C ***      B       = RIGHT HAND SIDE OF SPLINE EQUATIONS
C ***      TAUDP.  = SOLUTION OF SPLINE EQUATIONS
C ***      P       = EXPONENTIAL SPLINE TENSION PARAMETERS
C ***      H       = DELTA X
C ***      ALPHA   = SCALING PARAMETER
C ***
      SUBROUTINE EXPSPL(X,F,FPA,FPB,NP1,B,TAUDP,P,H,ALPHA)
      DIMENSION X(1),F(1),B(1),TAUDP(1),P(1),H(1)
      DIMENSION E(100),DIAG(100),ILIML(100),ILIMU(100),Q(100),U(100)
      REAL LAMDAR
C ***
C *** INITIALIZATION
C ***
      IOUT=1
      IPLOT=0
      CONST1=1./6.
      CONST2=1./3.
      CONST3=-7./60.
      CONST4=31./2520.
      CONST5=-1./15.
      CONST6=2./315.
      N=NP1-1
      EPS=1.E-6
      OMEGA=1.
      ETA=.07
      SIGMA=100.
      ALPHA=1.
      ITMAX=5
      SCALE=2./(X(NP1)-X(1))
      IF(IOUT.NE.0) WRITE(16,2) ITMAX
    2 FORMAT(/,' ITMAX = ',I5)
      IF(IOUT.NE.0) WRITE(16,3) EPS,OMEGA,ETA,SIGMA
    3 FORMAT(/,' EPS = ',E15.7,5X,'OMEGA = ',E15.7,5X,'ETA = ',E15.7,5X,
     1'SIGMA = ',E15.7)
      ICOUNT=0
      DO 4 I=1,N
      H(I)=X(I+1)-X(I)
    4 H(I)=SCALE*H(I)
      DO 5 I=1,N
    5 P(I)=0.
      IF(IOUT.NE.0) WRITE(16,6)
    6 FORMAT(/,'      I         H')
      IF(IOUT.NE.0) WRITE(16,7) (I,H(I),I=1,N)
    7 FORMAT(1X,I5,E15.7)
C ***
C *** DEFINE B'S
C ***
      B(1)=(F(2)-F(1))/H(1)-FPA/SCALE
      IF(N.EQ.1) GO TO 11
      DO 10 I=2,N
   10 B(I)=(F(I+1)-F(I))/H(I)-(F(I)-F(I-1))/H(I-1)
   11 B(NP1)=FPB/SCALE-(F(NP1)-F(N))/H(N)
      IF(IOUT.NE.0) WRITE(16,12)
   12 FORMAT(/,'      I         B')
      IF(IOUT.NE.0) WRITE(16,13) (I,B(I),I=1,NP1)
   13 FORMAT(1X,I5,E15.7)
```

```
C ***
C *** DETERMINE WHICH INTERVALS SHOULD BE FIT WITH
C *** LINE SEGMENTS AND WHICH INTERVALS SHOULD BE
C *** FIT WITH EXPONENTIAL SPLINES
C ***    K         = NUMBER OF EXPONENTIAL SPLINE INTERVALS
C ***    ILIML(I) = LEFT HAND ENDPOINT OF ITH SPLINE INTERVAL
C ***    ILIMU(I) = RIGHT HAND ENDPOINT OF ITH SPLINE INTERVAL
C ***
      K=0
      IL=1
   14 CONTINUE
      DO 20 I=IL,N
      TEMP=B(I)*B(I+1)
      IF(TEMP.EQ.0.) GO TO 20
      K=K+1
      ILIML(K)=I
      DO 16 J=I,N
      TEMP=B(J)*B(J+1)
      IF(TEMP.NE.0.) GO TO 15
      ILIMU(K)=J
      GO TO 29
   15 IF(J.NE.N) GO TO 16
      ILIMU(K)=N+1
      GO TO 30
   16 CONTINUE
   20 CONTINUE
      GO TO 30
   29 IL=J
      GO TO 14
   30 CONTINUE
      IF(IOUT.NE.0) WRITE(16,31) K
   31 FORMAT(/,' K = ',I3)
      IF(K.EQ.0) GO TO 54
      IF(IOUT.NE.0) WRITE(16,32)
   32 FORMAT(1X,'   I  ILIML ILIMU')
      IF(IOUT.NE.0) WRITE(16,33) (I,ILIML(I),ILIMU(I),I=1,K)
   33 FORMAT(I5,I5,1X,I5)
C ***
C *** SET UP SPLINE EQUATIONS
C ***
   34 DIM1=0.
      ICOUNT=ICOUNT+1
      IF(IOUT.NE.0) WRITE(16,1000)
      IF(IOUT.NE.0) WRITE(16,36) ICOUNT
   36 FORMAT(1X,'ICOUNT = ',I5)
      IF(IOUT.NE.0) WRITE(16,37)
   37 FORMAT(/,'     I         P              H')
      IF(IOUT.NE.0) WRITE(16,38) (I,P(I),H(I),I=1,N)
   38 FORMAT(1X,I5,E15.7,E15.7)
      DO 40 I=1,N
      PI=P(I)
      HI=H(I)
      PIHI=PI*HI
      PI2=PI*PI
      IF(PIHI.GT.ETA) GO TO 39
C *** USE POWER SERIES REPRESENTATION
      PI2HI2=PI2*HI*HI
      PI4HI4=PI2HI2*PI2HI2
      E(I)=(1.+CONST3*PI2HI2+CONST4*PI4HI4)*HI*CONST1
      DI=(1.+CONST5*PI2HI2+CONST6*PI4HI4)*HI*CONST2
      DIAG(I)=DIM1+DI
      GO TO 40
```

```
C *** USE HYPERBOLIC FUNCTIONS
   39 DO 40 I=1,NI
      SI=SINH(PIHI)
      CI=COSH(PIHI)
      PIDSI=PI/SI
      E(I)=(CNB*HI-PID*SI)/PI2
      DI=(PID*SI*(CI-CNB*HI)/PI2
      DIAG(I)=DIM1+DI
   40 DIM1=DI
      DIAG(NP1)=DI
      E(NP1)=0.
      IF(IOUT.NE.0) WRITE(16,41)
   41 FORMAT(/,'      I         E')
      IF(IOUT.NE.0) WRITE(16,42) (I,E(I),I=1,N)
   42 FORMAT(1X,I5,E15.7)
      IF(IOUT.NE.0) WRITE(16,43)
   43 FORMAT(/,'      I       DIAG')
      IF(IOUT.NE.0) WRITE(16,44) (I,DIAG(I),I=1,NP1)
   44 FORMAT(1X,I5,E15.7)
C ***
C *** ALTER SPLINE EQUATIONS
C ***
      IF(B(1).EQ.0.) E(1)=0.
      IF(N.EQ.1) GO TO 46
      DO 45 I=2,N
      IF(B(I).NE.0.) GO TO 45
      E(I-1)=0.
      E(I)=0.
   45 CONTINUE
   46 IF(B(NP1).EQ.0.) E(N)=0.
      IF(IOUT.NE.0) WRITE(16,47)
   47 FORMAT(/,'      I         E')
      IF(IOUT.NE.0) WRITE(16,48) (I,E(I),I=1,N)
   48 FORMAT(1X,I5,E15.7)
C ***
C *** SOLVE SPLINE EQUATIONS (SEE AHLBERG, NILSON AND WALSH)
C ***
      PK=DIAG(1)
      Q(1)=-E(1)/PK
      U(1)=B(1)/PK
      DO 49 KK=2,NP1
      PK=E(KK-1)*Q(KK-1)+DIAG(KK)
      Q(KK)=-E(KK)/PK
   49 U(KK)=(B(KK)-E(KK-1)*U(KK-1))/PK
      TAUDP(NP1)=U(NP1)
      DO 51 L=1,N
      KK=NP1-L
   51 TAUDP(KK)=Q(KK)*TAUDP(KK+1)+U(KK)
      IF(IOUT.NE.0) WRITE(16,52)
   52 FORMAT(/,'      I       TAUDP              B')
      IF(IOUT.NE.0) WRITE(16,53) (I,TAUDP(I),B(I),I=1,NP1)
   53 FORMAT(1X,I5,E15.7,E15.7)
C ***
C *** PLOT EXPONENTIAL SPLINE
C ***
   54 IF(IPLOT.NE.0) CALL ESPLOT(X,F,NP1,B,TAUDP,P,N,ALPHA)
      IF(K.EQ.0) GO TO 99
      IF(ICOUNT.GE.ITMAX) GO TO 2000
C ***
C *** UPDATE P'S
C ***
      IFLAG=0
      DO 60 I=1,K
      IL=ILINL(I)
      IU=ILINU(I)
      TEMP=TAUDP(IL)*B(IL)
      IF(TEMP.GE.0.) GO TO 64
```

```
C *** B*TAUDP<0 AT LEFT HAND ENDPOINT
      IFLAG=1
      LAMBAR=AMAX1(ABS(B(IL)),DIAG(IL)*ABS(TAUDP(IL)))/ABS(TAUDP(IL+1))
      PTILDA=1./SQRT(LAMBAR*H(IL))
      PTILDA=AMAX1(PTILDA,P(IL))
      P(IL)=P(IL)+OMEGA*(PTILDA-P(IL))
   64 ILP1=IL+1
      IUM1=IU-1
      IF(IUM1.LT.ILP1) GO TO 70
      DO 69 J=ILP1,IUM1
      IF(TAUDP(J).NE.0.) GO TO 66
C *** TAUDP=0 AT AN INTERIOR POINT
      IFLAG=1
      P(J-1)=P(J-1)+EPS
      P(J)=P(J)+EPS
      GO TO 68
   66 TEMP=TAUDP(J)*B(J)
      IF(TEMP.GT.0.) GO TO 68
C *** B*TAUDP<0 AT AN INTERIOR POINT
      IFLAG=1
      LAMBAR=AMAX1(ABS(B(J)),DIAG(J)*ABS(TAUDP(J)))
     1/(2.*AMAX1(ABS(TAUDP(J-1)),ABS(TAUDP(J+1))))
      PTILDA=1./SQRT(LAMBAR*H(J-1))
      PTILDA=AMAX1(PTILDA,P(J-1))
      P(J-1)=P(J-1)+OMEGA*(PTILDA-P(J-1))
      PTILDA=1./SQRT(LAMBAR*H(J))
      PTILDA=AMAX1(PTILDA,P(J))
      P(J)=P(J)+OMEGA*(PTILDA-P(J))
   68 CONTINUE
   70 TEMP=TAUDP(IU)*B(IU)
      IF(TEMP.GE.0.) GO TO 80
C *** B*TAUDP<0 AT RIGHT HAND ENDPOINT
      IFLAG=1
      LAMBAR=AMAX1(ABS(B(IU)),DIAG(IU)*ABS(TAUDP(IU)))/ABS(TAUDP(IUM1))
      PTILDA=1./SQRT(LAMBAR*H(IUM1))
      PTILDA=AMAX1(PTILDA,P(IUM1))
      P(IUM1)=P(IUM1)+OMEGA*(PTILDA-P(IUM1))
   80 CONTINUE
C ***
C *** CHECK FOR EXTRANEOUS INFLECTION POINTS
C ***
      IF(IFLAG.EQ.0) GO TO 99
C ***
C *** SCALE ABSCISSAE
C ***
      XMU=0.
      DO 93 I=1,N
   93 XMU=AMAX1(XMU,P(I)*H(I))
      ALPHAC=AMAX1(XMU/SIGMA,1.)
      ALPHA=ALPHA*ALPHAC
      DO 94 I=1,N
      H(I)=H(I)/ALPHAC
   94 B(I)=B(I)*ALPHAC
      B(NP1)=B(NP1)*ALPHAC
      IF(IOUT.NE.0) WRITE(16,95) XMU,ALPHAC,ALPHA
   95 FORMAT(/,'XMU = ',E15.7,5X,'ALPHAC = ',E15.7,5X,'ALPHA = ',E15.7)
      IF(IOUT.NE.0) WRITE(16,96)
   96 FORMAT(/,'     I          H                B')
      IF(IOUT.NE.0) WRITE(16,97) (I,H(I),B(I),I=1,N)
   97 FORMAT(1X,I5,2E15.7)
      IF(IOUT.NE.0) WRITE(16,98) NP1,B(NP1)
   98 FORMAT(1X,I5,15X,E15.7)
      GO TO 34
```

```
C ***
C *** NO EXTRANEOUS INFLECTION POINTS
C ***
   99 IF(IOUT.NE.0) WRITE(16,1000)
      IF(IOUT.NE.0) WRITE(16,100) ICOUNT
  100 FORMAT(' NO EXTRANEOUS INFLECTION POINTS: ICOUNT = ',I3)
 1000 FORMAT(///,'----------------------------------------',///)
      GO TO 3000
C ***
C *** MAXIMUM NUMBER OF ITERATIONS EXCEEDED
C ***
 2000 IF(IOUT.NE.0) WRITE(16,1000)
      IF(IOUT.NE.0) WRITE(16,2001)
 2001 FORMAT(' MAXIMUM NUMBER OF ITERATIONS EXCEEDED')
 3000 RETURN
      END
C ***
C ***
C *** ESPLOT: B. J. MCCARTIN  10/80
C ***
C ***
C *** FUNCTION: PLOTS THE EXPONENTIAL SPLINE
C ***
C *** DESCRIPTION OF CALLING ARGUMENTS:
C ***      X       = ABSCISSAE
C ***      F       = ORDINATES
C ***      NP1     = NUMBER OF ORDERED PAIRS OF DATA
C ***      B       = RIGHT HAND SIDE OF SPLINE EQUATIONS
C ***      TAUDP   = SOLUTION OF SPLINE EQUATIONS
C ***      P       = EXPONENTIAL SPLINE TENSION PARAMETERS
C ***      H       = DELTA X
C ***      ALPHA   = SCALING PARAMETER
C ***
      SUBROUTINE ESPLOT(X,F,NP1,B,TAUDP,P,H,ALPHA)
      DIMENSION X(1),F(1),B(1),TAUDP(1),P(1),H(1)
      DIMENSION XP(10000),FP(10000),Z(2),IPNUM(2),IPLINE(2),IPSYM(2),
     1IPINCR(2),ITITLE(60),XLBL(5),YLBL(5),IPSTRT(2)
      DATA BLANK/'     '/
      DATA XLBL/5*'     '/
      DATA YLBL/5*'     '/
      DATA ITITLE/5*'     ','EXPO','NENT','IAL ','SPLI','NE  ',50*'     '/
C ***
C *** INITIALIZATION
C ***
      IOUT=0
      N=NP1-1
      DO 1 M=1,2
    1 Z(M)=BLANK
      SCALE=2./(X(NP1)-X(1))
C ***
C *** SET UP ARRAYS
C ***
      K=0
      IF(IOUT.NE.0) WRITE(16,38)
   38 FORMAT(/,1X,'     K       XBAR           TAUBAR          TAUBP')
      DO 45 I=1,N
      DX=ALPHA*H(I)/99.
      DX=DX/SCALE
      XX=X(I)-DX
      DO 40 J=1,100
```

```
      K=K+1
      XP(K)=XX+DX
      IF(XP(K).LT.X(I)) XP(K)=X(I)
      IF(XP(K).GT.X(I+1)) XP(K)=X(I+1)
      XX=XP(K)
      CALL ESEVAL(X,F,NP1,B,TAUDP,P,H,XP(K),FP(K),TAUBP,ALPHA)
      IF(IOUT.NE.0) WRITE(10,39) K,XP(K),FP(K),TAUBP
   39 FORMAT(1X,I5,3E15.7)
   40 CONTINUE
   45 CONTINUE
      DO 50 I=1,NP1
      J=K+I
      XP(J)=X(I)
   50 FP(J)=F(I)
      NZ=2
      IPSTRT(1)=1
      IPSTRT(2)=K+1
      IPNUM(1)=K
      IPNUM(2)=NP1
      IPLINE(1)=4
      IPLINE(2)=-1
      IPSYM(1)=0
      IPSYM(2)=1
      IPINCR(1)=1
      IPINCR(2)=1
C ***
C *** PLOT
C ***
      CALL PLTEK(XP,FP,Z,NZ,IPSTRT,IPNUM,IPLINE,IPSYM,IPINCR,ITITLE,
     1XLBL,YLBL,DUM)
      RETURN
      END
C ***
C ***
C *** ESEVAL: B. J. MCCARTIN  10/80
C ***
C ***
C *** FUNCTION: EVALUATES THE EXPONENTIAL SPLINE
C ***
C *** DESCRIPTION OF CALLING ARGUMENTS:
C ***      X       = ABSCISSAE
C ***      F       = ORDINATES
C ***      NP1     = NUMBER OF ORDERED PAIRS OF DATA
C ***      B       = RIGHT HAND SIDE OF SPLINE EQUATIONS
C ***      TAUDP   = SOLUTION OF SPLINE EQUATIONS
C ***      P       = EXPONENTIAL SPLINE TENSION PARAMETERS
C ***      H       = DELTA X
C ***      XBAR    = ABSCISSA OF EVALUATION
C ***      TAUBAR  = ORDINATE OF EVALUATION
C ***      TAUBP   = DERIVATIVE OF EVALUATION
C ***      ALPHA   = SCALING PARAMETER
C ***
      SUBROUTINE ESEVAL(X,F,NP1,B,TAUDP,P,H,XBAR,TAUBAR,TAUBP,ALPHA)
      DIMENSION X(1),F(1),B(1),TAUDP(1),P(1),H(1)
C ***
C *** INITIALIZATION
C ***
      CONST1=1./6.
      CONST2=1./3.
      CONST3=-7./60.
      CONST4=31./2520.
      CONST5=-1./15.
```

```
      CONST6=2./315.
      CONST7=-7./195.
      CONST8=-.05
      CONST9=1./120.
      N=NP1-1
      ETA=.07
      SCALE=2./(X(NP1)-X(1))
C ***
C *** DETERMINE INTERVAL
C ***
      DO 10 J=1,N
      I=J
      IF(XBAR.LT.X(J+1)) GO TO 11
   10 CONTINUE
C ***
C *** DETERMINE TYPE OF FIT
C ***
   11 TEMP=B(I)*B(I+1)
      IF(TEMP.NE.0.) GO TO 20
C *** FIT WITH LINEAR FUNCTION
      TAUBAR=(F(I)*(X(I+1)-XBAR)+F(I+1)*(XBAR-X(I)))/(H(I)*ALPHA)
      TAUBAR=SCALE*TAUBAR
      TAUBP=(F(I+1)-F(I))/(H(I)*ALPHA)
      TAUBP=SCALE*TAUBP
      GO TO 30
C *** FIT WITH EXPONENTIAL SPLINE
   20 X1=SCALE*(X(I+1)-XBAR)/ALPHA
      X2=SCALE*(XBAR-X(I))/ALPHA
      PI2=P(I)*P(I)
      PIHI=P(I)*H(I)
      IF(PIHI.GT.ETA) GO TO 25
C *** USE POWER SERIES REPRESENTATION
      PI2HI2=PIHI*PIHI
      PI4HI4=PI2HI2*PI2HI2
      X12=X1*X1
      X22=X2*X2
      X14=X12*X12
      X24=X22*X22
      ONEHI2=1./(H(I)*H(I))
      TERM1=1.+CONST3*PI2HI2+CONST4*PI4HI4
      TERM2=CONST1*PI2-ONEHI2+CONST7*PI2*PI2HI2
      TERM3=(CONST8*ONEHI2+CONST9*PI2)*PI2
      TAUBAR=(F(I)*X1+F(I+1)*X2)/H(I)-CONST1*H(I)*
     1(X1*TAUDP(I)*(TERM1+TERM2*X12+TERM3*X14)+
     2X2*TAUDP(I+1)*(TERM1+TERM2*X22+TERM3*X24))
      TAUBP=(F(I+1)-F(I))/H(I)-CONST1*H(I)*
     1(TAUDP(I+1)*(TERM1+3.*TERM2*X22+5.*TERM3*X24)-
     2TAUDP(I)*(TERM1+3.*TERM2*X12+5.*TERM3*X14))
      TAUBP=SCALE*TAUBP/ALPHA
      GO TO 30
C *** USE HYPERBOLIC FUNCTIONS
   25 DENOM=PI2*SINH(P(I)*H(I))
      TAUBAR=(TAUDP(I)*SINH(P(I)*X1)+TAUDP(I+1)*SINH(P(I)*X2))/DENOM
     1+((F(I)-TAUDP(I)/PI2)*X1+(F(I+1)-TAUDP(I+1)/PI2)*X2)/H(I)
      TAUBP=(TAUDP(I+1)*COSH(P(I)*X2)-TAUDP(I)*COSH(P(I)*X1))*P(I)/DENOM
     1+(F(I+1)-F(I)+(TAUDP(I)-TAUDP(I+1))/PI2)/H(I)
      TAUBP=SCALE*TAUBP/ALPHA
   30 CONTINUE
      RETURN
      END
```

```
C ***
C ***
C *** DRIVER FOR PERIODIC PARAMETRIC EXPONENTIAL SPLINE PACKAGE:
C *** B. J. MCCARTIN  10/80
C ***
C ***
      DIMENSION XP(100),YP(100),S(100),BX(100),BY(100),TAUDPX(100),
     1TAUDPY(100),PX(100),PY(100),HX(100),HY(100)
      CALL ERRSET(207,256,-1,1,0)
      CALL ERRSET(208,256,-1,1,0)
      CALL TKINIT
      READ(15,10) NP1
   10 FORMAT(I3)
      WRITE(16,20) NP1
   20 FORMAT(1X,'NP1 = ',I3)
      WRITE(16,30)
   30 FORMAT('         XP              YP')
      DO 60 I=1,NP1
      READ(15,40) XP(I),YP(I)
   40 FORMAT(2E15.7)
      WRITE(16,50) XP(I),YP(I)
   50 FORMAT(1X,2E15.7)
   60 CONTINUE
      READ(15,70) EPS
   70 FORMAT(E15.7)
      WRITE(16,80) EPS
   80 FORMAT(1X,'EPS = ',E15.7)
      CALL PEREXP(XP,YP,NP1,EPS,S,BX,BY,TAUDPX,TAUDPY,PX,PY,HX,HY)
      RETURN
      END
C ***
C ***
C *** PEREXP: B. J. MCCARTIN  10/80
C ***
C ***
C *** FUNCTION: CONSTRUCTS THE PARAMETRIC EXPONENTIAL SPLINE
C ***           INTERPOLANT TO A SET OF POINTS IN THE PLANE
C ***           USING PERIODIC END CONDITIONS
C ***           NOTE: ARC LENGTH IS USED AS THE PARAMETER
C ***
C *** REFERENCE: B. J. MCCARTIN, NUMERICAL COMPUTATION OF
C ***            EXPONENTIAL SPLINES, COURANT MATHEMATICS
C ***            AND COMPUTING LABORATORY, OCTOBER 1980
C ***
C *** DESCRIPTION OF CALLING ARGUMENTS:
C ***      XP      = ABSCISSAE
C ***      YP      = ORDINATES
C ***      NP1     = NUMBER OF ORDERED TRIPLETS OF DATA.
C ***      EPS     = TOLERANCE FOR STOPPING CRITERIA IN
C ***                ARC LENGTH ITERATION
C ***      S       = ARC LENGTH
C ***      BX      = RIGHT HAND SIDE OF SPLINE EQUATIONS FOR X(S)
C ***      BY      = RIGHT HAND SIDE OF SPLINE EQUATIONS FOR Y(S)
C ***      TAUDPX  = SOLUTION OF SPLINE EQUATIONS FOR X(S)
C ***      TAUDPY  = SOLUTION OF SPLINE EQUATIONS FOR Y(S)
C ***      PX      = EXPONENTIAL SPLINE TENSION PARAMETERS FOR X(S)
C ***      PY      = EXPONENTIAL SPLINE TENSION PARAMETERS FOR Y(S)
C ***      HX      = DELTA S FOR X(S)
C ***      HY      = DELTA S FOR Y(S)
C ***
```

```
      SUBROUTINE PEREXPI(XP,YP,NP1,EPS,S,BX,BY,TAUDPX,TAUDPY,PX,PY,HX,HY)
      DIMENSION XP(1),YP(1),S(1),BX(1),BY(1),TAUDPX(1),TAUDPY(1),
     1PX(1),PY(1),HX(1),HY(1),SN(100)
C ***
C *** COMPUTE CHORD LENGTH
C ***
      S(1)=0.
      DO 10 I=2,NP1
   10 S(I)=S(I-1)+SQRT((XP(I)-XP(I-1))**2+(YP(I)-YP(I-1))**2)
   15 CONTINUE
C ***
C *** FIT X VS. ARC LENGTH
C ***
      CALL EXPSPL(S,XP,NP1,BX,TAUDPX,PX,HX,ALPHAX)
C ***
C *** FIT Y VS. ARC LENGTH
C ***
      CALL EXPSPL(S,YP,NP1,BY,TAUDPY,PY,HY,ALPHAY)
C ***
C *** PLOT Y VS. X
C ***
      CALL PLTFIT(S,XP,YP,NP1,BX,BY,TAUDPX,TAUDPY,PX,PY,HX,HY,
     1ALPHAX,ALPHAY)
C ***
C *** RECALCULATE ARC LENGTH
C ***
      CALL ARCLEN(S,NP1,BX,BY,TAUDPX,TAUDPY,PX,PY,HX,HY,ALPHAX,ALPHAY,
     1SN,XP,YP)
C ***
C *** DETERMINE SIGNIFICANCE OF RELATIVE CHANGE IN ARC LENGTH
C ***
      DO 30 I=2,NP1
      IF(ABS((SN(I)-S(I))/(S(I)-S(I-1))).GE.EPS) GO TO 31
   30 CONTINUE
      GO TO 35
   31 DO 32 I=2,NP1
   32 S(I)=SN(I)
      GO TO 15
   35 CONTINUE
      RETURN
      END
C ***
C ***
C *** ARCLEN: B. J. MCCARTIN  10/80
C ***
C ***
C *** FUNCTION: APPROXIMATE ARC LENGTH INTEGRAL USING
C ***           COMPOUND SIMPSON'S RULE
C ***
C *** DESCRIPTION OF CALLING ARGUMENTS:
C ***      S       = INPUT ARC LENGTH
C ***      NP1     = NUMBER OF ORDERED TRIPLETS OF DATA
C ***      BX      = RIGHT HAND SIDE OF SPLINE EQUATIONS FOR X(S)
C ***      BY      = RIGHT HAND SIDE OF SPLINE EQUATIONS FOR Y(S)
C ***      TAUDPX  = SOLUTION OF SPLINE EQUATIONS FOR X(S)
C ***      TAUDPY  = SOLUTION OF SPLINE EQUATIONS FOR Y(S)
C ***      PX      = EXPONENTIAL SPLINE TENSION PARAMETERS FOR X(S)
C ***      PY      = EXPONENTIAL SPLINE TENSION PARAMETERS FOR Y(S)
C ***      HX      = DELTA S X(S)
C ***      HY      = DELTA S FOR Y(S)
C ***      ALPHAX  = SCALING PARAMETER FOR X(S)
C ***      ALPHAY  = SCALING PARAMETER FOR Y(S)
C ***      SN      = OUTPUT ARC LENGTH
C ***      XP      = ABSCISSAE
C ***      YP      = ORDINATES
C ***
```

```
      SUBROUTINE ARCLEN(S,NP1,BX,BY,TAUDPX,TAUDPY,PX,PY,HX,HY,
     1ALPHAX,ALPHAY,SN,XP,YP)
      DIMENSION S(1),BX(1),BY(1),TAUDPX(1),TAUDPY(1),PX(1),PY(1),
     1HX(1),HY(1),SN(1),F(11),XP(1),YP(1)
C ***
C *** CALCULATE ARC LENGTH
C ***
      SN(1)=0.
      DO 10 K=2,NP1
      DS=S(K)-S(K-1)
      H=DS/10.
      DO 5 L=1,11
      S1=(L-1)*H+S(K-1)
      CALL ESEVAL(S,XP,NP1,BX,TAUDPX,PX,HX,S1,X1,DERIVX,ALPHAX)
      CALL ESEVAL(S,YP,NP1,BY,TAUDPY,PY,HY,S1,Y1,DERIVY,ALPHAY)
    5 F(L)=SQRT(DERIVX**2+DERIVY**2)
   10 SN(K)=SN(K-1)+(F(1)+4.*(F(2)+F(4)+F(6)+F(8)+F(10))
     1+2.*(F(3)+F(5)+F(7)+F(9))+F(11))*H/3.
      RETURN
      END
C ***
C ***
C *** PLTFIT: B. J. MCCARTIN  9/80
C ***
C ***
C *** FUNCTION: PLOTS THE PARAMETRIC EXPONENTIAL SPLINE
C ***
C *** DESCRIPTION OF CALLING ARGUMENTS:
C ***      S       = ARC LENGTH
C ***      XP      = ABSCISSAE
C ***      YP      = ORDINATES
C ***      NP1     = NUMBER OF ORDERED TRIPLETS OF DATA
C ***      BX      = RIGHT HAND SIDE OF SPLINE EQUATIONS FOR X(S)
C ***      BY      = RIGHT HAND SIDE OF SPLINE EQUATIONS FOR Y(S)
C ***      TAUDPX  = SOLUTION OF SPLINE EQUATIONS FOR X(S)
C ***      TAUDPY  = SOLUTION OF SPLINE EQUATIONS FOR Y(S)
C ***      PX      = EXPONENTIAL SPLINE TENSION PARAMETERS FOR X(S)
C ***      PY      = EXPONENTIAL SPLINE TENSION PARAMETERS FOR Y(S)
C ***      HX      = DELTA S FOR X(S)
C ***      HY      = DELTA S FOR Y(S)
C ***      ALPHAX  = SCALING PARAMETER FOR X(S)
C ***      ALPHAY  = SCALING PARAMETER FOR Y(S)
C ***
      SUBROUTINE PLTFIT(S,XP,YP,NP1,BX,BY,TAUDPX,TAUDPY,PX,PY,HX,HY,
     1ALPHAX,ALPHAY)
      DIMENSION S(1),XP(1),YP(1),BX(1),BY(1),TAUDPX(1),TAUDPY(1),
     1PX(1),PY(1),HX(1),HY(1),XPLT(11000),YPLT(11000),
     2IPSTRT(2),IPNUM(2),IPSYM(2),IPINCR(2),TITLE(60),
     3XLBL(5),YLBL(5),IPLINE(2),Z(2)
      DATA XLBL/5*'    '/
      DATA YLBL/5*'    '/
      DATA BLANK/'    '/
      DATA TITLE/4*'    ','PARA','METR','IC E','XPON','ENTI',
     1'AL S','PLIN','E   ',48*'    '/
      Z(1)=BLANK
      Z(2)=BLANK
      NZ=2
      IPSTRT(1)=1
      IPSTRT(2)=NP1+1
      IPNUM(1)=NP1
      IPNUM(2)=(NP1-1)*100+1
      IPLINE(1)=-1
      IPLINE(2)=4
      IPSYM(1)=1
      IPSYM(2)=0
      IPINCR(1)=1
      IPINCR(2)=1
```

```
      DO 10 I=1,NP1
      XPLT(I)=XP(I)
      YPLT(I)=YP(I)
   10 CONTINUE
      M=NP1+1
      NP=NP1-1
      DO 20 K=1,NP
      DS=S(K+1)-S(K)
      H=DS/100.
      DO 25 L=1,100
      SI=S(K)+(L-1)*H
      CALL ESEVAL(S,XP,NP1,BX,TAUDPX,PX,HX,SI,XPLT(M),DERIVX,ALPHAX)
      CALL ESEVAL(S,YP,NP1,BY,TAUDPY,PY,HY,SI,YPLT(M),DERIVY,ALPHAY)
   25 M=M+1
   20 CONTINUE
      XPLT(M)=XP(NP1)
      YPLT(M)=YP(NP1)
      DO 30 I=1,M
   30 CONTINUE
      CALL PLTER(XPLT,YPLT,Z,NZ,IPSTRT,IPNUM,IPLINE,
     1IPSTM,IPINCR,TITLE,XLBL,YLBL,DUM)
      RETURN
      END
C ***
C ***
C *** EXPSPL: B. J. MCCARTIN  10/80
C ***
C ***
C *** FUNCTION: CONSTRUCTS THE INTERPOLATORY EXPONENTIAL SPLINE
C ***           TO A SET OF POINTS IN THE PLANE WITH MONOTONICALLY
C ***           INCREASING ABSCISSAE USING PERIODIC END CONDITIONS
C ***
C *** REFERENCE: B. J. MCCARTIN, NUMERICAL COMPUTATION OF
C ***             EXPONENTIAL SPLINES, COURANT MATHEMATICS
C ***             AND COMPUTING LABORATORY, OCTOBER 1980
C ***
C *** DESCRIPTION OF CALLING ARGUMENTS:
C ***      X       = ABSCISSAE
C ***      F       = ORDINATES
C ***      NP1     = NUMBER OF ORDERED PAIRS OF DATA
C ***      B       = RIGHT HAND SIDE OF SPLINE EQUATIONS
C ***      TAUDP   = SOLUTION OF SPLINE EQUATIONS
C ***      P       = EXPONENTIAL SPLINE TENSION PARAMETERS
C ***      H       = DELTA X
C ***      ALPHA   = SCALING PARAMETER
C ***
      SUBROUTINE EXPSPL(X,F,NP1,B,TAUDP,P,H,ALPHA)
      DIMENSION X(1),F(1),B(1),TAUDP(1),P(1),H(1)
      DIMENSION E(100),DIAG(100),ILIML(100),ILIMU(100),
     1Q(100),S(100),T(100),U(100),V(100)
      REAL LAMBAR
C ***
C *** INITIALIZATION
C ***
      IOUT=1
      IPLOT=0
      CONST1=1./6.
      CONST2=1./3.
      CONST3=-7./60.
      CONST4=31./2520.
      CONST5=-1./15.
      CONST6=2./315.
      N=NP1-1
      NM1=N-1
      EPS=1.E-6
      ONE=1.
      BETA=.07
      SIGMA=100.
```

```
      ALPHA=1.
      ITMAX=5
      SCALE=0./(X(NP1)-X(1))
      IF(IOUT.NE.0) WRITE(16,2) ITMAX
    2 FORMAT(/,' ITMAX = ',I5)
      IF(IOUT.NE.0) WRITE(16,3) EPS,ONEGA,ETA,SIGMA
    3 FORMAT(/,' EPS = ',E15.7,5X,'ONEGA = ',E15.7,5X,'ETA = ',E15.7,5X,
     1'SIGMA = ',E15.7)
      ICOUNT=0
      DO 4 I=1,N
      H(I)=X(I+1)-X(I)
    4 H(I)=SCALE*H(I)
      DO 5 I=1,N
    5 F(I)=0.
      IF(IOUT.NE.0) WRITE(16,6)
    6 FORMAT(/,'      I         H')
      IF(IOUT.NE.0) WRITE(16,7) (I,H(I),I=1,N)
    7 FORMAT(1X,I5,E15.7)
C ***
C *** DEFINE B'S
C ***
      B(1)=(F(2)-F(1))/H(1)-(F(NP1)-F(N))/H(N)
      IF(N.EQ.1) GO TO 11
      DO 10 I=2,N
   10 B(I)=(F(I+1)-F(I))/H(I)-(F(I)-F(I-1))/H(I-1)
   11 B(NP1)=B(1)
      IF(IOUT.NE.0) WRITE(16,12)
   12 FORMAT(/,'      I         B')
      IF(IOUT.NE.0) WRITE(16,13) (I,B(I),I=1,NP1)
   13 FORMAT(1X,I5,E15.7)
C ***
C *** DETERMINE WHICH INTERVALS SHOULD BE FIT WITH
C *** LINE SEGMENTS AND WHICH INTERVALS SHOULD BE
C *** FIT WITH EXPONENTIAL SPLINES
C ***    K         = NUMBER OF EXPONENTIAL SPLINE INTERVALS
C ***    ILIML(I) = LEFT HAND ENDPOINT OF ITH SPLINE INTERVAL
C ***    ILIMU(I) = RIGHT HAND ENDPOINT OF ITH SPLINE INTERVAL
C ***
      K=0
      IL=1
   14 CONTINUE
      DO 20 I=IL,N
      TEMP=B(I)*B(I+1)
      IF(TEMP.EQ.0.) GO TO 20
      K=K+1
      ILIML(K)=I
      DO 16 J=I,N
      TEMP=B(J)*B(J+1)
      IF(TEMP.NE.0.) GO TO 15
      ILIMU(K)=J
      GO TO 29
   15 IF(J.NE.N) GO TO 16
      ILIMU(K)=N
      GO TO 30
   16 CONTINUE
   20 CONTINUE
      GO TO 30
   29 IL=J
      GO TO 14
   30 CONTINUE
      IF(IOUT.NE.0) WRITE(16,31) K
   31 FORMAT(/,' K = ',I3)
      IF(K.EQ.0) GO TO 54
      IF(IOUT.NE.0) WRITE(16,32)
   32 FORMAT(1X,'   I  ILIML ILIMU')
      IF(IOUT.NE.0) WRITE(16,33) (I,ILIML(I),ILIMU(I),I=1,K)
   33 FORMAT(I5,I5,1X,I5)
C ***
C *** SET UP SPLINE EQUATIONS
```

```
C ***
   34 DIM1=0.
      ICOUNT=ICOUNT+1
      IF(IOUT.NE.0) WRITE(16,1000)
      IF(IOUT.NE.0) WRITE(16,36) ICOUNT
   36 FORMAT(1X,'ICOUNT = ',I5)
      IF(IOUT.NE.0) WRITE(16,37)
   37 FORMAT(/,'      I         P                H')
      IF(IOUT.NE.0) WRITE(16,38) (I,P(I),H(I),I=1,N)
   38 FORMAT(1X,I5,E15.7,E15.7)
      DO 40 I=1,N
      PI=P(I)
      HI=H(I)
      PIHI=PI*HI
      PI2=PI*PI
      IF(PIHI.GT.ETA) GO TO 39
C *** USE POWER SERIES REPRESENTATION
      PI2HI2=PI2*HI*HI
      PI4HI4=PI2HI2*PI2HI2
      E(I)=(1.+CONST3*PI2HI2+CONST4*PI4HI4)*HI*CONST1
      DI=(1.+CONST5*PI2HI2+CONST6*PI4HI4)*HI*CONST2
      DIAG(I)=DIM1+DI
      GO TO 40
C *** USE HYPERBOLIC FUNCTIONS
   39 ONBYHI=1./HI
      SI=SINH(PIHI)
      CI=COSH(PIHI)
      PIBYSI=PI/SI
      E(I)=(ONBYHI-PIBYSI)/PI2
      DI=(PIBYSI*CI-ONBYHI)/PI2
      DIAG(I)=DIM1+DI
   40 DIM1=DI
      DIAG(1)=DIAG(1)+DI
      IF(IOUT.NE.0) WRITE(16,41)
   41 FORMAT(/,'      I         E')
      IF(IOUT.NE.0) WRITE(16,42) (I,E(I),I=1,N)
   42 FORMAT(1X,I5,E15.7)
      IF(IOUT.NE.0) WRITE(16,43)
   43 FORMAT(/,'      I      DIAG')
      IF(IOUT.NE.0) WRITE(16,44) (I,DIAG(I),I=1,N)
   44 FORMAT(1X,I5,E15.7)
C ***
C *** ALTER SPLINE EQUATIONS
C ***
      IF(B(1).EQ.0.) E(1)=0.
      IF(B(1).EQ.0.) E(N)=0.
      IF(N.EQ.1) GO TO 46
      DO 45 I=2,N
      IF(B(I).NE.0.) GO TO 45
      E(I-1)=0.
      E(I)=0.
   45 CONTINUE
   46 CONTINUE
      IF(IOUT.NE.0) WRITE(16,47)
   47 FORMAT(/,'      I         E')
      IF(IOUT.NE.0) WRITE(16,48) (I,E(I),I=1,N)
   48 FORMAT(1X,I5,E15.7)
C ***
C *** SOLVE SPLINE EQUATIONS (SEE AHLBERG, NILSON AND WALSH)
C ***
```

```
      PK=DIAG(1)
      Q(1)=-E(1)/PK
      U(1)=B(1)/PK
      S(1)=-E(N)/PK
      DO 49 KK=2,NM1
      PK=E(KK-1)*Q(KK-1)+DIAG(KK)
      Q(KK)=-E(KK)/PK
      U(KK)=(B(KK)-E(KK-1)*U(KK-1))/PK
   49 S(KK)=-E(KK-1)*S(KK-1)/PK
      T(N)=1.
      V(N)=0.
      DO 50 L=1,NM1
      KK=N-L
      T(KK)=Q(KK)*T(KK+1)+S(KK)
   50 V(KK)=Q(KK)*V(KK+1)+U(KK)
      TAUDP(N)=(B(N)-E(N)*V(1)-E(N-1)*V(N-1))/(E(N)*T(1)+E(N-1)*T(N-1)
     1+DIAG(N))
      DO 51 L=1,NM1
      KK=N-L
   51 TAUDP(KK)=Q(KK)*TAUDP(KK+1)+S(KK)*TAUDP(N)+U(KK)
      TAUDP(NP1)=TAUDP(1)
      IF(IOUT.NE.0) WRITE(16,52)
   52 FORMAT(/,'      I       TAUDP             B')
      IF(IOUT.NE.0) WRITE(16,53) (I,TAUDP(I),B(I),I=1,NP1)
   53 FORMAT(1X,I5,E15.7,E15.7)
C ***
C *** PLOT EXPONENTIAL SPLINE
C ***
   54 IF(IPLOT.NE.0) CALL ESPLOT(X,F,NP1,B,TAUDP,P,H,ALPHA)
      IF(K.EQ.0) GO TO 99
      IF(ICOUNT.GE.ITMAX) GO TO 2000
C ***
C *** UPDATE P'S
C ***
      IFLAG=0
      DO 80 I=1,K
      IL=ILIML(I)
      IU=ILIMU(I)
      TEMP=TAUDP(IL)*B(IL)
      IF(TEMP.GT.0.) GO TO 64
      IF(TAUDP(IL).EQ.0..AND.IL.NE.1) GO TO 64
      IF(TAUDP(IL).NE.0.) GO TO 60
C *** TAUDP=0 AT LEFT HAND ENDPOINT (FIRST POINT)
      IFLAG=1
      P(N)=P(N)+EPS
      P(1)=P(1)+EPS
      GO TO 64
   60 IF(IL.NE.1) GO TO 62
C *** B*TAUDP<0 AT LEFT HAND ENDPOINT (FIRST POINT)
      IFLAG=1
      LAMBAR=AMAX1(ABS(B(1)),DIAG(1)*ABS(TAUDP(1)))
     1/(2.*AMAX1(ABS(TAUDP(N)),ABS(TAUDP(2))))
      PTILDA=1./SQRT(LAMBAR*H(N))
      PTILDA=AMAX1(PTILDA,P(N))
      P(N)=P(N)+OMEGA*(PTILDA-P(N))
      PTILDA=1./SQRT(LAMBAR*H(1))
      PTILDA=AMAX1(PTILDA,P(1))
      P(1)=P(1)+OMEGA*(PTILDA-P(1))
      GO TO 64
C *** B*TAUDP<0 AT LEFT HAND ENDPOINT (OTHER THAN FIRST POINT)
   62 IFLAG=1
      LAMBAR=AMAX1(ABS(B(IL)),DIAG(IL)*ABS(TAUDP(IL)))/ABS(TAUDP(IL+1))
      PTILDA=1./SQRT(LAMBAR*H(IL))
      PTILDA=AMAX1(PTILDA,P(IL))
      P(IL)=P(IL)+OMEGA*(PTILDA-P(IL))
```

```
   64 ILP1=IL+1
      IUM1=IU-1
      IF(IUM1.LT.ILP1) GO TO 70
      DO 68 J=ILP1,IUM1
      IF(TAUDP(J).NE.0.) GO TO 66
C *** TAUDP=0 AT AN INTERIOR POINT
      IFLAG=1
      P(J-1)=P(J-1)+EPS
      P(J)=P(J)+EPS
      GO TO 68
   66 TEMP=TAUDP(J)*B(J)
      IF(TEMP.GT.0.) GO TO 68
C *** B*TAUDP<0 AT AN INTERIOR POINT
      IFLAG=1
      LAMBAR=AMAX1(ABS(B(J)),DIAG(J)*ABS(TAUDP(J)))
     1/(2.*AMAX1(ABS(TAUDP(J-1)),ABS(TAUDP(J+1))))
      PTILDA=1./SQRT(LAMBAR*H(J-1))
      PTILDA=AMAX1(PTILDA,P(J-1))
      P(J-1)=P(J-1)+OMEGA*(PTILDA-P(J-1))
      PTILDA=1./SQRT(LAMBAR*H(J))
      PTILDA=AMAX1(PTILDA,P(J))
      P(J)=P(J)+OMEGA*(PTILDA-P(J))
   68 CONTINUE
   70 TEMP=TAUDP(IU)*B(IU)
      IF(TEMP.GT.0.) GO TO 80
      IF(TAUDP(IU).EQ.0..AND.IU.NE.N) GO TO 80
      IF(TAUDP(IU).NE.0.) GO TO 75
C *** TAUDP=0 AT RIGHT HAND ENDPOINT (LAST POINT)
      IFLAG=1
      P(N-1)=P(N-1)+EPS
      P(N)=P(N)+EPS
      GO TO 80
   75 IF(IU.NE.N) GO TO 76
C *** B*TAUDP<0 AT RIGHT HAND ENDPOINT (LAST POINT)
      IFLAG=1
      LAMBAR=AMAX1(ABS(B(N)),DIAG(N)*ABS(TAUDP(N)))
     1/(1.*AMAX1(ABS(TAUDP(N-1)),ABS(TAUDP(2))))
      PTILDA=1./SQRT(LAMBAR*H(N-1))
      PTILDA=AMAX1(PTILDA,P(N-1))
      P(N-1)=P(N-1)+OMEGA*(PTILDA-P(N-1))
      PTILDA=1./SQRT(LAMBAR*H(N))
      PTILDA=AMAX1(PTILDA,P(N))
      P(N)=P(N)+OMEGA*(PTILDA-P(N))
      GO TO 80
C *** B*TAUDP<0 AT RIGHT HAND ENDPOINT (OTHER THAN LAST POINT)
   76 IFLAG=1
      LAMBAR=AMAX1(ABS(B(IU)),DIAG(IU)*ABS(TAUDP(IU)))/ABS(TAUDP(IUM1))
      PTILDA=1./SQRT(LAMBAR*H(IUM1))
      PTILDA=AMAX1(PTILDA,P(IUM1))
      P(IUM1)=P(IUM1)+OMEGA*(PTILDA-P(IUM1))
   80 CONTINUE
C ***
C *** CHECK FOR EXTRANEOUS INFLECTION POINTS
C ***
      IF(IFLAG.EQ.0) GO TO 99
C ***
C *** SCALE ABSCISSAE
C ***
      XMU=0.
      DO 93 I=1,N
   93 XMU=AMAX1(XMU,P(I)*H(I))
      ALPHAC=AMAX1(XMU/SIGMA,1.)
      ALPHA=ALPHA*ALPHAC
      DO 94 I=1,N
      H(I)=H(I)/ALPHAC
   94 B(I)=B(I)*ALPHAC
```

```
      B(NP1)=B(NP1)*ALPHAC
      IF(IOUT.NE.0) WRITE(16,95) XMU,ALPHAC,ALPHA
   95 FORMAT(/,'XMU = ',E15.7,5X,'ALPHAC = ',E15.7,5X,'ALPHA = ',E15.7)
      IF(IOUT.NE.0) WRITE(16,96)
   96 FORMAT(/,'     I        H               B')
      IF(IOUT.NE.0) WRITE(16,97) (I,H(I),B(I),I=1,N)
   97 FORMAT(1X,I5,2E15.7)
      IF(IOUT.NE.0) WRITE(16,98) NP1,B(NP1)
   98 FORMAT(1X,I5,15X,E15.7)
      GO TO 34
C ***
C *** NO EXTRANEOUS INFLECTION POINTS
C ***
   99 IF(IOUT.NE.0) WRITE(16,1000)
      IF(IOUT.NE.0) WRITE(16,100) ICOUNT
  100 FORMAT(' NO EXTRANEOUS INFLECTION POINTS: ICOUNT = ',I3)
 1000 FORMAT(///,'-------------------------------------------------',///)
      GO TO 3000
C ***
C *** MAXIMUM NUMBER OF ITERATIONS EXCEEDED
C ***
 2000 IF(IOUT.NE.0) WRITE(16,1000)
      IF(IOUT.NE.0) WRITE(16,2001)
 2001 FORMAT(' MAXIMUM NUMBER OF ITERATIONS EXCEEDED')
 3000 RETURN
      END
C ***
C ***
C *** ESPLOT: B. J. MCCARTIN  10/80
C ***
C ***
C *** FUNCTION: PLOTS THE EXPONENTIAL SPLINE
C ***
C *** DESCRIPTION OF CALLING ARGUMENTS:
C ***      X       = ABSCISSAE
C ***      F       = ORDINATES
C ***      NP1     = NUMBER OF ORDERED PAIRS OF DATA
C ***      B       = RIGHT HAND SIDE OF SPLINE EQUATIONS
C ***      TAUDP   = SOLUTION OF SPLINE EQUATIONS
C ***      P       = EXPONENTIAL SPLINE TENSION PARAMETERS
C ***      H       = DELTA X
C ***      ALPHA   = SCALING PARAMETER
C ***
      SUBROUTINE ESPLOT(X,F,NP1,B,TAUDP,P,H,ALPHA)
      DIMENSION X(1),F(1),B(1),TAUDP(1),P(1),H(1)
      DIMENSION XP(10000),FP(10000),Z(2),IPNUM(2),IPLINE(2),IPSYM(2),
     1IPINCR(2),ITITLE(60),XLBL(5),YLBL(5),IPSTRT(2)
      DATA BLANK/'    '/
      DATA XLBL/5*'    '/
      DATA YLBL/5*'    '/
      DATA ITITLE/5*'    ','EXPO','NENT','IAL ','SPLI','NE  ',50*'    '/
C ***
C *** INITIALIZATION
C ***
      IOUT=0
      N=NP1-1
      DO 1 M=1,2
    1 Z(M)=BLANK
      SCALE=2./(X(NP1)-X(1))
C ***
C *** SET UP ARRAYS
C ***
      K=0
      IF(IOUT.NE.0) WRITE(16,38)
   38 FORMAT(/,1X,'    K       XBAR           TAUBAR          TAUBP')
```

```
      DO 45 I=1,N
      DX=ALPHA*H(I)/99.
      DX=0.5*SCALE
      XX=X(I)-DX
      DO 50 J=1,100
      K=K+1
      XP(K)=XX+DX
      IF(XP(K).LT.X(I)) XP(K)=X(I)
      IF(XP(K).GT.X(I+1)) XP(K)=X(I+1)
      XX=XP(K)
      CALL ESEVAL(X,F,NP1,B,TAUDP,P,H,XP(K),FP(K),TAUBP,ALPHA)
      IF(ICOUT.NE.0) WRITE(16,39) K,XP(K),FP(K),TAUBP
   39 FORMAT(1X,I5,3E15.7)
   50 CONTINUE
   45 CONTINUE
      DO 55 I=1,NP1
      J=K+I
      XP(J)=X(I)
   55 FP(J)=F(I)
      NZ=2
      IPSTRT(1)=1
      IPSTRT(2)=K+1
      IPNUM(1)=K
      IPNUM(2)=NP1
      IPLINE(1)=4
      IPLINE(2)=-1
      IPSYM(1)=0
      IPSYM(2)=1
      IPINCR(1)=1
      IPINCR(2)=1
C ***
C *** PLOT
C ***
      CALL PLTEK(XP,FP,Z,NZ,IPSTRT,IPNUM,IPLINE,IPSYM,IPINCR,ITITLE,
     1XLBL,YLBL,DUM)
      RETURN
      END
C ***
C ***
C *** ESEVAL: B. J. MCCARTIN  10/80
C ***
C ***
C *** FUNCTION: EVALUATES THE EXPONENTIAL SPLINE
C ***
C *** DESCRIPTION OF CALLING ARGUMENTS:
C ***      X       = ABSCISSAE
C ***      F       = ORDINATES
C ***      NP1     = NUMBER OF ORDERED PAIRS OF DATA
C ***      B       = RIGHT HAND SIDE OF SPLINE EQUATIONS
C ***      TAUDP   = SOLUTION OF SPLINE EQUATIONS
C ***      P       = EXPONENTIAL SPLINE TENSION PARAMETERS
C ***      H       = DELTA X .
C ***      XBAR    = ABSCISSA OF EVALUATION
C ***      TAUBAR  = ORDINATE OF EVALUATION
C ***      TAUBP   = DERIVATIVE OF EVALUATION
C ***      ALPHA   = SCALING PARAMETER
C ***
      SUBROUTINE ESEVAL(X,F,NP1,B,TAUDP,P,H,XBAR,TAUBAR,TAUBP,ALPHA)
      DIMENSION X(1),F(1),B(1),TAUDP(1),P(1),H(1)
C ***
C *** INITIALIZATION
C ***
      CONST1=1./6.
      CONST2=1./3.
      CONST3=-7./60.
      CONST4=31./2520.
```

```
      CONST5=-1./15.
      CONST6=2./315.
      CONST7=-7./195.
      CONST8=-.05
      CONST9=1./120.
      N=NP1-1
      ETA=.07
      SCALE=2./(X(NP1)-X(1))
C ***
C *** DETERMINE INTERVAL
C ***
      DO 10 J=1,N
      I=J
      IF(XBAR.LT.X(J+1)) GO TO 11
   10 CONTINUE
C ***
C *** DETERMINE TYPE OF FIT
C ***
   11 TEMP=B(I)*B(I+1)
      IF(TEMP.NE.0.) GO TO 20
C *** FIT WITH LINEAR FUNCTION
      TAUBAR=(F(I)*(X(I+1)-XBAR)+F(I+1)*(XBAR-X(I)))/(H(I)*ALPHA)
      TAUBAR=SCALE*TAUBAR
      TAUBP=(F(I+1)-F(I))/(H(I)*ALPHA)
      TAUBP=SCALE*TAUBP
      GO TO 30
C *** FIT WITH EXPONENTIAL SPLINE
   20 X1=SCALE*(X(I+1)-XBAR)/ALPHA
      X2=SCALE*(XBAR-X(I))/ALPHA
      PI2=P(I)*P(I)
      PIHI=P(I)*H(I)
      IF(PIHI.GT.ETA) GO TO 25
C *** USE POWER SERIES REPRESENTATION
      PI2HI2=PIHI*PIHI
      PI4HI4=PI2HI2*PI2HI2
      X12=X1*X1
      X22=X2*X2
      X14=X12*X12
      X24=X22*X22
      ONEHI2=1./(H(I)*H(I))
      TERM1=1.+CONST3*PI2HI2+CONST4*PI4HI4
      TERM2=CONST1*PI2-ONEHI2+CONST7*PI2*PI2HI2
      TERM3=(CONST8*ONEHI2+CONST9*PI2)*PI2
      TAUBAR=(F(I)*X1+F(I+1)*X2)/H(I)-CONST1*H(I)*
     1(X1*TAUDP(I)*(TERM1+TERM2*X12+TERM3*X14)+
     2X2*TAUDP(I+1)*(TERM1+TERM2*X22+TERM3*X24))
      TAUBP=(F(I+1)-F(I))/H(I)-CONST1*H(I)*
     1(TAUDP(I+1)*(TERM1+3.*TERM2*X22+5.*TERM3*X24)-
     2TAUDP(I)*(TERM1+3.*TERM2*X12+5.*TERM3*X14))
      TAUBP=SCALE*TAUBP/ALPHA
      GO TO 30
C *** USE HYPERBOLIC FUNCTIONS
   25 DENOM=PI2*SINH(P(I)*H(I))
      TAUBAR=(TAUDP(I)*SINH(P(I)*X1)+TAUDP(I+1)*SINH(P(I)*X2))/DENOM
     1+((F(I)-TAUDP(I))/PI2)*X1+(F(I+1)-TAUDP(I+1)/PI2)*X2)/H(I)
      TAUBP=(TAUDP(I+1)*COSH(P(I)*X2)-TAUDP(I)*COSH(P(I)*X1))*P(I)/DENOM
     1+(F(I+1)-F(I)+(TAUDP(I)-TAUDP(I+1))/PI2)/H(I)
      TAUBP=SCALE*TAUBP/ALPHA
   30 CONTINUE
      RETURN
      END
```

Appendix III: Euler Solver

```
      PROGRAM MAIN(INPUT,OUTPUT,TAPE15=INPUT,TAPE16=OUTPUT,TAPE10,
     1TAPE11)
C ***
C ***
C *** EULER2D: B. J. MCCARTIN  2/81
C ***
C ***
      REAL JACOBN(40,14),XMNTM(40,14),YMNTM(40,14),MM(40,14),NN(40,14),
     1MINF,MACH(40,14),MSQD
      DIMENSION DENST(40,14),ENRGY(40,14),DD(40,14),EE(40,14),
     1SAVED(40,14),SAVEN(40,14),SAVEN(40,14),SAVEE(40,14),
     2X(40,14),Y(40,14),XP(40),YP(14),S(40),SP(40),
     3D(40),TAUDP(40),P(40),H(40),
     4XSUBX(40,14),XSUBY(40,14),YSUBX(40,14),YSUBY(40,14),
     5FP1(40,14),FP2(40,14),FP3(40,14),FP4(40,14),
     6FM1(40,14),FM2(40,14),FM3(40,14),FM4(40,14),
     7GP1(14,40),GP2(14,40),GP3(14,40),GP4(14,40),
     8GM1(14,40),GM2(14,40),GM3(14,40),GM4(14,40),
     9F1X(40,14),F2X(40,14),F3X(40,14),F4X(40,14),
     XG1Y(40,14),G2Y(40,14),G3Y(40,14),G4Y(40,14)
      COMMON CONST1,CONST2,CONST3,CONST4,CONST5,CONST6,CONST7,CONST8,CON
     1ST9
      DATA VAR/0.0/
      WRITE(16,1)
    1 FORMAT(1H1,//////////////////////////////,50X,
     125HEULER2D BY B. J. MCCARTIN)
      CALL PLOTSBL(6000,14HB. J. MCCARTIN)
      CONST1=1./6.
      CONST2=1./3.
      CONST3=-7./60.
      CONST4=31./2520.
      CONST5=-1./15.
      CONST6=2./315.
      CONST7=-7./945.
      CONST8=-.05
      CONST9=1./120.
      CALL INPUT(GAMMA,DELTAT,NPLOT,NSTEPS,NP1,MP1,MINF)
      CALL COORD(NP1,MP1,XP,YP,X,Y,S,SP,YMAX,DELTAX,DELTAY,
     1XSUBX,XSUBY,YSUBX,YSUBY,JACOBN)
      CALL INITFL(DENST,XMNTM,YMNTM,ENRGY,S,SP,YMAX,YP,NP1,MP1,MINF,
     1GAMMA)
      NPTS=NP1*MP1
      DO 10 J=1,MP1
      DO 10 I=1,NP1


      TERM=XMNTM(I,J)**2+YMNTM(I,J)**2
      MSQD=ABS(TERM/(GAMMA*(GAMMA-1.)*(DENST(I,J)*ENRGY(I,J)-.5*TERM)))
```

```
      MACH(I,J)=SQRT(MSQD)
   10 CONTINUE
      DO 100 K=1,NSTEPS
      DO 15 J=1,MP1
      DO 15 I=1,NP1
      SAVED(I,J)=DENST(I,J)
      SAVEM(I,J)=MMNTM(I,J)
      SAVEN(I,J)=NMNTM(I,J)
      SAVEE(I,J)=ENRGY(I,J)
      DD(I,J)=SAVED(I,J)
      MM(I,J)=SAVEM(I,J)
      NN(I,J)=SAVEN(I,J)
      EE(I,J)=SAVEE(I,J)
   15 CONTINUE
      CALL DEFINF(DD,MM,NN,EE,XP,YP,NP1,MP1,
     1YMAX,GAMMA,FP1,FP2,FP3,FP4,FM1,FM2,FM3,FM4,XSUBY,YSUBY)
      CALL DEFING(DD,MM,NN,EE,XP,YP,NP1,MP1,
     1YMAX,GAMMA,GP1,GP2,GP3,GP4,GM1,GM2,GM3,GM4,XSUBX,YSUBX)
      CALL COMPFP(XP,NP1,MP1,FP1,FP2,FP3,FP4,FM1,FM2,FM3,FM4,
     1B,TAUDP,P,H,F1X,F2X,F3X,F4X,DELTAX)
      CALL COMPGP(YP,NP1,MP1,GP1,GP2,GP3,GP4,GM1,GM2,GM3,GM4,
     1B,TAUDP,P,H,G1Y,G2Y,G3Y,G4Y,DELTAY)
      DO 20 J=1,MP1
      DO 20 I=1,NP1
      DELTA=DELTAT/JACOBN(I,J)
      DENST(I,J)=DENST(I,J)-(1./6.)*DELTA*(F1X(I,J)+G1Y(I,J))
      MMNTM(I,J)=MMNTM(I,J)-(1./6.)*DELTA*(F2X(I,J)+G2Y(I,J))
      NMNTM(I,J)=NMNTM(I,J)-(1./6.)*DELTA*(F3X(I,J)+G3Y(I,J))
      ENRGY(I,J)=ENRGY(I,J)-(1./6.)*DELTA*(F4X(I,J)+G4Y(I,J))
   20 CONTINUE
      DO 25 J=1,MP1
      DO 25 I=1,NP1
      DELTA=DELTAT/JACOBN(I,J)
      DD(I,J)=SAVED(I,J)-.5*DELTA*(F1X(I,J)+G1Y(I,J))
      MM(I,J)=SAVEM(I,J)-.5*DELTA*(F2X(I,J)+G2Y(I,J))
      NN(I,J)=SAVEN(I,J)-.5*DELTA*(F3X(I,J)+G3Y(I,J))
      EE(I,J)=SAVEE(I,J)-.5*DELTA*(F4X(I,J)+G4Y(I,J))
   25 CONTINUE
      CALL DEFINF(DD,MM,NN,EE,XP,YP,NP1,MP1,
     1YMAX,GAMMA,FP1,FP2,FP3,FP4,FM1,FM2,FM3,FM4,XSUBY,YSUBY)
      CALL DEFING(DD,MM,NN,EE,XP,YP,NP1,MP1,
     1YMAX,GAMMA,GP1,GP2,GP3,GP4,GM1,GM2,GM3,GM4,XSUBX,YSUBX)
      CALL COMPFP(XP,NP1,MP1,FP1,FP2,FP3,FP4,FM1,FM2,FM3,FM4,
     1B,TAUDP,P,H,F1X,F2X,F3X,F4X,DELTAX)
      CALL COMPGP(YP,NP1,MP1,GP1,GP2,GP3,GP4,GM1,GM2,GM3,GM4,
     1B,TAUDP,P,H,G1Y,G2Y,G3Y,G4Y,DELTAY)
      DO 30 J=1,MP1
      DO 30 I=1,NP1
      DELTA=DELTAT/JACOBN(I,J)
      DENST(I,J)=DENST(I,J)-(1./3.)*DELTA*(F1X(I,J)+G1Y(I,J))
      MMNTM(I,J)=MMNTM(I,J)-(1./3.)*DELTA*(F2X(I,J)+G2Y(I,J))
      NMNTM(I,J)=NMNTM(I,J)-(1./3.)*DELTA*(F3X(I,J)+G3Y(I,J))
      ENRGY(I,J)=ENRGY(I,J)-(1./3.)*DELTA*(F4X(I,J)+G4Y(I,J))
   30 CONTINUE
      DO 35 J=1,MP1
      DO 35 I=1,NP1
```

```
      DELTA=DELTAT/JACOBN(I,J)
      DD(I,J)=SAVED(I,J)-.5*DELTA*(F1X(I,J)+G1Y(I,J))
      MM(I,J)=SAVEM(I,J)-.5*DELTA*(F2X(I,J)+G2Y(I,J))
      NN(I,J)=SAVEN(I,J)-.5*DELTA*(F3X(I,J)+G3Y(I,J))
      EE(I,J)=SAVEE(I,J)-.5*DELTA*(F4X(I,J)+G4Y(I,J))
   35 CONTINUE
      CALL DEFINF(DD,MM,NN,EE,XP,YP,NP1,MP1,
     1YMAX,GAMMA,FP1,FP2,FP3,FP4,FM1,FM2,FM3,FM4,XSUBY,YSUBY)
      CALL DEFING(DD,MM,NN,EE,XP,YP,NP1,MP1,
     1YMAX,GAMMA,GP1,GP2,GP3,GP4,GM1,GM2,GM3,GM4,XSUBX,YSUBX)
      CALL COMPFP(XP,NP1,MP1,FP1,FP2,FP3,FP4,FM1,FM2,FM3,FM4,
     1B,TAUDP,P,H,F1X,F2X,F3X,F4X,DELTAX)
      CALL COMPGP(YP,NP1,MP1,GP1,GP2,GP3,GP4,GM1,GM2,GM3,GM4,
     1B,TAUDP,P,H,G1Y,G2Y,G3Y,G4Y,DELTAY)
      DO 40 J=1,MP1
      DO 40 I=1,NP1
      DELTA=DELTAT/JACOBN(I,J)
      DENST(I,J)=DENST(I,J)-(1./3.)*DELTA*(F1X(I,J)+G1Y(I,J))
      MMNTM(I,J)=MMNTM(I,J)-(1./3.)*DELTA*(F2X(I,J)+G2Y(I,J))
      NMNTM(I,J)=NMNTM(I,J)-(1./3.)*DELTA*(F3X(I,J)+G3Y(I,J))
      ENRGY(I,J)=ENRGY(I,J)-(1./3.)*DELTA*(F4X(I,J)+G4Y(I,J))
   40 CONTINUE
      DO 45 J=1,MP1
      DO 45 I=1,NP1
      DELTA=DELTAT/JACOBN(I,J)
      DD(I,J)=SAVED(I,J)-DELTA*(F1X(I,J)+G1Y(I,J))
      MM(I,J)=SAVEM(I,J)-DELTA*(F2X(I,J)+G2Y(I,J))
      NN(I,J)=SAVEN(I,J)-DELTA*(F3X(I,J)+G3Y(I,J))
      EE(I,J)=SAVEE(I,J)-DELTA*(F4X(I,J)+G4Y(I,J))
   45 CONTINUE
      CALL DEFINF(DD,MM,NN,EE,XP,YP,NP1,MP1,
     1YMAX,GAMMA,FP1,FP2,FP3,FP4,FM1,FM2,FM3,FM4,XSUBY,YSUBY)
      CALL DEFING(DD,MM,NN,EE,XP,YP,NP1,MP1,
     1YMAX,GAMMA,GP1,GP2,GP3,GP4,GM1,GM2,GM3,GM4,XSUBX,YSUBX)
      CALL COMPFP(XP,NP1,MP1,FP1,FP2,FP3,FP4,FM1,FM2,FM3,FM4,
     1B,TAUDP,P,H,F1X,F2X,F3X,F4X,DELTAX)
      CALL COMPGP(YP,NP1,MP1,GP1,GP2,GP3,GP4,GM1,GM2,GM3,GM4,
     1B,TAUDP,P,H,G1Y,G2Y,G3Y,G4Y,DELTAY)
      DO 50 J=1,MP1
      DO 50 I=1,NP1
      DELTA=DELTAT/JACOBN(I,J)
      DENST(I,J)=DENST(I,J)-(1./6.)*DELTA*(F1X(I,J)+G1Y(I,J))
      MMNTM(I,J)=MMNTM(I,J)-(1./6.)*DELTA*(F2X(I,J)+G2Y(I,J))
      NMNTM(I,J)=NMNTM(I,J)-(1./6.)*DELTA*(F3X(I,J)+G3Y(I,J))
      ENRGY(I,J)=ENRGY(I,J)-(1./6.)*DELTA*(F4X(I,J)+G4Y(I,J))
   50 CONTINUE
      TIME=K*DELTAT
      M=(K/NPLOT)*NPLOT-K
      NSUP=0
      DO 60 J=1,MP1
      DO 60 I=1,NP1
      TERM=MMNTM(I,J)**2+NMNTM(I,J)**2
      MSQD=ABS(TERM/(GAMMA*(GAMMA-1.)*(DENST(I,J)*ENRGY(I,J)-.5*TERM)))
      MACH(I,J)=SQRT(MSQD)
      IF(MACH(I,J).GE.1.) NSUP=NSUP+1
   60 CONTINUE
```

```
      SDENST=-1.
      SMMNTM=-1.
      SMMNTM=-1.
      SENRGY=-1.
      ADENST=0.
      AMMNTM=0.
      ANMNTM=0.
      AENRGY=0.
      DO 70 J=1,MP1
      DO 70 I=1,NP1
      DELTAD=ABS(DENST(I,J)-SAVED(I,J))
      DELTAM=ABS(MMNTM(I,J)-SAVEM(I,J))
      DELTAN=ABS(NMNTM(I,J)-SAVEN(I,J))
      DELTAE=ABS(ENRGY(I,J)-SAVEE(I,J))
      ADENST=ADENST+DELTAD
      AMMNTM=AMMNTM+DELTAM
      ANMNTM=ANMNTM+DELTAN
      AENRGY=AENRGY+DELTAE
      IF(SDENST.GE.DELTAD) GO TO 71
      SDENST=DELTAD
      ISD=I
      JSD=J
71    IF(SMMNTM.GE.DELTAM) GO TO 72
      SMMNTM=DELTAM
      ISM=I
      JSM=J
72    IF(SNMNTM.GE.DELTAN) GO TO 73
      SNMNTM=DELTAN
      ISN=I
      JSN=J
73    IF(SENRGY.GE.DELTAE) GO TO 70
      SENRGY=DELTAE
      ISE=I
      JSE=J
70    CONTINUE
      ADENST=ADENST/NPTS
      AMMNTM=AMMNTM/NPTS
      ANMNTM=ANMNTM/NPTS
      AENRGY=AENRGY/NPTS
      WRITE(16,78) K,NSUP
78    FORMAT(5H K = ,I5,3X,7HNSUP = ,I5)
      WRITE(16,74) SDENST,ISD,JSD,ADENST
74    FORMAT(1X,E15.7,I5,I5,E15.7)
      WRITE(16,75) SMMNTM,ISM,JSM,AMMNTM
75    FORMAT(1X,E15.7,I5,I5,E15.7)
      WRITE(16,76) SNMNTM,ISN,JSN,ANMNTM
76    FORMAT(1X,E15.7,I5,I5,E15.7)
      WRITE(16,77) SENRGY,ISE,JSE,AENRGY
77    FORMAT(1X,E15.7,I5,I5,E15.7)
      REWIND 10
      WRITE(10) K
      WRITE(10) ((DENST(I,J),MMNTM(I,J),NMNTM(I,J),ENRGY(I,J),I=1,NP1),
     1J=1,MP1)
      IF(M.NE.0) GO TO 90
      CALL CHKPTX(VAR)
      WRITE(16,81)
```

```
   81 FORMAT(6H INLET)
      WRITE(16,82) (J,DENST(1,J),MMNTM(1,J),NMNTM(1,J),ENRGY(1,J),
     1MACH(1,J),J=1,MP1)
   82 FORMAT(1X,I5,5E15.7)
      WRITE(16,83)
   83 FORMAT(7H OUTLET)
      WRITE(16,82) (J,DENST(NP1,J),MMNTM(NP1,J),NMNTM(NP1,J),ENRGY(NP1,J
     1),MACH(NP1,J),J=1,MP1)
      WRITE(16,84)
   84 FORMAT(11H UPPER WALL)
      WRITE(16,82) (I,DENST(I,MP1),MMNTM(I,MP1),NMNTM(I,MP1),ENRGY(I,MP1
     1),MACH(I,MP1),I=1,NP1)
      WRITE(16,85)
   85 FORMAT(11H LOWER WALL)
      WRITE(16,82) (I,DENST(I,1),MMNTM(I,1),NMNTM(I,1),ENRGY(I,1),
     1MACH(I,1),I=1,NP1)
      SCALX=.375
      SCALM=3.
      CALL FRAME
      IPEN=3
      DO 87 I=1,NP1
      CALL PLOT(SCALX*(X(I,1)+15.),SCALM*MACH(I,1),IPEN)
   87 IPEN=2
      CALL FRAME
      IPEN=3
      DO 89 I=1,NP1
      CALL PLOT(SCALX*(X(I,MP1)+15.),SCALM*MACH(I,MP1),IPEN)
   89 IPEN=2
   90 CALL ENDRYC(NP1,MP1,MINF,GAMMA,DENST,MMNTM,NMNTM,ENRGY,SP)
  100 CONTINUE
      CALL FRAME
      CALL PLOT(0.,0.,999)
 1000 STOP
      END



      SUBROUTINE INPUT(GAMMA,DELTAT,NPLOT,NSTEPS,NP1,MP1,MINF)
C *** INITIALIZE PARAMETERS
      REAL MINF
      GAMMA=1.4
      MINF=.675
      DELTAT=.05
      NP1=40
      MP1=14
      NSTEPS=750
      NPLOT=50
      WRITE(16,10)
   10 FORMAT(6H1INPUT)
      WRITE(16,11)
   11 FORMAT(1H ,7X,5HGAMMA,10X,4HMINF,10X,6HDELTAT,16X,3HNP1,12X,3HMP1,
     110X,5HNPLOT,10X,6HNSTEPS)
      WRITE(16,12) GAMMA,MINF,DELTAT,NP1,MP1,NPLOT,NSTEPS
   12 FORMAT(1X ,3E15.7,5I15)
```

```
      RETURN
      END



      SUBROUTINE COORD(NP1,MP1,XP,YP,X,Y,S,SP,YMAX,DELTAX,DELTAY,
     1XSUBX,XSUBY,YSUBX,YSUBY,JACOBN)
C *** GENERATE SHEARED COORDINATES
      REAL JACOBN(40,14)
      DIMENSION XP(40),YP(14),XS(14),YS(14),ES(14),TAUDPS(14),PS(14),
     1HS(14),S(40),SP(40),X(40,14),Y(40,14),
     2XSUBX(40,14),XSUBY(40,14),YSUBX(40,14),YSUBY(40,14)
      DIMENSION XT(3),YT(3),ST(3),TAUDPT(3),PT(3),HT(3)
      XMAX=15.
      XMIN=-15.
      YMAX=4.
      DELTAX=2.*XMAX/(NP1-1)
      DELTAY=YMAX/(MP1-1)
      DO 10 I=1,NP1
   10 XP(I)=XMIN+(I-1)*DELTAX
      DO 20 J=1,MP1
   20 YP(J)=(J-1)*DELTAY
      XBAR=5.
      NT=3
      XT(1)=0.
      XT(2)=XBAR+.5*(XMAX-XBAR)
      XT(3)=XMAX
      YT(1)=0.
      YT(2)=XBAR
      YT(3)=XMAX
      FPA=YT(2)/XT(2)-.15
      FPB=(YT(3)-YT(2))/(XT(3)-XT(2))+.1
      CALL EXPSPL(XT,YT,FPA,FPB,NT,ST,TAUDPT,PT,HT,ALPHAT)
      DO 25 I=1,NP1
      IF(XP(I).GE.0.) XEVAL=XP(I)
      IF(XP(I).LT.0.) XEVAL=-XP(I)
      IF(XEVAL.LT.XT(1)) XEVAL=XT(1)
      IF(XEVAL.GT.XT(NT)) XEVAL=XT(NT)
      CALL ESEVAL(XT,YT,NT,ST,TAUDPT,PT,HT,XEVAL,X(I,1),XSUBX(I,1),
     1DUMMY,DUMMY,ALPHAT)
      IF(XP(I).LT.0.) X(I,1)=-X(I,1)
   25 CONTINUE
      NS=7
      XS(1)=XMIN
      XS(2)=-5.
      XS(3)=-2.5
      XS(4)=0.
      XS(5)=2.5
      XS(6)=5.
      XS(7)=XMAX
      YS(1)=0.
      YS(2)=0.
      YS(3)=.1
      YS(4)=.4
```

```
      YS(5)=.1
      YS(6)=0.
      YS(7)=0.
      CALL EXPSPL(XS,YS,0.,0.,NS,BS,TAUDPS,PS,HS,ALPHAS)
      DO 30 I=1,NP1
      XEVAL=X(I,1)
      IF(XEVAL.LT.XS(1)) XEVAL=XS(1)
      IF(XEVAL.GT.XS(NS)) XEVAL=XS(NS)
      CALL ESEVAL(XS,YS,NS,BS,TAUDPS,PS,HS,XEVAL,S(I),SP(I),
     1DUMMY,DUMMY,ALPHAS)
   30 CONTINUE
      DO 40 J=1,MP1
      DO 40 I=1,NP1
      X(I,J)=X(I,1)
      Y(I,J)=YP(J)+S(I)*(1.-YP(J)/YMAX)
   40 CONTINUE
      WRITE(16,41)
   41 FORMAT(9H1GEOMETRY)
      WRITE(16,42)
   42 FORMAT(1H ,5H    I,5H    J,7X,1HX,14X,1HY)
      WRITE(16,43) ((I,J,X(I,J),Y(I,J),I=1,NP1),J=1,MP1)
   43 FORMAT(1H ,2I5,2E15.7)
      DO 50 J=1,MP1
      DO 50 I=1,NP1
      XSUBX(I,J)=XSUBX(I,1)
      XSUBY(I,J)=0.
      YSUBX(I,J)=SP(I)*(1.-YP(J)/YMAX)*XSUBX(I,J)
      YSUBY(I,J)=1.-S(I)/YMAX
      JACOBN(I,J)=XSUBX(I,J)*YSUBY(I,J)-XSUBY(I,J)*YSUBX(I,J)
   50 CONTINUE
      WRITE(16,51)
   51 FORMAT(11H    I    J,6X,5HXSUBX,10X,5HXSUBY,10X,5HYSUBX,10X,
     15HYSUBY,10X,5HJACOBN)
      WRITE(16,52) ((I,J,XSUBX(I,J),XSUBY(I,J),YSUBX(I,J),YSUBY(I,J),
     1JACOBN(I,J),I=1,NP1),J=1,MP1)
   52 FORMAT(1H ,2I5,5E15.7)
      RETURN
      END



      SUBROUTINE INITFL(DENST,MMNTM,NMNTM,ENRGY,S,SP,YMAX,YP,NP1,MP1,
     1MINF,GAMMA)
C *** CALCULATE INITIAL FLOW FIELD
      REAL MINF,MMNTM(40,14),NMNTM(40,14)
      DIMENSION DENST(40,14),ENRGY(40,14),S(40),SP(40),YP(14)
      DO 10 J=1,MP1
      DO 10 I=1,NP1
      DENST(I,J)=1.
      ENRGY(I,J)=.5+1./(GAMMA*(GAMMA-1.)*MINF*MINF)
   10 CONTINUE
      DO 20 I=1,NP1
      MMNTM(I,MP1)=1.
      NMNTM(I,MP1)=0.
```

```
      MMNTM(I,1)=1./SQRT(1.+SP(I)*SP(I))
      NMNTM(I,1)=SP(I)*MMNTM(I,1)
   20 CONTINUE
      M=MP1-1
      DO 30 J=2,M
      DO 30 I=1,NP1
      MMNTM(I,J)=(YP(J)/YMAX)*MMNTM(I,MP1)+(1.-YP(J)/YMAX)*MMNTM(I,1)
      NMNTM(I,J)=(YP(J)/YMAX)*NMNTM(I,MP1)+(1.-YP(J)/YMAX)*NMNTM(I,1)
   30 CONTINUE
      WRITE(16,31)
   31 FORMAT(19H1INITIAL FLOW FIELD)
      WRITE(16,32)
   32 FORMAT(1H ,12H     I     J   ,5X,5HDENST,10X,5HMMNTM,10X,5HNMNTM,
     110X,5HENRGY)
      WRITE(16,33) ((I,J,DENST(I,J),MMNTM(I,J),NMNTM(I,J),ENRGY(I,J),
     1I=1,NP1),J=1,MP1)
   33 FORMAT(1H ,2I5,4E15.7)
      RETURN
      END



      SUBROUTINE DEFINE(DD,MM,NN,EE,XP,YP,NP1,MP1,
     1YMAX,GAMMA,FP1,FP2,FP3,FP4,FM1,FM2,FM3,FM4,XSUBY,YSUBY)
C *** DEFINE FLUX VECTOR SPLITTING IN X-DIRECTION
      REAL MM(40,14),NN(40,14),LAMP1,LAMP3,LAMP4,
     1LAMM1,LAMM3,LAMM4,KKT,K1T,K2T
      DIMENSION DD(40,14),EE(40,14),XP(40),YP(14),
     1FP1(40,14),FP2(40,14),FP3(40,14),FP4(40,14),
     2FM1(40,14),FM2(40,14),FM3(40,14),FM4(40,14),
     3XSUBY(40,14),YSUBY(40,14)
      DO 100 J=1,MP1
      DO 100 I=1,NP1
      U=MM(I,J)/DD(I,J)
      V=NN(I,J)/DD(I,J)
      CSQD=ABS((EE(I,J)-.5*(MM(I,J)*MM(I,J)+NN(I,J)*NN(I,J))/DD(I,J))*
     1GAMMA*(GAMMA-1.)/DD(I,J))
      C=SQRT(CSQD)
      CONST1=YSUBY(I,J)*U-XSUBY(I,J)*V
      KKT=SQRT(YSUBY(I,J)**2+XSUBY(I,J)**2)
      CONST2=C*KKT
      LAMP1=0.
      LAMM1=0.
      IF(CONST1.GE.0.) LAMP1=CONST1
      IF(CONST1.LE.0.) LAMM1=CONST1
      LAMP3=LAMP1+CONST2
      LAMP4=LAMP1
      LAMM3=LAMM1
      LAMM4=LAMM1-CONST2
      K1T=YSUBY(I,J)/KKT
      K2T=-XSUBY(I,J)/KKT
      CONST3=DD(I,J)/(2.*GAMMA)
      W=((3.-GAMMA)*(LAMP3+LAMP4)*C*C)/(2.*(GAMMA-1.))
      FP1(I,J)=CONST3*(2.*(GAMMA-1.)*LAMP1+LAMP3+LAMP4)
```

```
      FP2(I,J)=CONST3*(2.*(GAMMA-1.)*LAMP1*U+LAMP3*
     1(U+C*K1T)+LAMP4*(U-C*K1T))
      FP3(I,J)=CONST3*(2.*(GAMMA-1.)*LAMP1*V+LAMP3*
     1(V+C*K2T)+LAMP4*(V-C*K2T))
      FP4(I,J)=CONST3*((GAMMA-1.)*LAMP1*(U*U+V*V)+.5*LAMP3*
     1((U+C*K1T)**2+(V+C*K2T)**2)+.5*LAMP4*((U-C*K1T)**2
     2+(V-C*K2T)**2)+R)
      R=((3.-GAMMA)*(LAMM3+LAMM4)*C*C)/(2.*(GAMMA-1.))
      FM1(I,J)=CONST3*(2.*(GAMMA-1.)*LAMM1+LAMM3+LAMM4)
      FM2(I,J)=CONST3*(2.*(GAMMA-1.)*LAMM1*U+LAMM3*
     1(U+C*K1T)+LAMM4*(U-C*K1T))
      FM3(I,J)=CONST3*(2.*(GAMMA-1.)*LAMM1*V+LAMM3*
     1(V+C*K2T)+LAMM4*(V-C*K2T))
      FM4(I,J)=CONST3*((GAMMA-1.)*LAMM1*(U*U+V*V)+.5*LAMM3*
     1((U+C*K1T)**2+(V+C*K2T)**2)+.5*LAMM4*((U-C*K1T)**2
     2+(V-C*K2T)**2)+R)
      IF(ABS(FP1(I,J)).LT.1.E-10) FP1(I,J)=0.
      IF(ABS(FP2(I,J)).LT.1.E-10) FP2(I,J)=0.
      IF(ABS(FP3(I,J)).LT.1.E-10) FP3(I,J)=0.
      IF(ABS(FP4(I,J)).LT.1.E-10) FP4(I,J)=0.
      IF(ABS(FM1(I,J)).LT.1.E-10) FM1(I,J)=0.
      IF(ABS(FM2(I,J)).LT.1.E-10) FM2(I,J)=0.
      IF(ABS(FM3(I,J)).LT.1.E-10) FM3(I,J)=0.
      IF(ABS(FM4(I,J)).LT.1.E-10) FM4(I,J)=0.
  100 CONTINUE
      RETURN
      END



      SUBROUTINE DEFING(DD,MM,NN,EE,XP,YP,NP1,MP1,
     1YMAX,GAMMA,GP1,GP2,GP3,GP4,GM1,GM2,GM3,GM4,XSUBX,YSUBX)
C *** DEFINE FLUX VECTOR SPLITTING IN Y-DIRECTION
      REAL MM(40,14),NN(40,14),LAMP1,LAMP3,LAMP4,
     1LAMM1,LAMM3,LAMM4,KKT,K1T,K2T
      DIMENSION DD(40,14),EE(40,14),XP(40),YP(14),
     1GP1(14,40),GP2(14,40),GP3(14,40),GP4(14,40),
     2GM1(14,40),GM2(14,40),GM3(14,40),GM4(14,40),
     3XSUBX(40,14),YSUBX(40,14)
      DO 100 J=1,MP1
      DO 100 I=1,NP1
      U=MM(I,J)/DD(I,J)
      V=NN(I,J)/DD(I,J)
      CSQD=ABS((EE(I,J)-.5*(MM(I,J)*MM(I,J)+NN(I,J)*NN(I,J))/DD(I,J))*
     1GAMMA*(GAMMA-1.)/DD(I,J))
      C=SQRT(CSQD)
      CONST1=-YSUBX(I,J)*U+XSUBX(I,J)*V
      KKT=SQRT(YSUBX(I,J)**2+XSUBX(I,J)**2)
      CONST2=C*KKT
      LAMP1=0.
      LAMM1=0.
      IF(CONST1.GE.0.) LAMP1=CONST1
      IF(CONST1.LE.0.) LAMM1=CONST1
      LAMP3=LAMP1+CONST2
```

```
      LAMP4=LAMP1
      LAMM3=LAMM1
      LAMM4=LAMM1-CONST2
      K1T=-YSUBX(I,J)/KKT
      K2T=XSUBX(I,J)/KKT
      CONST3=DD(I,J)/(2.*GAMMA)
      A=((3.-GAMMA)*(LAMP3+LAMP4)*C*C)/(2.*(GAMMA-1.))
      GP1(J,I)=CONST3*(2.*(GAMMA-1.)*LAMP1+LAMP3+LAMP4)
      GP2(J,I)=CONST3*(2.*(GAMMA-1.)*LAMP1*U+LAMP3*
     1(U+C*K1T)+LAMP4*(U-C*K1T))
      GP3(J,I)=CONST3*(2.*(GAMMA-1.)*LAMP1*V+LAMP3*
     1(V+C*K2T)+LAMP4*(V-C*K2T))
      GP4(J,I)=CONST3*((GAMMA-1.)*LAMP1*(U*U+V*V)+.5*LAMP3*
     1((U+C*K1T)**2+(V+C*K2T)**2)+.5*LAMP4*((U-C*K1T)**2
     2+(V-C*K2T)**2)+A)
      A=((3.-GAMMA)*(LAMM3+LAMM4)*C*C)/(2.*(GAMMA-1.))
      GM1(J,I)=CONST3*(2.*(GAMMA-1.)*LAMM1+LAMM3+LAMM4)
      GM2(J,I)=CONST3*(2.*(GAMMA-1.)*LAMM1*U+LAMM3*
     1(U+C*K1T)+LAMM4*(U-C*K1T))
      GM3(J,I)=CONST3*(2.*(GAMMA-1.)*LAMM1*V+LAMM3*
     1(V+C*K2T)+LAMM4*(V-C*K2T))
      GM4(J,I)=CONST3*((GAMMA-1.)*LAMM1*(U*U+V*V)+.5*LAMM3*
     1((U+C*K1T)**2+(V+C*K2T)**2)+.5*LAMM4*((U-C*K1T)**2
     2+(V-C*K2T)**2)+A)
      IF(ABS(GP1(J,I)).LT.1.E-10) GP1(J,I)=0.
      IF(ABS(GP2(J,I)).LT.1.E-10) GP2(J,I)=0.
      IF(ABS(GP3(J,I)).LT.1.E-10) GP3(J,I)=0.
      IF(ABS(GP4(J,I)).LT.1.E-10) GP4(J,I)=0.
      IF(ABS(GM1(J,I)).LT.1.E-10) GM1(J,I)=0.
      IF(ABS(GM2(J,I)).LT.1.E-10) GM2(J,I)=0.
      IF(ABS(GM3(J,I)).LT.1.E-10) GM3(J,I)=0.
      IF(ABS(GM4(J,I)).LT.1.E-10) GM4(J,I)=0.
  100 CONTINUE
      RETURN
      END



      SUBROUTINE COMPFP(XP,NP1,MP1,FP1,FP2,FP3,FP4,FM1,FM2,FM3,FM4,
     1B,TAUDP,P,H,F1X,F2X,F3X,F4X,DELTAX)
C *** COMPUTE DERIVITIVES IN X-DIRECTION
      DIMENSION XP(40),B(40),TAUDP(40),P(40),H(40),
     1FP1(40,14),FP2(40,14),FP3(40,14),FP4(40,14),
     2FM1(40,14),FM2(40,14),FM3(40,14),FM4(40,14),
     3F1X(40,14),F2X(40,14),F3X(40,14),F4X(40,14)
      DO 100 J=1,MP1
      CALL SPLEXP(XP,FP1(1,J),FPA,FPB,NP1,B,TAUDP,P,H,ALPHA)
      IFLAG=1
      CALL EVALF(XP,FP1(1,J),NP1,B,TAUDP,P,H,ALPHA,F1X,DELTAX,IFLAG,J)
      CALL SPLEXP(XP,FM1(1,J),FPA,FPB,NP1,B,TAUDP,P,H,ALPHA)
      IFLAG=2
      CALL EVALF(XP,FM1(1,J),NP1,B,TAUDP,P,H,ALPHA,F1X,DELTAX,IFLAG,J)
      CALL SPLEXP(XP,FP2(1,J),FPA,FPB,NP1,B,TAUDP,P,H,ALPHA)
      IFLAG=1
```

```
      CALL EVALF(XP,FP2(1,J),NP1,B,TAUDP,P,H,ALPHA,F2X,DELTAX,IFLAG,J)
      CALL SPLEXP(XP,FP2(1,J),FPA,FPB,NP1,B,TAUDP,P,H,ALPHA)
      IFLAG=2
      CALL EVALF(XP,FP2(1,J),NP1,B,TAUDP,P,H,ALPHA,F2X,DELTAX,IFLAG,J)
      CALL SPLEXP(XP,FP3(1,J),FPA,FPB,NP1,B,TAUDP,P,H,ALPHA)
      IFLAG=1
      CALL EVALF(XP,FP3(1,J),NP1,B,TAUDP,P,H,ALPHA,F3X,DELTAX,IFLAG,J)
      CALL SPLEXP(XP,FP3(1,J),FPA,FPB,NP1,B,TAUDP,P,H,ALPHA)
      IFLAG=2
      CALL EVALF(XP,FP3(1,J),NP1,B,TAUDP,P,H,ALPHA,F3X,DELTAX,IFLAG,J)
      CALL SPLEXP(XP,FP4(1,J),FPA,FPB,NP1,B,TAUDP,P,H,ALPHA)
      IFLAG=1
      CALL EVALF(XP,FP4(1,J),NP1,B,TAUDP,P,H,ALPHA,F4X,DELTAX,IFLAG,J)
      CALL SPLEXP(XP,FP4(1,J),FPA,FPB,NP1,B,TAUDP,P,H,ALPHA)
      IFLAG=2
      CALL EVALF(XP,FP4(1,J),NP1,B,TAUDP,P,H,ALPHA,F4X,DELTAX,IFLAG,J)
  100 CONTINUE
      RETURN
      END



      SUBROUTINE COMPGP(YP,NP1,MP1,GP1,GP2,GP3,GP4,GM1,GM2,GM3,GM4,
     1B,TAUDP,P,H,G1Y,G2Y,G3Y,G4Y,DELTAY)
C *** COMPUTE DERIVATIVES IN Y-DIRECTION
      DIMENSION YP(14),B(40),TAUDP(40),P(40),H(40),
     1GP1(14,40),GP2(14,40),GP3(14,40),GP4(14,40),
     2GM1(14,40),GM2(14,40),GM3(14,40),GM4(14,40),
     3G1Y(40,14),G2Y(40,14),G3Y(40,14),G4Y(40,14)
      DO 100 I=1,NP1
      CALL SPLEXP(YP,GP1(1,I),GPA,GPB,MP1,B,TAUDP,P,H,ALPHA)
      IFLAG=1
      CALL EVALG(YP,GP1(1,I),MP1,B,TAUDP,P,H,ALPHA,G1Y,DELTAY,IFLAG,I)
      CALL SPLEXP(YP,GM1(1,I),GPA,GPB,MP1,B,TAUDP,P,H,ALPHA)
      IFLAG=2
      CALL EVALG(YP,GM1(1,I),MP1,B,TAUDP,P,H,ALPHA,G1Y,DELTAY,IFLAG,I)
      CALL SPLEXP(YP,GP2(1,I),GPA,GPB,MP1,B,TAUDP,P,H,ALPHA)
      IFLAG=1
      CALL EVALG(YP,GP2(1,I),MP1,B,TAUDP,P,H,ALPHA,G2Y,DELTAY,IFLAG,I)
      CALL SPLEXP(YP,GM2(1,I),GPA,GPB,MP1,B,TAUDP,P,H,ALPHA)
      IFLAG=2
      CALL EVALG(YP,GM2(1,I),MP1,B,TAUDP,P,H,ALPHA,G2Y,DELTAY,IFLAG,I)
      CALL SPLEXP(YP,GP3(1,I),GPA,GPB,MP1,B,TAUDP,P,H,ALPHA)
      IFLAG=1
      CALL EVALG(YP,GP3(1,I),MP1,B,TAUDP,P,H,ALPHA,G3Y,DELTAY,IFLAG,I)
      CALL SPLEXP(YP,GM3(1,I),GPA,GPB,MP1,B,TAUDP,P,H,ALPHA)
      IFLAG=2
      CALL EVALG(YP,GM3(1,I),MP1,B,TAUDP,P,H,ALPHA,G3Y,DELTAY,IFLAG,I)
      CALL SPLEXP(YP,GP4(1,I),GPA,GPB,MP1,B,TAUDP,P,H,ALPHA)
      IFLAG=1
      CALL EVALG(YP,GP4(1,I),MP1,B,TAUDP,P,H,ALPHA,G4Y,DELTAY,IFLAG,I)
      CALL SPLEXP(YP,GM4(1,I),GPA,GPB,MP1,B,TAUDP,P,H,ALPHA)
      IFLAG=2
      CALL EVALG(YP,GM4(1,I),MP1,B,TAUDP,P,H,ALPHA,G4Y,DELTAY,IFLAG,I)
```

```
  100 CONTINUE
      RETURN
      END



      SUBROUTINE BNDRYC(NP1,MP1,MINF,GAMMA,DENST,MMNTM,NMNTM,ENRGY,SP)
C *** ENFORCE BOUNDARY CONDITIONS
      REAL MINF,MMNTM(40,14),NMNTM(40,14)
      DIMENSION DENST(40,14),ENRGY(40,14),SP(40)
      N=NP1-1
      M=MP1-1
      PINF=1./(GAMMA*MINF*MINF)
      EINF=.5+1./(GAMMA*(GAMMA-1.)*MINF*MINF)
C *** INLET
      DO 10 J=1,MP1
      RHO=DENST(1,J)
      U=MMNTM(1,J)/RHO
      V=NMNTM(1,J)/RHO
      P=(GAMMA-1.)*(ENRGY(1,J)-.5*(U*U+V*V)*RHO)
      CSQD=ABS(GAMMA*P/RHO)
      C=SQRT(CSQD)
C     RHOP=1.+(.5*RHO/C)*(1.-U)*(.5/CSQD)*(P-PINF)
      RHOP=1.
C     UP=.5*(1.+U)+(.5/(RHO*C))*(PINF-P)
      UP=1.
      VP=0.
C     PP=.5*RHO*C*(1.-U)+.5*(PINF+P)
      PP=P+RHO*C*(1.-U)
      DENST(1,J)=RHOP
      MMNTM(1,J)=RHOP*UP
      NMNTM(1,J)=RHOP*VP
      ENRGY(1,J)=PP/(GAMMA-1.)+.5*(UP*UP+VP*VP)*RHOP
   10 CONTINUE
C *** OUTLET
      DO 20 J=1,MP1
      RHO=DENST(NP1,J)
      U=MMNTM(NP1,J)/RHO
      V=NMNTM(NP1,J)/RHO
      P=(GAMMA-1.)*(ENRGY(NP1,J)-.5*(U*U+V*V)*RHO)
      CSQD=ABS(GAMMA*P/RHO)
      C=SQRT(CSQD)
C     RHOP=RHO+(.5*RHO/C)*(U-1.)+(.5/CSQD)*(PINF-P)
      RHOP=RHO+(PINF-P)/CSQD
C     UP=.5*(U+1.)+(.5/(RHO*C))*(P-PINF)
      UP=U+(P-PINF)/(RHO*C)
      VP=V
C     PP=.5*RHO*C*(U-1.)+.5*(P+PINF)
      PP=PINF
      DENST(NP1,J)=RHOP
      MMNTM(NP1,J)=RHOP*UP
      NMNTM(NP1,J)=RHOP*VP
      ENRGY(NP1,J)=PP/(GAMMA-1.)+.5*(UP*UP+VP*VP)*RHOP
   20 CONTINUE
```

```
C *** UPPER WALL
      DO 30 I=2,N
      RHO=DENST(I,MP1)
      U=MMNTM(I,MP1)/RHO
      V=NMNTM(I,MP1)/RHO
      P=(GAMMA-1.)*(ENRGY(I,MP1)-.5*(U*U+V*V)*RHO)
      CSQD=ABS(GAMMA*P/RHO)
      C=SQRT(CSQD)
      RHOP=RHO+(RHO/C)*V
C     RHOP=RHO
      UP=U
      VP=0.
      PP=P+RHO*C*V
C     PP=P
      DENST(I,MP1)=RHOP
      MMNTM(I,MP1)=RHOP*UP
      NMNTM(I,MP1)=RHOP*VP
      ENRGY(I,MP1)=PP/(GAMMA-1.)+.5*(UP*UP+VP*VP)*RHOP
   30 CONTINUE
C *** LOWER WALL
      DO 40 I=2,N
      RHO=DENST(I,1)
      U=MMNTM(I,1)/RHO
      V=NMNTM(I,1)/RHO
      P=(GAMMA-1.)*(ENRGY(I,1)-.5*(U*U+V*V)*RHO)
      CSQD=ABS(GAMMA*P/RHO)
      C=SQRT(CSQD)
      ALPHA=ATAN(SP(I))
      RHOP=RHO-(RHO/C)*(-SIN(ALPHA)*U+COS(ALPHA)*V)
C     RHOP=RHO
      UP=COS(ALPHA)**2*U+SIN(ALPHA)*COS(ALPHA)*V
      VP=SIN(ALPHA)*COS(ALPHA)*U+SIN(ALPHA)**2*V
      PP=P-RHO*C*(-SIN(ALPHA)*U+COS(ALPHA)*V)
C     PP=P
      DENST(I,1)=RHOP
      MMNTM(I,1)=RHOP*UP
      NMNTM(I,1)=RHOP*VP
      ENRGY(I,1)=PP/(GAMMA-1.)+.5*(UP*UP+VP*VP)*RHOP
   40 CONTINUE
      NMNTM(NP1,1)=0.
      NMNTM(NP1,MP1)=0.
      RETURN
      END



C ***
C ***
C *** EXPSPL: B. J. MCCARTIN  10/80
C ***
C ***
C *** FUNCTION: CONSTRUCTS THE INTERPOLATORY EXPONENTIAL SPLINE
C ***           TO A SET OF POINTS IN THE PLANE WITH MONOTONICALLY
C ***           INCREASING ABSCISSAE USING SPECIFIED DERIVATIVE
C ***           END CONDITIONS
C ***
```

```
C *** REFERENCE: B. J. MCCARTIN, NUMERICAL COMPUTATION OF
C ***            EXPONENTIAL SPLINES, COURANT MATHEMATICS
C ***            AND COMPUTING LABORATORY, OCTOBER 1980
C ***
C *** DESCRIPTION OF CALLING ARGUMENTS:
C ***      X       = ABSCISSAE
C ***      F       = ORDINATES
C ***      FPA     = LEFT HAND DERIVATIVE END CONDITION
C ***      FPB     = RIGHT HAND DERIVATIVE END CONDITION
C ***      NP1     = NUMBER OF ORDERED PAIRS OF DATA
C ***      B       = RIGHT HAND SIDE OF SPLINE EQUATIONS
C ***      TAUDP   = SOLUTION OF SPLINE EQUATIONS
C ***      P       = EXPONENTIAL SPLINE TENSION PARAMETERS
C ***      H       = DELTA X
C ***      ALPHA   = SCALING PARAMETER
C ***
      SUBROUTINE EXPSPL(X,F,FPA,FPB,NP1,B,TAUDP,P,H,ALPHA)
      DIMENSION X(1),F(1),B(1),TAUDP(1),P(1),H(1)
      DIMENSION E(100),DIAG(100),ILIML(100),ILIMU(100),Q(100),U(100)
      REAL LAMBAR
C ***
C *** INITIALIZATION
C ***
      IOUT=0
      IPLOT=0
      CONST1=1./6.
      CONST2=1./3.
      CONST3=-7./60.
      CONST4=31./2520.
      CONST5=-1./15.
      CONST6=2./315.
      N=NP1-1
      EPS=1.E-6
      OMEGA=1.
      ETA=.07
      SIGMA=100.
      ALPHA=1.
      ITMAX=5
      SCALE=2./(X(NP1)-X(1))
      ICOUNT=0
      DO 4 I=1,N
      H(I)=X(I+1)-X(I)
    4 H(I)=SCALE*H(I)
      DO 5 I=1,N
    5 P(I)=0.
C ***
C *** DEFINE B'S
C ***
      B(1)=(F(2)-F(1))/H(1)-FPA/SCALE
      IF(N.EQ.1) GO TO 11
      DO 10 I=2,N
   10 B(I)=(F(I+1)-F(I))/H(I)-(F(I)-F(I-1))/H(I-1)
   11 B(NP1)=FPB/SCALE-(F(NP1)-F(N))/H(N)
C ***
```

```
C *** DETERMINE WHICH INTERVALS SHOULD BE FIT WITH
C *** LINE SEGMENTS AND WHICH INTERVALS SHOULD BE
C *** FIT WITH EXPONENTIAL SPLINES
C ***     K         = NUMBER OF EXPONENTIAL SPLINE INTERVALS
C ***     ILIML(I) = LEFT HAND ENDPOINT OF ITH SPLINE INTERVAL
C ***     ILIMU(I) = RIGHT HAND ENDPOINT OF ITH SPLINE INTERVAL
C ***
      K=0
      IL=1
   14 CONTINUE
      DO 20 I=IL,N
      TEMP=B(I)*B(I+1)
      IF(TEMP.EQ.0.) GO TO 20
      K=K+1
      ILIML(K)=I
      DO 16 J=I,N
      TEMP=B(J)*B(J+1)
      IF(TEMP..E.0.) GO TO 15
      ILIMU(K)=J
      GO TO 29
   15 IF(J.NE.N) GO TO 16
      ILIMU(K)=N+1
      GO TO 30
   16 CONTINUE
   20 CONTINUE
      GO TO 30
   29 IL=J
      GO TO 14
   30 CONTINUE
      IF(K.EQ.0) GO TO 54
C ***
C *** SET UP SPLINE EQUATIONS
C ***
   34 DIM1=0.
      ICOUNT=ICOUNT+1
      DO 40 I=1,N
      PI=P(I)
      HI=H(I)
      PIHI=PI*HI
      PI2=PI*PI
      IF(PIHI.GT.ETA) GO TO 39
C *** USE POWER SERIES REPRESENTATION
      PI2HI2=PI2*HI*HI
      PI4HI4=PI2HI2*PI2HI2
      E(I)=(1.+CONST3*PI2HI2+CONST4*PI4HI4)*HI*CONST1
      DI=(1.+CONST5*PI2HI2+CONST6*PI4HI4)*HI*CONST2
      DIAG(I)=DIM1+DI
      GO TO 40
C *** USE HYPERBOLIC FUNCTIONS
   39 ONBYHI=1./HI
      SI=SINH(PIHI)
      CI=COSH(PIHI)
      PIBYSI=PI/SI
      E(I)=(ONBYHI-PIBYSI)/PI2
      DI=(PIBYSI*CI-ONBYHI)/PI2
      DIAG(I)=DIM1+DI
```

```
   40 DIM1=DI
      DIAG(NP1)=DI
      E(NP1)=0.
C ***
C *** ALTER SPLINE EQUATIONS
C ***
      IF(B(1).EQ.0.) E(1)=0.
      IF(N.EQ.1) GO TO 46
      DO 45 I=2,N
      IF(B(I).NE.0.) GO TO 45
      E(I-1)=0.
      E(I)=0.
   45 CONTINUE
   46 IF(E(NP1).EQ.0.) E(N)=0.
C ***
C *** SOLVE SPLINE EQUATIONS (SEE AHLBERG, NILSON AND WALSH)
C ***
      PK=DIAG(1)
      G(1)=-E(1)/PK
      U(1)=B(1)/PK
      DO 49 KK=2,NP1
      PK=E(KK-1)*G(KK-1)+DIAG(KK)
      G(KK)=-E(KK)/PK
   49 U(KK)=(B(KK)-E(KK-1)*U(KK-1))/PK
      TAUDP(NP1)=U(NP1)
      DO 51 L=1,N
      KK=NP1-L
   51 TAUDP(KK)=G(KK)*TAUDP(KK+1)+U(KK)
C ***
C *** PLOT EXPONENTIAL SPLINE
C ***
   54 CONTINUE
      IF(K.EQ.0) GO TO 99
      IF(ICOUNT.GE.ITMAX) GO TO 2000
C ***
C *** UPDATE P'S
C ***
      IFLAG=0
      DO 80 I=1,K
      IL=ILIML(I)
      IU=ILIMU(I)
      TEMP=TAUDP(IL)*B(IL)
      IF(TEMP.GE.0.) GO TO 64
C *** B=TAUDP<0 AT LEFT HAND ENDPOINT
      IFLAG=1
      LAMBAR=AMAX1(ABS(P(IL)),DIAG(IL)*ABS(TAUDP(IL)))/ABS(TAUDP(IL+1))
      PTILDA=1./SQRT(LAMBAR**4(IL))
      PTILDA=AMAX1(PTILDA,P(IL))
      P(IL)=P(IL)+OMEGA*(PTILDA-P(IL))
   64 ILP1=IL+1
      IUM1=IU-1
      IF(IUM1.LT.ILP1) GO TO 70
      DO 68 J=ILP1,IUM1
      IF(TAUDP(J).NE.0.) GO TO 68
C *** TAUDP=0 AT AN INTERIOR POINT
      IFLAG=1
```

```
      P(J-1)=P(J-1)+EPS
      P(J)=P(J)+EPS
      GO TO 68
   66 TEMP=TAUDP(J)*B(J)
      IF(TEMP.GT.0.) GO TO 68
C *** B*TAUDP<0 AT AN INTERIOR POINT
      IFLAG=1
      LAMBAR=AMAX1(ABS(B(J)),DIAG(J)*ABS(TAUDP(J)))
     1/(2.*AMAX1(ABS(TAUDP(J-1)),ABS(TAUDP(J+1))))
      PTILDA=1./SQRT(LAMBAR**4(J-1))
      PTILDA=AMAX1(PTILDA,P(J-1))
      P(J-1)=P(J-1)+OMEGA*(PTILDA-P(J-1))
      PTILDA=1./SQRT(LAMBAR**4(J))
      PTILDA=AMAX1(PTILDA,P(J))
      P(J)=P(J)+OMEGA*(PTILDA-P(J))
   68 CONTINUE
   70 TEMP=TAUDP(IU)*B(IU)
      IF(TEMP.GE.0.) GO TO 80
C *** B*TAUDP<0 AT RIGHT HAND ENDPOINT
      IFLAG=1
      LAMBAR=AMAX1(ABS(B(IU)),DIAG(IU)*ABS(TAUDP(IU)))/ABS(TAUDP(IUM1))
      PTILDA=1./SQRT(LAMBAR**4(IUM1))
      PTILDA=AMAX1(PTILDA,P(IUM1))
      P(IUM1)=P(IUM1)+OMEGA*(PTILDA-P(IUM1))
   80 CONTINUE
C ***
C *** CHECK FOR EXTRANEOUS INFLECTION POINTS
C ***
      IF(IFLAG.EQ.0) GO TO 99
C ***
C *** SCALE ABSCISSAE
C ***
      XMU=0.
      DO 93 I=1,N
   93 XMU=AMAX1(XMU,P(I)*H(I))
      ALPHAC=AMAX1(XMU/SIGMA,1.)
      ALPHA=ALPHA*ALPHAC
      DO 94 I=1,N
      H(I)=H(I)/ALPHAC
   94 B(I)=B(I)*ALPHAC
      B(NP1)=B(NP1)*ALPHAC
      GO TO 34
C ***
C *** NO EXTRANEOUS INFLECTION POINTS
C ***
   99 CONTINUE
      GO TO 3000
C ***
C *** MAXIMUM NUMBER OF ITERATIONS EXCEEDED
C ***
 2000 CONTINUE
 3000 RETURN
      END
```

```
C ***
C ***
C *** ESEVAL: B. J. MCCARTIN  10/80
C ***
C ***
C *** FUNCTION: EVALUATES THE EXPONENTIAL SPLINE
C ***
C *** DESCRIPTION OF CALLING ARGUMENTS:
C ***       X       = ABSCISSAE
C ***       F       = ORDINATES
C ***       NP1     = NUMBER OF ORDERED PAIRS OF DATA
C ***       B       = RIGHT HAND SIDE OF SPLINE EQUATIONS
C ***       TAUDP   = SOLUTION OF SPLINE EQUATIONS
C ***       P       = EXPONENTIAL SPLINE TENSION PARAMETERS
C ***       H       = DELTA X
C ***       XBAR    = ABSCISSA OF EVALUATION
C ***       TAUBAR  = ORDINATE OF EVALUATION
C ***       TAUBP   = DERIVATIVE OF EVALUATION
C ***       ALPHA   = SCALING PARAMETER
C ***
      SUBROUTINE ESEVAL(X,F,NP1,B,TAUDP,P,H,XBAR,TAUBAR,TAUBP,TAUBPP,TBP
     1PP,ALPHA)
      DIMENSION X(1),F(1),B(1),TAUDP(1),P(1),H(1)
C ***
C *** INITIALIZATION
C ***
      CONST1=1./6.
      CONST2=1./3.
      CONST3=7./60.
      CONST4=31./2520.
      CONST5=-1./15.
      CONST6=2./315.
      CONST7=-7./195.
      CONST8=-.05
      CONST9=1./120.
      N=NP1-1
      ETA=.07
      SCALE=2./(X(NP1)-X(1))
C ***
C *** DETERMINE INTERVAL
C ***
      DO 10 J=1,N
      I=J
      IF(XBAR.LT.X(J+1)) GO TO 11
   10 CONTINUE
C ***
C *** DETERMINE TYPE OF FIT
C ***
   11 TEMP=B(I)*B(I+1)
      IF(TEMP.NE.0.) GO TO 20
C *** FIT WITH LINEAR FUNCTION
      TAUBAR=(F(I)*(X(I+1)-XBAR)+F(I+1)*(XBAR-X(I)))/(H(I)*ALPHA)
      TAUBAR=SCALE*TAUBAR
      TAUBP=(F(I+1)-F(I))/(H(I)*ALPHA)
      TAUBP=SCALE*TAUBP
```

```
      TAUBPP=0.
      TBPPP=0.
      GO TO 30
C *** FIT WITH EXPONENTIAL SPLINE
   20 X1=SCALE*(X(I+1)-XBAR)/ALPHA
      X2=SCALE*(XBAR-X(I))/ALPHA
      PI2=P(I)*P(I)
      PIHI=P(I)*H(I)
      IF(PIHI.GT.ETA) GO TO 25
C *** USE POWER SERIES REPRESENTATION
      PI2HI2=PIHI*PIHI
      PI4HI4=PI2HI2*PI2HI2
      X12=X1*X1
      X22=X2*X2
      X14=X12*X12
      X24=X22*X22
      CONBHI2=1./(H(I)*H(I))
      TERM1=1.+CONST3*PI2HI2+CONST4*PI4HI4
      TERM2=CONST1*PI2+CONST6*PI2HI2+CONST7*PI2*PI2HI2
      TERM3=(CONST5*CONBHI2+CONST9*PI2)*PI2
      TAUBAR=(P(I)*X1+P(I+1)*X2)/H(I)-CONST1*H(I)*
     1(X1*TAUDP(I)*(TERM1+TERM2*X12+TERM3*X14)+
     2X2*TAUDP(I+1)*(TERM1+TERM2*X22+TERM3*X24))
      TAUBP=(P(I+1)-P(I))/H(I)-CONST1*H(I)*
     1(TAUDP(I+1)*(TERM1+3.*TERM2*X22+5.*TERM3*X24)-
     2TAUDP(I)*(TERM1+3.*TERM2*X12+5.*TERM3*X14))
      TAUBPP=SCALE*TAUBP/ALPHA
      TAUBPP=(SCALE/ALPHA)**2*CONST1*H(I)*(TAUDP(I)*X1*(6.*TERM2+20.*TER
     1M3*X12)-TAUDP(I+1)*X2*(6.*TERM2+20.*TERM3*X22))
      TBPPP=-(SCALE/ALPHA)**3*CONST1*H(I)*(6.*TAUDP(I)*(TERM2+
     110.*TERM3*X12)+6.*TAUDP(I+1)*(TERM2+10.*TERM3*X22))
      GO TO 30
C *** USE HYPERBOLIC FUNCTIONS
   25 DENOM=PI2*SINH(P(I)*H(I))
      TAUBAR=(TAUDP(I)*SINH(P(I)*X1)+TAUDP(I+1)*SINH(P(I)*X2))/DENOM
     1+((F(I)-TAUDP(I)/PI2)*X1+(F(I+1)-TAUDP(I+1)/PI2)*X2)/H(I)
      TAUBP=(TAUDP(I+1)*COSH(P(I)*X2)-TAUDP(I)*COSH(P(I)*X1))*P(I)/DENOM
     1+(F(I+1)-F(I)+(TAUDP(I)-TAUDP(I+1))/PI2)/H(I)
      TAUBP=SCALE*TAUBP/ALPHA
      TAUBPP=(SCALE*P(I)/ALPHA)**2/DENOM*(TAUDP(I+1)*SINH(P(I)*X2)+TAUDP
     1(I)*SINH(P(I)*X1))
      TBPPP=(SCALE*P(I)/ALPHA)**3/DENOM*(TAUDP(I+1)*COSH(P(I)*X2)
     1-TAUDP(I)*COSH(P(I)*X1))
   30 CONTINUE
      RETURN
      END



      SUBROUTINE EVALF(X,F,NP1,B,TAUDP,P,H,ALPHA,FP,DELTA,IFLAG,L)
      DIMENSION X(1),F(1),B(1),TAUDP(1),P(1),H(1),FP(40,14)
      COMMON CONST1,CONST2,CONST3,CONST4,CONST5,CONST6,CONST7,CONST8,CON
     1ST9
C ***
C *** INITIALIZATION
```

```
C ***
      N=NP1-1
      ETA=.07
      SCALE=2./(X(NP1)-X(1))
      DO 100 K=1,NP1
      XBAR=X(K)
C ***
C *** DETERMINE INTERVAL
C ***
      I=K
      IF(I.GE.NP1/2) I=I-1
      IF(I.EQ.NP1) I=N
C ***
C *** DETERMINE TYPE OF FIT
C ***
   11 TEMP=B(I)*B(I+1)
      IF(TEMP.NE.0.) GO TO 20
C *** FIT WITH LINEAR FUNCTION
      TAUBP=(P(I+1)-P(I))/(H(I)*ALPHA)
      TAUBP=SCALE*TAUBP
      GO TO 30
C *** FIT WITH EXPONENTIAL SPLINE
   20 X1=SCALE*(X(I+1)-XBAR)/ALPHA
      X2=SCALE*(XBAR-X(I))/ALPHA
      PI2=P(I)*P(I)
      PIHI=P(I)*H(I)
      IF(PIHI.GT.ETA) GO TO 25
C *** USE POWER SERIES REPRESENTATION
      PI2HI2=PIHI*PIHI
      PI4HI4=PI2HI2*PI2HI2
      X12=X1*X1
      X22=X2*X2
      X14=X12*X12
      X24=X22*X22
      CONSHI2=1./(H(I)*H(I))
      TERM1=1.+CONST3*PI2HI2+CONST4*PI4HI4
      TERM2=CONST1*PI2-CONST2*PI2HI2+CONST7*PI2*PI2HI2
      TERM3=(CONST5*CONST2*HI2+CONST9*PI2)*PI2
      TAUBP=(F(I+1)-F(I))/H(I)-CONST1*H(I)*
     1(TAUBP(I+1)*(TERM1+3.*TERM2*X22+5.*TERM3*X24)-
     2TAUBP(I)*(TERM1+3.*TERM2*X12+5.*TERM3*X14))
      TAUBP=SCALE*TAUBP/ALPHA
      GO TO 30
C *** USE HYPERBOLIC FUNCTIONS
   25 DENOM=PI2*SINH(P(I)*H(I))
      TAUBP=(TAUBP(I+1)*COSH(P(I)*X2)-TAUBP(I)*COSH(P(I)*X1))*P(I)/DENOM
     1+(F(I+1)-F(I)+(TAUBP(I)-TAUBP(I+1))/PI2)/H(I)
      TAUBP=SCALE*TAUBP/ALPHA
   30 CONTINUE
      IF(IFLAG.EQ.1) FP(K,L)=TAUBP
      IF(IFLAG.EQ.2) FP(K,L)=FP(K,L)+TAUBP
C     GO TO 100
   31 IF(K.EQ.1.OR.K.EQ.NP1) GO TO 100
      XBAR=X(K)+.5*DELTA
C ***
C *** DETERMINE INTERVAL
C ***
```

```
      I=K
C ***
C *** DETERMINE TYPE OF FIT
C ***
  111 TEMP=B(I)*B(I+1)
      IF(TEMP.NE.0.) GO TO 120
C *** FIT WITH LINEAR FUNCTION
      TAUBP=(F(I+1)-F(I))/(H(I)*ALPHA)
      TAUBP=SCALE*TAUBP
      TAUBPP=0.
      GO TO 130
C *** FIT WITH EXPONENTIAL SPLINE
  120 X1=SCALE*(X(I+1)-XBAR)/ALPHA
      X2=SCALE*(XBAR-X(I))/ALPHA
      PI2=P(I)*P(I)
      PIHI=P(I)*H(I)
      IF(PIHI.GT.ETA) GO TO 125
C *** USE POWER SERIES REPRESENTATION
      PI2HI2=PIHI*PIHI
      PI4HI4=PI2HI2*PI2HI2
      X12=X1*X1
      X22=X2*X2
      X14=X12*X12
      X24=X22*X22
      CONSHI2=1./(H(I)*H(I))
      TERM1=1.+CONST3*PI2HI2+CONST4*PI4HI4
      TERM2=CONST1*PI2-CONST2*HI2+CONST7*PI2*PI2HI2
      TERM3=(CONST8*CONSHI2+CONST9*PI2)*PI2
      TAUBP=(F(I+1)-F(I))/H(I)-CONST1*H(I)*
     1(TAUDP(I+1)*(TERM1+3.*TERM2*X22+5.*TERM3*X24)-
     2TAUDP(I)*(TERM1+3.*TERM2*X12+5.*TERM3*X14))
      TAUBP=SCALE*TAUBP/ALPHA
      TAUBPP=(SCALE/ALPHA)**2*CONST1*H(I)*(TAUDP(I)*X1*(6.*TERM2+20.*TER
     1M3*X12)-TAUDP(I+1)*X2*(6.*TERM2+20.*TERM3*X22))
      GO TO 130
C *** USE HYPERBOLIC FUNCTIONS
  125 DENOM=PI2*SINH(P(I)*H(I))
      TAUBP=(TAUDP(I+1)*COSH(P(I)*X2)-TAUDP(I)*COSH(P(I)*X1))*P(I)/DENOM
     1+(F(I+1)-F(I)+(TAUDP(I)-TAUDP(I+1))/PI2)/H(I)
      TAUBP=SCALE*TAUBP/ALPHA
      TAUBPP=(SCALE*P(I)/ALPHA)**2/DENOM*(TAUDP(I+1)*SINH(P(I)*X2)+TAUDP
     1(I)*SINH(P(I)*X1))
  130 CONTINUE
      EPS=DELTA*DELTA*ABS(TAUBPP)
      IF(EPS.GT..5) EPS=.5
      IF(IFLAG.EQ.1) FP(K,L)=FP(K,L)-EPS*TAUBP
      IF(IFLAG.EQ.2) FP(K,L)=FP(K,L)+EPS*TAUBP
      XBAR=X(K)-.5*DELTA
C ***
C *** DETERMINE INTERVAL
C ***
      I=K-1
C ***
C *** DETERMINE TYPE OF FIT
C ***
  211 TEMP=B(I)*B(I+1)
      IF(TEMP.NE.0.) GO TO 220
```

```
C *** FIT WITH LINEAR FUNCTION
      TAUDP=(F(I+1)-F(I))/(H(I)*ALPHA)
      TAUBP=SCALE*TAUDP
      TAUBPP=0.
      GO TO 230
C *** FIT WITH EXPONENTIAL SPLINE
  220 X1=SCALE*(X(I+1)-XBAR)/ALPHA
      X2=SCALE*(XBAR-X(I))/ALPHA
      PI2=P(I)*P(I)
      PIHI=P(I)*H(I)
      IF(PIHI.GT.ETA) GO TO 225
C *** USE POWER SERIES REPRESENTATION
      PI2HI2=PIHI*PIHI
      PI4HI4=PI2HI2*PI2HI2
      X12=X1*X1
      X22=X2*X2
      X14=X12*X12
      X24=X22*X22
      ONEHI2=1./(H(I)*H(I))
      TERM1=1.+CONST3*PI2HI2+CONST4*PI4HI4
      TERM2=CONST1*PI2-ONEHI2+CONST7*PI2*PI2HI2
      TERM3=(CONST5*ONEHI2+CONST9*PI2)*PI2
      TAUBP=(F(I+1)-F(I))/H(I)-CONST1*H(I)*
     1(TAUDP(I+1)*(TERM1+3.*TERM2*X22+5.*TERM3*X24)-
     2TAUDP(I)*(TERM1+3.*TERM2*X12+5.*TERM3*X14))
      TAUBP=SCALE*TAUBP/ALPHA
      TAUBPP=(SCALE/ALPHA)**2*CONST1*H(I)*(TAUDP(I)*X1*(6.*TERM2+20.*TER
     1M3*X12)-TAUDP(I+1)*X2*(6.*TERM2+20.*TERM3*X22))
      GO TO 230
C *** USE HYPERBOLIC FUNCTIONS
  225 DENOM=PI2*SINH(P(I)*H(I))
      TAUBP=(TAUDP(I+1)*COSH(P(I)*X2)-TAUDP(I)*COSH(P(I)*X1))*P(I)/DENOM
     1+(F(I+1)-F(I)+(TAUDP(I)-TAUDP(I+1))/PI2)/H(I)
      TAUBP=SCALE*TAUBP/ALPHA
      TAUBPP=(SCALE*F(I)/ALPHA)**2/DENOM*(TAUDP(I+1)*SINH(P(I)*X2)+TAUDP
     1(I)*SINH(P(I)*X1))
  230 CONTINUE
      EPS=DELTA*DELTA*ABS(TAUBPP)
      IF(EPS.GT..5) EPS=.5
      IF(IFLAG.EQ.1) FP(K,L)=FP(K,L)+EPS*TAUBP
      IF(IFLAG.EQ.2) FP(K,L)=FP(K,L)-EPS*TAUBP
  100 CONTINUE
      RETURN
      END



      SUBROUTINE EVALG(X,F,NP1,B,TAUDP,P,H,ALPHA,FP,DELTA,IFLAG,L)
      DIMENSION X(1),F(1),B(1),TAUDP(1),P(1),H(1),FP(40,14)
      COMMON CONST1,CONST2,CONST3,CONST4,CONST5,CONST6,CONST7,CONST8,CON
     1ST9
C ***
C *** INITIALIZATION
C ***
      N=NP1-1
```

```
      ETA=.07
      SCALE=2./(X(NP1)-X(1))
      DO 100 K=1,NP1
      XBAR=X(K)
C ***
C *** DETERMINE INTERVAL
C ***
      I=K
      IF(I.GE.NP1/2) I=I-1
      IF(I.EQ.NP1) I=N
C ***
C *** DETERMINE TYPE OF FIT
C ***
   11 TEMP=B(I)*B(I+1)
      IF(TEMP.NE.0.) GO TO 20
C *** FIT WITH LINEAR FUNCTION
      TAUBP=(F(I+1)-F(I))/(H(I)*ALPHA)
      TAUBP=SCALE*TAUBP
      GO TO 30
C *** FIT WITH EXPONENTIAL SPLINE
   20 X1=SCALE*(X(I+1)-XBAR)/ALPHA
      X2=SCALE*(XBAR-X(I))/ALPHA
      PI2=P(I)*P(I)
      PIHI=P(I)*H(I)
      IF(PIHI.GT.ETA) GO TO 25
C *** USE POWER SERIES REPRESENTATION
      PI2HI2=PIHI*PIHI
      PI4HI4=PI2HI2*PI2HI2
      X12=X1*X1
      X22=X2*X2
      X14=X12*X12
      X24=X22*X22
      ONEHI2=1./(H(I)*H(I))
      TERM1=1.+CONST3*PI2HI2+CONST4*PI4HI4
      TERM2=CONST1*PI2-ONEHI2+CONST7*PI2*PI2HI2
      TERM3=(CONST8*ONEHI2+CONST9*PI2)*PI2
      TAUBP=(F(I+1)-F(I))/H(I)-CONST1*H(I)*
     1(TAUDP(I+1)*(TERM1+3.*TERM2*X22+5.*TERM3*X24)-
     2TAUDP(I)*(TERM1+3.*TERM2*X12+5.*TERM3*X14))
      TAUBP=SCALE*TAUBP/ALPHA
      GO TO 30
C *** USE HYPERBOLIC FUNCTIONS
   25 DENOM=PI2*SINH(P(I)*H(I))
      TAUBP=(TAUDP(I+1)*COSH(P(I)*X2)-TAUDP(I)*COSH(P(I)*X1))*P(I)/DENOM
     1+(F(I+1)-F(I)+(TAUDP(I)-TAUDP(I+1))/PI2)/H(I)
      TAUBP=SCALE*TAUBP/ALPHA
   30 CONTINUE
      IF(IFLAG.EQ.1) FP(L,K)=TAUBP
      IF(IFLAG.EQ.2) FP(L,K)=FP(L,K)+TAUBP
C     GO TO 100
   31 IF(K.EQ.1.OR.K.EQ.NP1) GO TO 100
      XBAR=X(K)+.5*DELTA
C ***
C *** DETERMINE INTERVAL
C ***
      I=K
C ***
```

```
C *** DETERMINE TYPE OF FIT
C ***
  111 TEMP=B(I)*B(I+1)
      IF(TEMP.NE.0.) GO TO 120
C *** FIT WITH LINEAR FUNCTION
      TAUBP=(F(I+1)-F(I))/(H(I)*ALPHA)
      TAUBP=SCALE*TAUBP
      TAUBPP=0.
      GO TO 130
C *** FIT WITH EXPONENTIAL SPLINE
  120 X1=SCALE*(X(I+1)-XBAR)/ALPHA
      X2=SCALE*(XBAR-X(I))/ALPHA
      PI2=P(I)*P(I)
      PIHI=P(I)*H(I)
      IF(PIHI.GT.ETA) GO TO 125
C *** USE POWER SERIES REPRESENTATION
      PI2HI2=PIHI*PIHI
      PI4HI4=PI2HI2*PI2HI2
      X12=X1*X1
      X22=X2*X2
      X14=X12*X12
      X24=X22*X22
      ONEHI2=1./(H(I)*H(I))
      TERM1=1.+CONST3*PI2HI2+CONST4*PI4HI4
      TERM2=CONST1*PI2-ONEHI2+CONST7*PI2*PI2HI2
      TERM3=(CONST3*ONEHI2+CONST9*PI2)*PI2
      TAUBP=(F(I+1)-F(I))/H(I)-CONST1*H(I)*
     1(TAUDP(I+1)*(TERM1+3.*TERM2*X22+5.*TERM3*X24)-
     2TAUDP(I)*(TERM1+3.*TERM2*X12+5.*TERM3*X14))
      TAUBP=SCALE*TAUBP/ALPHA
      TAUBPP=(SCALE/ALPHA)**2*CONST1*H(I)*(TAUDP(I)*X1*(6.*TERM2+20.*TER
     1M3*X12)-TAUDP(I+1)*X2*(6.*TERM2+20.*TERM3*X22))
      GO TO 130
C *** USE HYPERBOLIC FUNCTIONS
  125 DENOM=PI2*SINH(P(I)*H(I))
      TAUBP=(TAUDP(I+1)*COSH(P(I)*X2)-TAUDP(I)*COSH(P(I)*X1))*P(I)/DENOM
     1+(F(I+1)-F(I)+(TAUDP(I)-TAUDP(I+1))/PI2)/H(I)
      TAUBP=SCALE*TAUBP/ALPHA
      TAUBPP=(SCALE*P(I)/ALPHA)**2/DENOM*(TAUDP(I+1)*SINH(P(I)*X2)+TAUDP
     1(I)*SINH(P(I)*X1))
  130 CONTINUE
      EPS=DELTA*DELTA*ABS(TAUBPP)
      IF(EPS.GT..5) EPS=.5
      IF(IFLAG.EQ.1) FP(L,K)=FP(L,K)-EPS*TAUBP
      IF(IFLAG.EQ.2) FP(L,K)=FP(L,K)+EPS*TAUBP
      XBAR=X(K)-.5*DELTA
C ***
C *** DETERMINE INTERVAL
C ***
      I=K-1
C ***
C *** DETERMINE TYPE OF FIT
C ***
  211 TEMP=B(I)*B(I+1)
      IF(TEMP.NE.0.) GO TO 220
C *** FIT WITH LINEAR FUNCTION
      TAUBP=(F(I+1)-F(I))/(H(I)*ALPHA)
```

```
      TAUBP=SCALE*TAUBP
      TAUBPP=0.
      GO TO 230
C *** FIT WITH EXPONENTIAL SPLINE
  220 X1=SCALE*(X(I+1)-XBAR)/ALPHA
      X2=SCALE*(XBAR-X(I))/ALPHA
      PI2=P(I)*P(I)
      PIHI=P(I)*H(I)
      IF(PIHI.GT.ETA) GO TO 225
C *** USE POWER SERIES REPRESENTATION
      PI2HI2=PIHI*PIHI
      PI4HI4=PI2HI2*PI2HI2
      X12=X1*X1
      X22=X2*X2
      X14=X12*X12
      X24=X22*X22
      ONEHI2=1./(H(I)*H(I))
      TERM1=1.+CONST3*PI2HI2+CONST4*PI4HI4
      TERM2=CONST1*PI2-CONST2*PI2HI2+CONST7*PI2*PI2HI2
      TERM3=(CONST5*ONEHI2+CONST9*PI2)*PI2
      TAUBP=(F(I+1)-F(I))/H(I)-CONST1*H(I)*
     1(TAUDP(I+1)*(TERM1+3.*TERM2*X22+5.*TERM3*X24)-
     2TAUDP(I)*(TERM1+3.*TERM2*X12+5.*TERM3*X14))
      TAUBP=SCALE*TAUBP/ALPHA
      TAUBPP=(SCALE/ALPHA)**2*CONST1*H(I)*(TAUDP(I)*X1*(6.*TERM2+20.*TER
     1M3*X12)-TAUDP(I+1)*X2*(6.*TERM2+20.*TERM3*X22))
      GO TO 230
C *** USE HYPERBOLIC FUNCTIONS
  225 DENOM=PI2*SINH(P(I)*H(I))
      TAUBP=(TAUDP(I+1)*COSH(P(I)*X2)-TAUDP(I)*COSH(P(I)*X1))*P(I)/DENOM
     1+(F(I+1)-F(I)+(TAUDP(I)-TAUDP(I+1))/PI2)/H(I)
      TAUBP=SCALE*TAUBP/ALPHA
      TAUBPP=(SCALE*P(I)/ALPHA)**2/DENOM*(TAUDP(I+1)*SINH(P(I)*X2)+TAUDP
     1(I)*SINH(P(I)*X1))
  230 CONTINUE
      EPS=DELTA*DELTA*ABS(TAUBPP)
      IF(EPS.GT..5) EPS=.5
      IF(IFLAG.EQ.1) FP(L,K)=FP(L,K)+EPS*TAUBP
      IF(IFLAG.EQ.2) FP(L,K)=FP(L,K)-EPS*TAUBP
  100 CONTINUE
      RETURN
      END



      SUBROUTINE SPLEXP(X,F,FPA,FPB,NP1,B,TAUDP,P,H,ALPHA)
      DIMENSION X(2),F(4),B(1),TAUDP(1),P(1),H(1)
      DIMENSION E(100),DIAG(100),ILIML(100),ILIMU(100),Q(100),U(100)
      COMMON CONST1,CONST2,CONST3,CONST4,CONST5,CONST6,CONST7,CONST8,CON
     1ST9
      REAL LAMBAR
C ***
C *** INITIALIZATION
C ***
      HH=1./(6.*(X(2)-X(1)))
```

```
      A0=-11.*HH
      A1=18.*HH
      A2=-9.*HH
      A3=2.*HH
      FPA=A0*F(1)+A1*F(2)+A2*F(3)+A3*F(4)
      FPB=-(A3*F(NP1-3)+A2*F(NP1-2)+A1*F(NP1-1)+A0*F(NP1))
      IF(ABS(FPA).LT.1.E-10) FPA=0.
      IF(ABS(FPB).LT.1.E-10) FPB=0.
      N=NP1-1
      EPS=1.E-6

      ETA=.07
      SIGMA=100.
      ALPHA=1.
      ITMAX=5
      SCALE=2./(X(NP1)-X(1))
      ICOUNT=0
      DO 4 I=1,N
      H(I)=X(I+1)-X(I)
    4 H(I)=SCALE*H(I)
      DO 5 I=1,N
    5 P(I)=0.
C ***
C *** DEFINE B'S
C ***
      B(1)=(F(2)-F(1))/H(1)-FPA/SCALE
      IF(N.EQ.1) GO TO 11
      DO 10 I=2,N
   10 B(I)=(F(I+1)-F(I))/H(I)-(F(I)-F(I-1))/H(I-1)
   11 B(NP1)=FPB/SCALE-(F(NP1)-F(N))/H(N)
C ***
C *** DETERMINE WHICH INTERVALS SHOULD BE FIT WITH
C *** LINE SEGMENTS AND WHICH INTERVALS SHOULD BE
C *** FIT WITH EXPONENTIAL SPLINES
C ***    K          = NUMBER OF EXPONENTIAL SPLINE INTERVALS
C ***    ILIML(I) = LEFT HAND ENDPOINT OF ITH SPLINE INTERVAL
C ***    ILIMU(I) = RIGHT HAND ENDPOINT OF ITH SPLINE INTERVAL
C ***
      K=0
      IL=1
   14 CONTINUE
      DO 20 I=IL,N
      TEMP=B(I)*B(I+1)
      IF(TEMP.GE.0.) GO TO 20
      K=K+1
      ILIML(K)=I
      DO 16 J=I,N
      TEMP=B(J)*B(J+1)
      IF(TEMP.NE.0.) GO TO 15
      ILIMU(K)=J
      GO TO 29
   15 IF(J.NE.N) GO TO 16
      ILIMU(K)=N+1
      GO TO 30
   16 CONTINUE
   20 CONTINUE
      GO TO 30
```

```
   29 IL=J
      GO TO 14
   30 CONTINUE
C  ***
C  *** SET UP SPLINE EQUATIONS
C  ***
   34 DIM1=0.
      ICOUNT=ICOUNT+1
      DO 40 I=1,N
      PI=P(I)
      HI=H(I)
      PIHI=PI*HI
      PI2=PI*PI
      IF(PIHI.GT.ETA) GO TO 39
C  *** USE POWER SERIES REPRESENTATION
      PI2HI2=PI2*HI*HI
      PI4HI4=PI2HI2*PI2HI2
      E(I)=(1.+CONST3*PI2HI2+CONST4*PI4HI4)*HI*CONST1
      DI=(1.+CONST5*PI2HI2+CONST6*PI4HI4)*HI*CONST2
      DIAG(I)=DIM1+DI
      GO TO 40
C  *** USE HYPERBOLIC FUNCTIONS
   39 ONBYHI=1./HI
      SI=SINH(PIHI)
      CI=COSH(PIHI)
      PIBYSI=PI/SI
      E(I)=(ONBYHI-PIBYSI)/PI2
      DI=(PIBYSI*CI-ONBYHI)/PI2
      DIAG(I)=DIM1+DI
   40 DIM1=DI
      DIAG(NP1)=DI
      E(NP1)=0.
C  ***
C  *** ALTER SPLINE EQUATIONS
C  ***
      IF(B(1).EQ.0.) E(1)=0.
      IF(N.EQ.1) GO TO 46
      DO 45 I=2,N
      IF(B(I).NE.0.) GO TO 45
      E(I-1)=0.
      E(I)=0.
   45 CONTINUE
   46 IF(B(NP1).EQ.0.) E(N)=0.
C  ***
C  *** SOLVE SPLINE EQUATIONS (SEE AHLBERG, NILSON AND WALSH)
C  ***
      PK=DIAG(1)
      Q(1)=-E(1)/PK
      U(1)=B(1)/PK
      DO 49 KK=2,NP1
      PK=E(KK-1)*Q(KK-1)+DIAG(KK)
      Q(KK)=-E(KK)/PK
   49 U(KK)=(B(KK)-E(KK-1)*U(KK-1))/PK
      TAUDP(NP1)=U(NP1)
      DO 51 L=1,N
      KK=NP1-L
   51 TAUDP(KK)=Q(KK)*TAUDP(KK+1)+U(KK)
```

```
C ***
C *** PLOT EXPONENTIAL SPLINE
C ***
   54 CONTINUE
      IF(K.EQ.0) GO TO 3000
      IF(ICOUNT.GE.ITMAX) GO TO 3000
C ***
C *** UPDATE P'S
C ***
      IFLAG=0
      DO 80 I=1,K
      IL=ILIML(I)
      IU=ILIMU(I)
      TEMP=TAUDP(IL)*B(IL)
      IF(TEMP.GE.0.) GO TO 64
C *** B*TAUDP<0 AT LEFT HAND ENDPOINT
      IFLAG=1
      LAMBAR=AMAX1(ABS(B(IL)),DIAG(IL)*ABS(TAUDP(IL)))/ABS(TAUDP(IL+1))
      PTILDA=1./SQRT(LAMBAR*H(IL))
      PTILDA=AMAX1(PTILDA,P(IL))
      P(IL)=P(IL)+OMEGA*(PTILDA-P(IL))
   64 ILP1=IL+1
      IUM1=IU-1
      IF(IUM1.LT.ILP1) GO TO 70
      DO 68 J=ILP1,IUM1
      IF(TAUDP(J).NE.0.) GO TO 66
C *** TAUDP=0 AT AN INTERIOR POINT
      IFLAG=1
      P(J-1)=P(J-1)+EPS
      P(J)=P(J)+EPS
      GO TO 68
   66 TEMP=TAUDP(J)*B(J)
      IF(TEMP.GT.0.) GO TO 68
C *** B*TAUDP<0 AT AN INTERIOR POINT
      IFLAG=1
      LAMBAR=AMAX1(ABS(B(J)),DIAG(J)*ABS(TAUDP(J)))
     1/(2.*AMAX1(ABS(TAUDP(J-1)),ABS(TAUDP(J+1))))
      PTILDA=1./SQRT(LAMBAR*H(J-1))
      PTILDA=AMAX1(PTILDA,P(J-1))
      P(J-1)=P(J-1)+OMEGA*(PTILDA-P(J-1))
      PTILDA=1./SQRT(LAMBAR*H(J))
      PTILDA=AMAX1(PTILDA,P(J))
      P(J)=P(J)+OMEGA*(PTILDA-P(J))
   68 CONTINUE
   70 TEMP=TAUDP(IU)*B(IU)
      IF(TEMP.GE.0.) GO TO 80
C *** B*TAUDP<0 AT RIGHT HAND ENDPOINT
      IFLAG=1
      LAMBAR=AMAX1(ABS(B(IL)),DIAG(IL)*ABS(TAUDP(IU)))/ABS(TAUDP(IUM1))
      PTILDA=1./SQRT(LAMBAR*H(IUM1))
      PTILDA=AMAX1(PTILDA,P(IUM1))
      P(IUM1)=P(IUM1)+OMEGA*(PTILDA-P(IUM1))
   80 CONTINUE
C ***
C *** CHECK FOR EXTRANEOUS INFLECTION POINTS
C ***
      IF(IFLAG.EQ.0) GO TO 3000
```

```
C ***
C *** SCALE ABSCISSAE
C ***
      XMU=0.
      DO 93 I=1,N
   93 XMU=AMAX1(XMU,P(I)*H(I))
      ALPHAC=AMAX1(XMU/SIGMA,1.)
      ALPHA=ALPHA*ALPHAC
      DO 94 I=1,N
      H(I)=H(I)/ALPHAC
   94 B(I)=B(I)*ALPHAC
      B(NP1)=B(NP1)*ALPHAC
      GO TO 34
C ***
C *** NO EXTRANEOUS INFLECTION POINTS
C ***
C ***
C *** MAXIMUM NUMBER OF ITERATIONS EXCEEDED
C ***
 3000 RETURN
      END
```