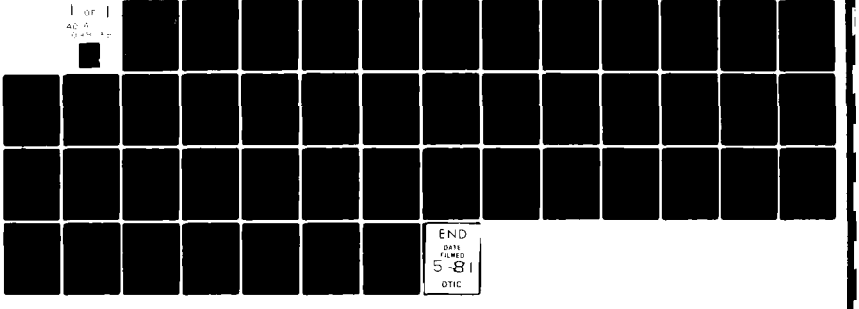
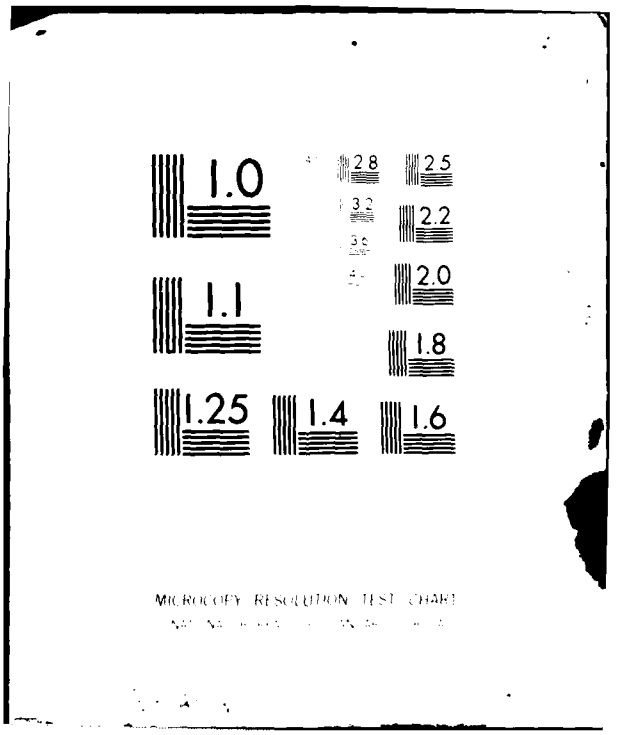
AD-A098 138 RENSSELAER POLYTECHNIC INST TROY NY DEPT OF MATHEMAT--ETC F/G 12/1
AN ADAPTIVE FINITE ELEMENT METHOD FOR INITIAL-BOUNDARY VALUE PR--ETC(U)
MAR 81 S F DAVIS, J E FLAHERTY AFOSR-80-0192

UNCLASSIFIED AFOSR-TR-81-0376 NL




1.0
2.8
2.5
3.2
2.2
3.6
2.0
1.1
1.8
1.25
1.4
1.6
MICROCOPY RESOLUTION TEST CHART

AN ADAPTIVE FINITE ELEMENT METHOD FOR INITIAL-BOUNDARY VALUE PROBLEMS FOR PARTIAL DIFFERENTIAL EQUATIONS*

Stephen F. Davis
Department of Mathematical Sciences
Rensselaer Polytechnic Institute
Troy, New York 12181

Joseph E. Flaherty
Department of Mathematical Sciences
Rensselaer Polytechnic Institute
Troy, New York 12181

and

Institute for Computer Applications in Science and Engineering
NASA Langley Research Center
Hampton, Virginia 23665

ABSTRACT

A finite element method is developed to solve initial-boundary value problems for vector systems of partial differential equations in one space dimension and time. The method automatically adapts the computational mesh as the solution progresses in time and is thus able to follow and resolve relatively sharp transitions such as mild boundary layers, shock layers, or wave fronts. This permits an accurate solution to be calculated with fewer mesh points than would be necessary with a uniform mesh.

(cont'd)

---

*This research was sponsored by the Air Force Office of Scientific Research, Air Force Systems Command, USAF, under Grant Number AFOSR 80-0192. Partial support for work performed under NASA contract NAS1-15810 was also provided while the second author was in residence at ICASE, NASA Langley Research Center. The first author's present address is Ballistic Research Laboratory, U. S. Army Armament Research and Development Command, Aberdeen Proving Ground, Maryland 21005. This work was submitted in partial fulfillment of his Ph.D requirements at the Rensselaer Polytechnic Institute. The United States Government is authorized to reproduce and distribute reprints for government purposes notwithstanding any copyright notation thereon.

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered) UNCLASSIFIED

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER: AFOSR-TR-81-0376	2. GOVT ACCESSION NO: AD-A098 138
3. RECIPIENT'S CATALOG NUMBER	
4. TITLE (and Subtitle): AN ADAPTIVE FINITE ELEMENT METHOD FOR INITIAL - BOUNDARY VALUE PROBLEMS FOR PARTIAL DIFFERENTIAL EQUATIONS	5. TYPE OF REPORT & PERIOD COVERED: Interim
	6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s): Stephen F. Davis and Joseph E. Flaherty	8. CONTRACT OR GRANT NUMBER(s): AFOSR-80-0192
9. PERFORMING ORGANIZATION NAME AND ADDRESS: Rensselaer Polytechnic Institute, Department of Mathematical Sciences, Troy, New York 12181	10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS: 61102F, 9749-03, 93
11. CONTROLLING OFFICE NAME AND ADDRESS: Air Force Office of Scientific Research (NM), Bolling AFB, Washington, DC 20332	12. REPORT DATE: Mar 81
	13. NUMBER OF PAGES: 43
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office)	15. SECURITY CLASS (of this report): UNCLASSIFIED
	15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Finite Elements, Adaptive Grids, Partial Differential Equations, Numerical Analysis, Boundary Layers.

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

A finite element method is developed to solve initial-boundary value problems for vector systems of partial differential equations in one space dimension and time. The method automatically adapts the computational mesh as the solution progresses in

DD FORM 1 JAN 73 1473

UNCLASSIFIED 408 15

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

81 4 22 053

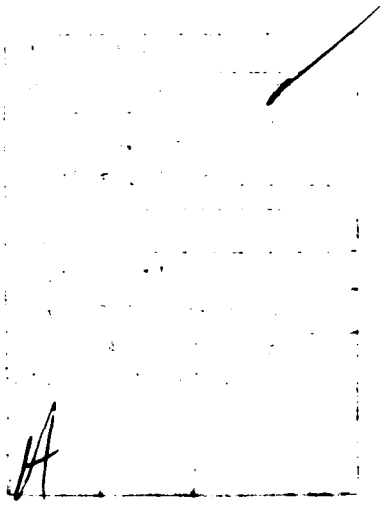
time and ~~is~~ thus able to follow and resolve relatively sharp transitions such as mild boundary layers, shock layers, or wave fronts. This permits an accurate solution to be calculated with fewer mesh points than would be necessary with a uniform mesh.

The overall method contains two parts, a solution algorithm and a mesh selection algorithm. The solution algorithm is a finite element-Galerkin method on trapezoidal space-time elements, using either piecewise linear or cubic polynomial approximations and the mesh selection algorithm builds upon similar work for variable knot spline interpolation.

A computer code implementing these algorithms has been written and applied to a number of problems. These computations confirm that the theoretical error estimates are attained and demonstrate the utility of variable mesh methods for partial differential equations.

The overall method contains two parts, a solution algorithm and a mesh selection algorithm. The solution algorithm is a finite element-Galerkin method on trapezoidal space-time elements, using either piecewise linear or cubic polynomial approximations and the mesh selection algorithm builds upon similar work for variable knot spline interpolation.

A computer code implementing these algorithms has been written and applied to a number of problems. These computations confirm that the theoretical error estimates are attained and demonstrate the utility of variable mesh methods for partial differential equations.



1. Introduction

In this paper we construct an adaptive grid finite element procedure to find numerical solutions of M-dimensional vector systems of partial differential equations having the form

$$\underset{\sim}{L}\underset{\sim}{u} := \underset{\sim}{u}_t + \underset{\sim}{f}(x,t,\underset{\sim}{u},\underset{\sim}{u}_x) - [\underset{\sim}{D}(x,t,\underset{\sim}{u})\underset{\sim}{u}_x]_x = 0,\ 0 < x < a,\ t > 0, \tag{1.1}$$

subject to the initial and linear separated boundary conditions

$$\underset{\sim}{u}(x,0) = \underset{\sim}{u}^0(x),\ 0 \le x \le a, \tag{1.2}$$

$$\begin{aligned} \underset{\sim}{B}_1\underset{\sim}{u}(0,t) &:= \underset{\sim}{A}_{11}(t)\underset{\sim}{u}(0,t) + \underset{\sim}{A}_{12}(t)\underset{\sim}{u}_x(0,t) = \underset{\sim}{b}_1(t), \\ \underset{\sim}{B}_2\underset{\sim}{u}(a,t) &:= \underset{\sim}{A}_{21}(t)\underset{\sim}{u}(a,t) + \underset{\sim}{A}_{22}(t)u_x(a,t) = \underset{\sim}{b}_2(t),\ t > 0. \end{aligned} \tag{1.3}$$

There are k_1 initial boundary conditions at $x = 0$ and k_2 terminal boundary conditions at $x = a$. We are primarily concerned with solving diffusion problems where $\underset{\sim}{D}$ is positive definite and $k_1 + k_2 = 2M$; however, we will not restrict ourselves to this case, but instead we assume that conditions are specified so that (1.1)-(1.3) has an isolated solution.

Problems of the above form arise in many applications which model problems as diverse as heat conduction (cf. Friedman [16]), determining bacterial motion (cf. Keller and Odell [25,30]), combustion (cf. Kapila [24]), chemical reactions (cf. Fife [14]), population dynamics (cf. Hoppensteadt [20]), and convecting flows (cf. Batchelor [4]). Therefore, a general purpose code to solve (1.1)-(1.3) numerically would be extremely useful.

Many of the problems mentioned above have solutions which contain sharp transitions such as boundary layers, shock layers, or wave fronts. In order to resolve such nonuniformities using a minimum number of mesh points it is desirable to concentrate the mesh within the transition layers. Since these transition layers can move, it is all the more desirable for the mesh to adapt itself with the evolving solution. To do this we develop methods that (i) discretize

(1.1)-(1.3) on a nonuniform mesh and (ii) determine the proper mesh point locations.

We discretize (1.1)-(1.3) using a finite element Galerkin method on trapezoidal space-time elements. This approach is similar to that of Jamet and Bonnerot [6,22] and it was chosen because it is generally easier to generate high order approximations to partial differential equations on a nonuniform mesh with finite element methods than with finite difference methods. The accuracy and order of convergence of our methods are analyzed in Davis [11] and are demonstrated experimentally in Section 4 of this paper.

Adaptive mesh selection strategies typically involve some recomputation of the solution. That is, an initial solution is computed on a coarse mesh and this is used to determine whether to add mesh points to some portion of the domain and redo the calculation, redo the calculation using a more accurate method, redo the computation using some combination of these methods, or accept the present computation. Algorithms of this general type have been developed and successfully applied to adaptive quadrature (cf. eg. Rice [33] and Lyness and Kaganove [28]), two-point boundary value problems (cf. eg. Childs et al [9]), elliptic boundary value problems (cf. eg. Carey [8] and Brandt [7]), and parabolic and hyperbolic problems (cf. eg. Berger et al [5] and White [37]).

Primarily because of the expense involved in recomputing the solution of the partial differential equations at possibly every time step we have developed an algorithm which initially places a fixed number of mesh points in optimal locations and then attempts to move them so that their locations remain optimal. Algorithms of this type have been used by Lawson [26], deBoor [12,13], and Jupp [23] for variable knot spline interpolation and it is their work that motivated our mesh selection algorithms.

A different approach to this problem was proposed by Miller and Miller [29] and later extended by Galinas et al [17]. They approximated the solution of

parabolic partial differential equations by piecewise linear polynomials where both the polynomial coefficients and the mesh on which they were defined were unknown functions of time. These functions were determined by minimizing the least squares residuals. They found that the mesh points could coalesce in certain situations and they avoided this by adding a number of spring and damping terms as constraints to the equations.

One advantage of the above approach is that it readily extends to higher dimensional problems. However, we are not convinced that it is necessary to couple the solution and mesh selection methods. This can dramatically increase the size of the discrete system without offering any corresponding increase in order of accuracy. Furthermore, the entire solution procedure must halt if an acceptable mesh cannot be calculated. Under the same circumstances our methods can continue to compute a solution on a suboptimal mesh. Since both methods are under development we have not attempted any detailed comparisons.

In Section 2 of this paper we develop a finite element Galerkin approximation to (1.1)-(1.3) using trapezoidal space-time elements. In Section 3 we describe a practical and efficient mesh selection procedure that approximately minimizes the L_2 error of the computed solution. In Section 4 we apply the method to a number of problems and discuss the computed results. Finally, in Section 5 we present an overall discussion of this effort and some suggestions for future work.

2. Finite Element Formulation

We discretize (1.1,2,3) using a finite element-Galerkin procedure. To this end, let S_n be the strip

$$S_n := \{(x,t) \mid 0 \le x \le a,\ t_n \le t \le t_{n+1}\}, \tag{2.1}$$

choose "test" or "weight" functions $v(x,t) \in C^0(S_n)$, multiply (1.1) by v, integrate over S_n, and integrate the time derivative and diffusive terms by parts to get

$$\begin{aligned} \underset{\sim}{F}(\underset{\sim}{u},v) := & \int_{t_n}^{t_{n+1}} \int_0^a \{-\underset{\sim}{u}\, v_t + \underset{\sim}{f}(x,t,\underset{\sim}{u},\underset{\sim}{u}_x)v + \underset{\sim}{D}(x,t,\underset{\sim}{u})\underset{\sim}{u}_x v_x\}dxdt \\ & + \int_0^a \underset{\sim}{u}\, v\, dx\Big|_{t=t_n}^{t_{n+1}} - \int_{t_n}^{t_{n+1}} \underset{\sim}{D}\, \underset{\sim}{u}_x v\, dt\Big|_{x=0}^{a} = 0. \end{aligned} \tag{2.2}$$

Equation (2.2) is called the Galerkin form of the problem and any function $\underset{\sim}{u}$ that satisfies (2.1) and the initial and boundary conditions (1.2,3) is called a "weak solution."

We introduce a mesh $\{0 = x_1^n < x_2^n < \ldots < x_N^n = a\}$ at $t = t_n$ and a different mesh $\{0 = x_1^{n+1} < x_2^{n+1} < \cdots < x_N^{n+1} = a\}$ at $t = t_{n+1}$. We connect the corresponding points x_i^n and x_i^{n+1} by straight lines and consequently divide the strip S_n into a set of $N - 1$ trapezoids. We let $x_i(t)$ denote the straight line connecting x_i^n and x_i^{n+1} and T_i^n denote the trapezoid with vertices (x_i^n,t_n), (x_{i+1}^n,t_n), (x_{i+1}^{n+1},t_{n+1}), (x_i^{n+1},t_{n+1}) (cf. Figure 1).

We approximate $\underset{\sim}{u}(x,t)$ on S_n by $\underset{\sim}{U}(x,t) \in \mathcal{U}_K(S_n)$ which has the form

$$\underset{\sim}{U}(x,t) = \sum_{i=1}^{K} \underset{\sim}{c}_i(t)\phi_i(x,t). \tag{2.3}$$

The "trial" functions $\phi_i(x,t)$, $i = 1,2,\ldots,K$, can be used to construct a basis for $\mathcal{U}_K(S_n)$. They are selected to be of class $C^0(S_n)$ and, in finite element methods, to have small support. Particular choices of ϕ_i are given in Section 2.1; herein, it suffices to note that ϕ_i is nonzero only on $T_{i-1}^n \cup T_i^n$ and that K must be at least N.

We determine $\underset{\sim}{U}$ on S_n by solving a discrete problem of the form

$$\underset{\sim}{P}\,\underset{\sim}{U}(x,0) = \underset{\sim}{P}\,\underset{\sim}{u}^{o}(x) \qquad , n = 0 \tag{2.4}$$

$$\hat{\underset{\sim}{F}}\,(\underset{\sim}{U},v) = 0, \forall v \in V_K, \tag{2.5}$$

$$\hat{\underset{\sim}{B}}_1\,\underset{\sim}{U}(0,t_{n+1}) = \hat{\underset{\sim}{b}}_1(t_{n+1}),\ \hat{\underset{\sim}{B}}_2\,\underset{\sim}{U}(a,t_{n+1}) = \hat{\underset{\sim}{b}}_2(t_{n+1}). \tag{2.6}$$

Here V_K is a finite dimensional space of $C^o(S_n)$ functions that depends on the boundary conditions (cf. Section 2.3), $\underset{\sim}{P}$ is an interpolation operator (cf. Section 2.3), $\hat{\underset{\sim}{B}}_1$, $\hat{\underset{\sim}{b}}_1$, $\hat{\underset{\sim}{B}}_2$, $\hat{\underset{\sim}{b}}_2$ are approximations of $\underset{\sim}{B}_1$, $\underset{\sim}{b}_1$, $\underset{\sim}{B}_2$, $\underset{\sim}{b}_2$ obtained by numerical integration (cf. Section 2.3), and $\underset{\sim}{F}(\underset{\sim}{U},v)$ is an approximation of $\hat{\underset{\sim}{F}}(\underset{\sim}{U},v)$ obtained by evaluating the integrals in (2.2) numerically (cf. Section 2.2). Equations (2.5,6) result in an MK dimensional nonlinear algebraic system for determining the Galerkin coordinates $\underset{\sim}{c}_i(t_{n+1})$, i = 1,2,...K, in terms of $\underset{\sim}{c}_i(t_n)$, i = 1,2,...,K. Since $\underset{\sim}{c}_i(0)$, i = 1,2,...,K, are determined from the initial conditions (2.4), equations (2.4-6) define a marching algorithm for determining $\underset{\sim}{U}(x,t)$ in successive strips S_n, n = 0,1,... .

If there were no boundary conditions we would select $\phi_i(x,t)$, i = 1,2,...,K, as a basis for V_K. This prescription has to be modified slightly for i = 1 and/or i = K (cf. Section 2.3) since boundary conditions are generally imposed; however, it is still appropriate to write $\underset{\sim}{F}(\underset{\sim}{U},v)$ as a sum of contributions from each trapezoid. Thus,

$$\begin{aligned}\underset{\sim}{F}(\underset{\sim}{U},v) = &\sum_{i=1}^{N-1} \iint_{T_i^n} \{-\underset{\sim}{U}\, v_t + \underset{\sim}{f}(x,t,\underset{\sim}{U},\underset{\sim}{U}_x)v + \underset{\sim}{D}(x,t,\underset{\sim}{U})\underset{\sim}{U}_x\, v_x\}\, dx\, dt \\ &+ \sum_{i=1}^{N-1} \Big[\int_{x_i(t)}^{x_{i+1}(t)} \underset{\sim}{U}\, v\, dt\Big]_{t=t_n}^{t_{n+1}} - \Big[\int_{t_n}^{t_{n+1}} \underset{\sim\sim}{DU}_x\, v\, dt\Big]_{x=0}^{a} = 0, \forall v \in V_K\end{aligned} \tag{2.7}$$

Since the bases for both the trial and test spaces have small support most of the integrals in (2.7) will be zero. The algebraic system (2.5,6) will be sparse and, hence, it may be solved effeciently.

2.1. Selection of a Basis

A simple way to construct a basis on trapezoidal elements that satisfies the necessary continuity requirements and has small support is to apply a local transformation that maps each trapezoid onto a rectangle. The inverse of this transformation on T_i^n is

$$x = x_i^n + (x_{i+1}^n - x_i^n)\left(\frac{\xi+1}{2}\right) + (x_i^{n+1} - x_i^n)\tau$$

$$+ (x_{i+1}^{n+1} - x_i^{n+1} - x_{i+1}^n + x_i^n)\left(\frac{\xi+1}{2}\right)\tau, \tag{2.8}$$

$$t = t_n + (t_{n+1} - t_n)\tau \ .$$

It maps the rectangle

$$R = \{(\xi,\tau) \mid -1 \le \xi \le 1,\ 0 \le \tau \le 1\} \tag{2.9}$$

in the (ξ,τ) plane onto T_i^n in the (x,y) plane.

We choose this basis so that $\phi_i(x,t)$ is a function of ξ only on T_j^n. To be specific, we currently allow $\phi_i(x,t)$ to be either a piecewise linear or a piecewise Hermite cubic polynomial in ξ on T_j^n.

For piecewise linear approximations we construct a basis in terms of the canonical basis function

$$\hat{\phi}(\xi) = (1-\xi)/2,\ -1 \le \xi \le 1, \tag{2.10}$$

by defining

$$\phi_i(x,t) = \begin{cases} \hat{\phi}(\xi) & , (x,t) \in T_i^n \\ \hat{\phi}(-\xi) & , (x,t) \in T_{i-1}^n \\ 0 & , \text{otherwise} \end{cases}, \quad i = 1,2,\ldots,K = N. \tag{2.11}$$

Thus, the dimension of the trial space U_K is $K = N$. Along the line $x_j(\tau)$ joining x_j^n and x_j^{n+1} we have

$$\phi_i(x_j(\tau),\ t(\tau)) = \delta_{ij},\ 0 \le \tau \le 1, \tag{2.12}$$

where δ_{ij} is the Kronecker delta. Using (2.3) this implies that

$$\underset{\sim}{c}_i(t) = \underset{\sim}{U}_i(\tau) := \underset{\sim}{U}(x_i(\tau), t(\tau)). \tag{2.13}$$

Thus, since only ϕ_i and ϕ_{i+1} are nonzero on T_i^n we have

$$\underset{\sim}{U}(x,t) = \underset{\sim}{U}_i(\tau)\hat{\phi}(\xi) + \underset{\sim}{U}_{i+1}(\tau)\hat{\phi}(-\xi), \quad (x,t) \in T_i^n. \tag{2.14}$$

For piecewise cubic Hermite approximations we construct a basis in terms of the two canonical basis functions

$$\hat{\psi}(\xi) = \frac{1}{4}(1-\xi)^2(2+\xi), \quad \hat{\chi}(\xi) = \frac{1}{4}(1+\xi)(1-\xi)^2, \quad -1 \le \xi \le 1, \tag{2.15}$$

by defining

$$\phi_{2i-1}(x,t) = \begin{cases} \hat{\psi}(\xi), & (x,t) \in T_i^n \\ \hat{\psi}(-\xi), & (x,t) \in T_{i-1}^n, \quad i = 1,2,\ldots,N \\ 0, & \text{otherwise} \end{cases} \tag{2.16a}$$

$$\phi_{2i}(x,t) = \begin{cases} \hat{\chi}(\xi), & (x,t) \in T_i^n \\ -\hat{\chi}(-\xi), & (x,t) \in T_{i-1}^n, \quad i = 1,2,\ldots,N \\ 0, & \text{otherwise} \end{cases} \tag{2.16b}$$

Thus, the dimension K of the trial space in 2N.

We note that $\phi_{2i-1}(x,t) \in C^1(S_n)$ with

$$\phi_{2i-1}(x_j(\tau),t(\tau)) = \delta_{ij}, \quad \phi_{2i-1_x}(x_j(\tau),t(\tau)) = 0, \quad 0 \le \tau \le 1, \tag{2.17a,b}$$

but $\phi_{2i}(x,t) \in C^0(S_n)$ with

$$\phi_{2i}(x_j(\tau),t(\tau)) = 0, \quad \phi_{2i_x}(x_j(\tau),t(\tau)) = \delta_{ij}/x_\xi(\tau), \quad 0 \le \tau \le 1. \tag{2.17c,d}$$

The function $x_\xi(\tau)$ is easily computed from (2.8) as

$$x_\xi(\tau) = \frac{1}{2}(x_{j+1}^n - x_j^n) + \frac{1}{2}(x_{j+1}^{n+1} - x_j^{n+1} - x_{j+1}^n + x_j^n)\tau, \text{ if } (x,t) \in T_j^n. \tag{2.18}$$

Thus, $x_\xi(\tau)$ is different on each trapezoid (unless the mesh is uniform and rectangular) and $\phi_{2i_x}(x,t)$ jumps as x crosses $x_i(\tau)$. However, using (2.3) and

(2.17) we can make $\underset{\sim}{U}(x,t)$ of class $C^1(S_n)$ by selecting

$$\underset{\sim}{c}_{2i-1}(t) = \underset{\sim}{U}_i(\tau) := \underset{\sim}{U}(x_i(\tau),t(\tau)), \tag{2.19a}$$

$$\underset{\sim}{c}_{2i}(t) = x_\xi(\tau)\underset{\sim}{U}_{x_i}(\tau) := x_\xi(\tau)\ \underset{\sim}{U}_x(x_i(\tau),t(\tau)), \tag{2.19b}$$

Thus, on T_i^n we have

$$\begin{aligned}\underset{\sim}{U}(x,t) = {} & \underset{\sim}{U}_i(\tau)\hat{\psi}(\xi) + \underset{\sim}{U}_{i+1}(\tau)\hat{\psi}(-\xi) + \underset{\sim}{U}_{x_i}(\tau)x_\xi(\tau)\hat{\chi}(\xi) \\ & - \underset{\sim}{U}_{x_{i+1}}(\tau)x_\xi(\tau)\hat{\chi}(-\xi).\end{aligned} \tag{2.20}$$

2.2 Numerical Integration

Ignoring the boundary conditions for the moment, we choose $v = \phi_j(x,t)$ according to either (2.11) or (2.16) and use (2.8) to transform (2.7) to

$$\underset{\sim}{F}(\underset{\sim}{U},\phi_j) = \sum_{i=1}^{N-1} \underset{\sim}{I}_i(\underset{\sim}{U},\phi_j) - \underset{\sim}{I}_B(\underset{\sim}{U},\phi_j) = 0,\ j = 1,2,\ldots,K, \tag{2.21}$$

where

$$\begin{aligned}\underset{\sim}{I}_i(\underset{\sim}{U},v) = {} & \int_{0-1}^{1}\int_{-1}^{1} \{-\underset{\sim}{U}\, v_\xi\, \xi_t + \underset{\sim}{f}(x,t,\underset{\sim}{U},\underset{\sim}{U}_\xi\xi_x)v \\ & + \underset{\sim}{D}(x,t,\underset{\sim}{U})\underset{\sim}{U}\, v_\xi\xi_x^2\}|J|d\xi d\tau + \int_{-1}^{1} \underset{\sim}{U}\, v\, x_\xi\, d\xi\Big|_{\tau=0}^{1}\end{aligned} \tag{2.22a}$$

$$\underset{\sim}{I}_B(\underset{\sim}{U},v) = \int_0^1 \underset{\sim}{D}(x,t,\underset{\sim}{U})\underset{\sim}{U}_\xi v\xi_x t_\tau d\tau\Big|_{\substack{x=0,\xi=-1\\ x=a,\xi=1}} \tag{2.22b}$$

The functions ξ_t, ξ_x, x_ξ, t_τ, and $|J|$ the Jacobian of the transformation can be computed from (2.8).

In order to complete the specification of our numerical method we need to select quadrature rules for evaluating the integrals in (2.22). We use the Trapezoidal rule to evaluate the τ integrals and a three point Gauss-Legendre rule (cf. Abromowitz and Stegen [1], Chap. 25) to evaluate the ξ integrals. The latter was chosen because it is known (cf. Strang and Fix [35]) to have the same order discretization error as our finite element method with cubic

approximations and the exact integration of (2.22). At present, we also use the three point Gauss-Legendre rule for linear approximations although it is more accurate than necessary in this case and therefore somewhat inefficient.

Upon use of the above mentioned quadrature rules equations (2.25) become

$$\hat{\underset{\sim}{F}}(\underset{\sim}{U},\phi_j) = \sum_{i=1}^{N-1} \hat{\underset{\sim}{I}}_i(\underset{\sim}{U},\phi_j) - \hat{\underset{\sim}{I}}_B(\underset{\sim}{U},\phi_j) = 0,\ j = 1,2,\ldots,K, \tag{2.23}$$

where $\hat{\underset{\sim}{I}}_i$ and $\hat{\underset{\sim}{I}}_B$ denote the approximations of (2.22) that are obtained by numerical integration.

2.3. Initial and Boundary Conditions. Solution Technique.

The solution $\underset{\sim}{U}$ is determined on S_n by solving (2.23) together with the initial and boundary conditions (2.4) and (2.6), respectively. We satisfy the initial conditions (and implicitly define the interpolation operator $\underset{\sim}{P}$ of (2.4)) by requiring

$$\underset{\sim}{U}_i^0 = \underset{\sim}{u}^0(x_i),\ i = 1,2,\ldots,N, \tag{2.24a}$$

for both linear and cubic approximations, and additionally

$$\underset{\sim}{U}_{x_i}^0 = \underset{\sim}{u}_x^0(x_i),\ i = 1,2,\ldots,N \tag{2.24b}$$

for cubic approximations. Here

$$\underset{\sim}{U}_i^n := \underset{\sim}{U}(x_i^n,t_n);\ U_{x_i}^n := U_x(x_i^n,t_n). \tag{2.25}$$

We obtain the approximate boundary conditions (2.6) by substituting (2.3) into (1.3), integrating the resulting equation from t_n to t_{n+1}, and evaluating the integrals by the Trapezoidal rule. Each boundary condition is associated with a particular partial differential equation in the vector system. The test space V_K is modified by setting the test functions ϕ_1 and ϕ_N (for linear approximations) or ϕ_{2N-1} (for cubic approximations) equal to zero for those partial differential equations associated with boundary conditions. This has

the effect of replacing the Galerkin approximation of a partial differential equation at either x = 0 or x = a by its corresponding approximate boundary condition.

The system (2.6), (2.23) is a nonlinear algebraic system for determining $\underset{\sim}{U}_i^{n+1}$, i = 1,2,...,N for linear approximations or $\underset{\sim}{U}_i^{n+1}$, $\underset{\sim}{U}_{x_i}^{n+1}$, i = 1,2,...,N, for cubic approximations given the same information at $t = t_n$. We solve this nonlinear system by Newton's method which requires the computation of the Jacobian of the vector $[\underset{\sim}{F}(\underset{\sim}{U},\phi_1),\underset{\sim}{F}(\underset{\sim}{U},\phi_2),\ldots,\underset{\sim}{F}(\underset{\sim}{U},\phi_K)]^T$ with respect to $[\underset{\sim}{U}_1^{n+1},\underset{\sim}{U}_2^{n+1},\ldots,\underset{\sim}{U}_N^{n+1}]^T$ for linear approximations or $[\underset{\sim}{U}_1^{n+1}, \underset{\sim}{U}_{x_1}^{n+1}, \underset{\sim}{U}_2^{n+1}, \underset{\sim}{U}_{x_2}^{n+1},\ldots,\underset{\sim}{U}_N^{n+1}, \underset{\sim}{U}_{x_N}^{n+1}]^T$ for cubic approximations. The Jacobian will be block tridiagonal because of the local nature of ϕ_i. The elements in the i th block of rows will be the M × M matrices

$$\frac{\partial \underset{\sim}{F}(\underset{\sim}{U},\phi_i)}{\partial \underset{\sim}{U}_j^{n+1}}, \quad j = i-1,\ i,\ i+1 \tag{2.26a}$$

for linear approximations and the 2M × 2M matrices

$$\begin{bmatrix} \dfrac{\partial \underset{\sim}{F}(\underset{\sim}{U},\phi_{2i-1})}{\partial \underset{\sim}{U}_j^{n+1}} & \dfrac{\partial \underset{\sim}{F}(\underset{\sim}{U},\phi_{2i-1})}{\partial \underset{\sim}{U}_{x_j}^{n+1}} \\ \dfrac{\partial \underset{\sim}{F}(\underset{\sim}{U},\phi_{2i})}{\partial \underset{\sim}{U}_j^{n+1}} & \dfrac{\partial \underset{\sim}{F}(\underset{\sim}{U},\phi_{2i})}{\partial \underset{\sim}{U}_{x_j}^{n+1}} \end{bmatrix}, \quad j = i-1,\ i,\ i+1, \tag{2.26b}$$

for cubic approximations. The elements of (2.16) are obtained from (2.22,23) in a relatively straightforward manner, but their computation requires users of our code to supply subroutines that define $\underset{\sim}{f}_{\underset{\sim}{u}}(x,t,\underset{\sim}{u},\underset{\sim}{u}_x)$, $\underset{\sim}{f}_{\underset{\sim}{u}_x}(x,t,\underset{\sim}{u},\underset{\sim}{u}_x)$, and $\underset{\sim}{D}_{\underset{\sim}{u}}(x,t,\underset{\sim}{u})$. Subroutines that define $\underset{\sim}{f}(x,t,\underset{\sim}{u},\underset{\sim}{u}_x)$ and $\underset{\sim}{D}(x,t,\underset{\sim}{u})$ must, of course, also be supplied. We calculate and factor the Jacobian once per time step and use $\underset{\sim}{U}(x,t_n)$ as an initial guess for $\underset{\sim}{U}(x,t_{n+1})$. The linearized Newton system is solved by an efficient block tridiagonal algorithm that uses pivoting both within and outside of blocks (cf. Davis [11]). Generally two iterations are performed per time step.

3. Adaptive Mesh Selection Strategy

In Section 2 we developed a finite element method to obtain numerical solutions to systems of partial differential equations on nonuniform trapezoidal grids. In this section we construct an algorithm to select a grid at $t = t_{n+1}$ so that the L_2 norm of the error at t_{n+1} is approximately minimized. This algorithm builds upon the work of deBoor [12], Lawson [26], and Jupp [23] on variable knot spline interpolation.

For most of this section we will be discussing approximations at a single time level, say $t = t_n$, so whenever there is no possibility of confusion we omit the n superscript on x_i^n and $\underset{\sim}{U}_i^n$ and surpress the t dependence when writing $\underset{\sim}{u}(x,t)$. We also present the development for scalar functions u and indicate the extensions to vector functions in Section 3.2.

It is well known (cf. [10,35,36]) that the errors in finite element-Galerkin methods for problems like (1.1-3) satisfy estimates of the form

$$||u-U||_{L_2} \leq C||u-\underset{\sim}{PU}||_{L_2}, \tag{3.1}$$

where $\underset{\sim}{PU} \in \mathcal{U}_K$ interpolates u. Thus, the error in the solution of the partial differential equation is bounded by an interpolation error. The following result (cf. e.g. Pereyra and Sewell [31]) indicates how to minimize this interpolation error for piecewise polynomial interpolants.

Lemma: Let $\Pi_N := \{0 = x_1 < x_2 < \cdots < x_N = a\}$ be a partition of $[0,a]$ into N-1 subintervals and let $u(x) \in C^{\ell+1}[0,a]$. The piecewise polynomial of degree ℓ on (x_i,x_{i+1}), $i = 1,2,\ldots,N-1$, that interpolates to u on Π_N has minimal L_2 error when the knots x_i, $i = 2,3,\ldots,N-1$ are chosen such that

$$h_i^{\ell+1}\left|u^{(\ell+1)}(\xi_i)\right| = E, \quad i = 1,2,\ldots,N-1, \tag{3.2}$$

where $u^{(\ell)}$ is the ℓ the derivative of u with respect to x, $\xi_i \in (x_i,x_{i+1})$, E is a constant, and

(3.3) $$h_i = x_{i+1} - x_i.$$

The Lemma states that the interpolation error is minimized by selecting the partition in such a way that the quantity $h_i^{\ell+1}|u^{(\ell+1)}(\xi_i)|$ is equidistributed. Considerable success has been achieved by using this result to implement adaptive grid algorithms for two-point boundary value problems (cf. Lentini and Pereyra [27], Ascher, Christiansen, and Russell [2], or Russell and Christiansen [34]). Nevertheless, some practical difficulties still remain and we discuss these and our solutions to them below.

Rather than work with (3.2) directly, we follow Lawson [27] and Jupp [23] and express (3.2) in the form

(3.4) $$p_i := h_{i+1}/h_i = [|u^{(\ell+1)}(\xi_i)/u^{(\ell+1)}(\xi_{i+1})|]^{1/(\ell+1)}, \quad i = 1,2,\ldots,N-2,$$

where $\ell = 1$ for piecewise linear and $\ell = 3$ for piecewise cubic approximations.

In addition to (3.4) we impose the constraint that

(3.5) $$h_1 + h_2 + \ldots + h_{N-1} = x_N - x_1.$$

This can be expressed in terms of the p_i's by defining

(3.6) $$z := 1 + (p_1) + (p_1p_2) + (p_1p_2p_3) + \ldots + (p_1p_2p_3 \cdots p_{N-2})$$

and observing that

(3.7) $$z = (h_1 + h_2 + h_3 + \ldots + h_{N-1})/h_1 = (x_N - x_1)/h_1$$

Equations (3.3,4,6,7) permit us to determine h_i, $i = 1,2,\ldots,N-1$, and x_i, $i = 1,2,\ldots,N$, in terms of $u^{(\ell+1)}$ without an explicit determination of E.

Of course, $u^{(\ell+1)}$ is unknown and must be approximated by $U^{(\ell+1)}$. The finite element procedure provides us with an approximate solution U and for cubics an approximate first derivative U_x. However, equation (3.4) requires a knowledge of second derivatives for linear approximations and fourth derivatives for cubic approximations. This situation typically arises in adaptive

mesh algorithms and it is usually resolved by using finite difference approximations for the necessary higher derivatives.

DeBoor [12] used finite difference approximations to choose mesh points for the solution of two-point boundary value problems by assuming that the $(\ell+1)$st derivative was constant on each subinterval. We modify this scheme slightly by assuming that $U^{(\ell+1)}$ is linear on each subinterval and takes on the following values at the nodes:

$$(3.8a)\quad U^{(\ell+1)}(x_i) = \begin{cases} \Delta U^{(\ell)}_{1/2}/(h_2+h_1), & i = 1 \\ 2\Delta U^{(\ell)}_{1/2}/(h_2+h_1), & i = 2 \\ \Delta U^{(\ell)}_{i-3/2}/(h_{i-1}+h_{i-2}) + \Delta U^{(\ell)}_{i-1/2}/(h_i+h_{i-1}), & i = 3,4,\ldots,N-1 \\ \Delta U^{(\ell)}_{N-3/2}/(h_{N-1}+h_{N-2}), & i = N \end{cases}$$

where

$$(3.8b)\qquad \Delta U^{(\ell)}_i := (U^{(\ell)}_{i+1} - U^{(\ell)}_i).$$

We use the approximation

$$(3.9a)\qquad U'_{i-1/2} = (U_i - U_{i-1})/h_{i-1}$$

for linear polynomials and

$$(3.9b)\qquad U'''_{i-1/2} = 12(U_{i-1}-U_i)/h^3_{i-1} + 6(U_{x_{i-1}}+U_{x_i})/h^2_{i-1}$$

for cubic polynomials.

We note that p_i becomes infinite or indeterminate when $u^{(\ell+1)}(\xi_i) = 0$ (cf. (3.4)); hence we can expect numerical difficulties when $u^{(\ell+1)}(x)$ is small on any subinterval. Indeed numerical experiments have shown that the mesh becomes very sensitive to small perturbations whenever $U^{(\ell+1)}(x)$ is of the same order of magnitude as the discretization error in the computed solution U.

We combat this problem by imposing a lower bound on $|U^{(\ell+1)}(x)|$. Thus, we let $\Delta t_n = (t_{n+1}-t_n)$ and $h = a/N$ denote the current time step and the average mesh spacing, respectively, and for linear approximations we calculate $|U''(x_i)|$ as the maximum of the value computed by (3.8,9) and $\max(\Delta t^2/h^2, h^2)$ while for cubic approximations we calculate $|U^{(iv)}(x_i)|$ as the maximum of the value computed by (3.8,9) and $12 \max(\Delta t/h, h^2) + 6 \max(\Delta t^4/h^3, h^2)$. These limits were determined empirically. They are small enough so that they do not affect the mesh adaption procedure when $U^{(\ell+1)}(x)$ is not small but large enough to avoid the numerical difficulties caused by vanishing values of $U^{(\ell+1)}(x)$. Observe that if $U^{(\ell+1)}(x)$ is uniformly small on [0,a] our limits assure that the solution of (3.3,4,6,7) is a uniform mesh, as it should be in this case.

The discussion thus far has concerned the computation of an optimal grid at a time level t_n where the solution U^n has already been computed. We would also like to estimate an optimal grid at time level t_{n+1} prior to computing the solution there. This can be done by extrapolating the optimal grids computed at a number of previous time levels to t_{n+1}. It was somewhat surprising that numerical experiments seemed to favor zero order extrapolation; i.e., the optimal grid computed at time level t_n is used at time level t_{n+1}. Multi-level extrapolation consistently overestimated the distance that a mesh point should move in one time step and then overcorrected this error in the next time step. In some cases this caused the mesh to oscillate wildly when in fact the solution changed very little. When we simply extrapolated the optimal mesh determined at the previous time level it tended to follow the solution even when rapidly moving fronts were present.

It is easy to show that the mesh selection strategy (3.3,4,5,6) maintains the knot ordering so that no two mesh points can cross. It does not however prohibit severely distorted trapezoidal elements. Ciarlet and Raviart [10] and Babuska and Aziz [3] have studied the effect of element distortion on the accuracy of the finite element method. They have shown that the error obtained

when computing on trapezoidal elements is a multiple of the error obtained when computing on rectangular elements. The multiplicative factor is proportional to a power of the magnitude of the derivatives of the transformation (2.8). Therefore we must control the magnitude of these derivatives in order to maintain acceptable accuracy. We let

$$h_i^n = x_{i+1}^n - x_i^n,\ \Delta t_n = t_{n+1} - t_n,\ \tan \omega_i = h_i^n/\Delta t_n. \tag{3.10}$$

Hence, ω_i is the angle between the line $x_i(t)$ and the positive t axis.

Differentiating (2.8) and using (3.10) we find

$$\begin{aligned} x_\xi &= [h_i^n + \tau(h_i^{n+1} - h_i^n)]/2, \\ x_\tau &= \Delta t_n [\tan \omega_i + (\tan \omega_{i+1} - \tan \omega_i)(\xi+1)/2], \\ t_\xi &= 0,\ t_\tau = \Delta t_n. \end{aligned} \tag{3.11}$$

Since the magnitudes of h_i^n and h_i^{n+1} are controlled by the bounds that we imposed on $|U^{(\ell+1)}|$ and Δt_n is prescribed, we can limit the magnitude of the derivatives in (3.11) by controlling the growth of $|\tan \omega_i|$. We found that the condition

$$\max_{1 \le i \le N} |\omega_i| \le 3\pi/8 \tag{3.12}$$

worked well in practice.

3.1. Mesh Selection Algorithm

In this section we discuss some details of a mesh selection algorithm based on the discussion of the previous section. The first algorithm uses the finite element solution $U(x,t_n)$ calculated on the mesh x_i^n, $i = 1,2,\ldots,N$, and equations (3.3,4,6,7) to find a new mesh at $t = t_n$ that satisfies the optimality condition (3.2). This is the mesh that should have been used to calculate $U(x,t_n)$. Instead we use it in the second algorithm to estimate an optimal mesh at $t = t_{n+1}$.

The difficulty in solving (3.3,4,6,7) for the optimal mesh is that these equations are nonlinear and must be solved iteratively. We use a relaxation scheme that is similar to one which has been analyzed by Isaacson and Keller [21, Chap. 3]. They give necessary convergence criteria, but, we chose not to incorporate these into our algorithm because they require too much additional computation. The following algorithm, which calculates the relaxation parameter heuristically, has not failed to converge in any of our tests.

1. Set the relaxation parameter $\Omega := 1$ and let $x_i^{(0)} := x_i^n$, $i = 1,2,\ldots,N$ be an initial guess for the optimal mesh. Calculate

$$z^{(0)} := (x_N^{(0)} - x_1^{(0)})/(x_2^{(0)} - x_1^{(0)})$$

$$z^{(1)} := z + 2\epsilon$$

$$\nu := 1$$

where ϵ is a convergence tolerance.

2. Compute

$$U^{(\ell+1)}(x_i^{(0)}),\ i = 1,2,\ldots,N$$

using equations (3.8,9).

3. <u>While</u> $|z^{(\nu)} - z^{(\nu+1)}| > \epsilon$ or $\nu \leq \nu_{max}$ <u>do</u>

4. Calculate $U^{(\ell+1)}(x_i^{(\nu-1)})$, $i = 1,2,\ldots,N$ by linear interpolation of $U^{(\ell+1)}(x_i^{(0)})$, $i = 1,2,\ldots,N$.

Calculate

$$p_i^{(\nu)} := \left|U^{(\ell+1)}(x_{i+1}^{(\nu-1)})/U^{(\ell+1)}(x_{i+2}^{(\nu-1)})\right|^{1/(\ell+1)},$$

$$i = 1,2,\ldots,N-2.$$

and

$$\hat{z}^{(\nu)} := 1 + (p_1^{(\nu)}) + (p_1^{(\nu)} p_2^{(\nu)}) + \ldots (p_1^{(\nu)} p_2^{(\nu)} \ldots p_{N-2}^{(\nu)}).$$

5. If $\nu > 1$ then

$$\text{If } |\hat{z}^{(\nu)} - z^{(\nu-1)}| \geq |z^{(\nu-1)} - z^{(\nu-2)}| \text{ then } \Omega := \Omega/2$$

6. Calculate

$$\left.\begin{aligned} h_1^{(\nu)} &:= (x_N^{(\nu-1)} - x_1^{(\nu-1)})/\hat{z}^{(\nu)}, \\ x_1^{(\nu)} &:= \hat{x}_1^{(\nu)} := x_1^{(\nu-1)}, \\ x_N^{(\nu)} &:= \hat{x}_N^{(\nu)} := x_N^{(\nu-1)}, \\ \hat{x}_{i+1}^{(\nu)} &:= \hat{x}_i^{(\nu)} + h_i^{(\nu)} \\ h_{i+1}^{(\nu)} &:= h_i^{(\nu)}\, p_i^{(\nu)} \\ x_{i+1}^{(\nu)} &:= \Omega \hat{x}_{i+1}^{(\nu)} + (1-\Omega) x_{i+1}^{(\nu-1)} \end{aligned}\right\} \quad i = 1,2,\ldots,N-2$$

$$z^{(\nu)} := (x_N^{(\nu)} - x_1^{(\nu)})/(x_2^{(\nu)} - x_1^{(\nu)}),$$

7. $\nu := \nu + 1$

For vector systems we need only to change the definition of $p_i^{(\nu)}$ used in step 3. We used

$$p_i^{(\nu)} := \sum_{j=1}^{M} \left| U_j^{(\ell+1)}(x_{i+1}^{(\nu-1)}) / U_j^{(\ell+1)}(x_{i+2}) \right|^{1/(\ell+1)},$$

where U_j is the j th component of $\underset{\sim}{U}$.

After we compute a convergent mesh, $\hat{x}_i^{n+1} = x_i^{(\nu)}$, $i = 1,2,\ldots,N$, we perform the following:

1. Compute

$$\Delta x_{max} = \max_{2 \leq i \leq N-1} |\hat{x}_i^{n+1} - x_i^n|$$

2. If $\Delta x_{max} \leq \Delta t_n \tan(3\pi/8)$

then $\Delta x_{fix} := \Delta x_{max}$

else $\Delta x_{fix} := \Delta t_n \tan(3\pi/8)$.

3. Compute corrected mesh $\underset{\sim}{x}^{n+1}$ as

$$x_i^{n+1} := x_i^n + (\hat{x}_i^{n+1} - x_i^n)\Delta x_{max}/\Delta x_{fix},$$

$$i = 2,3,\ldots,N-1$$

Steps 2-3 prevent the elements from becoming too distorted.

The algorithms contain several approximations and heuristic procedures. Derivatives are estimated by differences and are assumed to vary linearly between mesh points. Zero order extrapolation was used to predict optimal grids at subsequent time levels. Grids were restrained to prevent severe element distortion. Even with these approximations the mesh selection algorithms performed satisfactorily on all test examples that we considered. In addition we note that Rheinboldt [32] has shown that an order Δ error in the placement of the optimal mesh only produces an order Δ^2 change in the computed solution. Thus, it suffices to only be close to the optimal mesh in order to reap its benefits.

4. Computational Results

In this section we examine the performance of our method on four problems which are graded in difficulty such that each one exercises an additional facet of the method. The following norms are used to evaluate the performance of our method on examples where exact solutions are known.

$$(4.1a)\quad ||e(t)||_\infty := \max_{1\le i\le N} |\underset{\sim}{e}(x_i,t)|_\infty := \max_{1\le i\le N} |\underset{\sim}{u}(x_i,t) - \underset{\sim}{U}(x_i,t)|_\infty$$

$$(4.1b)\quad ||e(t)||^2_{L_2} := \sum_{i=1}^{N-1} (h_i/2)(|\underset{\sim}{e}(x_i,t)|^2_\infty + |e(x_{i+1},t)|^2_\infty),$$

where

$$(4.1c)\qquad |\underset{\sim}{v}|_\infty := \max_{1\le k\le M} |v_k|.$$

Example 1:

$$u_t = (1/\pi)^2 u_{xx}, \qquad 0 < x < 1,\ t > 0$$

$$(4.2)$$

$$u(x,0) = \sin \pi x,\ u(0,t) = u(1,t) = 0.$$

The exact solution is

$$u(x,t) = e^{-t} \sin \pi x.$$

Analysis presented in [11] indicates that the finite element method described in section 2 would have L_2 error of $O(h^2) + O(\Delta t^2)$ with linear elements and $O(h^4) + O(\Delta t^2)$ with cubic elements on a uniform spacial mesh of width h and a uniform time step of duration Δt. We created this simple constant coefficient example to verify that these errors are actually attained. Figures 2 and 3 present plots of the L_2 error at $t = 1$ as a function of h for linear and cubic approximations, respectively.

The analysis of [11] predicts that the points on Figure 2 for which $\Delta t = h$ and the points in Figure 3 for which $\Delta t = h^2$ should lie on straight lines having

slopes 2 and 4, respectively. These lines are shown confirming that the theoretical error bounds are actually attained.

Example 2:

(4.3a) $$u_t = \sigma u_{xx} + f(x),\ 0 < x < 1,\ t > 0,\ \sigma > 0.$$

The initial conditions, boundary conditions, and source f are chosen so that the exact solution is

(4.3b) $$u(x,t) = \tanh\ (r_1(x-1) + r_2 t)$$

The solution (4.3b) is a wave that travels in the negative x direction when r_1 and r_2 are positive. The values r_1 and r_2 determine the steepness of the wave and its speed of propagation. Thus, the problem can be made more or less difficult by adjusting r_1 and r_2.

We created this problem to study the effectiveness of our adaptive mesh algorithm at concentrating grid points in transition regions, following moving fronts, and reducing errors below those of uniform grid calculations.

We first solve problem (4.2) with $r_1 = r_2 = 5$, uniform time steps of $\Delta t = 0.01$, 10 elements per time step, and linear approximations. The mesh computed by our adaptive mesh algorithm is shown in Figure 4. The grid points are concentrated in the region of maximum curvature and move to the left with the wave. As the wave front passes out of the domain and u_{xx} becomes small, the grid points move toward a uniform distribution. It is clear that the grid adapts to the solution and follows its progress.

As a somewhat more difficult problem we solve (4.3a) with initial conditions, boundary conditions, and forcing function chosen so that the solution is given by (4.3b) with $r_1 = r_2 = 100$. The wave front is much steeper than in the previous test of (4.3).

In Table 1 we present a comparison at t = 1 of the results of computation using linear approximations on a variety of uniform and variably spaced meshes. These results are somewhat disappointing. At best the mesh moving scheme improves the accuracy of the solution only slightly. The improvement is greatest when Δt is small and in some cases, when Δt is large, the uniform mesh is more accurate. A closer examination explains these results and reveals something about the nature of this mesh moving scheme.

Table 1 shows that the solution of this problem was not computed accurately with either a uniform or a variable mesh. This can be explained by examining the time evolution of the solution at a fixed value of x, say x^*. The solution is approximately given by -1 until the time when the wave reaches the point x^*. It then jumps suddenly to a value near 1. If the time step Δt is too large to resolve this transition, we would expect large errors in the vicinity of the wave. The solid curve in Figure 5 confirms this prediction. The mesh selection procedure misinterprets these errors as being part of the solution and places too many points in the region outside of the wave front. Thus, a suboptimal mesh is selected and the expected decrease in the error is not obtained. When Δt becomes small enough to adequately resolve the passing wave the mesh selection procedure does improve the accuracy of the solution (cf. Table 1).

This points out the need for an algorithm to adaptively refine time steps in the vicinity of severe temporal gradients. Such a procedure was used by Berger et al. [5] to solve hyperbolic partial differential equations and we are currently studying its suitability for our code.

Table 2 summarizes the results of computations performed on the same problem using cubic approximations. In these cases the time steps Δt were small enough to resolve the transition of the solution and the cubic

approximations were accurate enough to provide us with reasonable estimates of the derivatives. As a result, the variable mesh scheme improved the solution significantly.

Figures 5 and 6 for linear and cubic approximations, respectively, show that the mesh selection algorithm tends to distribute the local error evenly over the domain and thus, as indicated in Section 3, approximately minimizes the error in L_2.

Example 3: (Burgers' Equation)

$$u_t = -uu_x + \varepsilon u_{xx}, \quad 0 < x < 1, \quad t > 0,$$

(4.4)

$$u(x,0) = \sin \pi x, \quad u(0,t) = u(1,t) = 0,$$

and $\varepsilon = 5 \times 10^{-3}$.

It is well known that the solution to this problem is a wave that steepens and moves to the right until a shock layer forms at $x = 1$. After a time of $O(1/\varepsilon)$ the wave dissipates and the solution decays to zero. Figures 7 and 8 show the results of computations on this problem using linear approximations on a uniform mesh and a variable mesh with a constant time step of $\Delta t = 0.1$ and 10 elements per time step. The results in Figure 7 are typical of finite difference or finite element calculations for this problem. Spurious oscillations develop in the computed solution unless the mesh width is of the same order as the width of the shock layer. which is $O(\sqrt{\varepsilon})$ for this example. The variable mesh results in Figure 8 largely surpress these oscillations by automatically concentrating the mesh in the shock region as the wave steepens.

When Example 3 is solved using cubic approximations on a uniform mesh we find that the solution U_i^n at the nodes is computed accurately; however, there are large errors in the slope of the solution $U_{x_i}^n$ at the nodes when the mesh is not suitably fine in the shock region. This effect is exhibited in Figure 9 where

the solution at t = 0.6 is shown for a calculation performed with $\Delta t = 0.1$ and N = 10. Equations (2.6,19,20,23) were used to calculate the solution between mesh points.

One possible explanation of this behavior was proposed by Miller and Miller [29], but they do not explain why the large error in the slopes do not feed back and cause large errors in the function values.

Once again, these problems are corrected when the mesh adapts with the solution. Figure 10 shows the result of a similar computation using cubic approximations on a variable mesh. Both the function values and slope values are computed accurately at the nodes.

Example 4:

$$b_t = [\mu(s)b_x]_x - [b\chi(s)s_x]_x,$$

(4.5)

$$s_t = -k(s)b, \quad 0 < x < 5, \quad t > 0.$$

This two component nonlinear system was studied by Keller and Odell [24,30] as a model for the chemotactic motion of bacteria. The quantity b(x,t) denotes the bacterial density and s(x,t) denotes the concentration of the critical substrate (bacterial food). If the functions μ,χ and k satisfy conditions derived by Keller and Odell [25], equations (4.5) have travelling wave solutions. These solutions have been computed by Odell and Keller [30] and are interpreted as travelling bands of bacteria. For our study we choose k(s) = 1, $\mu(s) = \mu_o$, and $\chi(s) = \delta_o/s$ where μ_o and δ_o are constants. The initial conditions are shown in Figure 11a and the boundary conditions are

(4.6) $$b(0,t) = b(5,t) = 0, \quad s(0,t) = 1.$$

We solved this problem for $\delta_o/\mu_o = 2$ using cubic approximations, uniform time steps of $\Delta t = 0.005$, and 50 elements per time step. The computed solutions at t = 0, 0.1, 0.5, and 1.0 are shown in Figures 11a,b,c, and d, respectively.

The method places the majority of the mesh points in the regions of the wave fronts and follows the bacterial motion. The results indicate that our adaptive mesh algorithm may be also used for vector systems of equations.

5. Discussion and Conclusions

The computations presented in the last section show that it is possible to construct an accurate and stable adaptive grid finite element method for nonlinear systems of partial differential equations and that such techniques offer advantages ver fixed grid techniques. In particular, we have shown that the error estimates obtained by Davis [11] are actually realized in practice and that the adaptive mesh algorithm correctly concentrates the mesh in a sharp transition and is able to follow moving fronts. Examples 3 and 4 of Section 4 indicate that our method is also useful for nonlinear equations and vector systems of equations.

In the present study we used piecewise polynomial functions for both the trial and test spaces. However, recent work of Flaherty and Mathon [15], Heinrich et al. [18], and Hemker [19] indicates that exponential and "upwinded" polynomial functions may give superior test spaces for singularly perturbed problems. We plan to incorporate these functions into our methods shortly.

All of our calculations were performed with a constant time step. Examples 3 and 4 of Section 4 indicate that it would be most desirable to be able to vary the time step during the calculation. Our code presently allows for this, but as yet we have not implemented an algorithm to adaptively alter the time step. We also plan to add this feature to our code shortly.

Other areas for future study include free boundary problems and higher dimensional problems. The present work has shown that it is possible to construct a practical adaptive grid finite element method. Future work must refine this method and apply it and test it on a greater variety of problems.

REFERENCES

1. M. Abromowitz and I. A. Stegan, Handbook of Mathematical Functions, Dover, New York, 1965.

2. U. Ascher, J. Christiansen, and R. D. Russell, Collocation software for boundary value ODE's, to appear in Trans. Math. Software, 1981.

3. I. Babuska and A. K. Aziz, On the angle condition in the finite element method, SIAM J. Numer. Anal., 13 (1976), pp. 214-226.

4. G. K. Batchelor, An introduction to fluid dynamics, Cambridge University Press, Cambridge, 1970.

5. M. Berger, W. Gropp, J. Oliger, Mesh generation for time dependent problems: Criteria and methods, in Proc. Workshop on Numerical Grid Generation Techniques for Partial Differential Equations, NASA Langley Research Center, Hampton, Va., October 1980.

6. R. Bonnerot and P. Jamet, Numerical computation of the free boundary for the two dimensional Stefan Problem by space-time finite elements, J. Comp. Phys., 25 (1977), pp. 161-181.

7. A. Brandt, Multi-level adaptive technique (MLAT) for fast numerical solution to boundary value problems, in Lecture Notes in Physics, 18, pp. 82-79, Springer Verlag, Berlin and New York, 1973.

8. G. F. Carey, A mesh refinement scheme for finite element computations, Computer Methods in Applied Mechanics and Engineering, 7 (1976), pp. 93-105.

9. B. Childs, et al. (Eds.), Codes for Boundary Value Problems in Ordinary Differential Equations: Proceedings of a working Conference, May 14-17, 1978, Lecture Notes in Computer Science No. 76, Springer Verlag, New York, 1979.

10. P. G. Ciarlet and P. A. Raviart, Interpolation theory over curved elements with applications to finite element methods, Computer Methods in Applied Mechanics and Engineering, 1 (1972), pp. 217-249.

11. S. F. Davis, An Adaptive Grid Finite Element Method for Initial-Boundary Value Problems, Ph.D. Dissertation, Rensselaer Polytechnic Institute, Troy, New York, 1980.

12. C. DeBoor, Good approximation by splines with variable knots II, Conference on the Numerical Solutions of Differential Equations, Lecture Notes in Math., vol. 363, Springer Verlag, New York, 1973.

13. C. DeBoor, A Practical Guide to Splines, Applied Mathematical Sciences, 27, Springer Verlag, New York, 1978.

14. P. C. Fife, Singular perturbation and wave front techniques in reaction-diffusion problems, SIAM-AMS Proceedings, 10 (1975), pp. 23-49.

15. J. E. Flaherty and W. Mathon, Collocation with polynomial and tension splines for singularly perturbed boundary value problems, SIAM J. Sci. & Stat. Comp., 1 (1980), pp. 260-289.

16. A. Friedman, The Stefan Problem in several space variables, Trans. Amer. Math. Soc., 133 (1968), pp. 51-87.

17. R. J. Galinas, S. K. Doss, and K. Miller, The moving finite element method: Application to general partial differential equations with multiple large gradients, to appear in J. Comp. Phys., 1980.

18. J. C. Heinrich, P. S. Huyakorn, O. C. Zienkiewicz, and A. R. Mitchell, An upwind finite element scheme for two-dimensional convective transport equations, Int. J. Numer. Meths. Engr., 11 (1977), pp. 131-143.

19. P. W. Hemker, A numerical study of stiff two-point boundary problems, Ph.D Dissertation, Mathematisch Centrum, Amsterdam, 1977.

20. F. Hoppensteadt, Mathematical Theories of Population: Demographies, Genetics and Epidemics, Regional Conference Series in Applied Math., 20, SIAM, Philadelphia, PA., 1975.

21. E. Isaacson and H. B. Keller, Analysis of Numerical Methods, J. Wiley and Sons, New York, 1966.

22. P. Jamet and R. Bonnerot, Numerical solution of the Eulerian equations of compressible flow by a finite element method which follows the free boundary and interfaces, J. Comp. Physics, 18 (1975), pp. 21-45.

23. D. L. Jupp, Approximation to data by splines with free knots, SIAM J. Numer. Anal., 15 (1978), pp. 328-343.

24. A. K. Kapila, Reactive-Diffusive systems with Arrhenius Kinetics: Dynamics of ignition, SIAM J. Appl. Math., 39 (1980), pp. 21-36.

25. E. F. Keller and G. M. Odell, Necessary and sufficient conditions for chemotactic bands, Math. Biosci., 27 (1975), pp. 309-317.

26. C. L. Lawson, Segmented rational minmax approximations, characteristic properties and computational methods, J. P. L. Tech. Rept., pp. 32-57, 1963.

27. M. Lentini and V. Pereyra, An adaptive finite difference solver for nonlinear two-point boundary problems with mild boundary layers, SIAM J. Numer. Anal., 14 (1977), pp. 91-111.

28. J. N. Lyness and J. J. Kaganove, Comments on the nature of automatic quadrature routines, ACM Trans. on Math. Software, 2 (1976), pp. 65-81.

29. K. Miller and R. Miller, Moving finite elements, to be published.

30. G. M. Odell and E. F. Keller, Traveling bands of chemotactic bacteria revisited, J. Theor. Biol., 56 (1976), pp. 243-247.

31. V. Pereyra and E. G. Sewell, Mesh selection for discrete solution of boundary problems in ordinary differential equations, Numer. Math., 23 (1975), pp. 261-268.

32. W. C. Reinboldt, Adaptive methods in numerical analysis, Invited lecture, SIAM 1979 Spring Meeting, Toronto, Canada.

33. J. R. Rice, A meta algorithm for adaptive quadrature, Journal of the A.C.M., 22 (1975), pp. 61-82.

34. R. D. Russell and J. Christensen, Adaptive mesh strategies for solving boundary value problems, SIAM, J. Numer. Anal., 15 (1978), pp. 59-80.

35. G. Strang and G. J. Fix, An analysis of the finite element method, Prentice Hall, Englewood Cliffs, New Jersey, 1973.

36. M. F. Wheeler, A. Priori, L_2 error estimates for Galerkin approximations to parabolic partial differential equations, SIAM J. Numer. Anal. 10 (1973), pp. 723-759.

37. A. B. White, Jr., On the numerical solution of initial/boundary value problems, to appear in SIAM J. Numer. Anal., 1980.

TABLE 1

Results of Computations at t = 1 for Example 2 with $r_1 = r_2 = 100$ using Linear Approximations on Uniform and Variably Spaced Grids.

N	Δt	Uniform Spacing		Variable Spacing	
		$\|\|e\|\|_{L_2}$	$\|\|e\|\|_\infty$	$\|\|e\|\|_{L_2}$	$\|\|e\|\|_\infty$
10	0.1	0.168	0.137	0.459	1.346
	0.05	1.107	1.708	0.492	0.949
	0.01	0.146	0.254	0.121	0.340
20	0.1	0.365	1.391	0.567	1.00
	0.05	0.177	0.392	0.155	0.746
	0.01	0.768 E-1	0.226	0.166 E-1	0.870 E-1
40	0.025	0.367 E-1	0.697 E-1	0.348 E-1	0.565 E-1
	0.01	0.342 E-1	0.158	0.106 E-1	0.105
100	0.01	0.701 E-2	0.703 E-2	0.493 E-2	0.158 E-1
				0.144 E-2	0.275 E-2

TABLE 2

Results of Computations at t = 1 for Example 2 with $r_1 = r_2 = 100$ using Cubic Approximations on Uniform and Varibly Spaced Grids.

		Uniform Spacing		Variable Spacing	
N	Δt	$\|\|e\|\|_{L_2}$	$\|\|e\|\|_\infty$	$\|\|e\|\|_{L_2}$	$\|\|e\|\|_\infty$
10	0.01	0.607 E-1	0.801 E-1	0.130 E-1	0.232 E-1
14	0.005	0.319 E-1	0.257 E-1	0.332 E-2	0.602 E-2
20	0.01	0.214 E-1	0.394 E-1	0.167 E-1	0.951 E-1
	0.005	0.185 E-1	0.309 E-1	0.483 E-2	0.950 E-2
	0.0025	0.138 E-1	0.405 E-2	0.353 E-3	0.170 E-2

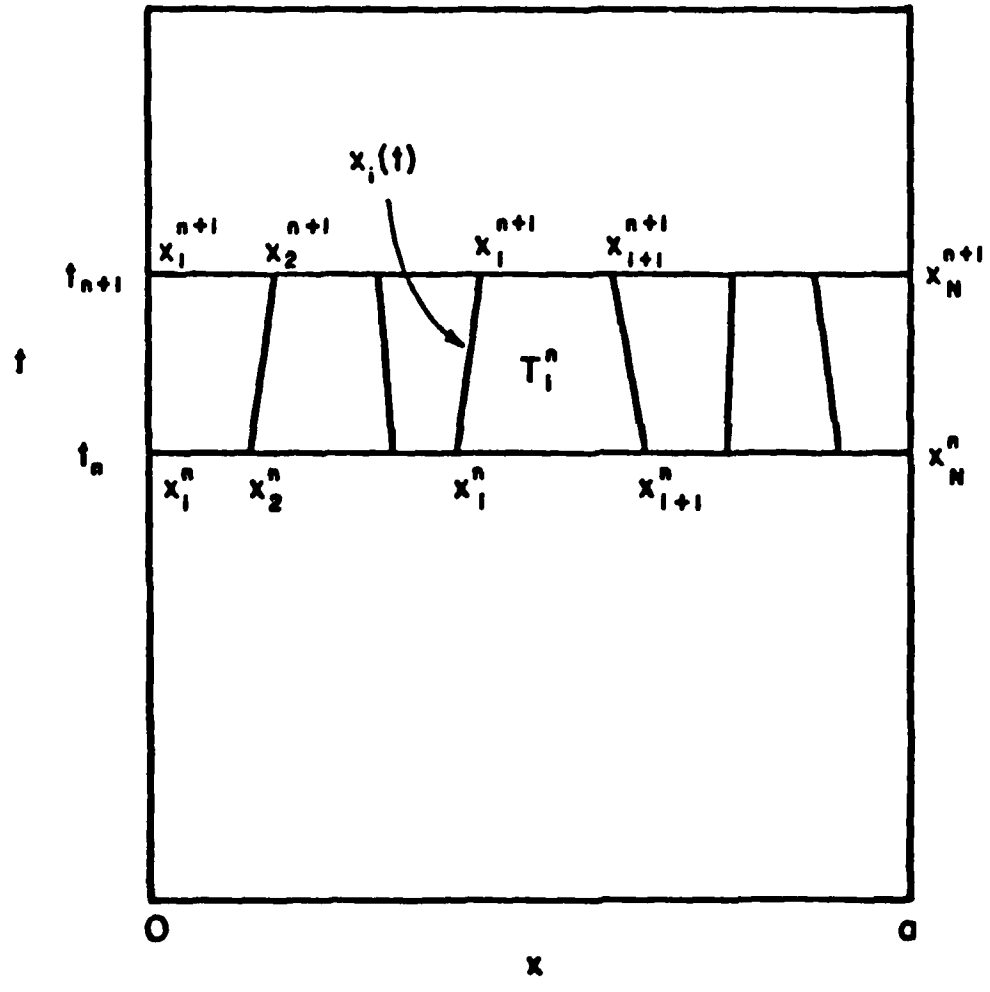


Figure 1: Space-time discretization for the time step $t_n \leq t \leq t_{n+1}$.

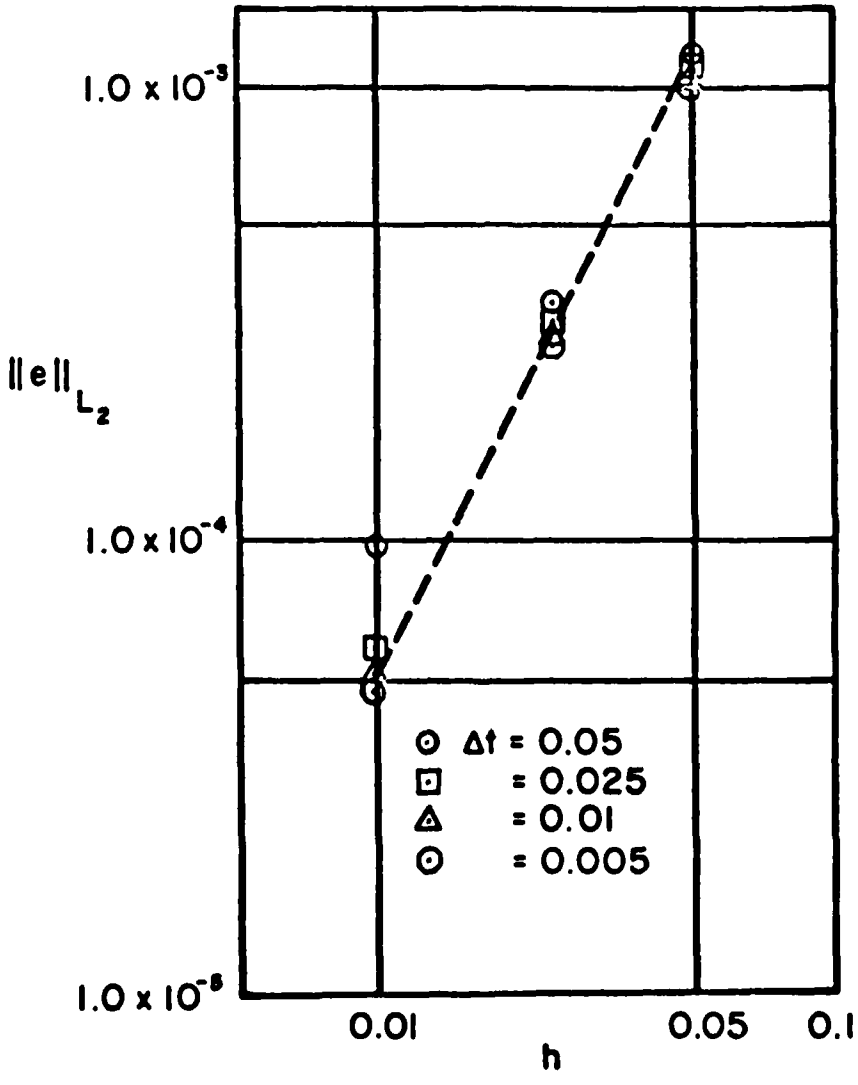


Figure 2: L_2 error vs. h for Example 1 computed on uniform meshes with linear approximations. The dotted line connects points for which $\Delta t = h$.

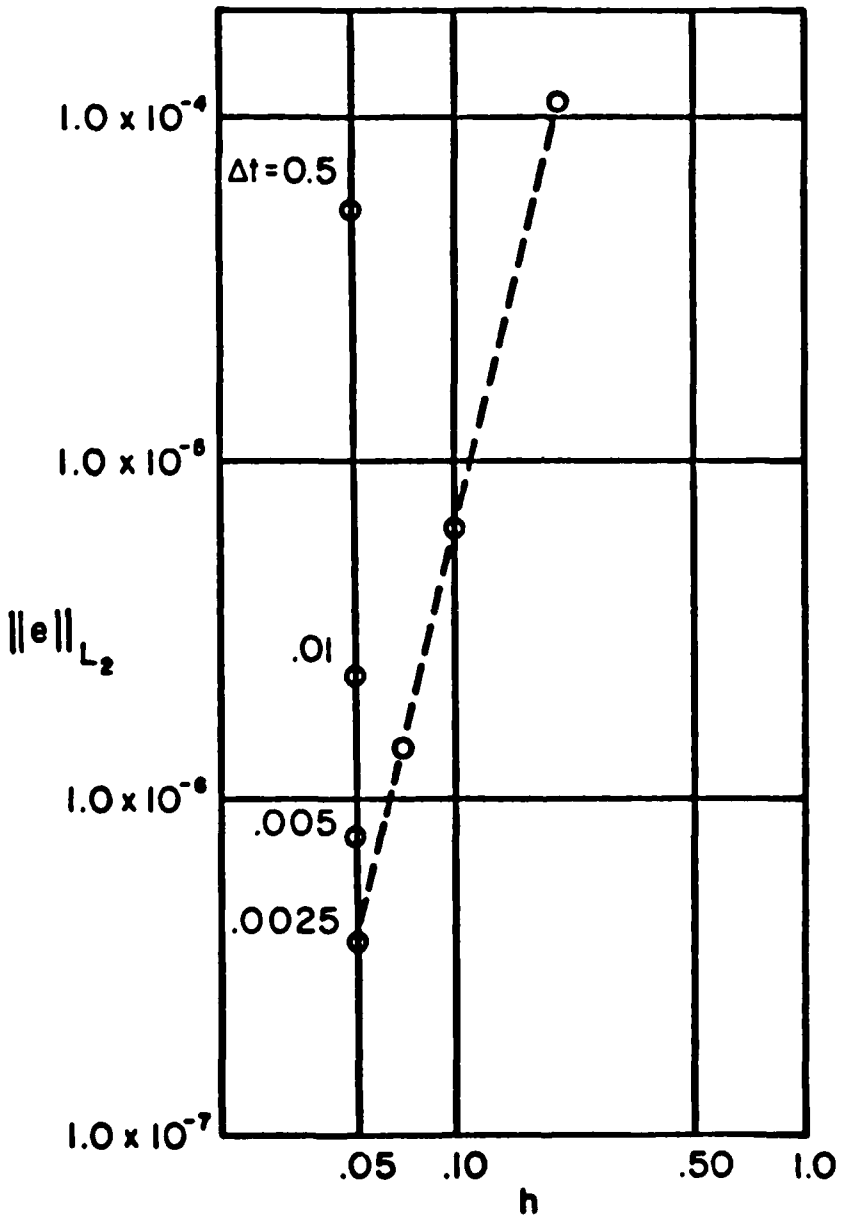


Figure 3: L_2 error vs. h for Example 1 computed on uniform meshes with cubic approximations. The dotted line connects points for which $\Delta t = h^2$.

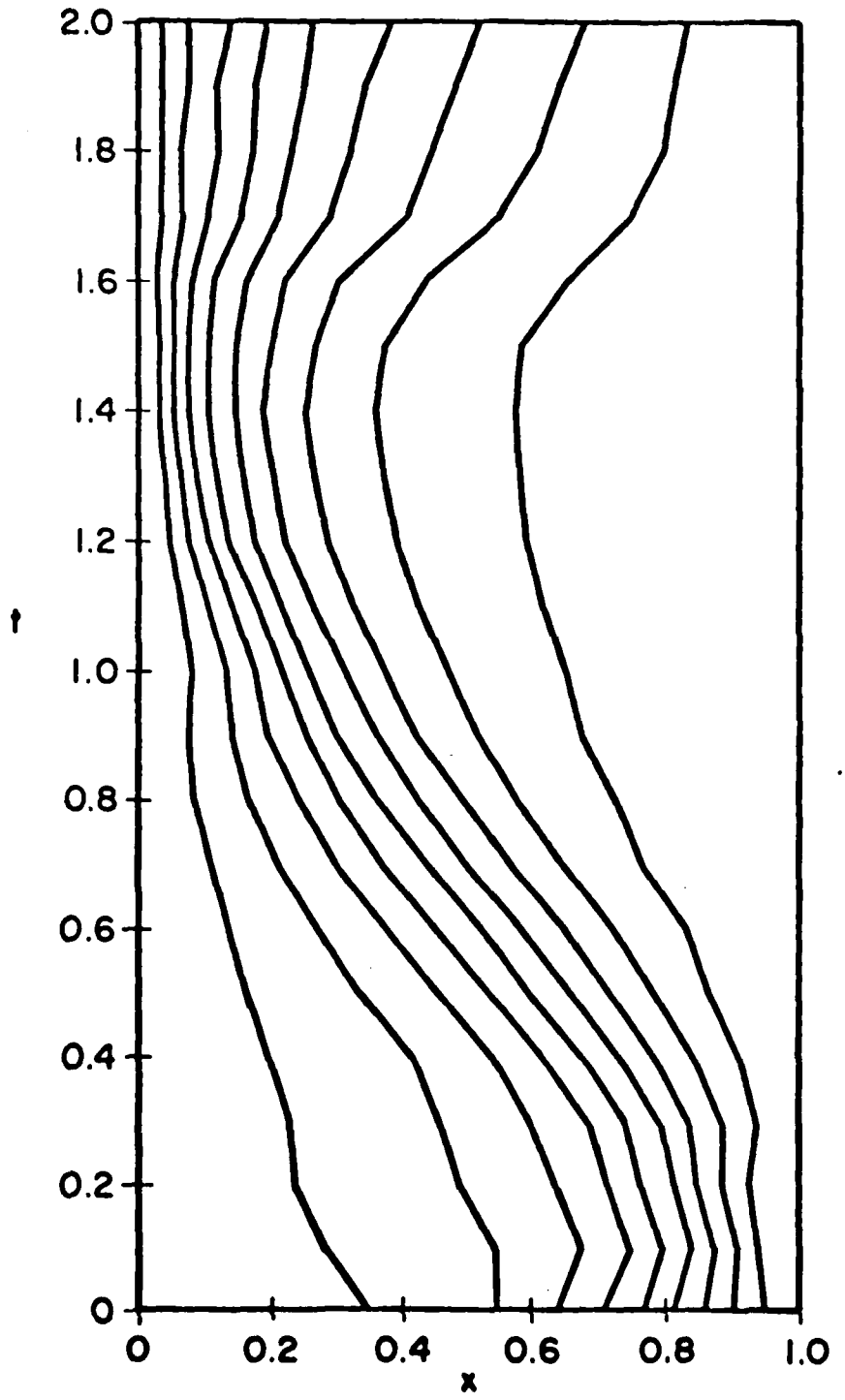


Figure 4: Mesh selected for Example 2 with $r_1 = r_2 = 5$, uniform time steps of $\Delta t = 0.1$, $N = 10$, and linear approximations.

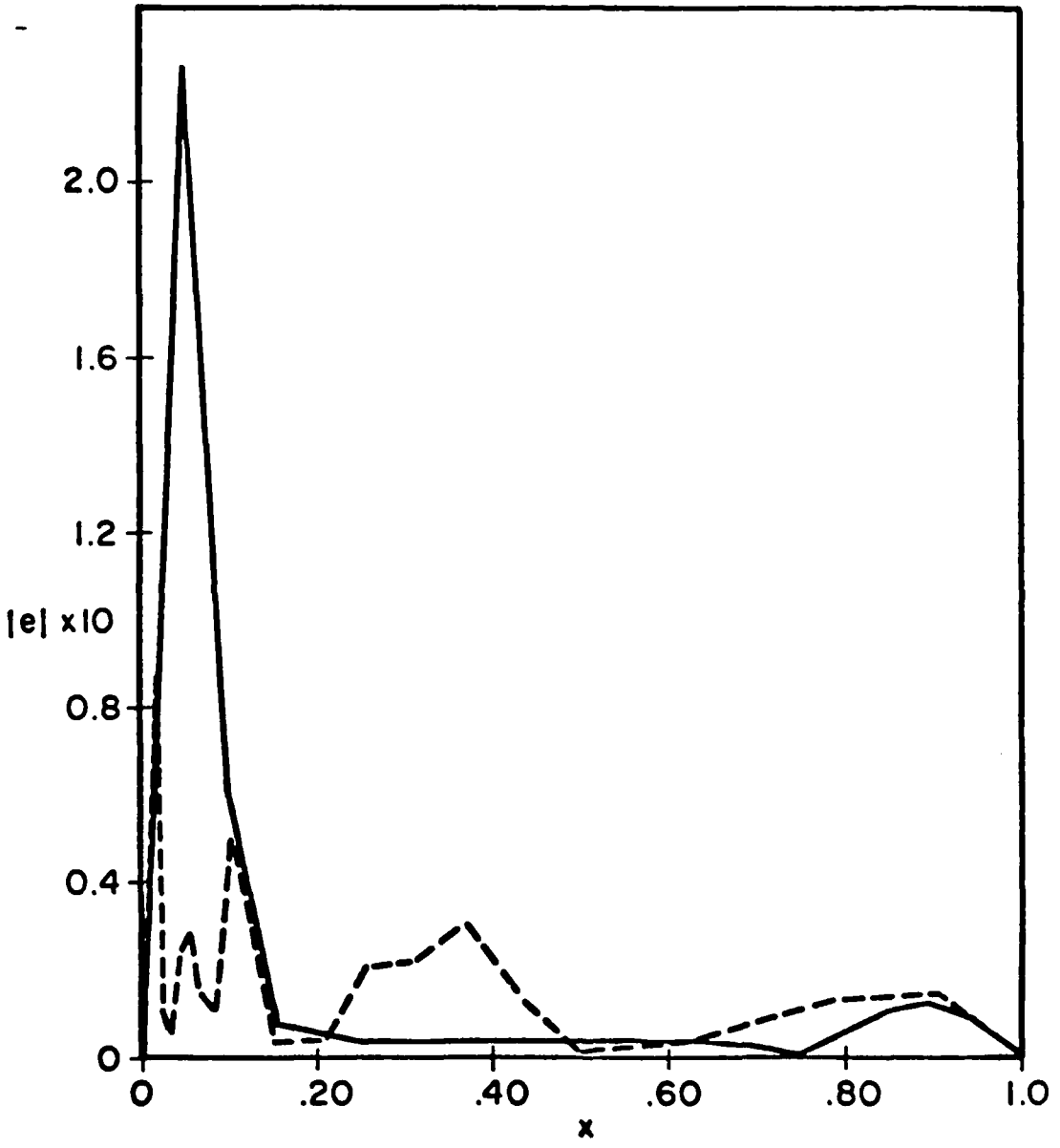


Figure 5: Local error at t = 1.0 for Example 2 with $r_1 = r_2 = 100$, uniform time steps of $\Delta t = 0.01$, N = 20, and linear approximations. The solid curve was computed on a fixed uniform mesh, the broken curve on a variable mesh.

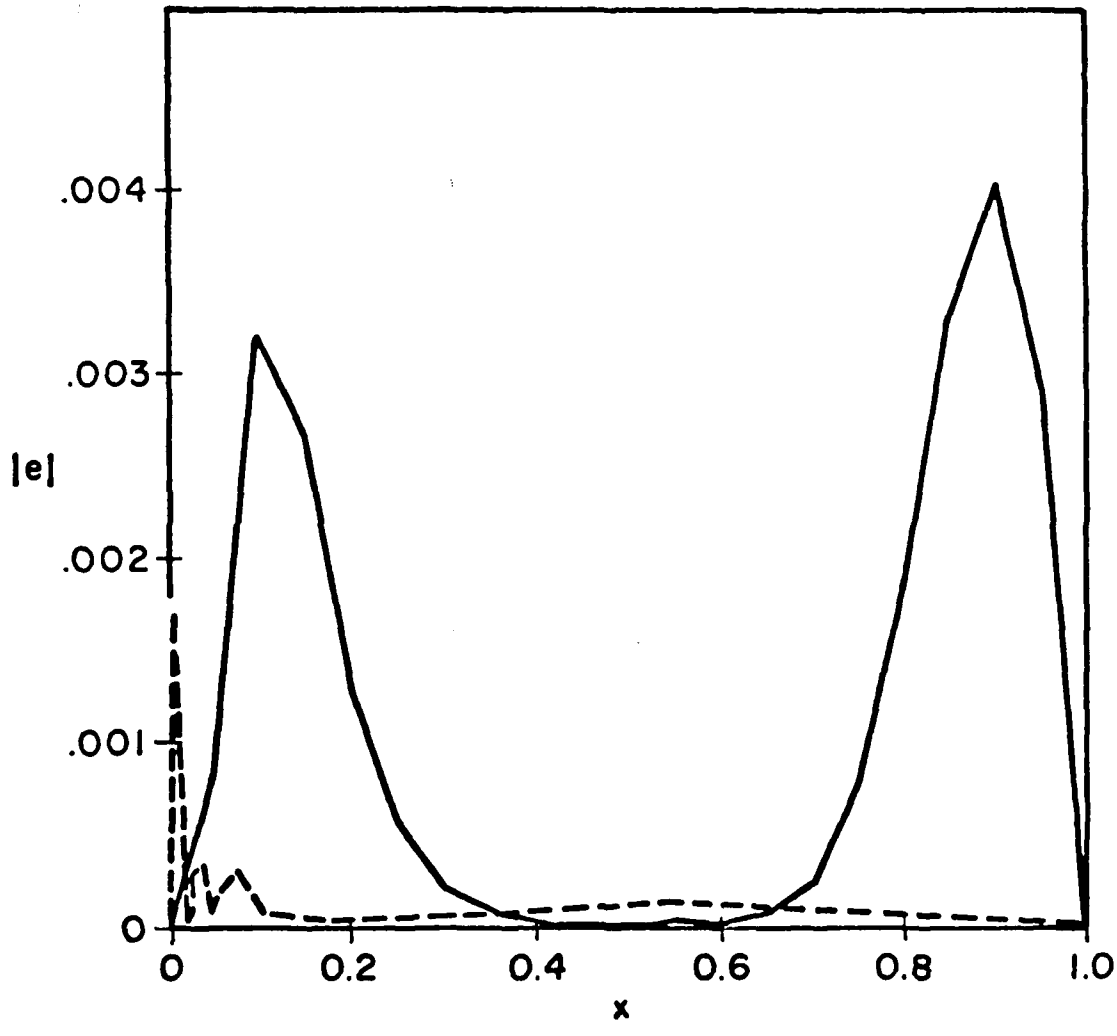


Figure 6: Local error at t = 1.0 for Example 2 with $r_1 = r_2 = 100$, uniform time steps of $\Delta t = 0.0025$, N = 20, and cubic approximations. The solid curve was computed on a fixed uniform mesh, the broken curve on a variable mesh.

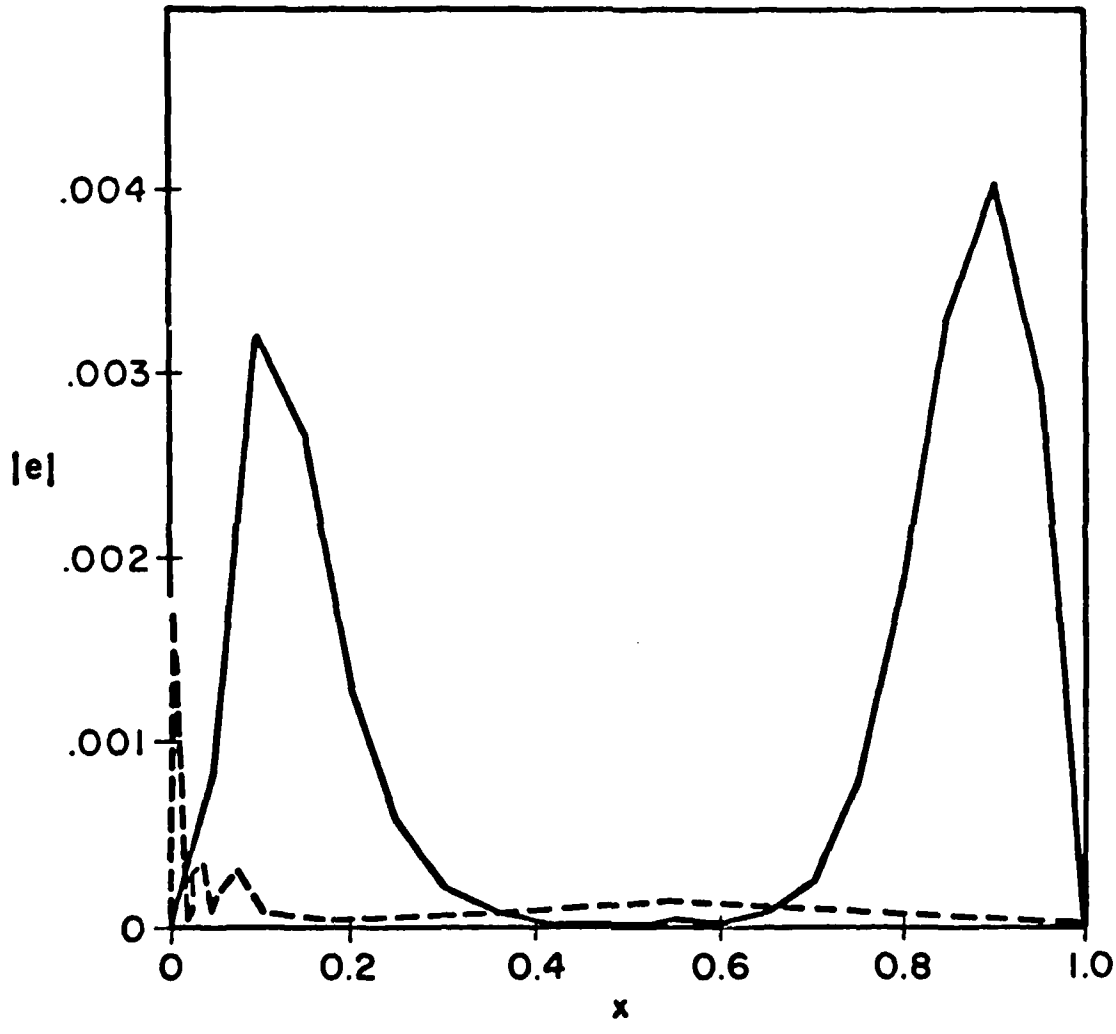


Figure 6: Local error at $t = 1.0$ for Example 2 with $r_1 = r_2 = 100$, uniform time steps of $\Delta t = 0.0025$, $N = 20$, and cubic approximations. The solid curve was computed on a fixed uniform mesh, the broken curve on a variable mesh.

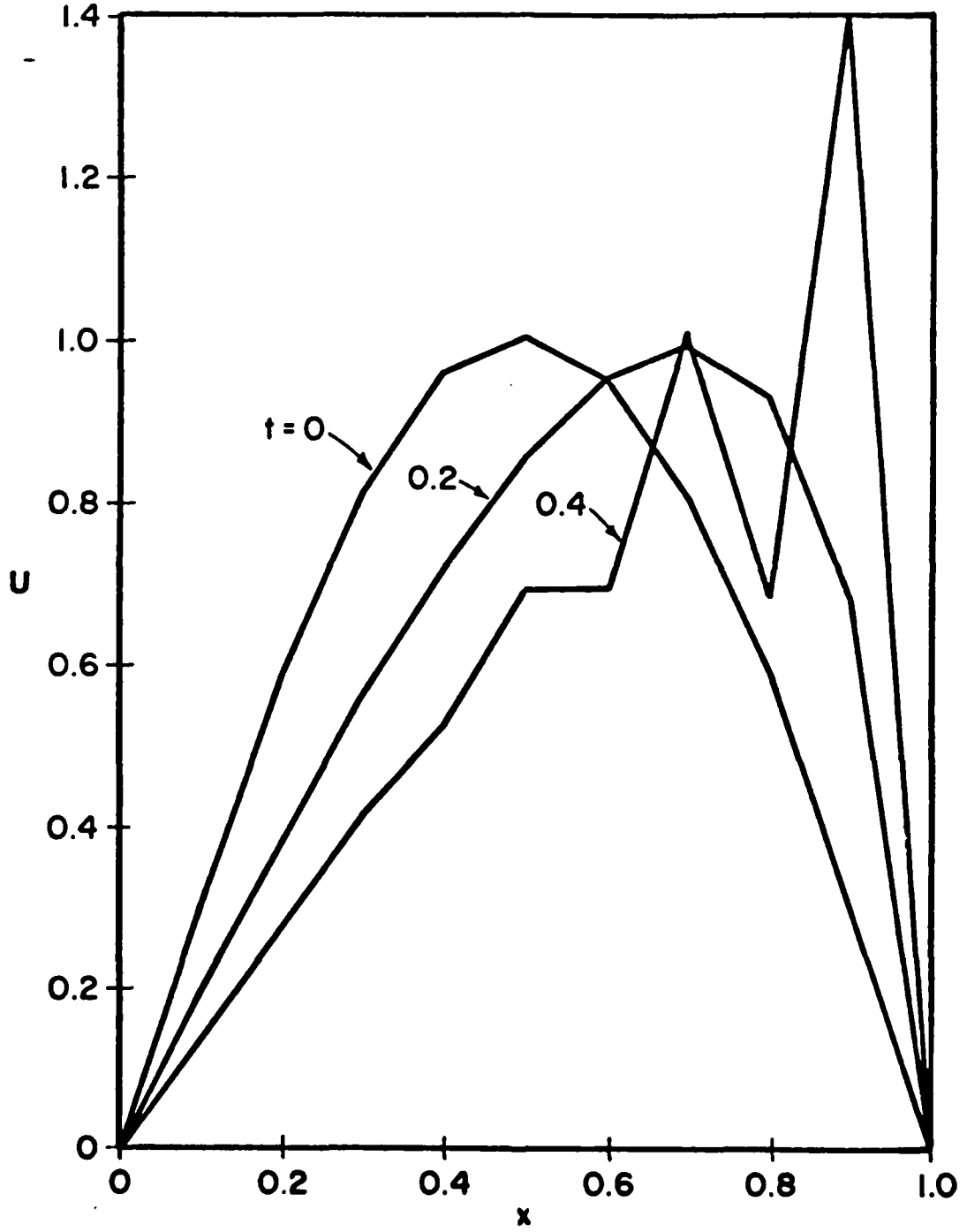


Figure 7: Solution of Example 3 for various values of t using linear approximations on a uniform mesh with $\Delta t = 0.1$ and $N = 10$.

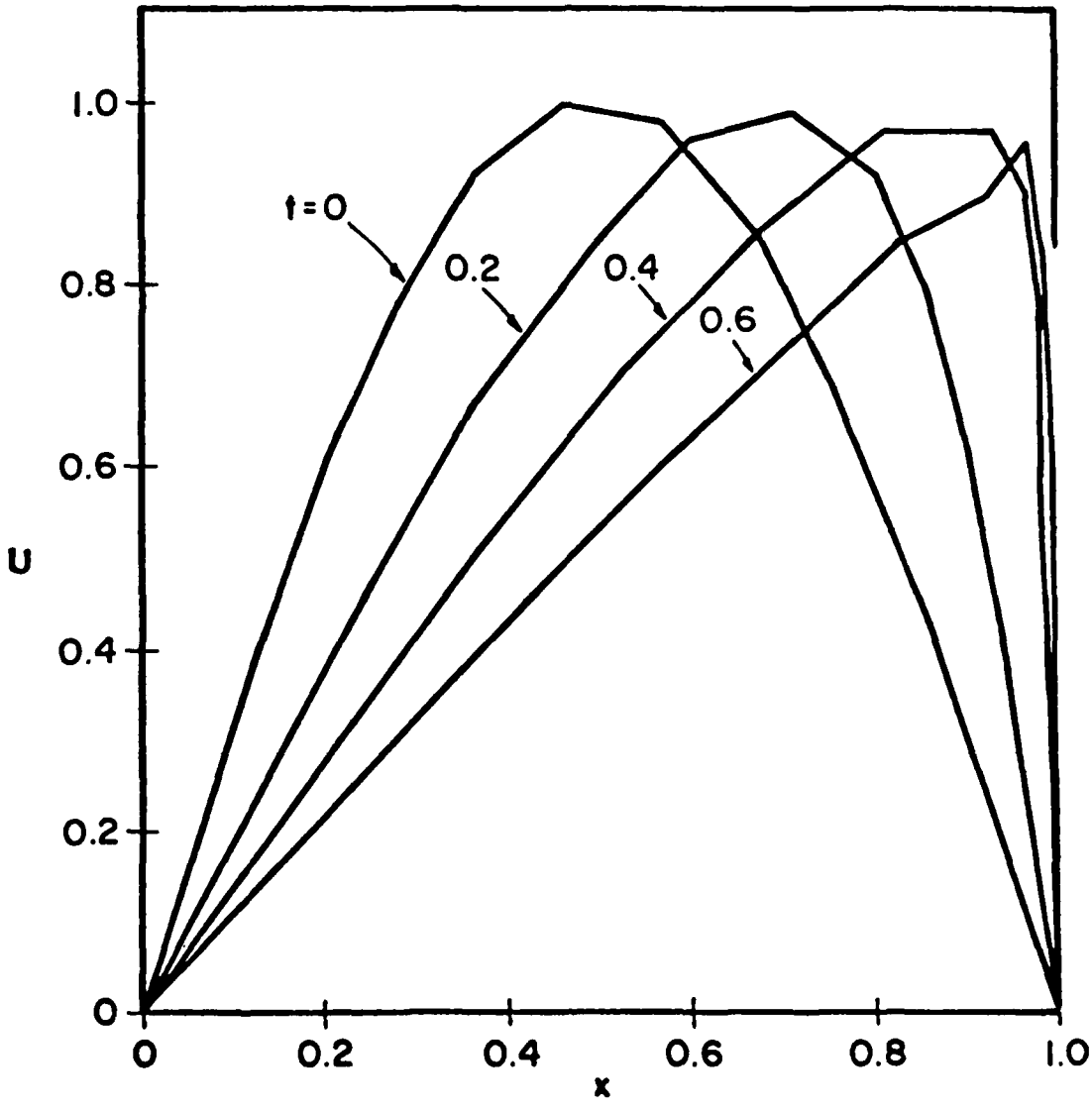


Figure 8: Solution of Example 3 for various values of t using linear approximations, uniform time steps of $\Delta t = 0.1$, and a variable mesh with N = 10 elements per time step.

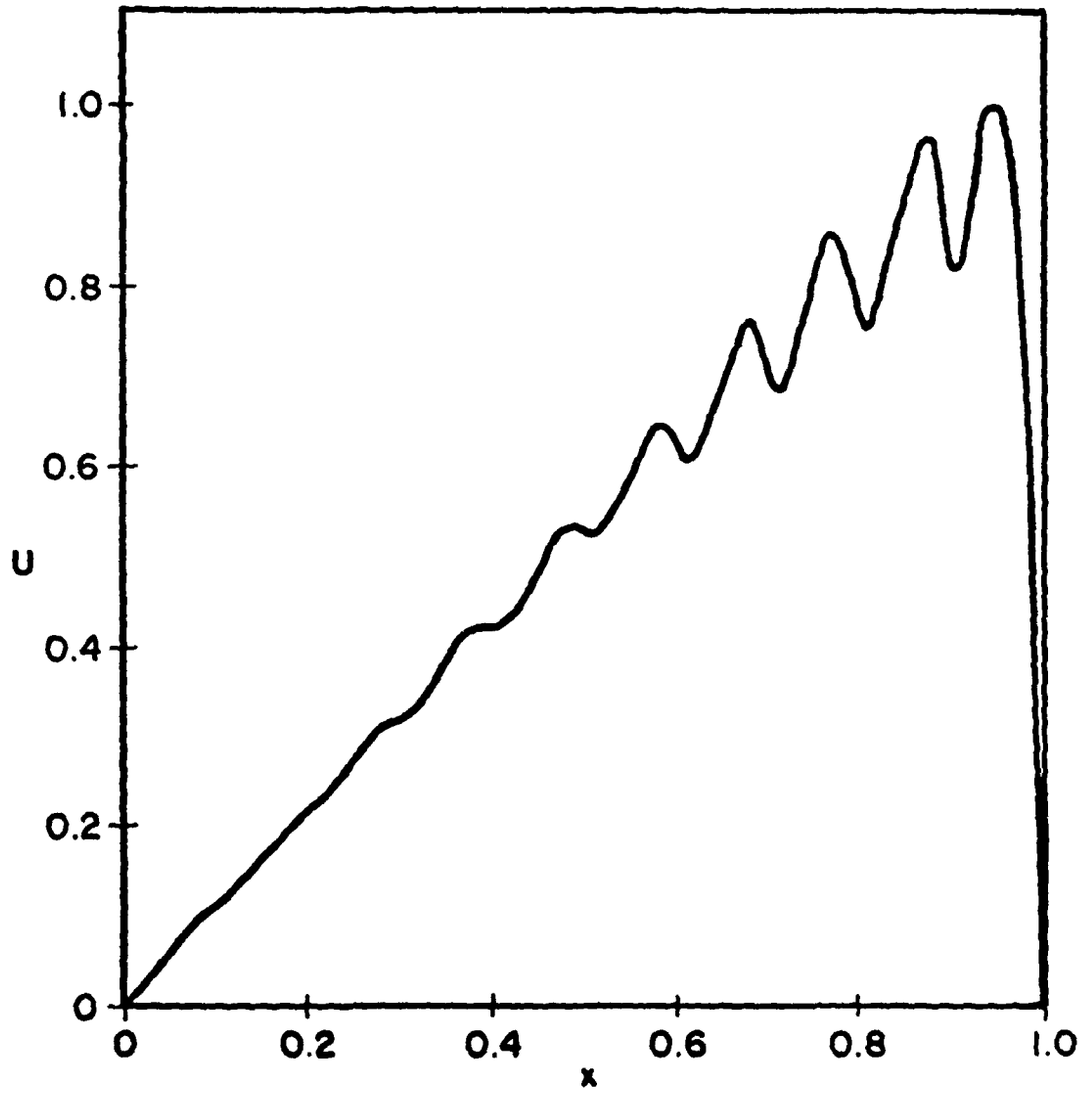


Figure 9: Solution of Example 3 at $t = 0.6$ using cubic approximations on a uniform mesh with $\Delta t = 0.01$ and $N = 10$.

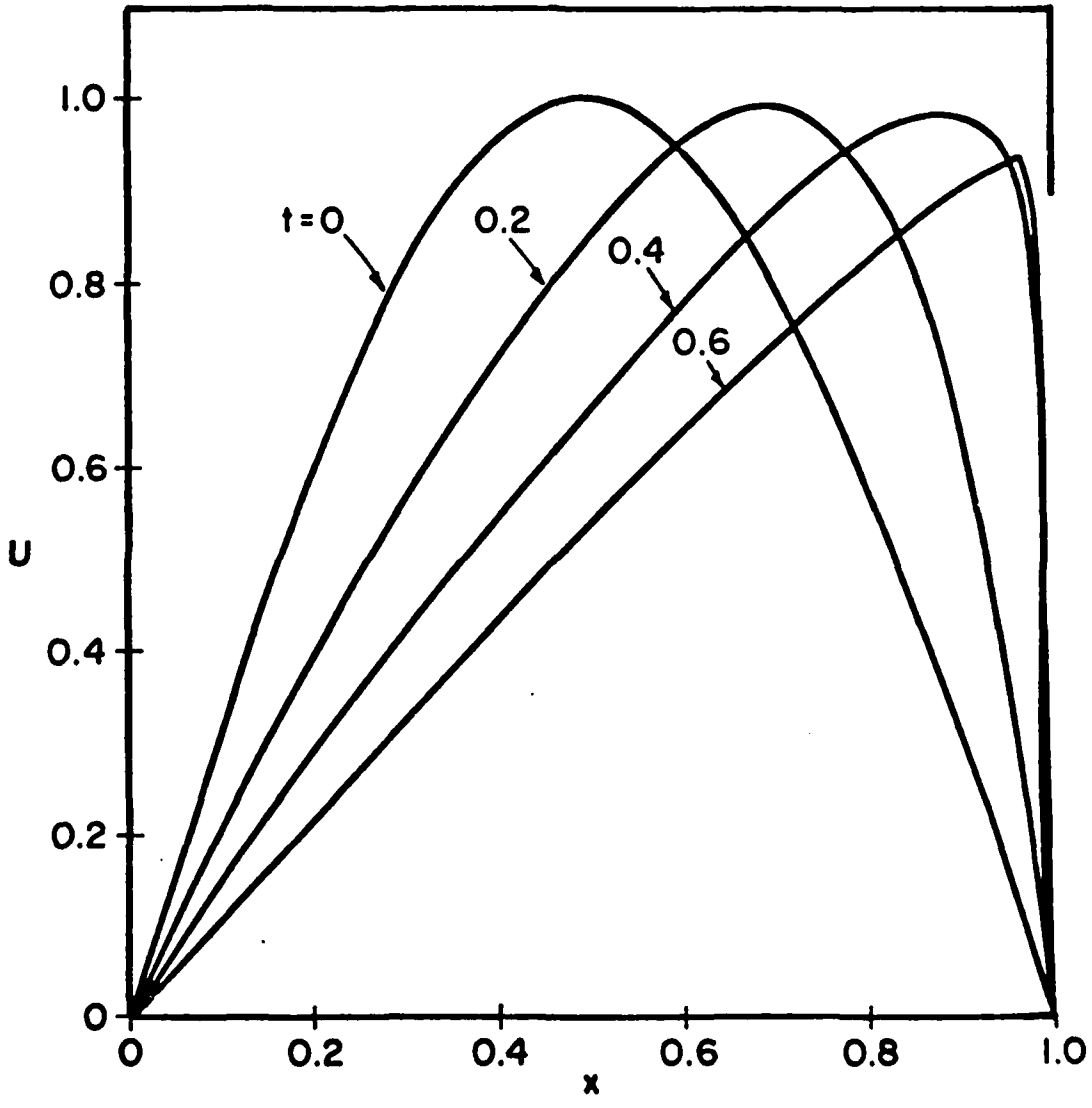


Figure 10: Solution of Example 3 for various values of t using cubic approximations, uniform time steps of $\Delta t = 0.01$, and a variable mesh with N = 10 elements per time step.

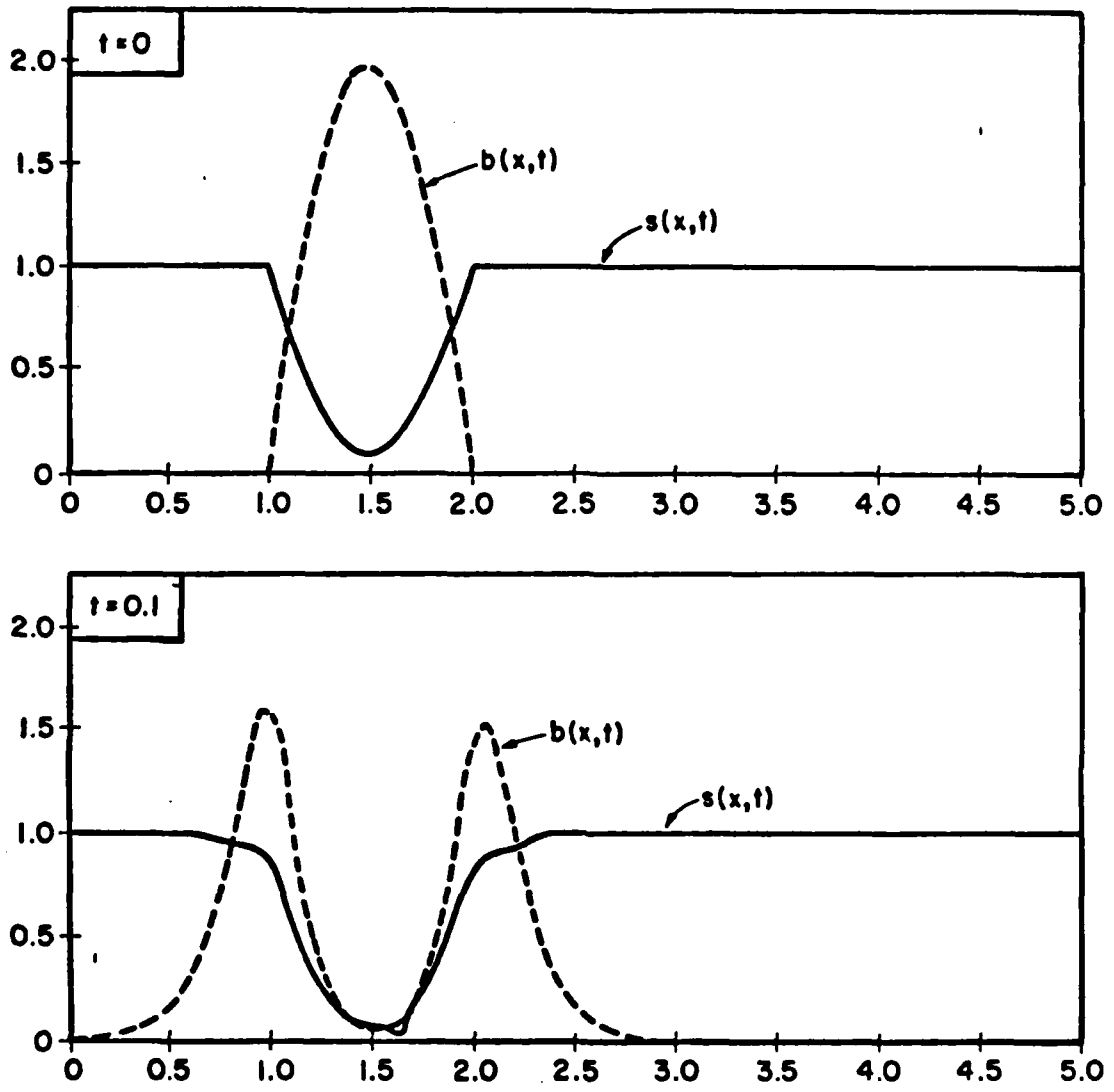
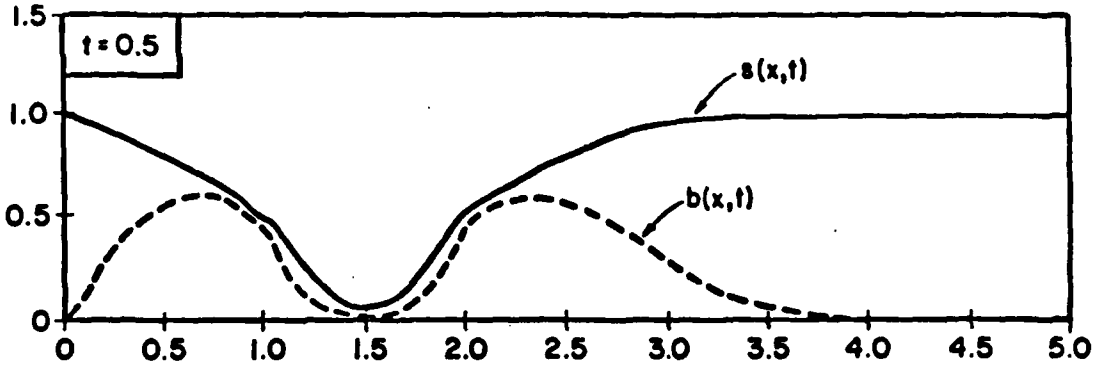
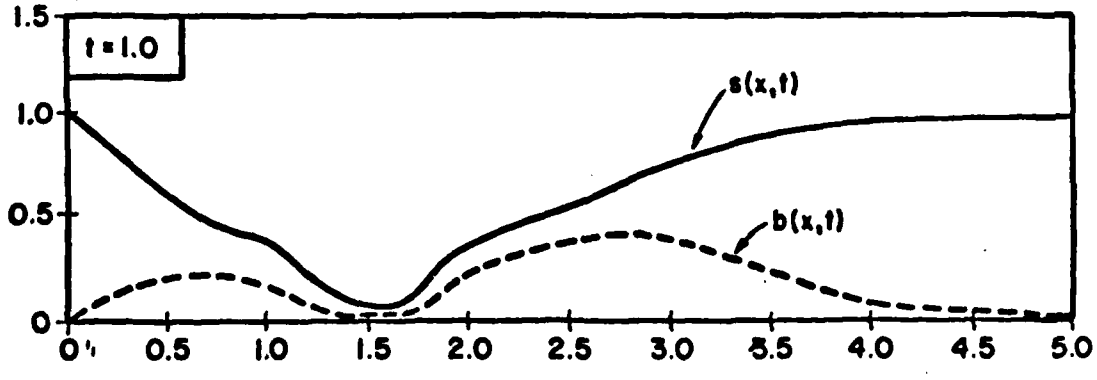
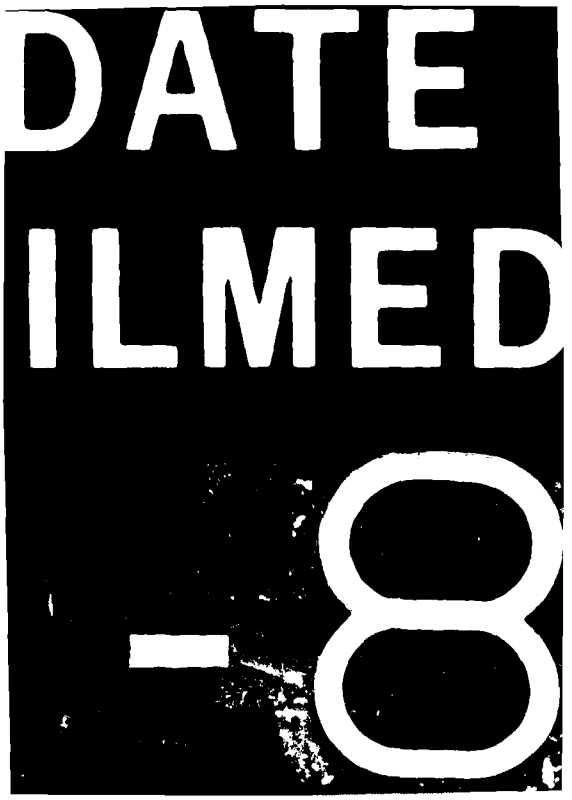


Figure 11: Computational results for Example 4 with $\delta_o/\mu_o = 2.0$ using uniform time steps of $\Delta t = 0.005$, $N = 50$, and cubic approximations at $t = 0$, 0.1, 0.5, and 1.0.


t = 0.5
s(x,t)
b(x,t)
1.5
1.0
0.5
0
0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0
4.5
5.0


t = 1.0
s(x,t)
b(x,t)
1.5
1.0
0.5
0
0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0
4.5
5.0


DATE
ILMED
-8