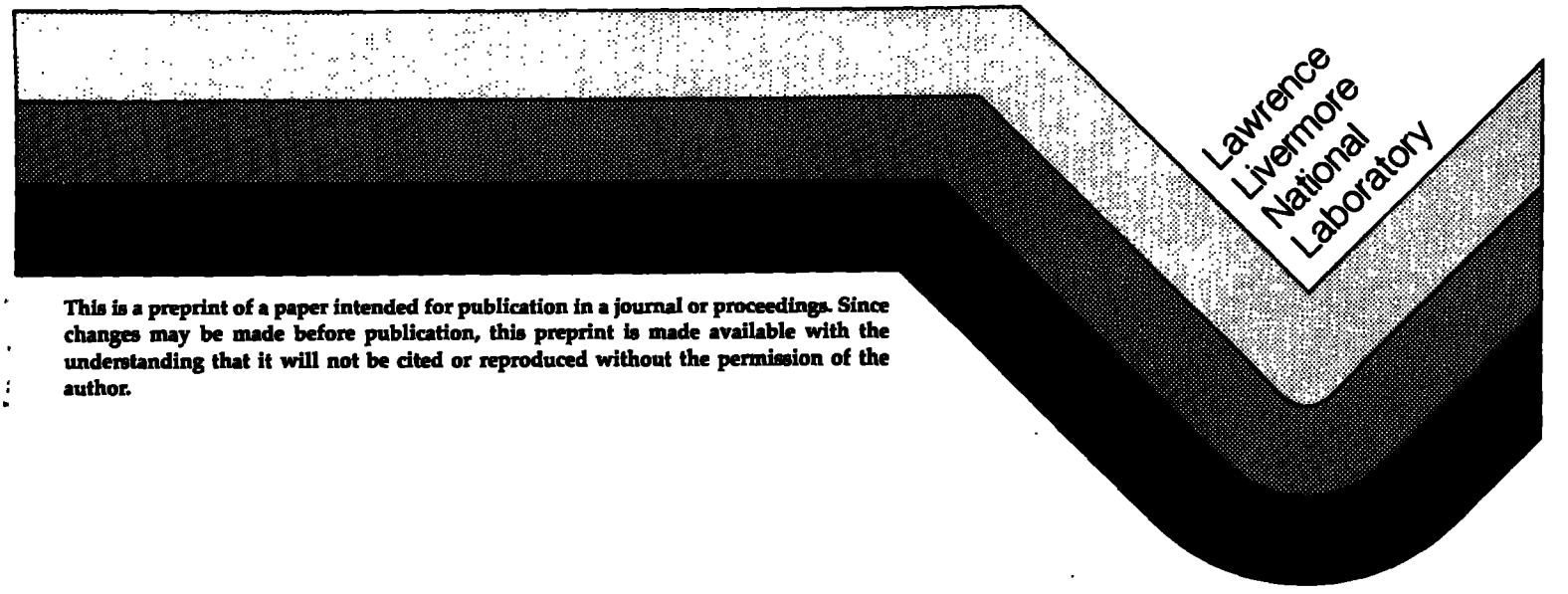
UCRL-94297
PREPRINT

STIFF ODE SOLVERS:
A REVIEW OF CURRENT AND COMING ATTRACTIONS

George D. Byrne
Alan C. Hindmarsh

This paper was prepared by invitation for
the Journal of Computational Physics.

March 1986



STIFF ODE SOLVERS:
A REVIEW OF CURRENT AND COMING ATTRACTIONS†

George D. Byrne* and Alan C. Hindmarsh**

† Prepared by invitation for Journal of Computational Physics

* Computing and Information Services Division
Exxon Research and Engineering Company
Clinton Township
Route 22 East
Annandale, New Jersey 08801

** Computing and Mathematics Research Division L-316
Lawrence Livermore National Laboratory
P.O. Box 808
Livermore, California 94550

This work was in part performed under the auspices of the U.S. Department of Energy by the Lawrence Livermore National Laboratory under contract No. W-7405-Eng-48.

DEDICATION

This work is dedicated to
the memory of Dennis William Byrne,
May 2, 1966 - April 16, 1985.

ABSTRACT

Stiff ordinary differential equations (ODE's) *can* now be solved more or less routinely. This review is intended for the user who is interested in learning how to solve these systems of equations. Consequently, this review explains what stiff ODE's are and where they arise. It tells what is expected of the software and sketches how it works. So, several tried and true, as well as experimental, numerical methods are outlined. Perhaps the most salient feature is a set of examples that have been compiled during the last fifteen years. These examples include several prototypical problems. The problems are deliberately tractable in size, yet display features of much larger problems in science and engineering. In each case, the choice of the software package, the ODE solver, is given. These choices are based on the authors' combined experience and take into account problem structure.

This then is a brief handbook which could be used to learn or to teach the pragmatics of numerically solving stiff ODE's. This work should be useful to line scientists, scientific computing managers, and to students.

TABLE OF CONTENTS

		Page Nos.
1.	**Introduction**	1
	A. ODEs in Normal Form and an Example	1
	B. The Notion of Stiffness	3
	C. The Connection Between Stiffness and Stability	6
	D. Stiffness and the Method of Lines	10
	E. Linearly Implicit ODEs, Differential Algebraic Systems, and the Method of Lines	11
	F. Problem Structure	13
	G. Active Time Scales	15
2.	**Survey of Methods**	17
	A. Linear Multistep Methods	18
	B. Runge-Kutta Methods	26
	C. Other Methods	28
	i. Collocation methods	29
	ii. Block and composite methods	29
	iii. Extrapolation	29
	iv. Multiderivative methods	30
	v. Blended methods and matrix coefficient methods	30
	vi. Averaging	30
	vii. Fitting	30
	viii. Hybrid methods	30
	ix. Partitioning	31
	x. One-leg methods	31
	D. Pros and Cons	32
3.	**Software**	37
	A. Brief Historical Background	37
	B. LSODE and Its Variants	40
	C. EPISODE and Its Variants	42
	D. Other Descendants of GEAR and EPISODE	42
	E. Differential-Algebraic System Solvers	44
	F. Runge-Kutta Codes	45
	G. Blended and Composite Multistep Codes	45
	H. Extrapolation Codes	46
	I. Second-Derivative Codes	46

Page Nos.

4. **Example Problems and Code Usage.** 47

A. The Example Problems 47
Problem 1 - Robertson's Problem. 47
Problem 2 - The Field-Noyes Chemical Oscillator. . . . 48
Problem 3 - Two Species Diurnal Kinetics 49
Problem 4 - A Kidney Model 51
Problem 5 - A Laser Oscillator Model 52
Problem 6 - Burgers' Equation. 54
Problem 7 - Two Species Diffusion-Diurnal Kinetics - One Dimensional. 57
Problem 8 - Two Species Diffusion-Diurnal Kinetics - Two Dimensional. 62
Problem 9 - A Two Phase Plug Flow Problem. 64
Problem10 - Troesch's Two Point Boundary Value Problem. 67

B. Code Usage and Computational Results 69

Problem 1 - Robertson's Problem. 69
Problem 2 - The Field-Noyes Chemical Oscillator. . . . 71
Problem 3 - Two Species Diurnal Kinetics 72
Problem 4 - A Kidney Model 73
Problem 5 - A Laser Oscillator Model 74
Problem 6 - Burgers' Equation. 76
Problem 7 - Two Species Diffusion - Diurnal Kinetics- One Dimensional. 77
Problem 8 - Two Species Diffusion - Diurnal Kinetics- Two Dimensional. 77
Problem 9 - A Two Phase Plug Flow Problem. 79
Problem10 - Troesch's Two Point Boundary Value Problem. 82

5. **Related Developments** 83

6. **Summary.** 86

7. **Acknowledgements** 88

8. **References** 89

Appendix 102

Figures. 104

1. INTRODUCTION

Software for stiff systems of ordinary differential equations (ODEs) has enjoyed a wide range of acceptance during the past decade. As a consequence, its impact has been substantial in the physical sciences and in engineering.

The intent of this paper is to give a review of modern methods and software solvers that are currently in use for stiff ODE systems. We also give examples showing both the challenges to the software and the capabilities of the solvers.

In the remainder of this section, we discuss ODE forms, stiffness, problem structures, and other features arising in stiff systems of interest. In Section 2, we survey the basic methods that are used, with some comments on their relative merits. Section 3 is a description of available software for stiff systems, as far as we are aware of it. In Section 4, we give ten example problems in detail, followed by descriptions of their solutions using available solvers. Section 5 mentions some related developments, and Section 6 summarizes the paper. We expect the reader to choose those parts of the paper that are of greatest interest.

A. ODEs in Normal Form and an Example

We begin with the canonical first order initial value problem and discuss problems in other forms later. We represent the first type as

$$\dot{y} = f(t,y),\ t_0 \leq t \leq t_{final} \tag{1.1}$$

$$y(t_0) = y_0 \tag{1.2}$$

where:

N is the number of scalar first order ODEs.

t is the time-like independent variable.

$y = [y^1, y^2, \ldots, y^N]^T$ is the column N-vector of dependent variables, and the superscript T denotes vector transpose.

$\dot{} = d/dt$ denotes differentiation with respect to t.

f is a N-vector valued function of y and t.

t_0 is the initial value or starting value and is given.

t_{final} is the final value of the interval of integration.

y_0 is the initial value N-vector.

In terms of the components of (1.1) and (1.2), we have

$$\dot{y}^i = f^i(t, y^1, y^2, \ldots, y^N)$$
$$y^i(t_0) = y^i{}_0$$

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

As an example of a system of stiff ODEs, we borrow a model of a chemical reaction which has been described in several places [Robertson (66)] [Byrne, et al. (77)] [Hindmarsh & Byrne (76a)] [Enright & Hull (76)] and can be described as a reaction of the type:

$$A_1 \longrightarrow A_2, \quad 0.04$$
$$A_2 + A_3 \longrightarrow A_1, \quad 10^4$$
$$2A_2 \longrightarrow 2A_3, \quad 1.5 \times 10^7$$

$$\left.\begin{aligned} \dot{y}^1 &= -0.04\, y^1 + 10^4 y^2 y^3 \\ \dot{y}^2 &= 0.04\, y^1 - 10^4 y^2 y^3 - 3\times10^7 y^2 y^2 \\ \dot{y}^3 &= 3\times10^7 y^2 y^2 \end{aligned}\right\} \qquad (1.3)$$

$$\left.\begin{aligned} y^1(0) &= 1 \\ y^2(0) &= 0 \\ y^3(0) &= 0 \end{aligned}\right\} \qquad (1.4)$$

This reaction is interesting because the reaction rate coefficients (the constants on the right hand side of (1.3)) vary over nine orders of magnitude. Moreover, it can be shown that as $t \to \infty$, $y^1 \to 0$, $y^2 \to 0$ and $y^3 \to 1$. Also, by looking at the system or by an appropriate computation [Hindmarsh & Byrne (76a)] we can see that the dominant equation at equilibrium is $\dot{y}^2 = -10^4 y^2$. Note that for any choice of initial value, the solution of this equation is a strongly damped exponential which is typical of stiff systems.

B. The Notion of Stiffness

We can now turn to the concept of stiffness. We will first give a rough notion and then (in the next subsection) a more precise one.

A prototypical stiff differential equation can be given by

$$\dot{y} = -10^3[y-\exp(-t)] - [\exp(-t)] \quad 0 \le t \le t_{final} \qquad (1.5)$$

$$y(0) = 0 \qquad (1.6)$$

where y is a scalar. The exact solution of this problem is,

$$y(t) = \exp(-t) - \exp(-10^3 t) \qquad (1.7)$$

and is seen to be comprised of two components, one of which ($\exp(-10^3 t)$) varies much more rapidly in t than the other ($\exp(-t)$). For this problem,

the notion of stiffness can be formalized somewhat if the time constants $\tau_1=10^{-3}$ and $\tau_2=1$ are introduced. Note that at $t = \tau_1$, the fast component is $\exp(-10^3\tau_1) = \exp(-1)$ and the slow component at $t = \tau_2$ is $\exp(-\tau_2) = \exp(-1)$. These time constants correspond to values of t for which their respective components have the value exp(-1). Stiffness in this problem is caused by the presence of a small decay time constant τ_1.

For t smaller than several times as large as τ_1, the fast component makes significant contributions to the value of the solution. This range of values of t is often called the transient interval. If the range of integration is restricted to the transient interval, we would not need to resort to any special numerical methods of integration, and the problem would *not* be considered stiff. However, beyond the transient interval, the value of the solution is essentially the value of the slow component. Yet the presence of the fast component (even though fully decayed) forces the use of either a very small step size (comparable to τ_1), if a traditional explicit method is used to solve (1.5) -(1.6), or else a stiff solver. Beyond the transient, the problem is stiff.

Thus the stiffness is determined by the range of integration, i.e., by t_{final}. The value of τ_2 would dictate that $t_{final} \geq 1$ for a complete picture of the solution, but a smaller value might be posed instead. In any case, a quantitative measure of the stiffness of this prototype problem is

$$S = t_{final}/\tau_1.$$

If S is on the order of 1000 or larger, we would certainly regard the problem as stiff. If S is less than 10, the problem would be non-stiff. The intermediate values of S would correspond to problems whose descriptions would range from non-stiff, through mildly stiff, to stiff. The numerial values are valid, but the transition from non-stiff to stiff is not sharply defined.

The same kind of observation can be made of the kinetics problem (1.3) - (1.4), but the stiffness is somewhat less transparent. Recall that near the equilibrium values, the second equation reduces approximately to the simple ODE $\dot{y}^2 = -10^4 y^2$, which has an exponential decay time constant of $\tau = 10^{-4}$. On the other hand, a complete picture of the approach to equilibrium turns out to require integrating to about $t = 10^7$. Thus we again have a rapid decay time constant that is much shorter (or smaller) than the time range t_{final}, and it is clear that we are looking at a stiff system of ODEs. In both cases, the essential features of a stiff system have been captured: disparate time constants, an interval of integration several times longer than the shortest time constants, and an approach to the steady state that does not involve rapid oscillations.

There have been several definitions and descriptions of stiff systems of ordinary differential equations given, e.g. [Shampine & Gear (79)] [Byrne & Hindmarsh (77)] [Price, et al. (66)] [Curtiss & Hirschfelder (52)]. Perhaps the most pragmatic way to determine the stiffness of a system of ODEs is simply to solve it with a non-stiff differential equations package such as ODE [Shampine & Gordon (75)], DVERK [Hull, et al. (76)] or DERKF [Shampine & Watts (80)], to name but a few. Then, record the cost of solving the problem. By the way, it would be prudent to impose a limit on CPU time or the number of function evaluations. Similarly, solve the problem with a stiff ODE package such as LSODE [Hindmarsh (80)], DGEAR [IMSL (82)] or EPISODE [Hindmarsh & Byrne (77)] or an appropriate relative. Upper bounds on the cost should again be imposed. Now compare the costs of the two solutions over the same time interval. If the stiff ODE solver was substantially less expensive to use than the non-stiff solver, then the problem was stiff. If the non-stiff solver was the less expensive, then the problem is non-stiff. Between these extremes are

mildly stiff problems and, perhaps, other categories. We have not addressed here the idea of appropriateness for non-stiff solvers on parts of the interval of integration and stiff solvers on other subintervals. However, this issue is now addressed by some packages that switch methodologies through a stiffness detector [Petzold (83b)].

To illustrate this empirical determination procedure, consider the solution of (1.5)-(1.6) by DERKF (a nonstiff/ODE solver) and by LSODE (with a stiff method option). We pose the same error tolerances (absolute error tolerance = 10^{-5}) for both, and ask for output at t = .001, .01, .1, 1, being careful to constrain the number of internal time steps to 500. The results are as follows: DERKF completed the problem at a cost of 1876 evaluations of f, while LSODE completed it with 136 (including those for evaluating $\partial f/\partial y$). The run times were in a ratio of about 2.4 to 1. Both solutions had five digits of accuracy at all output times. But the higher cost of the solution from the nonstiff solver clearly indicates that the problem is stiff. For other problems, the cost (run time) ratios may be significantly larger. Shampine (85) gives further empirical measures.

C. The Connection Between Stiffness and Stability

The notions of stiffness and stability are related. Let us briefly review how [Byrne & Hindmarsh (77)] [Shampine & Gear (79)]. Suppose we have two distinct solutions of (1.1), say y and w. Then

$$\dot{y} - \dot{w} = f(t,y) - f(t,w)$$

If we neglect higher order terms, then

$$\dot{y} - \dot{w} = f_y(t,w)\,(y-w)$$

If we assume that y-w is sufficiently small, in an approximate sense, then

$$\dot{y}-\dot{w} = J\,(y-w)$$

where $J = f_y$ is the Jacobian matrix given by

$$J_{ij} = \frac{\partial f^i}{\partial y^j}(t,w)$$

Here, J_{ij} is the element of J in row i, column j. We assume that J is locally a constant. If J is a stable matrix (all eigenvalues of J have negative real parts) then $y-w \to 0$ as $t \to \infty$. If we reevaluate J as t increases, and require that each J be locally constant and a stable matrix, then it follows that y and w tend to the same *finite* function as $t \to \infty$. That is, (1.1) is *stable*. By stable, we mean that given any two particular solutions y and w of (1.1), they tend to the same finite function as $t \to \infty$. (Other kinds of stability are also important, but this is the one needed here.) The connection between stiff ODEs and stable ODEs is this: Stiff ODEs are extremely stable, in that there is at least one eigenvalue with a large negative real part. In fact, they can be called *super-stable* [Shampine & Gear (79)].

A more rigorous definition of stiffness was also given by Shampine and Gear (79):

> "By a stiff problem we mean one for which no solution component is unstable (no eigenvalue [of the Jacobian matrix] has a real part which is at all large and positive) and at least some component is very stable (at least one eigenvalue has a real part which is large and negative). Further, we will not call a problem stiff unless its solution is slowly varying with respect to the most negative part of the eigenvalues....Consequently a problem may be stiff for some intervals and not for others."

By this definition, non-negative real parts in the spectrum of the Jacobian matrix are acceptable, as long as they reflect neutral or slowly growing modes in the mathematical model. Further, these modes must also stay within reasonable bounds over the time interval, $t_0 \leq t \leq t_{final}$, of interest, and must be slowly varying compared to the most strongly damped mode.

To illustrate this point, the scalar example (1.5) has one eigenvalue, $\partial\dot{y}/\partial y = -10^3$. By (1.7), the solution beyond the transient is essentially exp(-t). It is indeed slowly varying with respect to $\exp(-10^3 t)$. By contrast, if we replace the forcing function exp(-t) by (for example) sin(100t), the solution no longer varies slowly relative to the strongly damped mode, and the problem is not stiff.

The latter point relates closely to a common misunderstanding of stiffness. Problems which have undamped high frequency oscillations in the solution, whether attributable to forcing functions or to eigenvalues

with large imaginary parts, are called stiff by some authors. We (and most authors) do *not* call such problems stiff. One reason is that highly oscillatory problems require numerical approaches that differ radically from those for stiff problems.

The above definition leads to a quantitative definition of a stiffness ratio, matching that used in the simple example earlier. We simply need to identify the eigenvalue λ with the largest negative real part, define the smallest time constant to be $\tau = -1/\mathrm{Re}(\lambda)$, and define the stiffness ratio to be

$$S = (t_{final} - t_0)/\tau \qquad (1.8)$$

Unfortunately, this is also not a precise definition, because in general τ varies along the solution. We can only use (1.8) in a local sense, applying it to subintervals where τ is essentially a constant.

Now, let us return to the kinetics model (1.3)-(1.4). Some computations show that one associated eigenvalue is always 0. (The simplest way to see this is to note that $\dot{y}^1+\dot{y}^2+\dot{y}^3 = 0$, which in turn tells us that mass is conserved.) Moreover, at equilibrium (as $t \to \infty$), the eigenvalues are 0, 0, and -10^4-0.04. Thus the problem is certainly stiff if t_{final} is of order 1 or larger. By extending the notion of stability we examined earlier to include neutral stability (eigenvalues equal to zero), we can also show the problem is (neutrally) stable. If, however, we numerically perturb the asymptotically zero eigenvalue to a positive value, then the problem can become numerically unstable. This instability can arise through numerical round-off or insufficiently stringent error control. This feature makes this problem computationally challenging if we want to solve it for large t. [Hindmarsh & Byrne (76a)] [Curtis (78)]

D. Stiffness and the Method of Lines

So far, we have seen that stiff systems of differential equations arise directly from a model. However, they can also arise in another way -- the spatial discretization of parabolic partial differential equations. As an illustrative example, we take the one dimensional heat equation,

$$\left.\begin{aligned} &u_t = Du_{xx},\ 0 \le x \le 1,\ 0 < t \le t_{final} \\ &L(u(0,t),\ u_x(10,t)) = 0,\ R(u(1,t),\ u_x\ (1,t)) = 0 \\ &u(0,x) = \phi(x) \end{aligned}\right\} \quad (1.9)$$

Here subscripts denote differentiation, u is the dependent variable, t is time, D is the diffusion coefficient, and x denotes spatial position. Moreover, $L(0,t,u(0,t),u_x(0,t)) = 0$ is the left boundary condition, while $R(1,t,u(1,t),u_x(1,t)) = 0$ is the right boundary condition.

We can reduce (1.9) to a system of ODEs by a number of spatial discretization techniques, such as Galerkin's procedure in conjunction with B-splines, collocation in conjunction with B-splines, or other finite element techniques. Here, we simply replace the spatial derivative with a three point, second order difference scheme and use the N+2 uniformly spaced grid points,

$$x_i = i/(N+1),\ i = 0,1,\ldots,N+1$$

Also, for simplicity, let us take the boundary conditions to be of homogeneous Dirichlet type: $u = 0$ at $x = 0$ and $x = 1$.

The system of ODEs is then

$$\dot{y} = Jy, \quad (1.10)$$

$$y(0) = \phi \quad (1.11)$$

where

$$y = [y^1, y^2, \ldots, y^N]^T$$

and

$$y^1 = u(x_1,t)$$

$$J = [D/(\Delta x)^2] \text{ tridiag } [1,-2,1] \text{ (the NxN tridiagonal matrix)}$$

$$\Delta x = 1/(N+1)$$

$$\phi = [\phi(x_1), \phi(x_2),\ldots,\phi(x_N)]^T$$

The eigenvalues of J are given by [Varga (62)]

$$\lambda_k = \frac{-2D}{(\Delta x)^2}\,(1+\cos\,\frac{k\pi}{N+1}),\ k=1,2,\ldots,N$$

For N large, the largest eigenvalue (in magnitude) can be approximated by

$$\lambda_1 \approx -4N^2D$$

while the eigenvalue with smallest modulus can be estimated by

$$\lambda_N \approx -\pi^2D$$

The negative reciprocals of these eigenvalues correspond to the time constants for the system of ODEs. Again, if t_{final} denotes the length of the interval of integration, then the stiffness ratio for the system of ODEs is, by (1.8),

$$S = 4t_{final}DN^2$$

So, for example, if t_{final} D = 1 and N = 100, this problem would be stiff.

E. Linearly Implicit ODEs, Differential-Algebraic Systems and the Method of Lines

If we use a Galerkin or collocation procedure for (1.8) in conjunction with B-splines, then the system of ODEs has the general form [Strang & Fix (73)] [Leaf & Minkoff (84a), (84b)]

$$A\dot{y} = Jy \tag{1.12}$$

$$y(0) = \phi \tag{1.13}$$

Now, the vector y is comprised of the coefficients of the expansion of the approximate solution

$$u(x,t) = \sum_{i=1}^{N} y^i(t)\, B_i(x)$$

in terms of the B-spline basis functions B_i. The initial values for the coefficients are determined by a projection of the initial profile into the approximate solution space.

In the case that L and R in (1.9) prescribe essential boundary conditions, e.g. u(0,t) and u(1,t) are prescribed, A may be a singular matrix, with zero rows corresponding to equations that prescribe u(0,t) and u(1,t). Consequently (1.12) may be a differential-algebraic system, which is comprised of both implicit ordinary differential equations and algebraic equations. If L and R in (1.9) describe natural boundary conditions,

e.g., $\frac{\partial u}{\partial x}$ (0,t) and $\frac{\partial u}{\partial x}$ (1,t) are prescribed, then (1.12) is a system of

implicit ordinary differential equations and A is nonsingular. This is because the variational representation (the weak form representation) imposes no constraints on the approximate solution.

This example indicates the flexibility required in the underlying software in partial differential packages such as POST [Schryer (77)], DISPL2 [Leaf & Minkoff (84a) and (84b)], and PDECOL [Madsen & Sincovec (79)]. These and several other packages automatically implement spatial discretizations to the user's specification. This technique is called the numerical method of lines [Madsen & Sincovec (74)]. We also point out

that uniform grids are not required for hand discretizations or by the good method of lines codes. Indeed, some experimental codes dynamically adjust and/or insert grid points to model fronts and other phenomena. This dynamic adjustment may revise the form of (1.12).

Differential-algebraic systems arise in many other ways also. A few are described in [Sincovec et al. (81)].

F. Problem Structure

One key to effectively using stiff ODE solvers is the use of the appropriate package to take advantage of the structure of a problem --the coupling of the dependent variables. To give some idea of the significance of problem structure we have seen run times reduced by factors of 20 to 200. How? The user appropriately ordered the dependent variables and chose the software package to handle the resulting structure. We now address several structures in turn.

One type of problem we have talked about is the *dense system of equations in normal form.* That is, each ODE is coupled to most of the dependent variables and the system is in the form (1.1)-(1.2). An example of such a system is (1.3)-(1.4). Larger dense systems are quite common.

We have also seen a simple PDE (1.9) which led to a differential equation in normal form but with a tridiagonal Jacobian. In the case of a single nonlinear parabolic PDE, the Jacobian would generally be banded. (The bandwidth would depend on the discretization.) That is, $J_{ij} = 0$ for $i-j > M_L$ and $j-i > M_U$. Here, M_L and M_U represent the lower and upper half bandwidths of J, respectively. In the case of (1.10) $M_L=M_U=1$. The idea is fairly straightforward. We only need to store the elements within the bands formed by the M_L subdiagonals, the main diagonal, and the M_U

superdiagonals of J. We can also apply this idea to systems of PDEs in one spatial dimension. Such systems can also be treated with codes using block structured Jacobians.

Banded Jacobians also arise in systems of ODEs describing networks in which only a small, fixed number of near neighbors are coupled in a prescribed way. A good example of this is a series of stirred tank chemical reactors.

Linearly implicit ODEs in the form (1.12)-(1.13), generalized to nonlinear dependence on y, take the form

$$A\,\dot{y} = g\,(t,y),\ t_0 \le t \le t_{final} \tag{1.14}$$

$$y\,(t_0) = y_0. \tag{1.15}$$

In this form, A is an N x N coefficient matrix and g is an N-vector valued function. Here A = A (t,y) is allowed. Systems of this type do arise from finite element methods. However, they also arise from the finite difference treatment of linearly implicit PDEs, which occur in oil reservoir models [Gottfried (65)] [Douglas (70)] [Peaceman (77)]. In the case of PDEs in one spatial dimension, A and g_y are usually banded.

Block structured Jacobians arise in ODEs in normal form (1.1)-(1.2) in the solution of systems of PDEs in two or more spatial dimensions. By block structured, we mean that J can be partitioned into submatrices or blocks of size n x n. Most of the blocks are non-zero. However, the non-zero elements occupy a few blocks, usually in a discernable structure, but not necessarily tightly packed together. In two dimensional PDEs, there usually is a block-banded structure (blocks of nonzero elements are usually banded about the main diagonal), a few outlying block diagonals and, perhaps, a few block columns or rows have nonzero elements. Such systems can also arise in networks of various types.

We note that block structured linearly implicit ODEs (1.14)-(1.15) also arise in solving one dimensional PDEs by finite element methods.

Normal form ODEs with sparse Jacobians occur in PDEs on irregular geometries, certain loosely coupled networks, and some chemical kinetics models involving a large number of species or compositions. In general, a sparse matrix is a matrix with a very small percentage of nonzero elements. A general sparse description of the Jacobian is appropriate when the nonzero elements form no readily discernable pattern of the types discussed previously. Alternatively, it may be appropriate when the pattern is such that there is no neat way to take advantage of the structure, e.g. no handy software package.

G. Active Time Scales

Modern software packages for ODEs are capable of handling a broad spectrum of problems and problem features. So, let us see what some of these features are.

Contrary to the perceptions of some writers, stiff problems do not always have a single transient region. There may be several regions in which transient phenomena occur. That is, there may be recurring transient regions or regions where the problem is non-stiff followed by regions where the problem is stiff. We were introduced to problems of this type through the solution of chemical kinetics models of certain minor species in the upper atmosphere [Gelinas (74)] [Dickinson & Gelinas (76)] [Byrne & Hindmarsh (75)] [Byrne, et al. (77)]. The model is called the Chapman mechanism. Its features include very rapid changes in the concentrations of minor chemical species. These changes correspond to the rising and the setting of the sun, since the reactions are simply related to photodissociation. To our knowledge, Gelinas and Dickinson were the

first to solve this problem without restarts or averaging in 1973, using a prototype of EPISODE [Gelinas (74)].

Another model which is in some sense of a similar nature to the Chapman model is the Field-Noyes model of a chemical oscillator [Field & Noyes (74)]. A chemical oscillator is a chemical reaction which takes place in such a way that the concentrations of the chemical species in the system vary periodically in time. In the case of the Field-Noyes oscillator, the concentrations of the three species vary in such a way that the three time scales of the reactions are very evident and disparate over each period of the reaction.

Other forms of periodic transients do occur in ODEs. Consequently, solvers must accommodate these phenomena. They must also handle traveling waves. Let us see how traveling waves arise and why they must be handled.

In various applications of engineering and science we find time dependent PDEs involving convection, diffusion, and reactions. In this work, we shall assume that the cell Peclet numbers are small, say 10 or less. However, we have used MOL packages and techniques to solve non-diffusive systems. For many of these PDEs, the solutions have fronts in the variables which correspond to temperature, concentrations, saturations, density, or some other physical entity. We are quite often interested in the variation in time and/or space of a wave front. This could correspond to a reaction front, flame front, or leading edge of a phase. Although these spatial discretization and front tracking issues are important, they are not within the scope of this paper. The challenge, in general, is to develop an economical, reliable, universal strategy for tracking fronts. The challenge for ODE software, in particular, is to accommodate this strategy.

In the next section, we will discuss the numerical methods which are designed to handle these phenomena and features.

2. SURVEY OF METHODS

Software for ODE initial value problems has progressed to a point where, in most cases, the user needs to know little or nothing about the *methods* on which the software is based to get reliable answers economically. The user simply follows a set of usage instructions, perhaps experimenting a little with input parameters, until satisfactory results are obtained. However, these instructions and the requirements imposed on the user vary greatly among solvers. The reasons for this relate largely to properties of the underlying methods. Consequently, it is helpful for users to have some familiarity with the methods in the software packages. Moreover, ODE problems with special features often cannot be fit into the available software without modifications to the latter, and this situation certainly requires a knowledge of methods. Finally, we recognize that the last word on ODE methods has not been said, and a familiarity with current methods is valuable in assessing new methods that appear in the literature from time to time. Some of the methods we mention, which are not yet available in production software form, may someday appear in highly effective software.

Other surveys of ODE methods have appeared occasionally. Two recent ones are [Seward et al. (84)] and [Gupta et al. (85)].

For these reasons, we give here a brief survey of the numerical ODE methods that are currently used in the more popular and successful ODE (initial value) solvers, both stiff and non-stiff. We will write the problem in the simple general form

$$\dot{y} = f(t,y),\ y(t_0) = y_0 \tag{2.1}$$

where y is a vector of length N, t_0 and y_0 are given, and f is an arbitrary vector-valued function. Modifications for the linearly implicit form

$$A\dot{y} = g(t,y),\ y(t_0) = y_0 \tag{2.2}$$

will also be described.

A. Linear Multistep Methods

The class of linear multistep methods is large and varied. We begin with it because it contains some of the most useful methods for stiff problems and also for non-stiff problems. This fact is reflected in both the available software and the frequency of its use.

These and the other methods discussed here are *discrete*, in the sense that what is produced is basically a sequence $y_0, y_1, \ldots y_n \ldots$ of values of y which approximate the solution $y(t)$ at discrete t values $t_0, t_1, \ldots, t_n \ldots$ In the linear multistep case, these discrete values of y are defined by a formula of the form

$$y_n = \sum_{i=1}^{K_1} \alpha_i \, y_{n-i} + h \sum_{i=0}^{K_2} \beta_i \, \dot{y}_{n-i}. \tag{2.3}$$

Here $\dot{y}_j$ denotes $f(t_j, y_j)$, h is a constant step size in t, i.e. $h = t_n - t_{n-1}$. The coefficients α_i and β_i and the nonnegative integer constants K_1 and K_2 are fixed for a given method. The number $K = \max (K_1, K_2)$ is the step number, i.e. the number of past values involved, and (2.3) is referred to as a K-step method. Note that the y_j and $\dot{y}_j$ occur linearly in (2.3); hence, the name. These formulas can also be written in a form which accounts for varying step sizes $h_j = t_j - t_{j-1}$. In that case, they are written as

$$y_n = \sum_{i=1}^{K_1} \alpha_{ni} y_{n-i} + h_n \sum_{i=0}^{K_2} \beta_{ni} \, \dot{y}_{n-i}, \tag{2.4}$$

where the α and β coefficients now depend on $h_n, h_{n-1}, \ldots, h_{n-K+1}$.

The simplest examples of linear multistep methods are the Euler (forward Euler) method,

$$y_n = y_{n-1} + h\dot{y}_{n-1}$$

and the backward Euler method,

$$y_n = y_{n-1} + h\dot{y}_n.$$

These are one-step methods (K=1), but are nevertheless included in the linear multistep class, as degenerate cases, for convenience.

The class of methods now most heavily used for stiff problems is that of Backward Differentiation Formulas (BDFs). These are characterized by putting $K_2 = 0$ and $K_1 = q$ in (2.3) or (2.4) above. Thus, the fixed-step BDF of order q is

$$y_n = \sum_{i=1}^{q} \alpha_i y_{n-i} + h\beta_0 \dot{y}_n, \tag{2.5}$$

and the variable step BDF of order q is

$$y_n = \sum_{i=1}^{q} \alpha_{ni} y_{n-i} + h_n \beta_{n0} \dot{y}_n, \tag{2.6}$$

with α_{ni} and β_{n0} depending on $h_n/h_{n-1}, \ldots, h_n/h_{n-q+1}$. The case q=1 is the backward Euler method. The name BDF comes from writing (2.5) or (2.6) in a form that gives $\dot{y}_n$ as a combination of the y_{n-i} (approximately).

Among the most popular nonstiff methods are the Adams methods, which are characterized by having only one term, y_{n-1}, in the first sum in (2.3) or (2.4). Thus, the explicit Adams method of order q is given by either

$$y_n = y_{n-1} + h \sum_{i=1}^{q} \beta_i \dot{y}_{n-i} \tag{2.7}$$

or the variable-step analog of this formula, and the implicit Adams method of order q is given by either

$$y_n = y_{n-1} + h \sum_{i=0}^{q-1} \beta_i \dot{y}_{n-i} \tag{2.8}$$

or its variable-step analog. The familiar Trapezoid Rule,

$$y_n = y_{n-1} + (h/2)(\dot{y}_n + \dot{y}_{n-1}), \tag{2.9}$$

is the case q = 2 in (2.8). (Some refer to (2.9) as the Crank-Nicholson formula in the context of partial differential equations.)

The term *order* is well-defined for linear multistep methods. For (2.3), it is the largest integer q for which

$$y(t_n) - \sum_{i=1}^{K_1} \alpha_i y(t_{n-i}) - h \sum_{i=0}^{K_2} \beta_i \dot{y}(t_{n-i}) = O(h^{q+1})$$

as $h \to 0$, when y(t) is an arbitrary smooth function. It can be equivalently defined as the largest integer q for which the *local error* $y_n - y(t_n) = O(h^{q+1})$ when (2.3) is used to take one step with all past values exact ($y_{n-i} = y(t_{n-i})$ for $i \geq 1$). In general, a method of order q yields *global errors* $y_n - y(t_n) = O(h^q)$ when integrating from t_0 to a

fixed $t_n = t$ with $h \to 0$ ($n = (t-t_0)/h \to \infty$). The analogous definitions and results for (2.4) are straightforward, although the theory is complicated by the variability of h_j.

In the practical implementation of linear multistep methods, the most significant distinction among them is between *implicit* and *explicit* methods. A method given by (2.3) or (2.4) is explicit if $\beta_0 = 0$ ($\dot{y}_n$ is absent), and is implicit otherwise. For an implicit method, an algebraic system of the form

$$y_n = h\beta_0 f(t_n,y_n) + \sum_{i\geq 1} (\alpha_i y_{n-i} + h\beta_i \dot{y}_{n-i}) = h\beta_0 f(t_n,y_n) + a_n \qquad (2.10)$$

must be solved for y_n at each step. The choice of methods for doing this has a profound impact on the efficiency of the resulting algorithm or solver. As f is in general nonlinear, an iterative procedure of some type is usually used. The simplest such procedure is functional (or fixed-point) iteration,

$$y_{n(m+1)} = h\beta_0 f(t_n,y_{n(m)}) + a_n \quad (m = 0,1,\ldots),$$

where $y_{n(0)}$ is an initial guess for y_n and m denotes the interation count. This works reasonably well for nonstiff problems, but for stiff problems it converges only when h is smaller than or comparable to the smallest time constant in the system, and such a restriction on h is unacceptable because of excessive computer run time. (For this reason, an implicit formula combined with functional iteration is usually referred to as an explicit method. Because of the restriction on the step size, it is certainly a nonstiff method.)

For stiff problems, the choices most often made for solving (2.10) are based on Newton's method. For the problem in the form

$$F(y_n) = y_n - h\beta_0 f(t_n,y_n) - a_n = 0,$$

Newton iteration takes the form

$$y_{n(m+1)} = y_{n(m)} - \left[F_y(y_{n(m)})\right]^{-1} F(y_{n(m)})$$

or

$$\left[I - h\beta_0 f_y(t_n, y_{n(m)})\right](y_{n(m+1)} - y_{n(m)}) = - F(y_{n(m)}). \qquad (2.11)$$

This is in general a powerful method, but has some considerable costs associated with it. The first is that of computing and storing the Jacobian matrix f_y, and the second is in the solution of the N by N linear system for the correction $y_{n(m+1)} - y_{n(m)}$. Both costs can be reduced greatly by modifying the iteration and paying close attention to matrix structures. For one thing, f_y need not be recomputed in (2.11) at every iteration; little is lost in speed of convergence with the use of a fixed value of the Jacobian in the iterations for one time step. By the same reasoning, f_y can be kept fixed for several steps, provided a test can be made to decide when to recompute it. For a given problem, f_y may well have a sparse structure which can be exploited to reduce the costs of computing and storing it and of solving the system. Once a value of the matrix

$$F_y = P = I - h\beta_0 f_y \qquad (2.12)$$

is computed (or approximated), suitable preprocessing of it (such as LU factorization) can be done, depending on the structure assumed, so that the subsequent solutions of linear systems

$$P\,\Delta y = -F$$

are as inexpensive as possible. By the way, some current work brought to our attention warrants the following caveat: if the number of equations in the system is greater than one, then the explicit inverse $[F_y(y_{n(m)})]^{-1}$ should *not* be computed. Rather, the form (2.11) should be solved by

modern numerical linear algebraic methods. The reason is efficiency.

Regardless of the choice of iteration, an initial guess $y_{n(0)}$ is always needed. This is easily obtained by appealing to any of the explicit linear multistep methods.

Linear multistep methods can also be applied to the implicit ODE problem (2.2) with relatively little additional effort. An approach that is usually highly *uneconomical* is to apply them to the equivalent system

$$\dot{y} = f \equiv A^{-1}g \quad .$$

This replaces evaluations of g with more costly evaluations of f and, what is much worse, it replaces matrices A and g_y, which typically have a sparse structure, with a matrix f_y which is most likely dense (not sparse). Instead, if we multiply any of the basic formulas (2.3) or (2.4) by $A(t_n, y_n)$ and use (2.10), we get an algebraic system of the form

$$G(y_n) \equiv A(t_n y_n)y_n - h\beta_0 g(t_n, y_n) - A(t_n, y_n)a_n = 0 \qquad (2.13)$$

with a_n as before (known). The various ways of treating (2.10) by Newton-like methods in the case of an explicit ODE also apply here, in terms of iterations of the form

$$G_y[y_{n(m+1)} - y_{n(m)}] = - G(y_{n(m)}). \qquad (2.14)$$

Thus, for example, if A is at most weakly dependent on y, a good approximation to G_y is

$$G_y \approx A - h\beta_0 g_y$$

evaluated at some convenient point. Note the similarity to the matrix P in the explicit ODE case (2.12). If A depends strongly on y, G_y must include other terms that involve $\partial A/\partial y$. In either case, it is clear that the cost of the major operations in a step is only marginally greater for this approach to $A\dot{y} = g$ than they would be for the corresponding stiff method applied to a similar problem of the form $\dot{y} = f$.

Formally, these techniques also apply to the case where A is singular, i.e. when (2.2) is a differential-algebraic system (DAS), provided that the matrix $G_{\dot{y}}$ is nonsingular and that all of the initial data $y(t_0)$, $\dot{y}(t_0)$ is available. However, this is a very risky approach to DASs in general, even those of the form (2.2), without an understanding of the DAS in question [Petzold (82)], [Gear & Petzold (84)]. The problem may be of a type for which these ODE-based methods work well, or it may be of a type that is numerically ill-posed for these methods, or even mathematically ill-posed (independent of the method). Fortunately, it is often possible to reformulate ill-posed DAS problems so that they are solvable with these methods.

Returning to the Newton or modified Newton iterations (2.11) and (2.14), the manner in which this linear system is treated is extremely important. In fact, the technique for storing and processing the matrix $F_{\dot{y}}$ or $G_{\dot{y}}$ often makes the difference between being able to solve a stiff ODE system in the fast core of a computer and being unable to do so. For example, if the structure of this matrix is banded, then only the elements in this band need to be stored and used. Other matrix structures such as general sparse structure and block banded structure have also been taken advantage of.

For general use, linear multistep formulas are of little value without a means of selecting values of the step size h and method order q for which the method is reasonably accurate and efficient. The algorithms for doing this constitute a major distinction between modern ODE software and its obsolete counterparts. These algorithms are the result of considerable research and development efforts, which are still continuing, and their impact on the accuracy and economy achievable with modern codes is

often quite dramatic. A key ingredient here is that the step and order selection is based on estimates of the actual *errors* committed by the numerical method, rather than *ad hoc* rules often used.

The basic features of these error control algorithms are quite simple. On taking a step to t_n, of size h at order q, the error E(q,h) committed on the step, according to the local error theory, is given asymptotically by $C\ h^{q+1}y^{(q+1)}(t_n)$ for some known constant C, independent of the problem being solved. Now we (typically) impose a condition on h and q that some norm of E(q,h) satisfy

$$||E(q,h)|| \leq \varepsilon \tag{2.15}$$

for a user specified error tolerance parameter ε. The derivative $y^{(q+1)}$ can be easily estimated by finite differences using quantities already generated in the computation. Thus we can compute a step size h_q' at which $||E(q,h)||$ is about equal to ε. This is the step size considered appropriate at order q for the current step and the next few steps. Similarly, E(q',h) can be estimated for other orders q' (typically restricted to q±1), and values $h_{q'}$ obtained on the same basis. Now the code simply chooses the order q' or q which gives the largest step size, and uses that maximum step size. If the step just taken failed the error test (2.15), the step is redone accordingly. At the beginning of the problem, the order q is usually set to 1, for which no past values prior to y_0 are needed, and the order then increases to whatever value is found to be most efficient. The details of the various decisions, approximations, fudge factors, etc., vary considerably, and are highly heuristic in nature. But extensive use has shown that these ideas work well in practice.

The selection process for h and q that we just sketched is incorporated in various codes. An additional constraint, the requirement of monotone decreasing differences (with respect to order), is imposed in

others [Shampine & Gordon (75)]. This illustrates the concept that different underlying algorithms lead to different computational strategies.

B. Runge-Kutta Methods

The class of Runge-Kutta methods is also wide and varied, and has a long history. However, it is less often used in current software, especially for stiff problems, and so will be summarized only briefly here.

Runge-Kutta methods are one-step methods, but involve intermediate stages in a step. They can be either explicit or implicit. The general r-stage explicit Runge-Kutta method for $\dot{y} = f$ can be written as

$$k_1 = hf(t_{n-1}, y_{n-1}) \tag{2.16}$$

$$k_i = hf\left(t_{n-1} + c_i h,\ y_{n-1} + \sum_{j=1}^{i-1} a_{ij} k_j\right) \quad (i = 2,\ldots,r) \tag{2.17}$$

$$y_n = y_{n-1} + \sum_{i=1}^{r} b_i k_i \tag{2.18}$$

where the a_{ij}, b_i, and c_i are constants satisfying $c_i = \sum_{j=1}^{i-1} a_{ij}$.

Through rather tedious calculations, we could determine the order of accuracy q of such a method, and arrive at coefficient values which yield given orders (q never exceeds r). At the same time, it is often possible to imbed a method of order q-1 within the method of order q, and this makes a dynamic error control possible, with little added effort, based on the difference between the two y_n values.

For stiff problems, explicit Runge-Kutta methods are inappropriate, and analogous implicit methods have been developed. The general r-stage implicit Runge-Kutta method can be written

$$k_i = hf(t_{n-1} + c_i h,\ y_{n-1} + \sum_{j=1}^{r} a_{ij} k_j) \quad (i = 1,\ldots,r) \tag{2.19}$$

$$y_n = y_{n-1} + \sum_{i=1}^{r} b_i k_i \ . \tag{2.20}$$

So, we need to solve an algebraic system in rN unknowns. Several special cases have been studied in which this algebraic problem is less formidable. One is the semi-implicit case where the matrix (a_{ij}) is lower triangular and so each k_i involves solving an algebraic system of equations of size N. A Newton-like solution of the equation for k_i involves a matrix of the form $I-ha_{ii}f_y$, and a further reduction in algebraic effort can be achieved if we take all a_{ii} equal - the so-called diagonally implicit case [Alexander (77)]. Other (more complicated) approaches have been used to reduce the cost of solving the fully implicit case to a feasible level. One of these is that of singly-implicit Runge-Kutta method [Burrage (78)] characterized by the

fact that the matrix (a_{ij}) has a single r-fold eigenvalue, thereby permitting a linear transformation to an algebraic system that resembles that of the diagonally implicit case.

Runge-Kutta methods have been generalized in another direction, in the form of Rosenbrock methods and so-called ROW [Norsett & Wolfbrandt (79)], [Kaps & Wanner (81)]. Here terms involving the Jacobian matrix are added to (2.19), so that it has the form

$$k_i = hf_i(t_{n-1} + c_i h,\ y_{n-1} + \sum_{j=1}^{r} a_{ij}\, k_j) + hJ \sum_{j=i}^{i} d_{ij} k_i .$$

Here J is either $\partial f/\partial y$ evaluated at y_n or some approximation to that Jacobian, and the coefficients d_{ij} are chosen to optimize order and stability properties. Setting all the d_{ii} equal reduces the required matrix computation to a minimum.

Another variation on implicit Runge-Kutta methods is the class of mono-implicit RK methods [Cash & Singhal (82)], in which a term involving the unknown y_n is added to the y argument of f in (2.17), within the equations for an explicit RK method.

C. Other Methods

The special challenges posed by stiff ODE problems have led to searches for suitable methods outside of the traditional linear multistep and Runge-Kutta classes. One reason for including these methods is completeness. Another is that for some problems BDF does not work very well. In particular, for highly oscillatory problems, BDF methods with order greater than 2 do not work well. That said, we also warn the user

that the exotic may not be robust. We will only mention the other methods here, as their value has not yet been fully assessed. The list below is intended to be neither exhaustive nor in any particular order.

i. *Collocation methods.* If the solution function $y(t)$ is approximated by a piecewise polynomial function $p(t)$, then collocation methods arise by posing conditions of the form

$$\dot{p}(t) = f(t,p(t))$$

for a set of discrete values of t [Hulme (72)], [Hulme and Daniel (74a-b)]. These methods can also be regarded as implicit Runge-Kutta or block methods.

ii. *Block and composite methods.* If one linear multistep method is used to advance from t_0 to t_1, another from t_1 to t_2, and so on up to t_L, after which the cycle is repeated, the result is a cyclic composite multistep method [Bickart & Picel (73)] [Tischer & Sacks-Davis (83)] [Tendler, et al. (78a-b)]. If instead the values at all of the points $t_1,\ldots,t_L$ are defined by a coupled set of L equations, the method is called a block or block-implicit method [Watts & Shampine (72)] [Andria, et al. (73)]. (See also [Rosser (67)]), who noted the similarity of these methods to Runge-Kutta methods.)

iii. *Extrapolation.* Suppose that a given basic method (usually a one-step method) is used to approximate $y(t)$ whenever $y(t-H)$ is given, using n steps of size $h=H/n$. Then an extrapolation method arises by considering the result, denoted $y(t,h)$, as a function of h which can be approximated well (usually by a polynomial or a rational function) by means of the data obtained from several values of h. This approximation is evaluated at $h=0$ to get the final extrapolated approximation to $y(t)$ [Lindberg (74)] [Bader & Deuflhard (83)]. These can be thought of as multistage, one-step methods, akin to Runge-Kutta. For a recent review of

extrapolation methods, see [Deuflhard (85)].

iv. *Multiderivative methods.* Linear multistep methods that involve derivatives of order two or more are of interest, even though the ODEs are first order [Enright (74)]. Their implementation requires a means of accurately computing the higher order derivatives as well as solving the implicit relation defining the step.

v. *Blended methods and matrix-coefficient methods.* Motivated by the attractive features of certain second-derivative methods, a new class of methods - blended multistep methods - was developed in which the formula is a linear combination of two first-derivative formulas (e.g. Adams and BDF), involving the Jacobian matrix in the coefficients [Skeel & Kong (77)]. Blended formulas of an extended type are given by Cash (83). More generally, first-derivative multistep formulas with matrix-valued coefficients have been studied [Lambert & Sigurdsson (72)].

vi. *Averaging.* An averaging method is one in which an integration step (of size h) is taken with each of several linear multistep (or other) methods, and the final answer taken to be a linear combination of the individual answers [Liniger (76)].

vii. *Fitting.* If an integration method (such as a linear multistep method) has one or more free parameters in it, and if a corresponding number of the dominant eigenvalues of the problem can be estimated, then the free parameters can be set so that the method integrates exactly the exponential modes corresponding to those eigenvalues. The result is an exponential fitting method [Liniger & Willoughby (70)], [Cash (81)].

viii. *Hybrid methods.* The features of linear multistep methods and of Runge-Kutta methods can be combined in hybrid multistage-multistep methods [Butcher (73), (81), (85)], [Byrne & Lambert (66)]. One can even include multiderivative methods in such a hybrid class [Hairer & Wanner

(73)]. A large number of free parameters then has to be dealt with by way of accuracy and stability criteria. Application-oriented hybrid methods are also common, wherein some carefully selected components or terms of a system are treated implicitly, and the rest explicitly.

ix. *Partitioning.* For problems where the stiff eigenvalues (those with $-\mathrm{Re}(\lambda)$ very large) are well separated from the rest, a number of approaches have been studied to separate out the corresponding modes and treat the problem as composed of nonstiff and stiff subsystems. (This is a decoupling by equation type.) Some involve automatic determination of a suitable linear transformation and partition [Alfeld & Lambert (77)] [Bjorck (83)] [Enright & Kamel (79)] [Watkins & Hansonsmith (83)], but these are limited to the case of relatively few stiff eigenvalues. Others are suitable when the user can specify an appropriate partition [Eitelberg (82)] [Hofer (76)].

x. *One-leg methods.* A class of methods that resembles that of classical linear multistep methods was introduced by Dahlquist (83), who calls them one-leg methods [Dahlquist (83)] [Watanabe & Sheikh (84)]. They are based on formulas of the type

$$-\alpha_0 y_n = \sum_{i=1}^{K_1} \alpha_i y_{n-i} + hf\left(\sum_{i=0}^{K_2} \beta_i t_{n-i}, \sum_{i=0}^{K_2} \beta_i y_{n-i}\right) \qquad (2.21)$$

with a normalization $\sum \beta_i = 1$.

If f does not depend on t and is linear in y, this is the same as a linear multistep method, but for general f it may have some advantages over linear multistep methods.

D. Pros and Cons

Without going into too much detail about the various methods and their implementations, it is nevertheless possible to state some advantages and disadvantages of the various method classes of stiff problems. Naturally, these lists depend on the problem environment. As a general rule, for problems which are small in size and inexpensive in function evaluations (of f or g, etc.), there is little difference among methods in performance, and the main criterion for a choice should be the convenience of accessing a solver and setting up the problem for the use of the solver. At the other extreme, problems with large sizes and expensive functions can display vast performance differences among the various methods and solvers. The comments below are aimed at the environment in which the size and/or the expense is considerable.

Implicit linear multistep methods possessing suitable stiff stability properties, including the BDF methods, have some very attractive features for stiff systems:

(1) The method order is easily varied in a dynamic manner.

(2) The estimation and control of local error can be done in a very inexpensive way.

(3) The algebraic system to be solved at each step is only of size N (not a multiple of N), and this system, namely (2.10) or (2.13), is highly amenable to a wide class of iterative nonlinear system methods, notably Newton's method and its variants. [Byrne & Hindmarsh (85)]

(4) In a Newton or Newton-like solution of the algebraic system, the Newton matrix F_y or G_y involved is related very closely to the functions defining the problem, and only one such N x N matrix needs to be stored at any one time.

(5) The Newton matrix, and thus the problem-related matrices (such as f_y) of which it is composed, do not need to be very accurate. Considerable errors can often be tolerated, with only a modest compensating price in convergence speed. One consequence of this is that terms in the Jacobian which are costly to compute but numerically small can be discarded. Another is that the Newton matrix need not be evaluated at every step. (See [Byrne & Hindmarsh (85)] where these ideas were tested.)

(6) The direct relationship of the Newton matrix to the functions defining the problem allows for the exploitation of sparse structure in the latter, with tremendous advantages in both storage and cost for large and/or expensive problems.

(7) An accurate initial guess for use in a modified Newton iteration is available by way of an explicit formula, with the result that the number of Newton iterations per step is typically less than 2.

Very few other methods have all of these advantages.

The chief disadvantages of stiff multistep methods stem from their inherent multistep nature. High order accuracy requires high step number K, hence high storage requirements, and numerous steps of smaller size have to be taken to build up to the required order. (Alternatively, another method could be used for starting [Gear (80)].) A multistep method can also lose much of its efficiency advantage if the problem contains frequent discontinuities; then a one-step method, which has no memory of past solution values, has a distinct advantage. Finally, the high-order stiff multistep methods (especially the BDFs) have relatively poor stability properties when the problem has highly oscillatory modes.

Among the Runge-Kutta methods, only those of implicit type should be considered for stiff problems. For the sake of efficiency in solving the algebraic system of size rN (for an r-stage method), only certain implicit RK methods are of interest. For $\dot{y} = f$ with a diagonally implicit method, a Newton-like iteration involves only a single N x N matrix of the form $I - h\beta f_y$, but r different f values and right-hand side vectors. The same result holds for the singly-implicit methods. This is much more economical than other choices of implicit Runge-Kutta methods, but considerably more costly (per step) than the typical BDF algorithm, where on average less than two f evaluations and right-hand side vectors are involved per step.

In favor of the implicit Runge-Kutta methods is their one-step nature, an advantage in starting up and when crossing points of discontinuity. More importantly, they are able to achieve high orders and good stiff stability properties simultaneously.

Most Runge-Kutta algorithms are of fixed order, but not all [Burrage et al.(80)]. We simply note that comparison tests, e.g. [Enright et al. (75)], have shown that variability of order can be very important in stiff ODE solvers.

Collocation methods and block methods suffer from the same inefficiencies in the algebraic system solution as the implicit Runge-Kutta methods. In some cases, storage of several N x N matrices is required. For composite multistep methods, a very similar difficulty arises because of the different individual methods used. However, it is possible to avoid this difficulty, at least for 2-stage composite methods, by choosing the coefficients β_0 to be the same in each stage [Tischer & Sacks-Davis (83)]. With the inclusion of a variable order, algorithms based on these methods appear to have some features superior to BDF algorithms.

Extrapolation methods offer a natural way of achieving arbitrary accuracy orders, but again at a high cost in the algebraic system solution. In solving $\dot{y} = f$, the Newton matrices for the individual steps all have the same form, $I - h\beta_0 J$, but the values of $h\beta_0$ vary widely among the step sequences used within each major step. Thus the costs in storage and/or matrix operations is necessarily considerably higher (per step) than for ordinary linear multistep methods. However, the larger step sizes often outweigh this cost. Exactly the same comments apply to averaging methods.

Multiderivative methods, especially second derivative methods, can be formed with very attractive order and stiff stability properties. The price one must pay is in dealing with the second derivative, which for $\dot{y} = f$ is given by

$$y = df/dt = f_y\dot{y} + f_t = f_y f + f_t,$$

and possibly with higher derivatives, if any are involved. The effect of this on the Newton iteration is that the Newton matrix is, in general, a complicated combination of the various partial derivatives of f. If the system is autonomous ($f_t = 0$), or is made autonomous, then an approximate Newton matrix for a second-derivative method can be formed as a quadratic polynomial in $J = f_y$, and a considerable reduction in the effort to do the Newton iterations is possible. However, the effort and storage are still greater than for, say, BDF methods.

Blended multistep methods composed from Adams and BDF formulas appear to have all the desirable accuracy and stability properties of second-derivative methods, but almost none of the obstacles. For $\dot{y} = f$, the formulas and the Newton iteration to solve them tolerate errors in the

Jacobian matrix, and allow for the use of sparse structure. Each Newton-like iteration requires only one f evaluation, but two linear system solutions (with the same matrix $I - \gamma J$).

Fitting methods appear to be useful only when the number of stiff eigenvalues is quite small, say less than 10, and only when fairly good estimates of those eigenvalues are available. The process of fitting the free parameters to the given exponential modes is rather complicated and must be repeated frequently throughout the integration, generally.

Hybrid methods include so many different possibilities that it is presently impossible to make general comments about them. But a few isolated studies of hybrid methods have shown some promise.

Partitioning methods appear to be suitable for general use only if they include a reliable automatic way of constructing the linear transformation and the partition of the transformed dependent variable vector. So far, methods for doing this require that the number of stiff modes be a fairly small fraction of the problem size N.

One-leg methods are closely related to linear multistep methods and share most of the properties which make the latter attractive for stiff systems. Moreover, the variable step forms of one-leg methods seem to be more stable. When algorithmic details are fully worked out, certain one-leg methods are likely to be fully competitive with present linear multistep methods.

3. SOFTWARE

We now turn to software for solving stiff ordinary differential equations. In so doing, it is appropriate to give some caveats and a brief history of stiff solvers. Then each of several groups of solvers will be described.

The following does not constitute an endorsement of the listed software. Nor does it necessarily imply that unnamed solvers are not worth trying. However, we can say that if you are using a twelve line solver for differential equations on anything bigger than a hand calculator, you should consider using one of the cited packages instead. Recently, we have noticed that there is commercially available "software" for differential equations with no error control, a user-specified fixed step size, no warning messages, and so on. We strongly advise against using such programs, even on a personal computer. The reasons are straightforward. For all but trivial problems, such programs cannot be sufficiently reliable for accurate computational results. In short, consider one of the solvers mentioned here.

A. A Brief Historical Background

We now turn to a short history of backward differentiation formula based ODE solvers. This is an attempt to answer some questions which are frequently asked of us. We also intend to give the reader some historical perspective of ODE software.

To our knowledge, the first notion of stiffness and the first formal methodology for solving stiff ODEs was reported by Curtiss and Hirschfelder (52). They used the term *stiff* for ODEs because the corresponding servomechanism felt stiff. C. W. Gear became interested in stiff

problems while visiting Argonne National Laboratory. He developed a software package that used backward differentiation formulas (BDFs), which had not enjoyed much favor among numerical analysts, e.g. [Henrici (62)]. Gear's pioneering code was called DIFSUB [Gear (68), (71a)].

Subsequently, Gear revised DIFSUB while visiting Stanford University in 1969. The new code was called STIFF. R. J. Gelinas, at Lawrence Livermore National Laboratory, had been having trouble with some chemical kinetics problems. He acquired STIFF and found that it could solve his kinetics models. By this time, he had enlisted his co-worker, Hindmarsh, as a collaborator. With consultation from Gear, they rewrote STIFF and called the new package GEAR. By 1974, the code GEAR had gone through two revisions by Hindmarsh [Hindmarsh (74)].

All of these packages use BDF for the stiff solver option, and explicit Adams predictors and implicit Adams correctors for the nonstiff option. They also use a fixed time step size h for several steps. Then, they test to see if h should be changed dynamically to effect efficient and accurate solutions. They can also dynamically change the order of the formula of integration for efficiency and accuracy. These codes differ from one another in several ways -- tuning parameters, code structure, linear algebra routines, user interfaces, and overall robustness.

Several variants of GEAR had been developed by 1976. GEARB, GEARS, and GEARBI differed from GEAR in the linear algebra routines [Hindmarsh (77), (76c)], [Sherman & Hindmarsh (80)]. Consequently, problems of various structures could be attacked with economy of both storage and computer time. Other variants of GEAR were designed to take advantage of computer architecture and/or problem structure --GEARBIL, GEARIB, GEARV, GEARST [Hindmarsh (76 a-c), (79)] [Morris et al. (77)].

In 1973 Byrne was a summer visitor at Lawrence Livermore National Laboratory. There Hindmarsh, R. P. Dickinson, Jr., R. J. Gelinas and others were concerned with diurnal chemical kinetics problems. In these, the chemical reactions among minor species were turned on by the rising of the sun and turned off at sunset. They felt that averaging, pseudo-steady-state methods, and periodic restarts were not the answer. The GEAR package used fixed step size for several time steps. Then, it adjusted the step size by interpolating previously computed values. For the diurnal kinetics model, the implementation of this fixed-step-interpolate strategy was not stable. One consequence was the initiation of a project to develop an integration package for stiff ODE systems with the capability of adjusting its time step after *each* integration step. Because this ability to change the step size at each time step is built into the formula, we call this a variable-step method.

The variable step BDF methods were incorporated in EPISODE and its variants. These developments were intermingled with the final revisions of GEAR and its variants [Hindmarsh (79)].

In 1976 we collaborated in a report on calling sequences for stiff ODE solvers [Hindmarsh & Byrne (76b)]. This report was based on several discussions and workshops held during 1975 and 1976. This evolved into a project to develop the package called ELSODE. Subsequently, as a result of a wider effort to standardize the user interface for ODE solvers [Hindmarsh (78)], this evolved into LSODE [Hindmarsh (80)]. In many ways, LSODE is similar to GEAR, Rev. 3. However, LSODE has a user interface that is much more flexible than GEAR, Rev. 3. LSODE also uses the LINPACK linear algebra packages, dynamic storage allocation, more extensive modularization and a wide range of types of error controls.

Of course variants of LSODE have been developed to handle problems of

various structures, as we shall see.

So far, we have given a rather quick sketch of ODE software along just one path. Even in the BDF tree, there are other computer codes and developments. Valuable contributions were also made by R. W. Klopfenstein and F. T. Krogh [Klopfenstein (71)], [Klopfenstein & Davis (71)], [Pellos & Klopfenstein (72)], [Krogh (68)]. For variable step BDF, Brayton et al. (72) and Hachtel, et al. (71) predated EPISODE with their papers on a method for solving differential-algebraic systems. Gear (71b) had looked at solving differential-algebraic systems with DIFSUB. Rübner-Peterson (73) had also developed a BDF scheme for solving differential-algebraic systems. Curtis (78) developed FACSIMILE, which solves certain kinds of differential-algebraic systems with a BDF method. Carver installed a sparse solver (MA28) in GEAR, Rev. 2 within FORSIM [Carver (79)]. Krogh and Stewart (84) developed a new implementation of BDF methods based on stability with respect to Newton matrix errors. There are other such developments, too numerous to mention here.

Now, let us see what software is currently readily available and the type or types of problems each package can solve.

B. LSODE and Its Variants

LSODE [Hindmarsh (80)] is the basic member of the LSODE family called ODEPACK [Hindmarsh (83)]. LSODE is designed to solve stiff and non-stiff problems in the canonical form (1.1)-(1.2). For these problems, the Jacobian matrix may be either dense (very few zero elements) or banded.

LSODI [Hindmarsh (80), (81), (82)] is intended to solve linearly implicit ODEs of the form (1.14)-(1.15). LSODI allows A and g_y to both be either dense or banded.

If we want to solve problems of the form (1.1)-(1.2) with a sparse

Jacobian matrix, then we could use LSODES. This package uses components of the Yale Sparse Package [Eisenstat et al. (77), (82)].

LSODA has a novel feature. It automatically switches between stiff (BDF) and non-stiff (Adams) methods according to an algorithm developed by Petzold [Petzold (83b)]. The basic purpose is to relieve the user of the responsibility of determining whether, and also where, a problem is stiff or non-stiff. For example, LSODA would select the non-stiff method in transient regions and the stiff methods elsewhere. As for general problem structure, the problem class addressed is essentially the same as that for LSODE (full and banded Jacobians).

LSODAR is based on LSODA, but includes a rootfinder. It gives the user the capability of computing the zeros of a set of functions $\{Z_i(t,y): 1 \le i \le m\}$. This is called the *g-stop* capability by F. T. Krogh, whom we believe to have coined the term. For example, we could set $Z_i = y_i - C_i$ in the simulation of a continuously stirred tank reactor (chemical) or $C*$, where the idea is to stop the reactions and recover the product when one or more components y_i have reached a certain mole fraction C_i. Other examples might include changing the system of differential equations when a particle reaches a container wall in a tracking problem. Krogh has used the example of extending an antenna and revising the center of gravity of a space vehicle when it reaches a prescribed position.

LSOIBT is designed for problems of the form (1.14)-(1.15). However, LSOIBT assumes that A and g_y are both block-tridiagonal in structure. By this, we simply mean that these matrices can be partitioned into n x n blocks. These blocks in turn form three stripes --the main block diagonal, and the principal upper and lower block diagonals. As we noted previously, this problem structure arises in the finite element solution of one dimensional PDEs and elsewhere.

This LSODE family, also known as ODEPACK, is available from the National Energy Software Center, whose address is given in the Appendix.

C. EPISODE and Its Variants

For some problems, changes occur frequently or dramatically. Consequently, the ability to change the step size at each integration step can be advantageous. That is precisely why we developed the EPISODE family. EPISODE [Byrne & Hindmarsh (75)], [Hindmarsh & Byrne (77)] is intended for problems of the form (1.1)-(1.2) with dense Jacobian matrices. EPISODE does have a non-stiff option. We generally suggest that EPISODE be tried after LSODE has failed on a stiff problem with occasional fronts, because the overhead for EPISODE is frequently higher than that for LSODE. Moreover, the user interface is not as flexible as that for LSODE. Finally, EPISODE does not use the modern linear algebra routines that LSODE does.

EPISODEB [Byrne & Hindmarsh (76)] treats problems with banded Jacobians of the same type as EPISODE. We would use EPISODEB after the banded option of LSODE failed.

EPISODEIB [Byrne (79)] is designed to solve banded problems from the class that LSODI solves. As another member of the EPISODE family, it is intended for use on problems with fronts.

EPISODE, EPISODEB and EPISODEIB are all available from the National Energy Software Center.

D. Other Descendants of GEAR and EPISODE

One of the GEAR family has not yet been superseded by a corresponding member of the LSODE family. That package is GEARBI [Hindmarsh (76c)], which is based on GEAR, Rev. 3. GEARBI is designed to solve problems with a general block structure by block successive overrelaxation (block SOR).

DGEAR is the stiff ODE solver in the IMSL library [IMSL (82)]. This package is based on GEAR, Rev. 3, which is a precursor of LSODE. DGEAR handles both stiff and non-stiff problems. For stiff problems, the structure of J may be banded or dense.

The NAG (Numerical Algorithm Group) library [NAG (82)] lists five stiff ODE solvers. To some extent, their purposes correspond to those of the members of ODEPACK (the LSODE family). The codes and purposes are as follows:

- D02EAF - Integration over an interval
- D02EBF - Integration over an interval, with intermediate output
- D02EGF - Integration until a component of the solution reaches a prescribed value
- D02EHF - Integration until a function of the solution is equal to zero
- D02QBF - A comprehensive integration package (called by all of the above codes)

These routines are based on GEAR, Rev. 3 [Gladwell (79)].

DEBDF is a driver, which calls a modified version of LSODE. The complete DEBDF package is a member of the SLATEC library, which may be obtained from the National Energy Software Center. The DEBDF package is also a member of DEPAC [Shampine & Watts (80)].

A recent descendant of EPISODE is a code called TORANAGA [Axelrod et al. (83)]. It uses variable-step variable-order BDF methods, but differs from most other stiff solvers because it is designed for an environment of large scale problems. It uses a memory management package, requires the user to supply whatever linear system solver is appropriate, has an elaborate dump/restart feature, and numerous optional inputs and outputs.

The software package SPRINT [Berzins et al. (84)] is designed for both ODE and PDE systems. SPRINT is derived from LSODI and LSODES.

E. Differential-Algebraic System Solvers

Software for differential-algebraic systems is now readily available. Earlier, we saw that systems such as (1.12)-(1.13) can arise from solving parabolic PDEs by the numerical method of lines via a Galerkin procedure. They can also arise in solving mixed parabolic-elliptic systems of PDEs or directly in certain models of reactive flows or electronic networks. In the past a version of GEAR, called GEARIB [Hindmarsh (76a)], was used to solve differential and differential-algebraic systems of the linearly implicit form in numerical method of lines codes for PDEs. LSODI has also been used for this purpose. Earlier work [Gear (71b)] [Hachtel, et al. (71)] [Brayton, et al. (72)] [Rübner-Peterson (73)] [Sincovec, et al. (79)] [Klopfenstein (74)] focused on similar systems of differential-algebraic equations that arise directly in the simulation of complex electrical circuits.

More recent work on differential-algebraic systems has led to the development of DASSL [Petzold (83a)], a differential-algebraic system solver. DASSL is intended for the solution of problems of the general form

$$g(t,y,\dot{y}) = 0 \tag{3.1}$$

$$y(t_0) = y_0 \tag{3.2}$$

$$\dot{y}(t_0) = p_0 \tag{3.3}$$

Here, the data (3.3) may be either prescribed or computed from (3.1)-(3.2). In any case, (3.1)-(3.3) must be consistent, i.e. $g(t_0, y_0, p_0) = 0$. It is certainly important to note that differential-algebraic systems are not as straightforward as we might suppose. [Petzold (82)], [Gear & Petzold (84)].

F. Runge-Kutta Codes

Runge-Kutta codes for stiff ODEs are not generally available in the more common software libraries in the United States. Moreover, some of the Runge-Kutta codes reported on elsewhere are listed as experimental [Gaffney (84)].

Three codes based on implicit Runge-Kutta methods of the traditional type have become well known. DIRK [Alexander (77)] uses diagonally-implicit RK methods with fixed (selectable) orders up to four. STRIDE [Burrage et al. (80)] uses singly-implicit methods in a variable-order manner, up to order 15. An early code, COLODE [Hulme & Daniel (74a-b)] uses fully implicit RK methods.

A number of solvers based on Rosenbrock-type methods are mentioned in the literature. Kaps and Rentrop (79) mention GRK4A and GRK4T. Gottwald and Wanner (81) mention ROW4A. Shampine (82a) mentions DEGRK. All of these have imbedded methods of orders 3 and 4. Comparison tests on these and other stiff solvers are given in [Kaps et al. (85)].

G. Blended and Composite Multistep Codes

Blended linear multistep methods are best represented by the code BLEND [Skeel & Kong (77)]. It uses a variable-order blend of Adams and BDF methods, of order up to 7.

An early cyclic composite multistep code is STINT [Tendler et al. (78a-b)]. It is also variable-order with orders up to 7. More recent work on cyclic composite methods has focused on practical implementation issues, and has resulted in a code called ODIOUS [Tischer & Sacks-Davis (83), Tischer & Gupta (84)]. However, the authors of ODIOUS appear to regard it as experimental, and are still testing various coefficient

choices in it.

H. Extrapolation Codes

An early extrapolation code is IMPEX2 [Lindberg (74)], which uses an extrapolated implicit midpoint rule. A more recent code is METAN1 [Bader & Deuflhard (83)], which uses a semi-implicit midpoint rule. A variant of the latter was also developed for the case of a sparse Jacobian, and called METAS1 [Deuflhard (81)]. The code LIMEX [Deuflhard et al. (85)] is intended to solve differential-algebraic systems in the linearly implicit form (1.14), with **A** singular and constant.

I. Second-Derivative Codes

An early implementation of second-derivative methods was the solver SDBASIC [Enright (74)]. A more recent and more efficient implementation is SDSTEP [Sacks-Davis (80)]. Both are variable-order, with orders up to 9, and require the user to supply the Jacobian matrix exactly.

4. EXAMPLE PROBLEMS AND CODE USAGE

Here we present several problems for computation. Each is easily described. However, we feel these examples, as presented, represent a cross section of real problems we have seen over the span of our careers. In what follows, we generally do not rewrite or rescale the ODEs, because we believe the average user would not. Some problems even have a closed form (analytic solution), e.g. Burgers equation. Despite their simplicity, these examples often present interesting lessons - e.g. significance of features in a solution, importance of repeated regions of stiffness, diurnal kinetics, incompatible boundary and initial conditions, and so on. The basic problems are described in Subsection A. The numerical results and further discussion are given in Subsection B.

A. The Example Problems.

PROBLEM 1 - Robertson's Problem

We have already seen an example of a neutrally (conditionally) stable, dense system of ODEs in normal form. For our first problem, we use Robertson's problem (1.3)-(1.4) on the time interval $0 \leq t \leq 4 \times 10^7$. By choosing this interval, we place severe demands on the error control and step size control of the solvers. The reason is this: if a zero or near-zero eigenvalue of J goes positive in the numerical calculations, the system becomes unstable. Moreover, as t gets larger, we expect the step size h to increase dramatically for efficiency. The need to increase step size for efficiency and the simultaneous need to maintain strict error control run somewhat counter to one another. Yet, this is precisely the type of performance we look for in high quality ODE software.

PROBLEM 2 - The Field Noyes Chemical Oscillator

This is another small, dense system in normal form. However, it is potentially iconoclastic, because there are periodic transients followed by regions of stiffness. This model represents a chemical oscillator, a chemical reaction which takes place in such a way that the concentrations of the three chemical species vary periodically in time. In dimensionless form the system is [Field & Noyes (74)]:

$$\left.\begin{aligned} \dot{y}^1 &= s(y^2 - y^1 y^2 + y^1 - q[y^1]^2) \\ \dot{y}^2 &= (y^3 - y^2 - y^1 y^2)/s \\ \dot{y}^3 &= w(y^1 - y^3) \end{aligned}\right\} \qquad (4.1)$$

where

$$s = 77.27,\ w = 0.1610,\ q = 8.375 \times 10^{-6}. \qquad (4.2)$$

The initial data we use are

$$y^1(0) = 4.0, \qquad y^2(0) = 1.1, \qquad y^3(0) = 4.0$$

and are due to Enright and Hull (76).

To connect this model with the chemistry somewhat, we note that y^1 is the scaled concentration of bromous acid [$HBrO_2$], y^2 is the scaled concentration of the bromide ion [Br^-], and y^3 is the scaled concentration of cerium IV [Ce (IV)]. By looking at the reaction rate coefficients in (4.1) and (4.2), we expect to see three disparate time scales in the solution. By virtue of hindsight, we know that if the output points are too far apart, we can expect to loose some features in the solution of this problem.

PROBLEM 3 - Two Species Diurnal Kinetics

This is also a small, dense problem in normal form. This problem is meaningful to us because it and similar problems led us to begin our collaboration and the development of the EPISODE family. This model represents the Chapman mechanism for the generation of ozone and the oxygen singlet. It can be a severe test for a stiff ODE package. The symbolic representation for the four reactions in this model are [Dickinson & Gelinas (76)]:

$$\left.\begin{array}{ll} O + O_2 \xrightarrow{M} O_3, & k_1 \\ O + O_3 \rightarrow 2O_2, & k_2 \\ O_2 \xrightarrow{h\nu} 2O, & k_3 \\ O_3 \xrightarrow{h\nu} O + O_2, & k_4 \end{array}\right\} \qquad (4.4)$$

where k_i denotes the reaction rate for (i=1,2,3,4), M denotes some molecule required to carry off excess energy, hν indicates a photochemical reaction, and O, O_2, and O_3 represent the oxygen singlet, oxygen, and ozone, respectively. In the example, the concentration of O_2, denoted by $[O_2]$, will be held constant, the rates k_1 and k_2 are fixed and k_3 and k_4 vary diurnally. If $y^1 = [O]$, $y^2 = [O_3]$ and $y^3 = [O_2]$, the system of ordinary differential equations is

$$\left.\begin{aligned} \dot{y}^1 &= R^1(y^1,y^2,t) = -k_1 y^1 y^3 - k_2 y^1 y^2 + 2k_3(t)y^3 + k_4(t)y^2 \\ \dot{y}^2 &= R^2(y^1,y^2,t) = k_1 y^1 y^3 - k_2 y^1 y^2 - k_4(t)y^2 \end{aligned}\right\} \quad (4.5)$$

with

$$y^3 = 3.7 \times 10^{16}$$

$$k_1 = 1.63 \times 10^{-16}$$

$$k_2 = 4.66 \times 10^{-16}$$

$$k_i = \left\{ \begin{array}{l} \exp[-a_i/\sin\omega t],\ \sin\omega t > 0 \\ 0\ ,\ \sin\omega t \leq 0 \end{array} \right\} \ i = 3,4 \qquad (4.6)$$

$$a_3 = 22.62,\ a_4 = 7.601,$$

$$\omega = \pi/43200$$

and

$$y^1(0) = 10^6,\ \ y^2(0) = 10^{12}. \qquad (4.7)$$

The constant 43,200 is twelve hours measured in seconds. Graphs of the solution of this problem appear in Figs. 4.3 and 4.4. The former is given on a shifted logarithmic scale and contrasts the behavior of y^1 and y^2, while the latter shows how y^2 (or $[O_3]$) increases slowly. Note how y^1 (or [O]) oscillates between large daylight values and small nighttime values. Although this problem only involves three chemical species and just two of these have concentrations varying in time, it does have features of larger problems:

- The Jacobian matrix is not a constant.
- The diurnal effect is present.
- The oscillations are fast.
- The time interval used is fairly long, $0 \leq t \leq 8.64 \times 10^5$, or 10 days.

PROBLEM 4. A KIDNEY MODEL

The following example was posed as a two point boundary value problem in [Scott & Watts (76)] and is attributed to Ivo Babuska. In [Bader & Deuflhard (83)], the problem is given as

$$\begin{aligned} \dot{y}^1 &= a\,(y^3 - y^1)\,y^1/y^2 \\ \dot{y}^2 &= -a\,(y^3 - y^1) \\ \dot{y}^3 &= [b - c(y^3 - y^5) - a y^3 (y^3 - y^1)]/y^4 \\ \dot{y}^4 &= a(y^3 - y^1) \\ \dot{y}^5 &= -c(y^5 - y^3)/d \end{aligned} \tag{4.8}$$

with

$$a = 100,\ b = 0.9,\ c = 1000,\ d = 10. \tag{4.9}$$

The initial data here are:

$$\left.\begin{aligned} y^1(0) &= y^2(0) = y^3(0) = 1.0 \\ y^4(0) &= -10 \end{aligned}\right\} \tag{4.10}$$

In the original problem, the remaining condition was $y^3(1) = y^5(1)$. However, for $0 \leq t \leq 1$, we take

$$y^5(0) = 0.990268835 \tag{4.11}$$

$$y^5(0) = 0.99 \tag{4.12}$$

$$y^5(0) = 0.9 \tag{4.13}$$

in turn. For the last two choices of the initial value, the problem is reported to be stiff. For the first, it is reported to be non-stiff [Bader & Deuflhard (83)].

Babuska and B. Kellogg have told us that this model is indeed similar to a three tube model of a kidney. Solute and water are exchanged through the walls of the tubes. Here y^1, y^5, and y^3 represent the concentration of the solute in tubes 1, 2, and 3, respectively. y^2 and y^4 represent the flow rates of tubes 1 and 3. We expect this problem to behave like other first order two point boundary value problems.

PROBLEM 5 - A LASER OSCILLATOR MODEL

This pair of coupled equations represents a model of a ruby laser oscillator. If we let ϕ denote photon density and n denote dimensionless population inversion, then we can write

$$\left.\begin{aligned} \dot{n} &= -n\,(\alpha\phi + \beta) + \gamma \\ \dot{\phi} &= \phi\,(\rho n - \sigma) + \tau\,(1+n) \end{aligned}\right\} \tag{4.14}$$

where the parameters are as follows:

$$\alpha = 1.5 \times 10^{-18} \qquad \beta = 2.5 \times 10^{-6}$$

$$\gamma = 2.1 \times 10^{-6} \qquad \rho = 0.6 \tag{4.15}$$

$$\sigma = 0.18 \qquad \tau = 0.016$$

The initial conditions are:

$$\left.\begin{aligned} n(0) &= -1 \\ \phi(0) &= 0 \end{aligned}\right\} \tag{4.16}$$

This problem is challenging because it is stiff initially, but mildly damped and oscillatory later. It can be shown that as $t \to \infty$, $n \to 0.3\text{-}1.155 \times 10^{-14}$ and $\phi \to 3 \times 10^{12} + 0.1798$, the steady state values. It suffices here to solve for $0 \leq t \leq 0.7 \times 10^{6}$ ns = 0.7 ms. Time t is in nanoseconds.

PROBLEM 6 - BURGERS' EQUATION

We have seen the basic idea of the numerical method of lines in Section 1. It is not particularly difficult to see that the numerical method of lines can impose quite a few requirements on a stiff ODE solver. Among them are the need to track traveling waves.

With this in mind, we now describe a partial differential equation with traveling wave solutions. Burgers' equation [Benton & Platzman (72)] for $u = u(x,t)$ is

$$u_t + uu_x = \nu u_{xx} \qquad 0 \leq x \leq 1,\ t \geq 0 \qquad (4.17)$$

with subscripts denoting partial differentiation. An exact solution can be shown to be

$$u(x,t) = \left[\ 1 + \exp\left(\frac{x}{2\nu} - \frac{t}{4\nu}\right)\ \right]^{-1} . \qquad (4.18)$$

The initial and Dirichlet boundary conditions are taken directly from (4.18). Note that the solution is a travelling wave whose speed is dx/dt = 1/2.

By the way, Burgers equation is a very good example for several reasons:

- It is nonlinear.
- The exact solution of the PDE is known. [Benton & Platzman (72)].
- It can be thought of as a hyperbolic problem with artificial diffusion for small ν. [Chin et al. (79)]
- It is sometimes used in boundary layer calculations for the flow of viscous fluids.
- It is very nearly a standard test problem for PDE solvers.

The simplest method of spatial discretization is to discretize along the x-axis with a uniform mesh and to replace all spatial derivatives in (4.17) by (say) centered finite difference analogues. Thus, if we take

$$\Delta = \frac{1}{N+1}$$

$$u_i(t) = u(i\Delta,t), \quad i= 0,1,\ldots,N+1 \tag{4.19}$$

then a system of ODEs for the method of lines (MOL) approach to solving (4.17) is

$$\dot{u}_i = -(u_i/2\Delta)(u_{i+1}-u_{i-1}) + (\nu/\Delta^2)(u_{i+1}-2u_i+u_{i-1}), i=1,2,\ldots,N \tag{4.20}$$

$$u_i(0) = [1 + \exp(i\Delta/2\nu)]^{-1}, \quad i=1,2,\ldots,N \tag{4.21}$$

$$u_0(t) = [1 + \exp(-t/4\nu)]^{-1} \tag{4.22}$$

$$u_{N+1}(t) = \left[\; 1 + \exp\left(\frac{1}{2\nu} - \frac{t}{4\nu}\right) \; \right]^{-1} \tag{4.23}$$

where (4.21)-(4.23) are taken directly from (4.18) and where $u_i = u_i(t)$. Although the problem (4.20)-(4.23) is of the desired form, its exact solution is not known. The exact solution is known only for the PDE.

Finally, we can note by inspection that the system (4.20) has a tridiagonal Jacobian matrix. The subdiagonal elements of the Jacobian matrix are:

$$\frac{\partial \dot{u}_i}{\partial u_{i-1}} = \frac{u_i}{2\Delta} + \frac{\nu}{\Delta^2} \tag{4.24}$$

while the diagonal elements are

$$\frac{\partial \dot{u}_i}{\partial u_i} = - \left(\frac{u_{i+1} - u_{i-1}}{2\Delta} \right) - \frac{2\nu}{\Delta^2} \tag{4.25}$$

and the superdiagonal elements are

$$\frac{\partial \dot{u}_i}{\partial u_{i+1}} = - \frac{u_i}{2\Delta} + \frac{\nu}{\Delta^2} \tag{4.26}$$

for $i=1,2,\ldots,N$, with appropriate exclusions and substitutions of (4.22) and (4.23).

Another method to reduce Burgers equation (4.17) to a system of ODEs was described by Chin et al (79). Their simplified Galerkin method uses piecewise linear B-splines or chapeau functions as basis elements for both test and trial functions. The inner product is taken by applying Simpson's rule with the quadrature points taken as the break points for the basis functions. The system of ODEs then has the form:

$$A\dot{u} = g(t,u)$$

with

$A = (1/6).\{3 \text{ diag } [1, 4, 1]\}$, an NxN tridiagonal matrix.

If $u_i(t)$ is the numerical solution of (4.17) at x_i, then

$$g^i = -[u^2_{i+1} - u^2_{i-1}]/2\Delta + (\nu/\Delta^2)\,[u_{i+1}-2u_i+u_{i-1}],\ i=1,2,\ldots N.$$

Note that this is almost the same as the right hand side of the finite difference equation (4.20). The boundary and initial data can be taken

from (4.21) -(4.23). The Jacobian matrix J for g follows from the above and is described by

$$\frac{\partial g^i}{\partial u_{i-1}} = u_{i-1}/\Delta + \upsilon/\Delta^2, \quad \frac{\partial g^i}{\partial u_i} = -2\upsilon/\Delta^2, \text{ and}$$

$$\frac{\partial g^i}{\partial u_{i+1}} = -u_{i+1}/\Delta + \upsilon/\Delta^2. \quad \text{The Newton Matrix is}$$

$$A - h\beta_0 J$$

since it is a linearly implicit system of ODEs. Note that this procedure can also be thought of as a collocation procedure. One of the tricks was to interpolate the u^2 term rather than to work with u. [Swartz & Wendroff (69)].

PROBLEM 7 - Two Species Diffusion-Diurnal Kinetics-
One Dimensional

The main idea of this problem is to combine some of the features of Problems 3 and 6. This example is another from the general area of transport [Bird et al (60)] and is rather similar to one addressed by [Chang et al (74)]. This is a diffusion-reaction problem and has no convective term. Such problems are fairly common. A description of c^i, the concentration of the i-th minor chemical species in the upper atmosphere, is represented by

$$\frac{\partial c^i}{\partial t} = \frac{\partial}{\partial z}\left[K(z)\frac{\partial c^i}{\partial z}\right] + R^i(c,t), \quad i=1,2,\ldots,I \qquad (4.27)$$

where z denotes the elevation above the earth in km, and $\frac{\partial}{\partial z}\left[K(z)\frac{\partial c^i}{\partial z}\right]$,

the diffusive term, accounts for vertical transport by turbulence. Horizontal convection is neglected in this simple one dimensional model. The term $R^i(c,t)$ is the reaction term in this system, where $c = [c^1,c^2,\ldots c^I]^T$ is the vector of concentrations. Systems of this type have been discussed by [Chang et al. (74)] and solved in the manner described below.

In this prototypical example, we take $I = 2$, $30 \leq x \leq 50$, $0 \leq t \leq 8.64 \times 10^4$ (1 day measured in seconds), and $K(z) = \exp(z/5\cdot 10^{-8})$ (km^2/s), subject to the initial conditions

$$c^1(z,0) = 10^6\,\gamma(z) \qquad c^2(z,0) = 10^{12}\,\gamma(z)$$
$$\gamma(z) = 1 - \left(\frac{z-40}{10}\right)^2 + \frac{1}{2}\left(\frac{z-40}{10}\right)^4 \tag{4.28}$$

Boundary conditions are taken to be

$$\frac{\partial c^i}{\partial z}(30,t) = \frac{\partial c^i}{\partial z}(50,t) = 0, \quad i = 1,2. \tag{4.29}$$

The reaction terms $R^i(c^1,c^2,t) = R^i(c,t)$ are taken to be identical to those in Problem 3, given by (4.5) - (4.6), with $c^1 = [O]$, $c^2 = [O_3]$, and a constant third species concentration $c^3 = [O_2] = 3.7 \times 10^6$ (denoted y^3 in (4.5))

To generate a system of ordinary differential equations, we will discretize the interval $30 \leq z \leq 50$ and replace all of the spatial derivatives in (4.27) with centered finite differences. Let $M = 50$, say, set $\Delta z = 20/M$, and set $z_j = 30 + j(\Delta z)$ for $0 \leq j \leq M$. Next, let $c_j^i(t)$ be the approximation to $c^i(z_j,t)$ obtained by solving

$$\dot{c}_j^i = (\Delta z)^{-2}\left[K_{j+1/2}c_{j+1}^i - (K_{j+1/2} + K_{j-1/2})c_j^i + K_{j-1/2}c_{j-1}^i\right] + R^i(c,t) \tag{4.30}$$

for $i = 1,2$; $j = 1,2\ldots,M$ and with $K_{j\pm1/2} = K(z_{j\pm1/2})$ $= K(30 + [j\pm1/2]\Delta z)$. The boundary conditions are to be replaced by $c_0^i = c_2^i$ in the ODE's for c_1^i and by $c_{M-1}^i = c_{M+1}^i$ in those for c_M^i. The system of $N = 2M$ ODEs can be specified by setting $y(t) = [c_1^1(t), c_1^2(t), c_2^1(t), c_2^2(t),\ldots,c_M^1(t), c_M^2(t)]^T$. This procedure leads to the following system of ODE's.

At the left hand boundary, we obtain:

$$\dot{y}^1 = (\Delta z)^{-2}\left[K_{3/2}y^3 - \left(K_{3/2} + K_{1/2}\right)y^1 + K_{1/2}y^3\right] + R^1(y^1,y^2,t) \tag{4.31}$$

$$\dot{y}^2 = (\Delta z)^{-2}\left[K_{3/2}y^4 - \left(K_{3/2} + K_{1/2}\right)y^2 + K_{1/2}y^4\right] + R^2(y^1,y^2,t) \tag{4.32}$$

For $2 \le \ell \le M - 1$, (i.e. on the interior of the interval):

$$\dot{y}^{2\ell-1} = (\Delta z)^{-2}\left[K_{\ell+1/2}y^{2\ell+1} - \left(K_{\ell+1/2} + K_{\ell-1/2}\right)y^{2\ell-1} + K_{\ell-1/2}y^{2\ell-3}\right] + R^1(y^{2\ell-1}, y^{2\ell}, t) \tag{4.33}$$

$$\dot{y}^{2\ell} = (\Delta z)^{-2}\left[K_{\ell+1/2}y^{2\ell+2} - \left(K_{\ell+1/2} + K_{\ell-1/2}\right)y^{2\ell} + K_{\ell-1/2}y^{2\ell-2}\right] + R^2(y^{2\ell-1}, y^{2\ell}, t) \tag{4.34}$$

At the right hand boundary, we obtain:

$$\dot{y}^{2M-1} = (\Delta z)^{-2}\left[K_{M+1/2}y^{2M-3} - \left(K_{M+1/2} + K_{M-1/2}\right)y^{2M-1} + K_{M-1/2}y^{2M-3}\right] + R^1(y^{2M-1}, y^{2M}, t) \tag{4.35}$$

$$\dot{y}^{2M} = (\Delta z)^{-2}\left[K_{M+1/2}y^{2M-2} - \left(K_{M+1/2} + K_{M-1/2}\right)y^{2M} + K_{M-1/2}y^{2M-2}\right] + R^2(y^{2M-1}, y^{2M}, t) \tag{4.36}$$

This system is in the desired form $\dot{y} = f(y,t)$ and is subject to the initial conditions taken from (4.28):

$$\left.\begin{aligned} y^{2i-1}(0) &= 10^6\gamma(30 + i\,\Delta z) \\ y^{2i}(0) &= 10^{12}\gamma(30 + i\,\Delta z) \end{aligned}\right\} \quad i = 1,2,\ldots,M \tag{4.37}$$

In summary, we have reduced the system of two parabolic PDEs (4.27), subject to the initial and boundary conditions (4.28) and (4.29), to a system of 2M ODEs (4.31) - (4.36), subject to the initial conditions (4.37). The key step was the discretization or chopping up of the regime in the z direction. In particular the approximation

$$\frac{\partial}{\partial z}\left[K(z_j)\frac{\partial c^i(z_j)}{\partial z}\right] = (\Delta z)^{-2}\left[K(z_{j+1/2})(c^i_{j+1} - c^i_j) - K(z_{j-1/2})(c^i_j - c^i_{j-1})\right]$$

is important.

Finally, we remark that the Jacobian matrix for this system is a 5-diagonal matrix - the main diagonal, the two adjacent super diagonals, and the two adjacent subdiagonals contain all of the nonzero elements. This can be verified by computing J or simply by noting the couplings in the ODE's. The structure of this matrix is very important in the solution of large systems as we noted earlier in Section 1. In particular the centered finite difference discretization of two coupled parabolic PDE's of type (4.27) always leads to a 5-diagonal matrix. Note that ordering by grid point and then by species (the reverse of the order above) destroys this structure.

PROBLEM 8 - Two Species Diffusion - Diurnal Kinetics -
Two Dimensional

This example is based on a pair of PDEs in two dimensions, representing a simple model of ozone production in the stratosphere with diurnal kinetics. (See also [Hindmarsh (83)] for comparison tests on this problem.) There are two dependent variables c^i, representing concentrations of O_1 (the oxygen singlet) and O_3 (ozone) in moles/cm^3, which vary with altitude z and horizontal position x, both in km, with $0 \le x \le 20$, $30 \le z \le 50$, and with time t in sec, $0 \le t \le 86400$ (one day). These obey a pair of coupled reaction-diffusion equations:

$$\frac{\partial c^i}{\partial t} = K_h \frac{\partial^2 c^i}{\partial x^2} + \frac{\partial}{\partial z}\left(K_v(z) \frac{\partial c^i}{\partial z} \right) + R^i(c^1,c^2,t) \quad (i=1,2), \tag{4.38}$$

$$K_h = 4 \times 10^{-6}, \qquad K_v(z) = 10^{-8} e^{z/5}, \tag{4.39}$$

where the $R^i(c^1,c^2,t)$ are identical to those in Problems 3 and 7 (see (4.5) - (4.6)).

We impose homogeneous Neumann boundary conditions:

$$\partial c^i/\partial x = 0 \text{ at } x = 0 \text{ and } x = 20; \quad \partial c^i/\partial z = 0 \text{ at } z = 30 \text{ and } z = 50. \tag{4.40}$$

The initial conditions are given by polynomials that are slightly peaked in the center and consistent with the boundary conditions:

$$c^1(x,z,0) = 10^6\, \alpha(x)\, \beta(z), \quad c^2(x,z,0) = 10^{12}\, \alpha(x)\, \beta(z),$$
$$\alpha(x) = 1 - (.1x - 1)^2 + (.1x - 1)^4/2, \tag{4.41}$$
$$\beta(z) = 1 - (.1z - 4)^2 + (.1z - 4)^4/2.$$

These initial values agree with observations fairly well.

We reduce the PDEs to ODEs using spatial central differences and a

rectangular grid with uniform spacings, $\Delta x = 20/(J-1)$ and $\Delta z = 20/(K-1)$, as in Problem 7. If $c^i_{j,k}$ denotes the approximation to $c^i(x_j, z_k, t)$, where $x_j = (j-1)\Delta x$, $z_k = 30 + (k-1)\,\Delta z$, $1 \le j \le J$, $1 \le k \le K$, then we obtain the following ODE's:

$$\dot{c}^i_{j,k} = R^i(c^1_{j,k}, c^2_{j,k}, t) + (K_h/\Delta x^2)\,(c^i_{j+1,k} - 2c^i_{j,k} + c^i_{j-1,k}) + (\Delta z)^{-2}\,[K_v(z_{k+1/2})\,(c^i_{j,k+1} - c^i_{j,k}) - K_v(z_{k-1/2})\,(c^i_{j,k} - c^i_{j,k-1})]. \tag{4.42}$$

At the boundaries, we take:

$$\left.\begin{aligned} &c^i_{0,k} = c^i_{2,k},\ c^i_{J+1,k} = c^i_{J-1,k} \quad \text{(all k), and} \\ &c^i_{j,0} = c^i_{j,2},\ c^i_{j,K+1} = c^i_{j,K-1} \quad \text{(all j).} \end{aligned}\right\} \tag{4.43}$$

The size of the ODE system is $N = 2JK$. The variables are indexed first by species, then by x position, and finally by z position. Thus in

$\dot{y} = f(t,y)$, we have $c^i_{j,k} = y_m$ with $m = i + 2(j-1) + 2J(k-1)$

The underlying assumption is that J is no bigger than K to keep the bandwidth minimal.

A strategy similar to this was described by [Chang et al. (74)] who solved a system of over 14,000 ODEs in a study of the effect of supersonic transports on the ozone layer. The form of the PDEs was (4.38).

PROBLEM 9 - A Two Phase Plug Flow Problem

Here we are concerned with a pair of coupled, implicit differential-algebraic equations for an unusual pipeline problem [Byrne & Ho (83)]. Briefly, we are interested in piping a stable foam from a holding tank to a processing plant. The foam is gas bubbles dispersed in a liquid phase. If the foam (core phase) is to be successfully piped, it must be surrounded by an incompressible lubricating film (annular phase). Moreover, the pipeline pressure must be sufficiently high to keep the core from expanding to touch the wall.

We can develop this model by using the universal velocity law for very large Reynolds number flow through a smooth pipe for the annular phase. For the viscous core phase, we assume plug flow and expansion in the radial direction only when pressure decreases. We also assume a no slip condition at the interface between the two phases.

The equations describing the problem outlined above are:

$$\pi [R/(2\rho)]^{1/2} (R - y_c)^2 (-dP/dx)^{1/2} \cdot$$

$$\left\{ 2.5 \ln [(\rho R/2)^{1/2} (y_c/\mu) (-dP/dx)^{1/2} - 5] \right.$$

$$\left. + 10.5 \right\} - [bQ_{co} + P_o Q_{co} (1 - b)/P] = 0 \qquad (4.44)$$

$$2\pi [R/(2\rho)]^{1/2} (-dP/dx)^{1/2} \left\{ (2.5 Ry_c - 1.25 y_c^2) \cdot \right.$$

$$\ln [(\rho R/2)]^{1/2} (y_c/\mu) (-dP/dx)^{1/2} - 5] +$$

$$3Ry_c - 2.125 y_c^2 - 13.6 \cdot$$

$$\left. R\mu [2/R\rho)]^{1/2} (-dP/dx)^{-1/2} \right\} - Q_a = 0 \qquad (4.45)$$

In this system, the prescribed parameters are:

R = pipe radius (cm)

ρ = density of the annular phase (g/cm^3)

μ = viscosity of the annular phase (poise)

P_0 = inlet or initial pressure ($dynes/cm^2$)

Q_a = inlet flow rate for annulus or wetting agent (cm^3/s)

Q_{co} = inlet flow rate for the core or emulsion (cm^3/s)

b = inlet volumetric fraction of the liquid in the foam

The equations are to be solved for $0 \leq x \leq L$ where L is a prescribed length (cm), corresponding to several kilometers. The values to be computed at various distances down the pipe are: pressure P ($dynes/cm^2$) and the thickness of the annular phase y_c (cm). It is also convenient to know the pressure gradient, but not essential.

There are several interesting features in these equations. The pressure gradient dP/dx appears only with a negative sign and under radicals. (If $dP/dx \geq 0$, the system breaks down as we would expect when invoking Darcy's law.) The radicals appear both in the arguments of natural logarithms and outside the arguments. Neither the initial value for y_c nor for dP/dx is prescribed. Flow choking corresponds to vanishing or negative arguments of the natural logarithms. That is, choking would occur for a prescribed foam if the pressure gradient were too small in magnitude, the pipe radius were too narrow for the length L, or the initial annulus thickness were too small. In particular, we note that these equations are a coupled, implicit differential-algebraic system, which appears to be non-stiff. Some typical data are as follows:

Case 1. This is an example of a normal flow.

$R = 4.572.10^{1}$ cm

$\rho = 8.14.10^{-1}$ g/cm^3

$\mu = 9.8.10^{-2}$ poise

$b = 6.06.10^{-1}$

$Q_{co} = 1.1531 \times 10^{6}$ cm^3/s

$Q_a = 2.035 \times 10^{5}$ cm^3/s

$P_o = 1.457 \times 10^{8}$ dynes/cm^2

$L = 8.047 \times 10^{6}$ cm

Case 2. This is an example of a flow which choked. Parameters not listed have the values specified in Case 1.

$b = 3.45 \times 10^{-1}$

$Q_{co} = 1.7153 \times 10^{6}$

$Q_a = 3.027 \times 10^{5}$

$P_o = 1.378 \times 10^{8}$

$L = 3.2188 \times 10^{7}$

PROBLEM 10 - Troesch's Two Point Boundary Value Problem

It may seem unusual to see a two point boundary value problem listed as a stiff ODE. We will use essentially the same technique as in [Sincovec & Madsen (75)], which is in some sense tantamount to using time t as a continuation parameter to solve an elliptic problem.

The problem to be solved is

$$0 = \frac{\partial^2 u}{\partial x^2} - 10 \sinh (10u) \tag{4.46}$$

for $0 \leq x \leq 1$ with

$$\left.\begin{aligned} u(0) &= 0 \\ u(1) &= 1 \end{aligned}\right\} \tag{4.47}$$

We simply replace this problem with the related time dependent problem

$$\frac{\partial u}{\partial t} = \frac{\partial^2 u}{\partial x^2} - 10 \sinh (10u) \tag{4.48}$$

use the boundary conditions, and take an initial value

$$u(0,x) = 0 \quad \text{for } 0 \leq x \leq 1 \text{ and} \tag{4.49}$$
$$u(0,1) = 0$$

Again, we can use central differences to replace the second order spatial derivative in (4.48). (See Problem 7.)

The use of a uniform grid for this problem is soon found to be unwise, because the solution has a thin boundary layer near x = 1. Thus, a modest uniform grid misses this feature, and a sufficiently fine

one is inefficient outside of the boundary layer. Thus, following [Sincovec & Madsen (75)], we pose a nonuniform grid of 51 points, with 14 equal intervals on [0,.4], 13 on [.4,.7], 12 on [.7,.9], and 11 on [.9, 1]. Alternatively, we expect that a good dynamic grid or moving finite element algorithm would overcome this difficulty automatically.

Finally, we again point out that the last four examples are treated as partial differential equations by the numerical method of lines. We have indeed carried out the discretizations by hand and have used uniform mesh spacing in all but the last case. We do not advocate hand discretizations in general and we do generally recommend high quality method of lines codes. We have illustrated the last four examples as we did simply to show the requirements imposed on high quality ODE software, as well as its use.

B. Code Usage and Computational Results

We now turn to some of the pragmatics associated with the problems described in Subsection A of this section.

PROBLEM 1 - Robertson's Problem

The numerical solution of this problem illustrates the nearly logarithmic increase in step size and the control of a neutrally stable problem. The step size did increase dramatically. In fact, observed values ranged from about 4.5×10^{-1} to 1.7×10^{6}. The magnitude of the largest step size may be mildly surprising if we think in terms of asymptotic ($h \rightarrow 0$) numerical results.

The CPU times on a CRAY 1S for 10 output points was 0.06 s for both MF = 21 and for MF = 22. In LSODE, the software package we used -- MF = 21-- uses an analytic, user supplied, dense Jacobian. On the other hand, MF = 22 uses an internally generated, divided difference version of the dense Jacobian. A 2 in the first digit of MF signifies the choice of BDF or stiff option. It is not surprising that on a small, not very complicated problem, the run times and results would be very similar.

The graphical results in Figure 4.1 show how y^2 starts at 0, builds to about 3.6×10^{-4} at about $t = 2 \times 10^{-3}$ s and decays. This phenomena would be, at best, hard to capture by using absolute or relative error control alone. Note that y^1 decays from 1 on about the same time scale as y^3 builds from 0 to 1. Note that steady state is not reached until t is

in the millions. We used 100 data points for each component to generate Figure 4.1. Note that good quality graphics and a reasonable choice for scaling the dependent variables help to understand the chemistry.

The results shown in Figure 4.1 were obtained by setting the relative error tolerance (RTOL) to 10^{-6} and the absolute error tolerance (ATOL) to the vector $[10^{-6}, 10^{-10}, 10^{-6}]^T$

As a general rule of thumb, we like to set the relative error tolerance by asking how many digits of accuracy are required. If the answer is r digits, then we set RTOL = $1.0 \times 10^{-(r+1)}$ or less. (In the Robertson example, fairly small RTOL helps to control the stability problem.)

To set ATOL, we ask what the noise level is for each component of the solution. The noise level is the size of the largest number that may be neglected for that component. (In the national budget, 10^6 appears to be small enough.)

The selection of the error tolerances is very important and yet fairly straightforward. The penalties for loose tolerances are incorrect solutions and for tight tolerances the penalties are high cost.

For RTOL = 10^{-3}, ATOL = $[10^{-3}, 10^{-7}, 10^{-3}]^T$ the results are acceptable. The Cray 1S CPU time is about .015 s for MF=21 and 22.

PROBLEM 2 - The Field-Noyes Chemical Oscillator

The graphical results for this problem are shown in Figure 4.2 on the time interval 0 to 610.0747. It illustrates several interesting features:

- Disparate time scales
- A trigger notch, and
- The periodicity of the solution

The time scales are evident, since the graph of y^1 looks like 3 sharp upward direct spikes, y^3 has a sharp rise followed by a decay for about 90 s, and y^2 has a gentle rise and a decay for the remainder of the period. The sharp downward spike in y^2 has been called the trigger notch. To resolve these features, we used 484 data points for each component of the solutions. For these results, we used LSODE with RTOL = ATOL = 10^{-6}, and MF = 21. The problem features noted earlier are not major obstacles.

The tolerances of 10^{-6} may sound a bit academic. We should recall that LSODE does not control global error directly. It controls local error. With only 10 output points, the CPU time was 0.25 s on a CRAY 1s. With tolerances of 10^{-3}, the run times were lower, but the answers were highly inaccurate.

PROBLEM 3 - Two Species Diurnal Kinetics

This problem illustrates several points:

- Sharp fronts can be accommodated with modern ODE software.
- If negative results for the solution are smaller than ATOL and if these results do not make the solution unstable, then we should not worry about them.
- This problem requires the setting of a maximum time step size, HMAX. In this case HMAX = 3.6×10^3 (1 hour).

In particular, the last point warrants the caveat that neither ODE software packages nor their designers are omniscient. Intuition tells us that if the time step is too large in this problem, the solver can go past a major event--sunrise or sunset--and miss the feature we are after--the sharp buildup or decay of a species.

To solve this problem, we used LSODE with ATOL = 10^{-4}, RTOL = 10^{-6}, and MF = 22. The output in the timed run was taken every 6 hours. This run took 0.84 s on a CRAY 1S. With MF = 21, the CPU time was 0.81 s for these error tolerances. With RTOL = 10^{-3} and ATOL = 10^{-7}, CPU time was 0.29 s and 0.31 s for MF = 21 and 22, respectively. However, the results were not of as high a quality. With few exceptions, higher quality numerical results take more CPU time than low quality results. We solved this problem with EPISODE, too. That code should run well on a problem of this type because the time step length can be varied at each step. We used a relative error control of the following type. The error in Y^i was controlled relative to the quantity $\max(|y^i|, \text{FLOOR}^i)$. For this particular problem, we chose FLOOR = 10^{-4}. Consequently, when $|y^i| \geq \text{FLOOR}^i$, EPISODE tries to keep the magnitude of the local error in y^i less than $|y^i| * \text{EPS}$.

(Here, EPS is the user specified error tolerance.) When $|y^i| <$ FLOOR^i, EPISODE tries to keep the magnitude of the local error in y^i less than EPS * FLOOR^i. (See [Hindmarsh & Byrne (76a).]

For EPS = 10^{-6}, FLOOR = 10^{-4}, and MF = 21 the EPISODE run time was 0.55 CPU seconds and for MF = 22, 0.59 seconds. For this problem, EPISODE was faster than LSODE for all tolerances that we tried.

The graphical results in Figure 4.3 were obtained with 401 output points to resolve the solution adequately. Note that $y^2 = [O_3]$ looks like a staircase with a rise at mid-day every day. In Figure 4.3, $y^1 = [O]$ looks like a spike with its amplitude increasing each day. A logarithmic scale is used for the vertical axis. We cut off the bottom of the graph of y^1 to illustrate other features, such as the increases in peak values of y^1 and y^2. The t-axis is scaled in days with each day beginning at dawn, daylight lasting a half day and night lasting the remaining half day.

PROBLEM 4 - A Kidney Model

The graphical results (with 201 data points for each species) in Figures 4.4a, 4.4b, and 4.4c correspond to the initial values $y^5_0 =$ 0.990268335, 0.99, 0.9, respectively. In the graphical results, we used the observation that for t beyond 0.1, y^1, y^3, and y^5 were virtually equal. So, we did not plot y^3 and y^5. Furthermore, in Figures 4.4b and 4.4c, we show $|y^4|$. The figures illustrate the sensitivity of the solution to small changes in the choices of y^5_0. This is a vivid illustration that two point boundary value problems for first order ODEs can be challenging. However, the solution curves are not overly exciting. By actual computation with a non-stiff solver and with a stiff solver, we can

compare the cost for these options with LSODE. In all three cases, the cost for MF = 10 is at least twice as great as the cost for MF = 21 or 22. In LSODE, MF = 10 causes LSODE to use Adams method with functional iteration. Looking at the graphs probably does little to give insight to stiffness. However, the timing results indicate all 3 cases are stiff. We used RTOL = 10^{-6} and ATOL = 10^{-6} and 10 output points for the timed runs. The CPU times are in Table 4.4.

y_0^5	MF 10	21	22
0.990268835	.031	.012	.012
0.99	*	.020	.021
0.9	*	.055	.054

Table 4.4 CPU time in s for problem 4.

PROBLEM 5 - A Laser Oscillator Model

This is a challenging problem. (See Figure 4.5a for a 901 point plot for each ϕ and n.) It is initially stiff and then has a damped oscillatory structure with a period of about 7000 ns. Perhaps the earlier remarks about omniscience ought to be recalled. In any case, intuition suggests that an automatic method switching code such as LSODA would do well on a problem such as this. It does not because it chose to continue with the BDF method during the highly oscillatory part of the solution. In fact, the LSODA performance is comparable to a straight application of LSODE with MF = 21. What works most cheaply? Starting out with MF = 21 (BDF, analytic Jacobian) and switching to MF = 10 (Adams method,

functional iteration) is the cheapest in CPU time. We made the switch at $t = 4.9 \times 10^5$ because that corresponded to several times the fastest time constant. The catch is that either quite a little analysis to observe this is required or some numerical computation must be done. Frankly, neither may be realistic when results are needed quickly, staffing is short, or budgets are small. It is far more likely that the user would solve the problem with a straight application of LSODE.

A comparison of CPU times for 87 output points is given in Table 4.5 for several methods we tried. The results in Table 4.5 give CPU times for the various choices of method.

RTOL = 10^{-6}, ATOL = $[10^{-9}, 10^{-6}]^T$.

Code	MF	CPU(s)
LSODE	21	0.63
LSODE	21 → 10	0.47
LSODA	(JT = 1)	0.74

Table 4.5 Run times for problem 5

Turning back to Figure 4.5a, the amplitudes of the oscillations in ϕ (plotted on a logarithmic scale) are not represented by only 901 output points. However the key features of oscillation and damping are captured. Figure 4.5b gives somewhat more resolution (401 data points on a shorter t interval) and depicts less variation in peak values of ϕ.

PROBLEM 6 - Burgers' Equation

Reasons for including this example were noted earlier. Another is to illustrate the two solution techniques for this mildly stiff problem. Elsewhere we have worked with the traveling square pulse version of this problem [Byrne (79)]. We again note that the cell Peclet number must be fairly small for the numerical method of lines to work well on convective problems. Finally, in applying both the finite element and finite difference strategies, we use the banded version of the solver. The run times for 50 interior grid points and 4 output times are given in Table 4.6.

Solver	MF	Method	CPU(s)
LSODE	14	Finite Differences	0.032
LSODI	14	Simplified Galerkin	0.030

Table 4.6

The MF = 14 setting is for implicit Adams formulas with banded, analytic user supplied Jacobian. We again used LSODE, with RTOL = 0, ATOL = 10^{-3}. Similar run times were obtained for other banded options.

The graphical results are given in Figures 4.6a and b. The solutions for t = 0, 1, and 2 are shown in Figure 4.6. When t = 3, the solution fits the upper right corner of the u-x coordinate system. To the eye,

graphical results for the simplified Galerkin (Figure 4.6b) and finite difference method (Figure 4.6a) were identical. Moreover, results with 50 and 100 interior points were identical to the eye.

Note that we used MF = 14, *implicit* Adams formula and the analytic, banded Jacobian in the modified Newton iteration. Run times for BDF were comparable. This is the only test problem in the set for which we believe this to be true. Again, we refer to this as a mildly stiff problem, because functional iteration would be expensive.

PROBLEM 7 - Two Species Diffusion-Diurnal Kinetics - One Dimensional

This problem features both diffusion and kinetics. Consequently it is reasonable to include it in a problem set of this type. Moreover, as we mentioned earlier, it is similar to some early large scale method of line problems. There is also a pedagogical reason for including this problem. It is just a one dimensional mixing version of Problem 3.

There are some other points, too. The analytic Jacobian version is about twice as fast as the finite difference version.

For 10 output times, RTOL = 10^{-3}, ATOL = 10^{-1}, and 50 interior grid points, the run time with LSODE is 0.57 s with MF = 24. Figures 4.7a and b give the results for c^1 and c^2, respectivelly, at various values of t (hours).

PROBLEM 8 - Two Species Diffusion-Diurnal Kinetics - Two Dimensional

The lessons learned from one-dimensional problems can help with two dimensional problems. However, there are aspects of the game that are quite different. These include storage requirements for realistic resolu-

tion, selection of linear algebraic methods, and, of course, speed of solution. Pedagogically, it makes sense to add another degree of complexity to PROBLEM 7.

In keeping with these remarks we break out GEARBI, modified for two dimensional differencing problems. We give a brief comparison of GEARBI with LSODE. Some data are given in Table 4.8

Code	MF	ATOL	RTOL	CPU(s)
LSODE	24	10^{-1}	10^{-3}	10.4
LSODE	25	10^{-1}	10^{-3}	29.6
GEARBI	--	10^{-1}	10^{-3}	11.6

Table 4.8

Graphical results are given in Figures 4.8a-m. In these figures, $\Delta x = 20/19$.

For tighter error tolerances, GEARBI is faster than LSODE. These data are for a 20 x 20 grid and 12 output times.

PROBLEM 9 - A TWO PHASE PLUG FLOW PROBLEM

We actually solved this problem in three different ways. The first way was picked for such practical reasons as available software and severe constraints on the required turnaround time for the parameter study. The nonlinear implicit nature of the problem precluded the direct use of LSODI. The package DASSL was not initially at hand, so we used a combination of Newton's method and LSODE as follows.

- The spatial position x and pressure P are known values. (Initially, $x = 0$, $P = P_0$ and estimates for y_c and dP/dx were also made.) Call the integrator, which only uses or requires *discrete* values of P, dP/dx, and y_c.

- In the function subroutine called by LSODE, we treated (4.44) and (4.45) as two nonlinear equations in y_c and $u = (-dP/dx)^{1/2}$. These equations were solved using Newton's method.

- The value of $dP/dx = -u^2$ was passed from the function routine to the integrator, which in turn computed P at the next value of x, and so on. (The most recent available values of y_c and u were used to start Newton's method.)

- When output was requested at x = xout, the value of P(xout) was made available to the main program by the integrator. So Newton's method was used to find u(xout) (and hence dP/dx) and y_c(xout). It was useful, but hardly necessary, to have dP/dx as an output parameter.

In the second method of solution, we just used DASSL in the most obvious ways. (That is, we used the DASSL examples and preamble for the problem setup and coding.) The initial values and slopes were computed as before. Table 4.9 has CPU times for the normal flow case.

	IBM 3033 AP	Cray 1S
Newton/LSODE	0.04	*
DASSL (divided difference Jacobian)	0.03	0.027
DASSL (user-supplied Jacobian)	0.04	0.027
LSODI	*	0.024

Table 4.9

The IBM 3033 AP runs were with a Newton method taken from a continuation package. The Cray 1S runs used DZERX, an IMSL nonlinear system solver. The timings for the two machines were made in as nearly comparable ways as possible. However, the IBM runs were made in an interactive environment, and the timings are rather rough. There are some other algorithmic differences, too. We do not believe these two computers are generally comparable in speed. All runs are for Case 1 only. It is clear that the computation of the starting values has significant impact on the timing. Finally, the simplest codes to implement were those using DASSL.

There is a third way to solve this problem. By way of a sketch, it involves the following:

- Recognize that the logarithmic term is the same in both (4.44) and (4.45).
- Eliminate the logarithmic term to get a quadratic equation in $U = (-dP/dx)^{1/2}$.
- Rearrange the system to get:
  - + An explicit ODE for P
  - + An algebraic equation in P and y_c, with care taken to pick the right root of the quadratic equation.
- Solve the resulting system with LSODI.

It turns out that computationally this is faster than the other two methods.

Now we can look at some pragmatics. It is far simpler to use DASSL directly. With today's staffing costs, it is surely less expensive to apply DASSL directly. Less creative? Perhaps. The real life setting for the solution of this problem left little cushion for experimentation and importing of codes. The point is this. There are clever ways to solve problems. Occasionally, pragmatics preclude their discovery or use. With some of today's software, the risk is kept low. On the Cray 1S, the DASSL CPU time was .075 s with RTOL = ATOL = 10^{-6}, while the time for the LSODI trick was .069 s. Both include the cost of finding initial guesses, and of calling ZEROIN to find starting values. We can also get a feeling for the difference between Cray 1s and an IBM 3033 AP.

The graphical results for Case 1 (normal flow) are given in Figure 4.9a, while those for choked flow are given in Figure 4.9b. The normal flow solution is fairly linear. The dramatic change in the pressure gradient in the choked flow case is perhaps not surprising. We used 101 and 69 data points to generate Figures 4.9a and 4.9b, respectively.

One feature that this problem has that is in no other problem in this set is that we expected an abnormal termination of some kind for the case of choked flow (Case 2). In the original runs, we did not know when to expect choked flow and trapped for non-positive arguments of the logarithm or error flags from the integrator. According to the method used, we observed both types of conditions.

PROBLEM 10 - Troesch's Two Point Boundary Value Problem

The graphical results for this problem are given in Figures 4.10a and 4.10b. The runs were made with LSODE and the banded Jacobian options for both the analytic (MF = 24) and divided difference (MF = 25) Jacobians. The error tolerances were set with RTOL = 0 and ATOL = 10^{-3}. For 50 interior grid points, the run times were 0.06 and 0.07 s for MF = 24 and 25, respectively.

The graphs show how the initial guess relaxes. The graphical solutions for t = 0.1 and t = 1 overlapped. We used 51 and 24 points to generate each curve in 4.10a and 4.10b, respectively.

5. RELATED DEVELOPMENTS

There are many ongoing or recent projects of potential interest here. For example, the active work on Krylov subspace methods could mean that the BDF solvers would need but minimal storage for certain types of PDEs. At this time it is not quite clear to which classes of PDEs this work will be applicable [Brown & Hindmarsh (85)]. Other storage reduction methods for MOL solution of PDEs have also been investigated. These include several adaptations of Newton's method -- Newton/successive over relaxation (SOR), SOR/Newton, using only diagonal blocks of the Jacobian and so on. Again, the extent of the applicability of these methods is not well understood [Byrne & Hindmarsh (85)]. In the broadest sense, the moving finite element methods and dynamic grid methods could also be regarded as storage reduction methods. So far, most test results are available for only one or two PDEs in one spatial dimension. The worth of these techniques will be fully realized for reasonably sized systems of PDEs in one and two spatial dimensions or problems in three spatial dimensions.

It seems likely to us that these methods will be effective on reactive, diffusive, convective flows. If this is to be, the cell Peclet numbers will be kept low by the adaptive technique.

We have alluded to automatic method switching elsewhere in this paper. The idea is for the code to pick a stiff or a non-stiff ODE method automatically and dynamically. In this way, the more efficient method is automatically applied to each phase of a problem. So far, efforts along these lines have been few. They have, however, met with some success [Petzold (83b) Shampine (81), (82b)]. The extent of the consequences of such a code is not clear. However, there is clear potential market for an automatic method switching code, because the user does not need to choose a method.

The trend toward scientific/engineering workstations will impact ODE solvers. We can expect to see an even stronger trend toward non-Fortran front ends and graphical output in this setting. It is likely that the smaller problems will be run on the workstation and the larger ones uploaded to a large scale computer or supercomputer. In this supercomputer setting, the workstation would serve as a pre- and post processor.

The idea of having a good, inexpensive global error estimate has drawn a fair amount of attention. In most cases, this amounts to a clever interpretation or solution of a fairly simple differential equation (variational equation). It appears that most of these estimates are based on an asymptotic analysis ($h \to 0$) and are currently fairly crude. Also, there is not much numerical evidence to show that the global techniques are superior to the current local estimates. However, global estimates may prove to be practical as they stand or as they evolve in the future. Some work in this area for stiff ODEs includes: [Dew & West (79)], [Robinson & Prothero (77)], [Prothero (80)], [Dahlquist (81a), (81b)], and [Shampine (82)]. The value in computing global error is this. Global error is what the user really wants to control. One issue is whether the global error estimate will be simply supplied to the user or whether the estimate will be used to control order and step size selection. Another is the need to carry a differential equation or its reduced cost solution for each ODE in the system. Finally, there is some disagreement regarding the required quality of this estimate.

One drawback of using a variable step BDF lies in the retention of the matrix

$$P = [I - h\beta_0 f_y]$$

in the corrector phase of the code (See 2.12)). The parameter β_0 changes with h. Consequently, the matrix P gets out of date, then needs to be re-evaluated

and factored. The basic idea is this. If the coefficient β_0 can be held fixed, then the matrix P does not need to be recomputed and factored as often as when β_0 varies freely. The cost of recomputing and factoring can be high. The scheme becomes a little more apparent if the system (2.11) is multiplied by $b = 1/\beta_0$ so that bI and hf_y can, in some sense be treated separately. By using interpolation, b can remain fixed when h changes. The consequence is that a fixed leading coefficient method can be more stable than a fixed step-interpolate strategy (GEAR, LSODE) and less expensive, and a little less stable than a variable step method (EPISODE). Indeed, Petzold used the fixed leading coefficient approach in DASSL. [Petzold (83a)]. The pioneering work in this area was done by Jackson and Sacks-Davis (80). The idea is promising since it works well in DASSL and performed well in the Jackson and Sacks-Davis prototypical revision of EPISODE.

6. SUMMARY

We have discussed the notion of stiffness, where it arises, and how to pragmatically test for it in Section 1. There, we also looked at an example of a neutrally (conditionally) stable system of ODEs, looked at eigenvalues of both ODEs and spatially discretized PDEs, particularly the heat equation. We then turned to the various structures of systems of ODEs and talked about the origins of the structures. With this discussion, we associated the importance of the structure of systems of ODEs and how we might take advantage of them. Finally, we discussed some of the features of the *solutions* of systems of ODEs and how high quality software must handle them.

In Section 2, we presented some of the underlying methods for solving stiff ODEs. These included BDF, Runge-Kutta, and other methods. In the category of other methods were: averaging, extrapolation, one-leg, multi-derivative, partitioning, composite, block, fitting, collocation, and blended methods. Along the way, we also discussed error and step size control.

In Section 3, we gave a brief historical perspective of the development of some stiff ODE software. Then, we turned to a discussion of the readily available stiff ODE solvers. (Much of the highest quality software is available at low or no cost. See the Appendix for sources of software.)

Section 4 is where we gave a number of examples. One word of caution. These examples are not very large or time consuming, in general. For example, see [Chang, et al. (74)], [Byrne & Hindmarsh (85)] for problems that are larger in scale. We mention this because problem size can and does bias test results dramatically. Nonetheless, we believe these examples are fairly representative with respect to many features. We did not, however, give

examples using zeros of functions (g-stops) or problems involving extensive constraints. These types of problems do occur and with some frequency [Thompson & Tuttle (82)].

We believe that this review will be helpful to line scientists and engineers faced with the need to solve a large number of problems quickly and efficiently. Consequently, it could be (and has been) used in the classroom or as a basis for a workshop. We also believe that this review indicates some of the more promising areas of research and development for the solution of stiff ODEs.

Finally, managers of scientific computing units can use this paper for an overview of the field.

7. ACKNOWLEDGEMENTS

We gratefully acknowledge Dr. Raymond C. Y. Chin, Lawrence Livermore National Laboratory, for his patient encouragement and suggestion that we write such a review paper.

We also thank the management at Lawrence Livermore National Laboratory, especially Dr. Nora Smiriga, Computing and Mathematics Research Division Leader, for making facilities available to us. It was there that we sketched and drafted the key elements of this paper and did most of our computing. The constructive criticism of Drs. Linda Petzold and Scott Weidman and Professor L. E. Scriven were very helpful.

Finally, we thank the management at Exxon Research and Engineering Company for making it possible for us to reunite to write this paper and for its processing.

8. REFERENCES

[Alexander (77)]
R. Alexander, "Diagonally Implicit Runge-Kutta Methods for Stiff ODEs," SIAM J. Numer. Anal., 14 (1977), pp. 1006-1021.

[Alfeld & Lambert (77)]
P. Alfeld and J. D. Lambert, "Correction in the Dominant Space: A Numerical Technique for a Certain Class of Stiff Initial Value Problems," Math. Comp., 31 (1977), pp. 922-938.

[Andria, et al. (73)]
G. D. Andria, G. D. Byrne and D. R. Hill, "Natural Spline Block Implicit Methods," BIT, 13 (1973), pp. 131-144.

[Axelrod et al. (83)]
T. S. Axelrod, P. F. Dubois, R. B. Hickman, A. C. Hindmarsh, and J. F. Painter, "TORANAGA," Lawrence Livermore Laboratory report UCID-30190 (Jan 1983).

[Bader & Deuflhard (83)]
G. Bader and P. Deuflhard, "A Semi-Implicit Midpoint Rule for Stiff Systems of Ordinary Differential Equations," Numer. Math., 41 (1983), pp. 373-398.

[Benton & Platzman (72)]
E. R. Benton and G. W. Platzman, "A Table of Solutions of the One-Dimensional Burgers Equation," Quart. Appl. Math., 30 (1972), pp. 195-212.

[Berzins et al. (84)]
M. Berzins, P. M. Dew, and R. M. Furzeland, "Software for Time-Dependent Problems in PDE Software, Modules, Interfaces, and Systems." B. Engquist and T. Smedsaas, Eds., North-Holland, Amsterdam, 1984, pp. 309-322.

[Bickart & Picel (73)]
T. A. Bickart and Z. Picel, "High Order Stiffly Stable Composite Multistep Methods for Numerical Integration of Stiff Differential Equations," BIT, 13 (1973), pp. 272-286.

[Bird, et al. (60)]
R. B. Bird, W. E. Stewart, E. N. Lightfoot, Transport Phenomena, John Wiley & Sons, New York, 1960.

[Björck (83)]
Åke Björck, "Some Methods for Separating Stiff Components in Initial Value Problems," in Proceedings of Dundee Numerical Analysis Conference (1983), Springer-Verlag, to appear.

[Brayton, et al. (72)]
R. K. Brayton, F. G. Gustauson and H. D. Hachtel, "A New Efficient Algorithm for Solving Differential-Algebraic Systems Using Backward Differentiation Formulas," Proc. IEEE, 60 (1972), pp. 98-108.

[Brown & Hindmarsh (85)]
P. N. Brown and A. C. Hindmarsh, "Matrix-Free Methods for Stiff Systems of ODEs," SIAM J. Num. Anal., to appear; also Lawrence Livermore National Laboratory report UCRL-90770, Rev.1 (Feb. 1985).

[Burrage (78)]
K. Burrage, "A Special Family of Runge-Kutta Methods for Solving Stiff Differential Equations," BIT, 18 (1978), pp. 22-41.

[Burrage et al. (80)]
K. Burrage, J. C. Butcher, and F. H. Chipman, "An Implementation of Singly-Implicit Runge-Kutta Methods," BIT, 20 (1980), pp. 326-340.

[Butcher (73)]
J. C. Butcher, "The Order of Numerical Methods for Ordinary Differential Equations," Math. Comp., 27 (1973), pp. 793-806.

[Butcher (81)]
J. C. Butcher, "General Linear Methods: Prospects as Stiff Solvers," in Numerical Methods for Solving Stiff Initial Value Problems, G. Dahlquist and R. Jeltsch, Eds., Proceedings of Conference at Oberwolfach, 1981; Rheinisch-Westfälische Technische Hochschule Aachen, Inst. f. Geometrie u. Praktische Mathematik (1981).

[Butcher (85)]
J. C. Butcher, "General Linear Methods: A Survey," Appl. Numer. Math., 1 (1985), pp. 273-284.

[Byrne, et al. (77)]
G. D. Byrne, A. C. Hindmarsh, K. R. Jackson and H. G. Brown, "Comparative Test Results for Two ODE Solvers -- EPISODE and GEAR," Argonne National Laboratory report ANL 77-19 (March, 1977).

[Byrne (79)]
G. D. Byrne, "The ODE Solver EPISODE, Its Variants and Their Use," in American Nuclear Society Proceedings of the Topical Meeting on Computational Methods in Nuclear Engineering, Vol. 3, American Nuclear Society, Virginia Section, Lynchburg, VA, April 1979, pp. 8-17 to 8-52.

[Byrne & Hindmarsh (75)]
G. D. Byrne and A. C. Hindmarsh, "A Polyalgorithm for the Numerical Solution of Ordinary Differential Equations," ACM Trans. Math. Software, 1 (1975), pp. 71-96.

[Byrne & Hindmarsh (76)]
G. D. Byrne and A. C. Hindmarsh, "EPISODEB: An Experimental Package for the Integration of Systems of Ordinary Differential Equations with Banded Jacobians," Lawrence Livermore Laboratory Report UCID-30132 (April 30, 1976).

[Byrne & Hindmarsh (77)]
G. D. Byrne and A. C. Hindmarsh, "Numerical Solution of Stiff Ordinary Differential Equations," AIChE Today Series, American Institute of Chemical Engineers, New York (1977).

[Byrne & Hindmarsh (85)]
G. D. Byrne and A. C. Hindmarsh, "Experiments in Numerical Methods for a Problem in Combustion Modeling," Appl. Numer. Math., 1 (1985), pp. 29-57.

[Byrne & Ho (83)]
G. D. Byrne and W. S. Ho, "A Solution of a Differential-Algebraic System: A Trick," Exxon Research and Engineering Company report CTSD.2DI.83, (May 1983.)

[Byrne & Lambert (66)]
G. D. Byrne and R. J. Lambert, "Pseudo-Runge-Kutta Methods Involving Two Points," J. Assoc. Comput. Mach., 13 (1966), pp. 114-123.

[Carver (79)]
M. B. Carver, "FORSIM VI: A Program Package for the Automated Solution of Arbitrarily Defined Differential Equations," in Comput. Phys. Comm., 17 (1979), pp. 239-282.

[Cash (81)]
J. R.Cash, "On the Design of High Order Exponentially Fitted Formulae for the Numerical Integration of Stiff Systems," Numer. Math., 36 (1981), pp. 253-266.

[Cash (83)]
J. R. Cash, "Blended Extended Linear Multistep Methods for the Accurate Numerical Integration of Stiff Initial Value Problems," Appl. Numer. Math., 1 (1985), pp. 195-216.

[Cash & Singhal (82)]
J. R. Cash and A. Singhal, "Mono-Implicit Runge-Kutta Formulae for the Numerical Integration of Stiff Differential Systems," IMA J. Numer. Anal., 2 (1982), pp. 211-227.

[Chang, et al. (73)]
J. S. Chang, A. C. Hindmarsh and N. K. Madsen, "Simulation of Chemical Kinetics Transport in the Stratosphere," in Stiff Differential Systems, R. A. Willoughby, Ed., Plenum Press, New York 1973, pp. 51-65.

[Chin, et al. (79)]
R. C. Y. Chin, G. W. Hedstrom and K. E. Karlsson, "A Simplified Galerkin Method for Hyperbolic Equations," Math. Comp., 33 (1979), pp. 647-658.

[Curtis (78)]
A. R. Curtis, "Solution of Large, Stiff Initial Value Problems - The State of the Art," in Numerical Software - Needs and Availability, D. A. H. Jacobs (Ed.), Academic Press, London, 1978, pp. 257-278.

[Curtiss & Hirschfelder (52)]
C. F. Curtiss and J. O. Hirschfelder, "Integration of Stiff Equations," Proc. Nat. Acad. Sci. U.S.A., 38 (1952), pp. 235-243.

[Dahlquist (81a)]
G. Dahlquist, "On the Control of Global Error in Stiff Initial Value Problems," Dept. of Numerical Analysis and Computing Science, Royal Institute of Technology, Stockholm (1981).

[Dahlquist (81b)]
G. Dahlquist, "On the Local and Global Errors of One-Leg Methods," Dept. of Numerical Analysis and Computing Science, Royal Institute of Technology, Stockholm, report TRITA-NA-81110. (1981).

[Dahlquist (83)]
G. Dahlquist, "On One-Leg Multistep Methods," SIAM J. Numer. Anal., 20 (1983), pp. 1130-1138.

[Deuflhard et al. (81)]
P. Deuflhard, G. Bader, and U. Nowak, "LARKIN - A Software Package for the Numerical Simulation of LARge Systems Arising in Chemical Reaction KINetics," in Modelling of Chemical Reaction Systems, K. H. Ebert et al., Eds., Springer-Verlag, 1981, pp. 38-55.

[Deuflhard (85)]
P. Deuflhard, "Recent Progress in Exrapolation Methods for Ordinary Differential Equations," SIAM Rev., 27(85), pp. 505-535.

[Deuflhard et al. (85)]
P. Deuflhard, E. Hainer, and J. Zugck, "One Step Extrapolation Methods for Differential-Algebraic Systems," Institut fur Angewandte Mathematik, Universitat Heidelbeg preprint nr. 318 (June 1985).

[Dew & West (79)]
P. M. Dew and M. R. West, "Estimating and Controlling the Global Error in Gear's Method," BIT, 19 (1979), pp. 135-137.

[Dickinson & Gelinas (76)]
R. P. Dickinson and R. J. Gelinas, "Sensitivity Analysis of Ordinary Differential Systems - A Direct Variational Method," J. Comput. Phys., 21 (1976), pp. 123-143.

[Douglas (70)]
J. Douglas, Jr., "The Numerical Solution of a Composition Model in Petroleum Reservoir Engineering," Numerical Solution of Field Problems in Continuum Physics, SIAM-AMS Proceedings, Vol. II, G. Birkhoff and R. S. Varga, Eds., American Mathematical Socity, Providence, Rhode Island, 1970, pp. 54-59.

[Eisenstat et al. (77)]
S. C. Eisenstat, M. C. Gursky, M. H.Schultz, and A. H. Sherman, "Yale Sparse Matrix Package: II. The Nonsymmetric Codes," Yale University Computer Sciences Dept. report no. 114, (1977).

[Eisenstat et al. (82)]
S. C. Eisenstat, M. C. Gursky, M. H. Schultz, and A. H. Sherman, "Yale Sparse Matrix Package: I. The Symmetric Codes," Int. J. Num. Meth. Eng., 18 (1982), pp. 1145-1151.

[Eitelberg (82)]
E. Eitelberg, "Modular Simulation of Large Stiff Systems," in Progress in Modelling and Simulation, F. E. Cellier, Ed., Academic Press, 1982, pp. 281-292.

[Enright (74)]
W. H. Enright, "Second Derivative Multistep Methods for Stiff Ordinary Differential Equations," SIAM J. Numer. Anal., 11 (1974), pp. 321-331.

[Enright et al. (75)]
W. H. Enright, T. E. Hull, and B. Lindberg, "Comparing Numerical Methods for Stiff Systems of ODEs," BIT, 15 (1975), pp. 10-48.

[Enright & Hull (76)]
W. H. Enright and T. E. Hull, "Comparing Numerical Methods for the Solution of Stiff Systems of ODEs Arising in Chemistry," in Numerical Methods for Differential Systems, Recent Developments in Algorithms, Software and Applications, L. Lapidus and W. E. Schiesser, Eds., Academic Press, New York, 1976, pp. 45-66.

[Enright & Kamel (79)]
W. H. Enright and M. S. Kamel, "Automatic Partitioning of Stiff Systems and Exploiting the Resultant Structure," ACM Trans. Math. Software, 5 (1979), pp. 374-385.

[Field & Noyes (74)]
R. J. Field and R. M. Noyes, "Oscillations in Chemical Systems. IV. Limit Cycle Behavior in a Model of a Chemical Reaction," J. Chem. Phys., 60 (1974), pp. 1877-1884.

[Gaffney (84)]
Patrick W. Gaffney, "A Performance Evaluation of Some FORTRAN Subroutines for the Solution of Stiff Oscillatory Ordinary Differential Equations," ACM Trans. Math. Software, 10 (1984), pp. 58-72.

[Gear (68)]
C. W. Gear, "The Automatic Integration of Stiff Ordinary Differential Equations," Proceedings of the 1968 IFIP Congress held in Edinburgh, Scotland, A. J. H. Morrel, Ed., North Holland Publishing Co., Amsterdam (1968), pp. 187-193.

[Gear (71a)]
C. W. Gear, Numerical Initial Value Problems in Ordinary Differential Equations, Prentice-Hall, Englewood Cliffs, New Jersey, 1971.

[Gear (71b)]
C. W. Gear, "Simultaneous Numerical Solution of Differential-Algebraic Equations," IEEE Trans. Circuit Theory, CT-18 (1971), pp. 89-95.

[Gear (80)]
C. W. Gear, "Runge-Kutta Starters for Multistep Methods," ACM Trans. Math. Software, 6 (1980), pp. 263-279.

[Gear & Petzold (84)]
C. W. Gear and L. R. Petzold, "ODE Methods for the Solution of Differential-Algebraic Systems, SIAM J. Numer. Anal., 21 (1984), pp. 716-728.

[Gelinas (74)]
R. J. Gelinas, "Diurnal Kinetic Modeling," Lawrence Livermore Laboratory report UCRL-75373 (Jan. 1974).

[Gladwell (79)]
I. Gladwell, "Initial Value Routines in the NAG Library," ACM Trans. Math. Software, 5 (1979), pp. 386-400.

[Gottfried (65)]
B. S. Gottfried, "A Mathematical Model of Thermal Oil Recovery in Linear Systems," Soc. Pet. Eng. J., 5 (1965), pp. 196-210.

[Gottwald & Wanner (81)]
B. A. Gottwald and G. Wanner, "A Reliable Rosenbrock Integrator for Stiff Differential Equations," Computing, 26 (1981), pp. 355-360.

[Gupta et al. (85)]
G. K. Gupta, R. Sacks-Davis, and P. E. Tischer, "A Review of Recent Developments in Solving ODEs," ACM Computing Surveys, 17 (1985), pp. 5-47.

[Hachtel, et al. (71)]
G. D. Hachtel, R. K. Brayton, and F. G. Gustavson, "The Sparse Tableau Approach to Network Analysis and Design," IEEE Trans. Circuit Theory, CT-18 (1971), pp. 101-113.

[Hairer & Wanner (73)]
E. Hairer and G. Wanner, "Multistep Multistage Multiderivative Methods for Ordinary Differential Equations," Computing, 11 (1973), pp. 287-303.

[Henrici (62)]
P. Henrici, "Discrete Variable Methods in Ordinary Differential Equations," John Wiley & Sons, NY, 1962.

[Hindmarsh (74)]
A. C. Hindmarsh, "GEAR: Ordinary Differential Equation System Solver," Lawrence Livermore Laboratory report UCID-30001, Rev. 3 (Dec. 1974).

[Hindmarsh (76a)]
A. C. Hindmarsh, "Preliminary Documentation of GEARIB: Solution of Implicit Systems of Ordinary Differential Equations with Banded Jacobian," Lawrence Livermore Laboratory report UCID-30130 (Feb. 1976).

[Hindmarsh (76b)]
A. C. Hindmarsh, "The LLL Family of Ordinary Differential Equation Solvers," Lawrence Livermore Laboratory report UCRL-78129 (April 1976).

[Hindmarsh (76c)]
A. C. Hindmarsh, "Preliminary Documentation of GEARBI: Solution of ODE Systems with Block-Iterative Treatment of the Jacobian," Lawrence Livermore Laboratory report UCID-30149 (Dec. 1976).

[Hindmarsh (77)]
A. C. Hindmarsh, "GEARB: Solution of Ordinary Differential Equations Having Banded Jacobian," Lawrence Livermore Laboratory report UCID-30059, Rev. 2 (June 1977).

[Hindmarsh (78)]
A. C. Hindmarsh, "A Tentative User Interface Standard for ODEPACK, Lawrence Livermore Laboratory report UCID-17954 (Oct. 1978).

[Hindmarsh (79)]
A. C. Hindmarsh, "A Collection of Software for Ordinary Differential Equations," in American Nuclear Society Proceedings of the Topical Meeting on Computational Methods in Nuclear Engineering, Vol. 2. American Nuclear Society Virginia Section, Lynchburg, VA, April 1979, pp. 8-1 to 8-15. [Also UCRL-82091, Lawrence Livermore Laboratory, Livermore, CA (Jan. 1979)].

[Hindmarsh (80)]
A. C. Hindmarsh, "LSODE and LSODI, Two New Initial Value Ordinary Differential Equation Solvers," ACM SIGNUM Newsletter, 15, No. 4 (1980), pp. 10-11.

[Hindmarsh (81)]
A. C. Hindmarsh, "ODE Solvers for Use with the Method of Lines," in Advances in Computer Methods for Partial Differential Equations-IV, R. Vichnevetsky and R. S. Stepleman, Eds., IMACS, Rutgers University, New Brunswick, NJ, 1981, pp. 312-316.

[Hindmarsh (83)]
A. C. Hindmarsh, "ODEPACK, a Systematized Collection of ODE Solvers," in Scientific Computing, R. S. Stepleman et al., Eds., North-Holland, Amsterdam, 1983, pp. 55-64.

[Hindmarsh & Byrne (76a)]
A. C. Hindmarsh and G. D. Byrne, "Applications of EPISODE: An Experimental Package for the Integration of Systems of Ordinary Differential Equations," in Numerical Methods for Differential Systems, Recent Developments in Algorithms, Software, and Applications, L. Lapidus and W. E. Schiesser, Eds., Academic Press, New York, 1976, pp. 147-166.

[Hindmarsh & Byrne (76b)]
A. C. Hindmarsh and G. D. Byrne, "A Proposed ODEPACK Calling Sequence," Lawrence Livermore Laboratory report UCID-30134, May 1976.

[Hindmarsh & Byrne (77)]
A. C. Hindmarsh and G. D. Byrne, "EPISODE: An Effective Package for the Integration of Systems of Ordinary Differential Equations," Lawrence Livermore Laboratory report UCID-30112, Rev. 1 (April 1977).

[Hofer (76)]
E. Hofer, "A Partially Implicit Method for Large Stiff Systems of ODEs with Only Few Equations Introducing Small Time-Constants," SIAM J. Numer. Anal., 13 (1976), pp. 645-663.

[Hull, et al. (76)]
T. E. Hull, W. H. Enright, and K. R. Jackson, "User's Guide for DVERK - A Subroutine for Solving Non-Stiff ODEs," Department of Computer Science, University of Toronto, report TR No. 100 (Oct. 1976).

[Hulme (72)]
B. L. Hulme, "Discrete Galerkin and Related One-Step Methods for Ordinary Differential Equations," Math. Comp., 26 (1972), pp. 881-891.

[Hulme & Daniel (74a)]
B. L. Hulme and S. L. Daniel, "COLODE: A Collocation Subroutine for Ordinary Differential Equations", Sandia Laboratories report SAND74-0380 (Dec. 1974).

[Hulme & Daniel (74b)]
B. L. Hulme and S. L. Daniel, "Using STFODE/COLODE to Solve Stiff Ordinary Differential Equations," Sandia Laboratories report SAND74-0381 (Dec. 1974).

[IMSL (82)]
IMSL Library Reference Manual 1, IMSL, Houston, 1982.

[Jackson & Sacks - Davis (80)]
K. R. Jackson and R. Sacks - Davis,"An Alternative Implementation of Variable Step-Size Multistep Formulas for Stiff ODEs," ACM Trans. Math. Software 6 (1980), pp. 295-318."

[Kaps & Rentrop (79)]
P. Kaps and P. Rentrop,"Generalized Runge-Kutta Methods of Order Four with Stepsize Control for Stiff Ordinary Differential Equations," Numer. Math., 33 (1979), pp. 55-68.

[Kaps & Wanner (81)]
P. Kaps and G. Wanner, "A Study of Rosenbrock-Type Methods of High Order," Numer. Math., 38 (1981), pp. 279-298.

[Kaps et al. (85)]
P. Kaps, S. W. H. Poon, and T. D. Bui, "Rosenbrock Methods for Stiff ODEs: A Comparison of Richardson Extrapolation and Embedding Technique," Computing, 34 (1985), pp.17-40.

[Klopfenstein (71)]
R. W. Klopfenstein, "Numerical Differentiation Formulas for Stiff Systems of Ordinary Differential Equations," RCA Rev., 32 (1971), pp. 447-462.

[Klopfenstein (74)]
R. W. Klopfenstein, RCA David Sarnoff Laboratory, private communication on IMPEQ (1974).

[Klopfenstein & Davis (71)]
R. W. Klopfenstein and C. B. Davis, "PECE Algorithms for the Solution of Stiff Systems of Ordinary Differential Equations," Math. Comp., 25 (1971), pp. 457-473.

[Krogh (68)]
F. T. Krogh, "The Numerical Integration of Stiff Differential Equations," TRW report 99900-6573-R000 (1968).

[Krogh & Stewart (84)]
F. T. Krogh and K. Stewart, "Asymptotic ($h\to\infty$) Absolute Stability for BDFs Applied to Stiff Differential Equations," ACM Trans. Math. Software, 10 (1984), pp. 45-57.

[Lambert & Sigurdsson (72)]
J. D. Lambert and S. T. Sigurdsson, "Multistep Methods with Variable Matrix Coefficients," SIAM J. Numer. Anal., 9 (1972), pp. 715-733.

[Leaf & Minkoff (84a)]
G. K. Leaf and M. Minkoff, "DISPL2: A Software Package for Solving One and Two Spatially Dimensioned Convection-Diffusion Kinetics Nonlinear PDEs," Advances in Computer Methods for Partial Differential Equations-V, IMACS, 1984, pp. 429-432.

[Leaf & Minkoff (84b)]
G. K. Leaf and M. Minkoff, "DISPL1: A Software Package for One and Two Spatially Dimensioned Kinetics-Diffusion Problems," Argonne National Laboratory report ANL-84-56 (Sept. 1984).

[Lindberg (74)]
B. Lindberg, "A Stiff System Package Based on the Implicit Midpoint Method with Smoothing and Extrapolation," in Stiff Differential Systems, Ralph A. Willoughby, Ed., Plenum Press, New York, 1974, pp. 201-215.

[Liniger (76)]
W. Liniger, "High-Order A-Stable Averaging Algorithms for Stiff Differential Systems," in Numerical Methods for Differential Systems," L. Lapidus and W. E. Schiesser, Eds., Academic Press, New York, 1976, pp. 1-23.

[Liniger & Willoughby (70)]
W. Liniger and R. A. Willoughby, "Efficient Integration Methods for Stiff Systems of Ordinary Differential Equations," SIAM J. Numer. Anal., 7 (1970), pp. 47-66.

[Madsen & Sincovec (74)]
N. K. Madsen and R. F. Sincovec, "The Numerical Method of Lines for the Solution of Nonlinear Partial Differential Equations," in Computational Methods in Nonlinear Mechanics, J. T. Oden, et al., Eds., Texas Institute for Computational Mechanics, University of Texas at Austin, 1974, pp. 371-380.

[Madsen & Sincovec (79)]
N. K. Madsen and R. F. Sincovec, "Algorithm 540, PDECOL, General Software for Partial Differential Equations [D3]:," ACM Trans. Math. Software, 5 (1979), pp. 326-351.

[Morris et al. (77)]
D. B. Morris, A. C. Hindmarsh, and P. F. Dubois, "GEARV and GEARST: Vectorized Ordinary Differential Equation Solvers for the 7600 and STAR Computers," Lawrence Livermore Laboratory report UCID-30119, Rev. 1 (Dec. 1977).

[NAG (82)]
NAG Library, Mark 9, Oxford, 1982.

[Norsett & Wolfbrandt (79)]
S. P. Norsett and A. Wolfbrandt, "Order Conditions for Rosenbrock Type Methods," Numer. Math. 32 (1979), pp. 1-15.

[Peaceman (77)]
Donald W. Peaceman, "Fundamentals of Reservoir Simulation," Elsevier Scientific Publishing Co., New York, 1977.

[Pelios & Klopfenstein (72)]
A. Pelios and R. W. Klopfenstein, "Minimal Error Constant Numerical Differentiation (N.D.) Formulas," Math. Comp., 26 (1972), pp. 467-475

[Petzold (82)]
L. R. Petzold, "Differential/Algebraic Equations are not ODEs," SIAM J. Sci. Stat. Comput., 3 (1982), pp. 367-384.

[Petzold (83a)]
L. R. Petzold, "A Description of DASSL: A Differential-Algebraic System Solver," in Scientific Computing, R. S. Stepleman et al., Eds., North Holland, Amsterdam, 1983, pp. 65-68.

[Petzold (83b)]
L. R. Petzold, "Automatic Selection of Methods for Solving Stiff and Nonstiff Systems of Ordinary Differential Equations," SIAM J. Sci. Stat. Comput., 4 (1983), pp. 136-148.

[Price, et al. (66)]
H. S. Price, R. S. Varga, and J. E. Warren, "Application of Oscillation Matrices to Diffusion-Convection Equations," J. Math. and Phys., 45 (1966), pp. 301-311.

[Prothero (80)]
A. Prothero, "Estimating the Accuracy of Numerical Solutions to Ordinary Differential Equations," in Computational Techniques for Ordinary Differential Equations, I. Gladwell and D. K. Sayers, eds., Academic Press, London, 1980, pp. 103-128.

[Robertson (66)]
H. H. Robertson, "The Solution of a Set of Reaction Rate Equations," in Numerical Analysis, An Introduction, J. Walsh, Ed., Academic Press, 1966, pp. 178-182.

[Robinson & Prothero (77)]
A. Robinson and A. Prothero, "Global Error Estimates for Solutions to Stiff Systems of Ordinary Differential Equations," contributed paper, Dundee Numerical Analysis Conference, 1977.

[Rosser (67)]
J. B. Rosser, "A Runge-Kutta for All Seasons," SIAM Rev., 9 (1967), pp. 417-452.

[Rübner-Peterson (73)]
T. Rübner-Peterson, "An Efficient Algorithm Using Backward Time-Scaled Differences for Solving Stiff Differential-Algebraic Systems," Institute of Circuit Theory and Telecommunications, Building 326, Technical University of Denmark, 2800 Lyngby, Denmark, Sept. 1973.

[Sacks-Davis (80)]
R. Sacks-Davis, "Fixed Leading Coefficient Implementation of SD-Formulas for Stiff ODEs," ACM Trans. Math. Software, 6 (1980), pp. 540-562.

[Schryer (77)]
N. L. Schryer, "Numerical Solution of Time-Varying Partial Differential Equations in One Space Variable," Bell Laboratories Computer Science Technical Report No. 53, Bell Laboratories, Murray Hill, NJ (1977).

[Scott & Watts (76)]
M. R. Scott and H. A. Watts, "A Systematized Collection of Codes for Solving Two-Point Boundary-Value Problems," in Numerical Methods for Differential Systems, L. Lapidus and W. E. Schiesser, Eds., Academic Press, 1976, pp. 197-227.

[Seward et al. (84)]
W. L. Seward, G. Fairweather, and R. L. Johnston, "A Survey of Higher-order Methods for the Numerical Integration of Semidiscrete Parabolic Problems," IMA J. Numer. Anal., 4 (1984), pp. 375-425.

[Shampine (81)]
L. F. Shampine, "Type-Insensitive ODE Codes Based on Implicit A-Stable Methods", Math. Comp., 36 (1981), pp. 499-510

[Shampine (82)]
L. F. Shampine, Global Error Estimation for Stiff ODEs, Sandia National Laboratory report SAND 82-2517 (Dec. 1982).

[Shampine (82a)]
L. F. Shampine, "Implementation of Rosenbrock Methods," ACM Trans. Math. Software, 8 (1982), pp. 93-113.

[Shampine (82b)]
L. F. Shampine, "Type-Insensitive Codes Based on A(α)-Stable Formulas," Math. Comp., 39 (1982), pp. 109-123.

[Shampine (85)]
L. F. Shampine, "Measuring Stiffness," Appl. Numer. Math., 1 (1985), pp. 107-119.

[Shampine & Gear (79)]
L. F. Shampine and C. W. Gear, "A User's View of Solving Stiff Ordinary Differential Equations," SIAM Rev., 21 (1979), pp. 1-17.

[Shampine & Gordon (75)]
L. F. Shampine and M. K. Gordon, "Computer Solution of Ordinary Differential Equations. The Initial Value Problem," W. H. Freeman and Co., San Francisco, 1975.

[Shampine & Watts (80)]
L. F. Shampine and H. A. Watts, "DEPAC--Design of a User Oriented Package of ODE Solvers," Sandia National Laboratories Report SAND79-2374, Albuquerque, NM (Sept. 1980).

[Sherman & Hindmarsh (80)]
A. H. Sherman and A. C. Hindmarsh, "GEARS: A Package for the Solution of Sparse, Stiff Ordinary Differential Equations," in Electrical Power Problems: The Mathematical Challenge, SIAM, Philadelphia, 1980, pp. 190-200.

[Sincovec & Madsen (75)]
R. F. Sincovec and N. K. Madsen, "Software for Nonlinear Partial Differential Equations," ACM Trans. Math. Software, 1 (1975), pp. 232-260.

[Sincovec, et al. (81)]
R. F. Sincovec, B. Dembart, M. A. Epton, A. M. Erisman, S. W. Manke, and E. L. Yip, "Solvability of Large Scale Descriptor Systems," IEEE Trans. on Auto. Control, 26 (1981), pp. 139-147.

[Skeel & Kong (77)]
R. D. Skeel and A. K. Kong, "Blended Linear Multistep Methods," ACM Trans. Math. Software, 3 (1977), pp. 326-345.

[Strang & Fix (73)]
G. Strang and G. J. Fix, An Analysis of the Finite Element Method, Prentice-Hall, Englewood Cliffs, NJ, 1973.

[Swartz & Wendroff (69)]
B. Swartz and B. Wendroff, "Generalized Finite Difference Schemes," Math. Comp., 23 (1969), pp.37-49.

[Tendler, et al. (78a)]
J. M. Tendler, T. A. Bickart, and Z. Picel, "A Stiffly Stable Integration Process Using Cyclic Composite Methods," ACM Trans. Math. Soft., 4 (1978), pp. 339-368.

[Tendler, et al. (78b)]
J. M. Tendler, T. A. Bickart, and Z. Picel, "Algorithm 534, STINT: STiff (Differential Equations) INTegrator [D2]," ACM Trans. Math. Software, 4 (1978), pp. 399-403.

[Thompson & Tuttle (82)]
S. Thompson and P. G. Tuttle, "Automatic ODE Software for Use in an Industrial Environment," Proceedings of the 10th IMACS World Congress on System Simulation and Scientific Computation, 1982, pp. 440-442.

[Tischer & Gupta (84)]
P. E. Tischer and G. K. Gupta, "Considerations in Designing a Cyclic Method Stiff ODE Solver," Monash University Computer Science Dept. report, Clayton, Victoria, Australia (1984).

[Tischer & Sacks-Davis (83)]
R. Tischer and R. Sacks-Davis, "A New Class of Cyclic Multistep Formulae for Stiff Systems," SIAM J. Sci. Stat. Comput., 4 (1983), pp. 733-747.

[Varga (62)]
R. S. Varga, Matrix Iterative Analysis, Prentice-Hall, Englewood Cliffs, NJ, 1962, Ch. 6.

[Watanabe & Sheikh (84)]
D. S. Watanabe and Q. M. Sheikh, "One-Leg Formulas for Stiff Ordinary Differential Equations," SIAM J. Sci. Stat. Comput., 5 (1984), pp. 489-496.

[Watkins & Hansonsmith (83)]
D. S. Watkins and R. W. Hansonsmith, "The Numerical Solutions of Separably Stiff Systems by Precise Partitioning," ACM Trans. Math. Software, 9 (1983), pp. 293-301.

[Watts & Shampine (72)]
H. A. Watts and L. F. Shampine, "A-Stable Block Implicit One-Step Methods," BIT, 12 (1972), pp. 252-266.

APPENDIX

HOW TO OBTAIN STIFF ODE CODES

Here we have focused on ODE software that is readily available. We simply list known sources for such software. The ODE solvers in the National Energy Software Center (NESC) are known by name and number.

Source: National Energy Software Center

Argonne National Laboratory
9700 South Cass Avenue
Argonne, IL 60439

Code	Catalog Number
DASSL	9918
EPISODE	675
EPISODEB	705
EPISODEIB	Not issued at this time
FORSIM	514
GEARBI	Not issued at this time
LSODE	592
LSODI	9939
LSODA	9937
LSODAR	9936
LSODES	9938
LSOIBT	9832
SLATEC Library (DEBDF)	820
STFODE/COLODE	652
TORANAGA	Not issued at this time

We strongly recommend the following. If you want to obtain a code from NESC, contact your in-house NESC representative. If you do not have an NESC representative, telephone the NESC to determine prices, procedures, and timing. Note that the SLATEC library can be obtained directly from NESC.

The IMSL library is often available on the in-house mainframe computer. We recommend that you contact your in-house user representative, if you are interested in using IMSL software. For further information about this library, write or telephone:

IMSL
Sixth Floor, NBC Building
7500 Bellaire Boulevard
Houston, TX 77036-5085

The NAG library may be on your in-house mainframe. Elements may also be available to you for use on a personal computer. Again, see your in-house user support staff. You may obtain further information about the NAG library from

NAG Inc.
1131 Warren Avenue
Downers Grove, Illinois 60515
USA

or

NAG Central Office
Mayfield House
256 Banbury Road
Oxford OX2 7DE
United Kingdom

It is possible that some experimental codes are available from their authors for trial use and beta testing. Some of the papers cited include statements of availability.

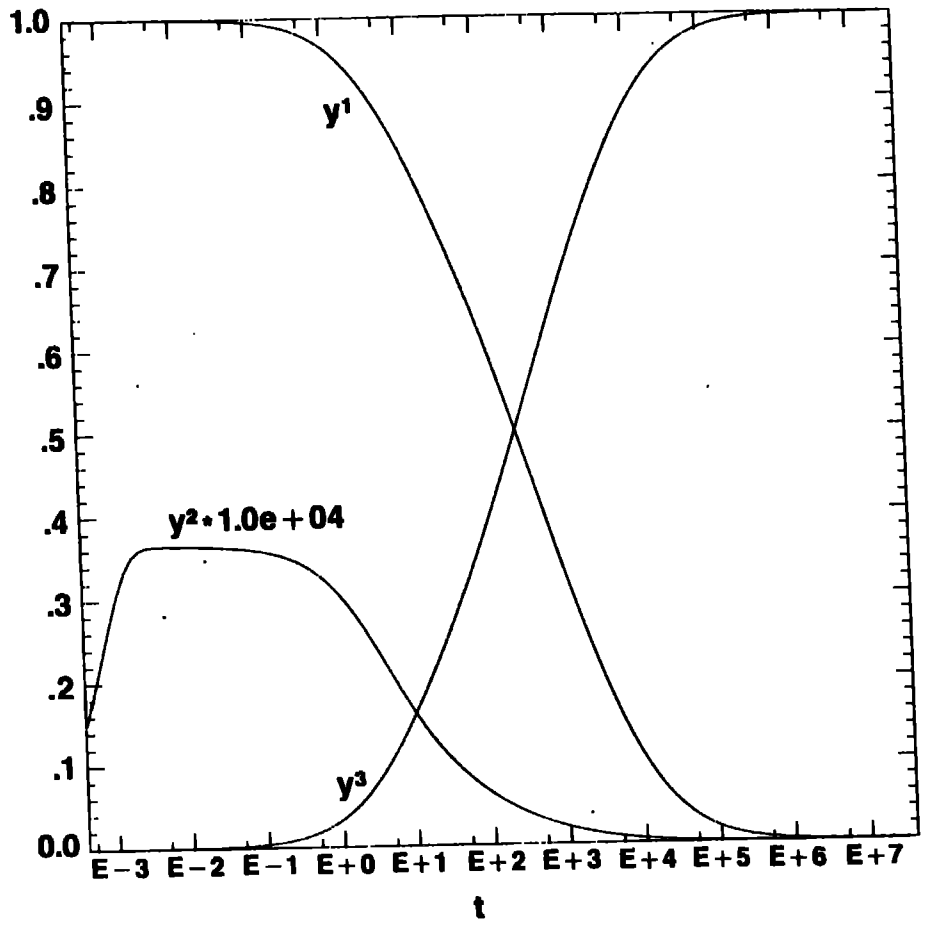


Figure 4.1. Robertson's Problem

0186-38C-055

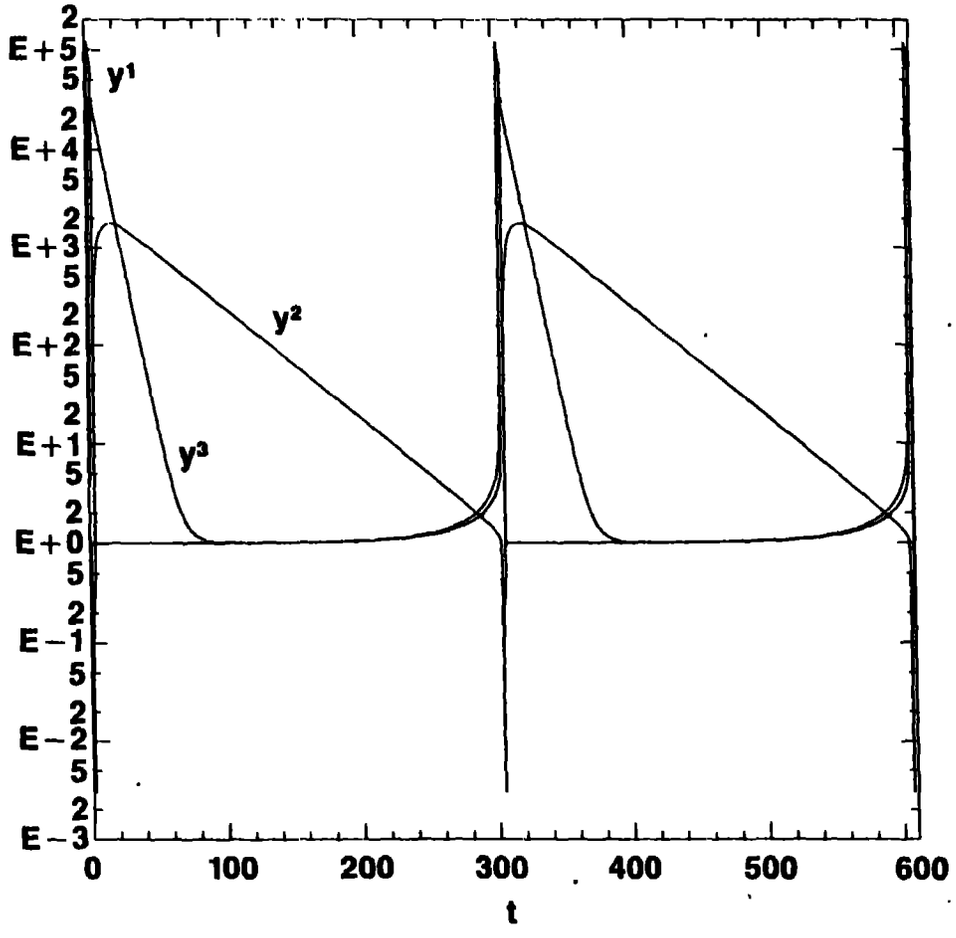


Figure 4.2. The Field-Noyes Chemical Oscillator - Logarithmic y Axis

0186-38C-056

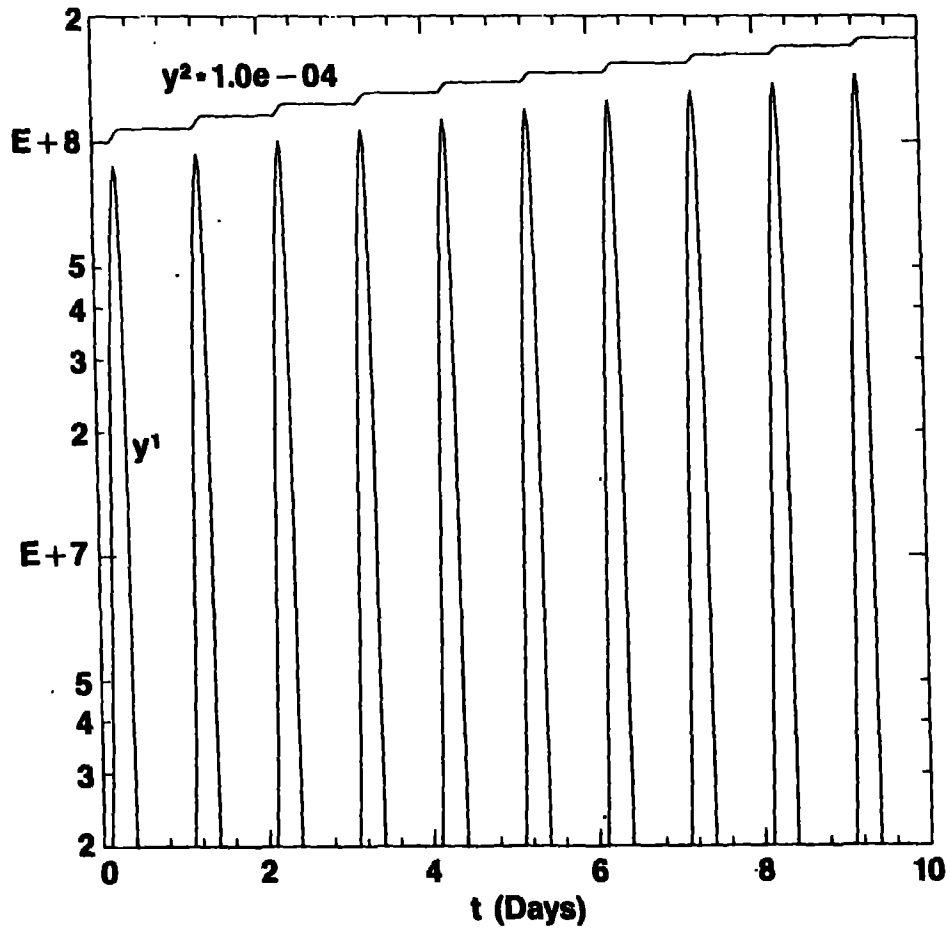


Figure 4.3. Two Species Diurnal Kinetics - Truncated, Logarithmic y-Axis

0186-38C-057

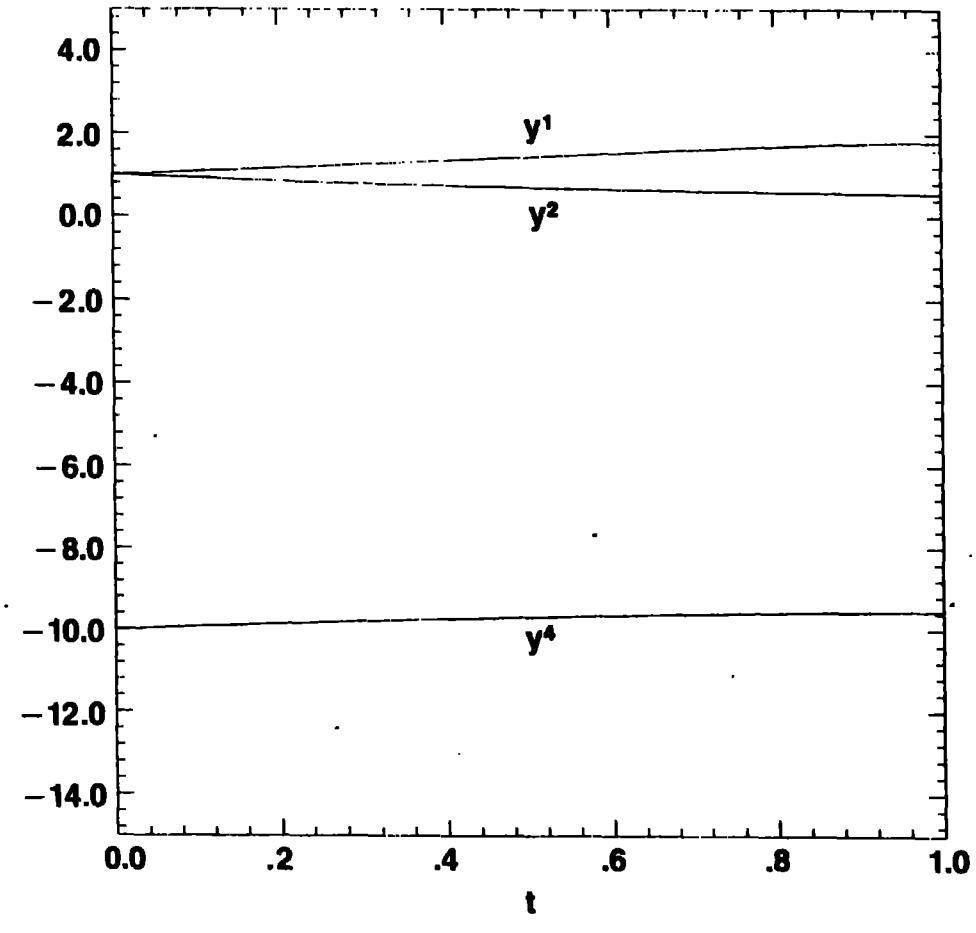


Figure 4.4a. A Kidney Model - $y_o^5 = 0.990268835$

0186-38C-058

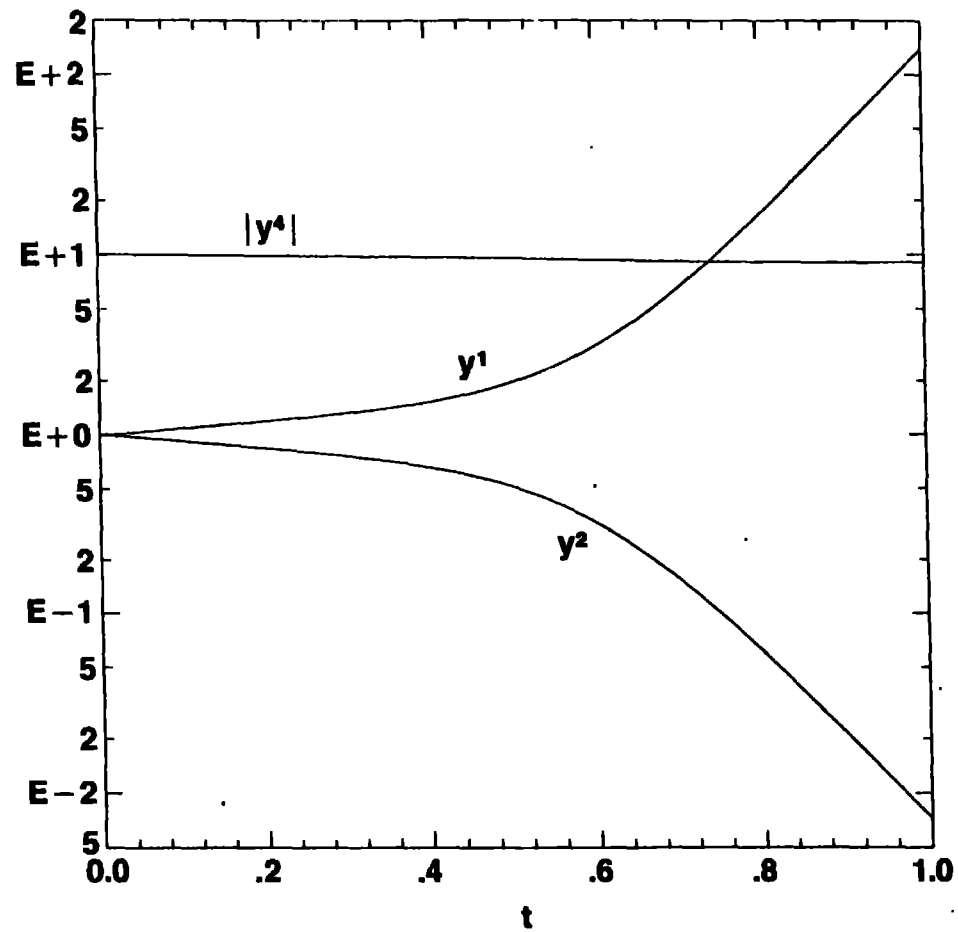


Figure 4.4b. A Kidney Model - Logarithmic y Axis, $y_0^5 = 0.99$

0186-38C-059

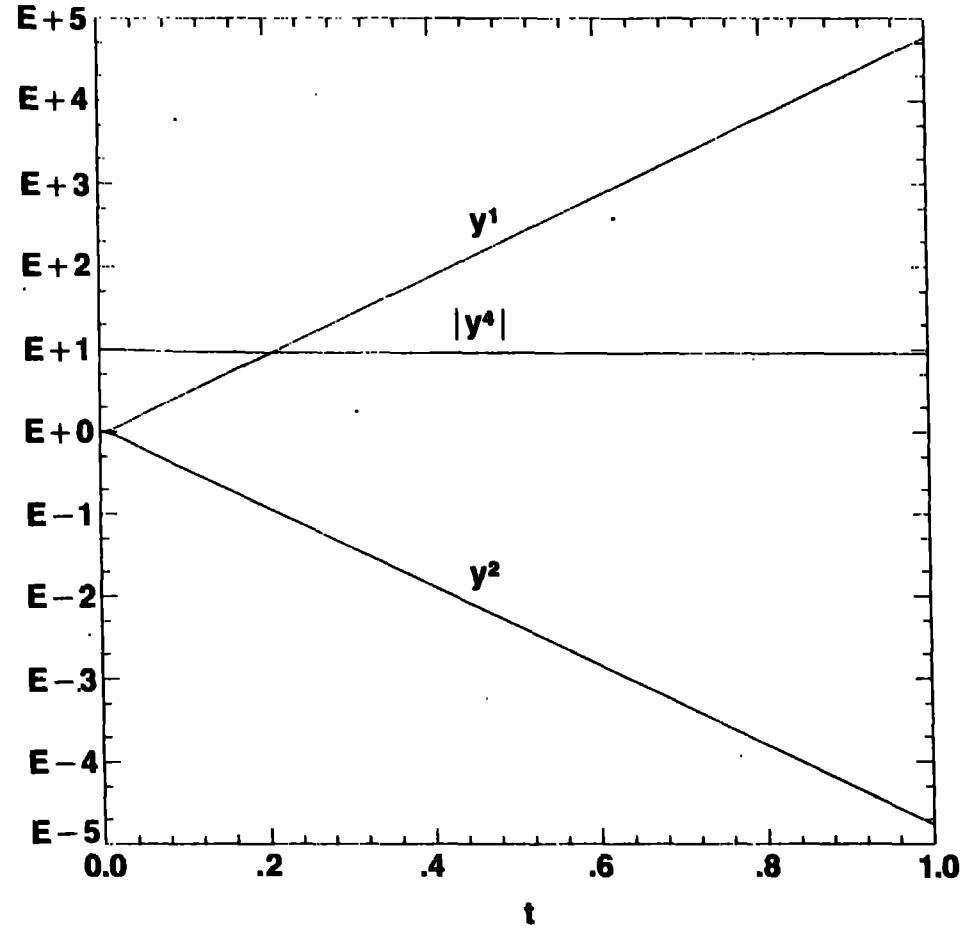


Figure 4.4c. A Kidney Model - Logarithmic y Axis, $y_o^5 = 0.9$

0186-38C-060

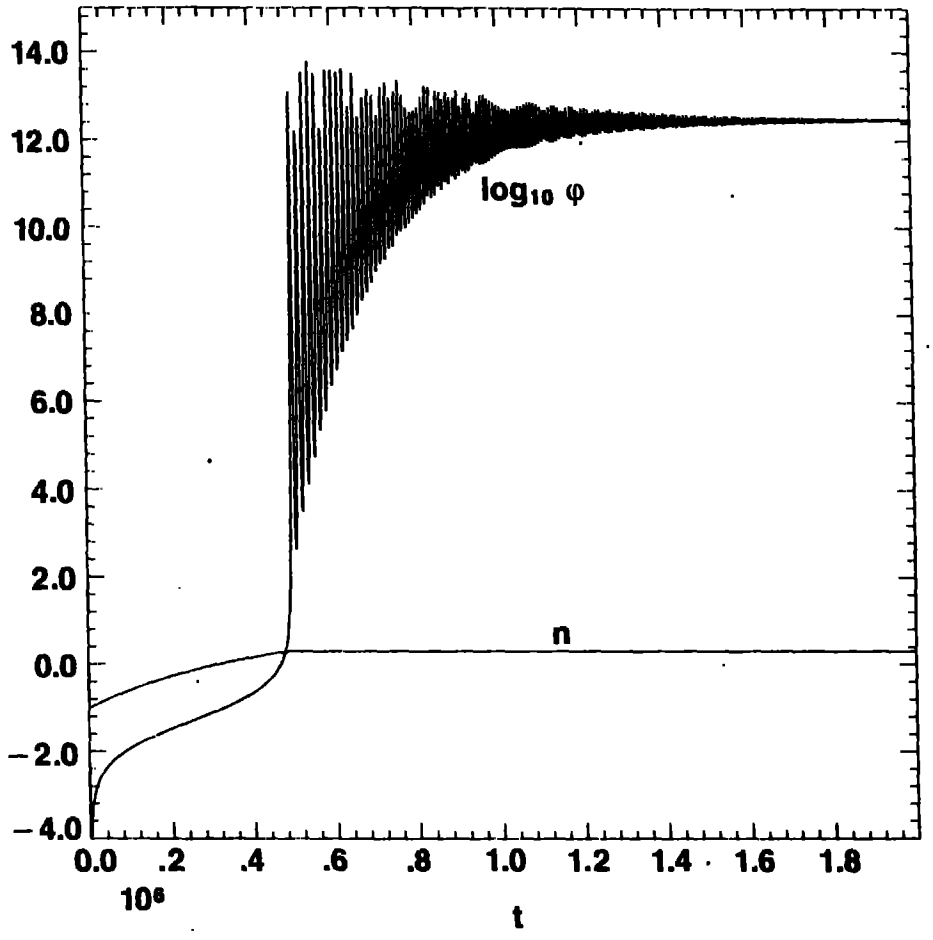


Figure 4.5a. A Laser Oscillator Model

0186-38C-061

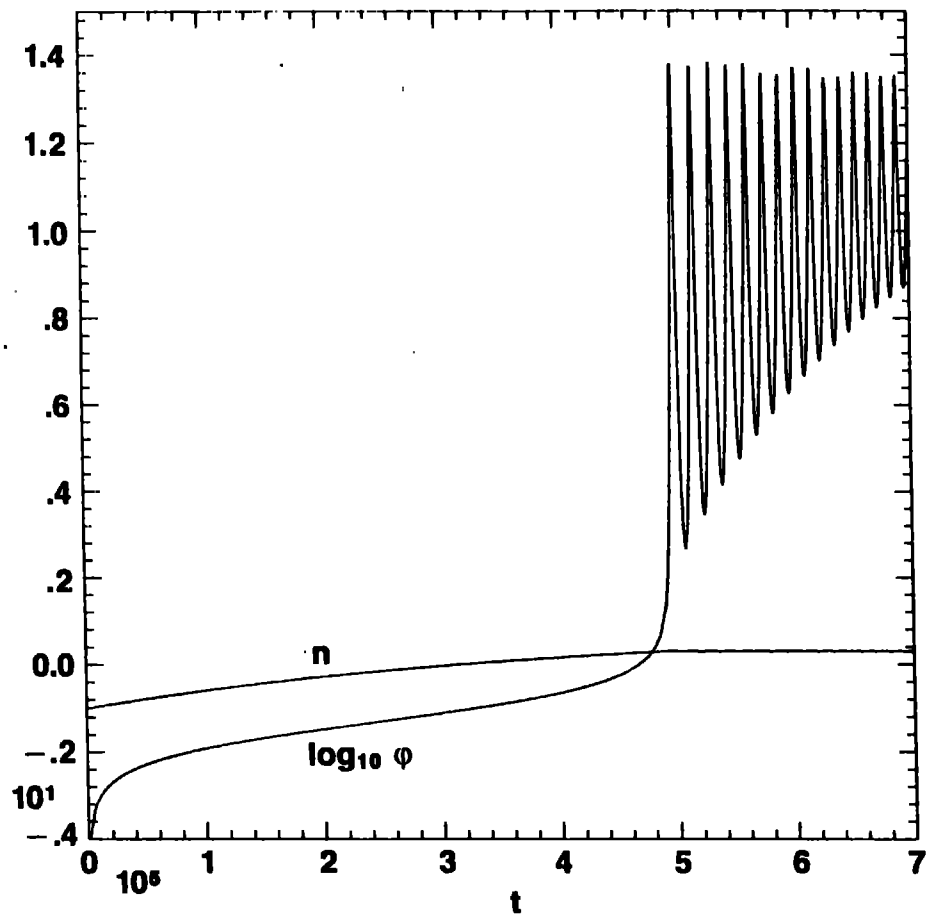


Figure 4.5b. A Laser Oscillator Model

0186-38C-062

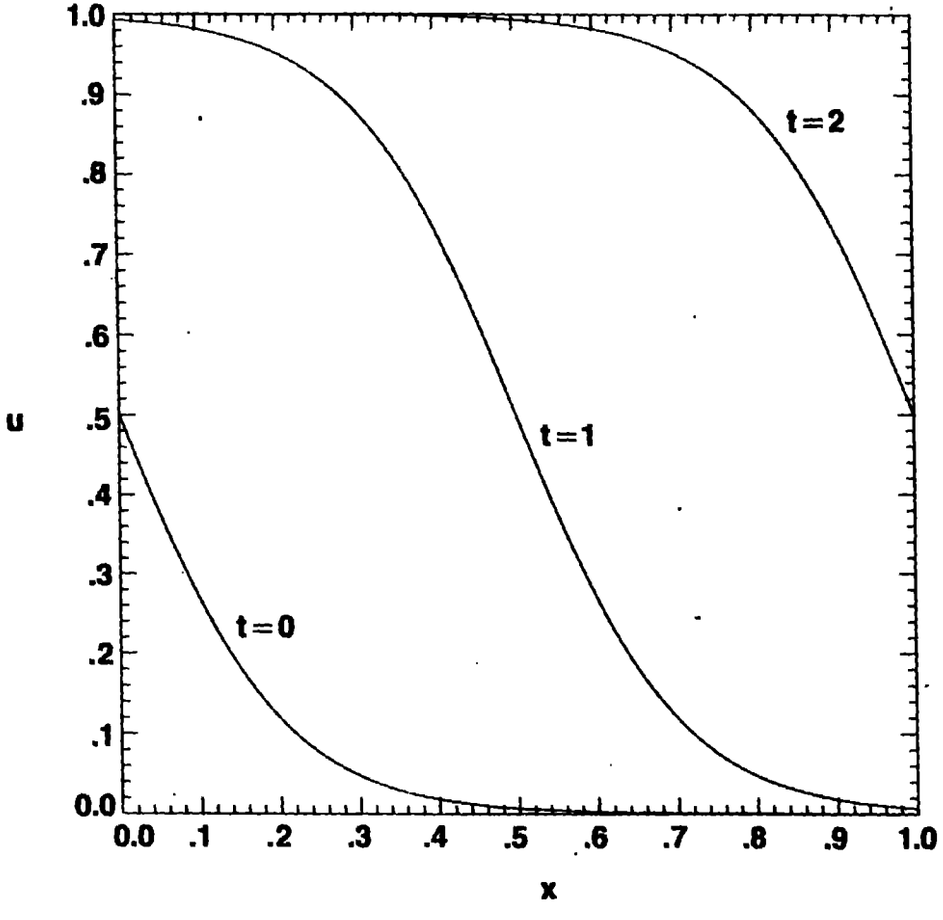


Figure 4.6a. Burgers' Equation - Finite Differences, 102 Grid Points - u vs. x at Various t

0186-38C-063

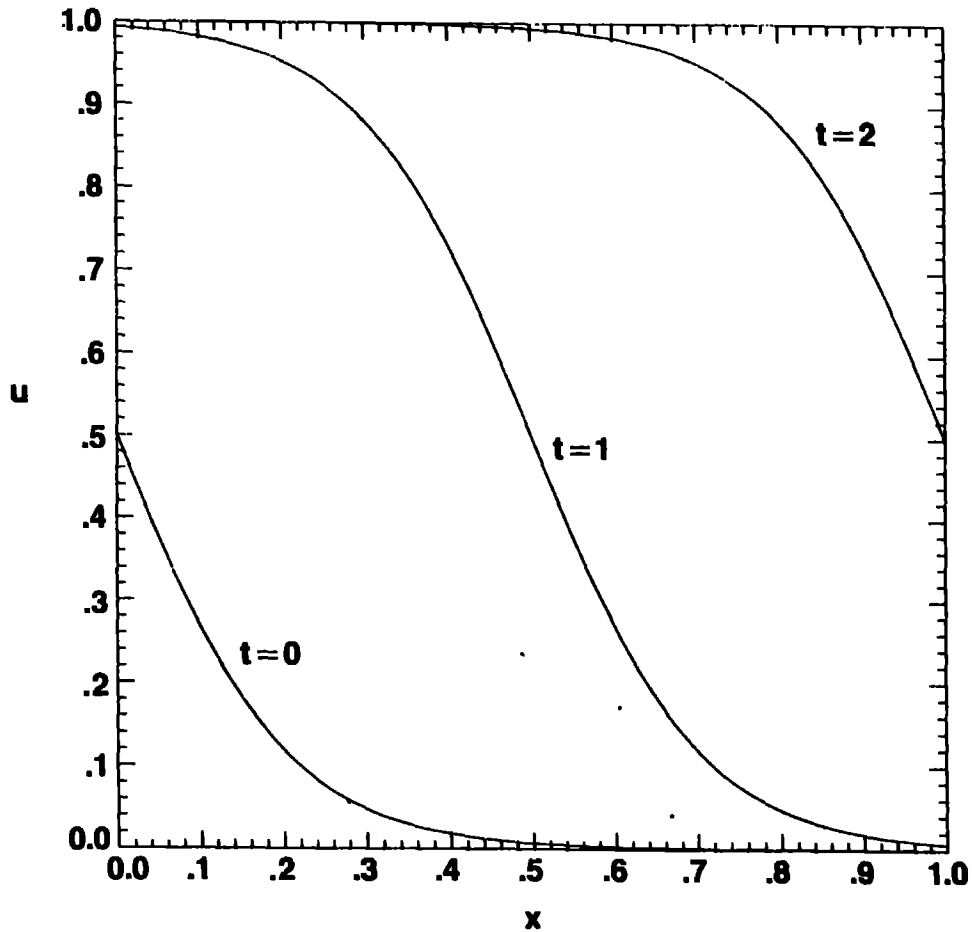


Figure 4.6b. Burgers' Equation - Simplified Galerkin Procedure, 102 Grid Points - u vs. x at Various t

0186-38C-064

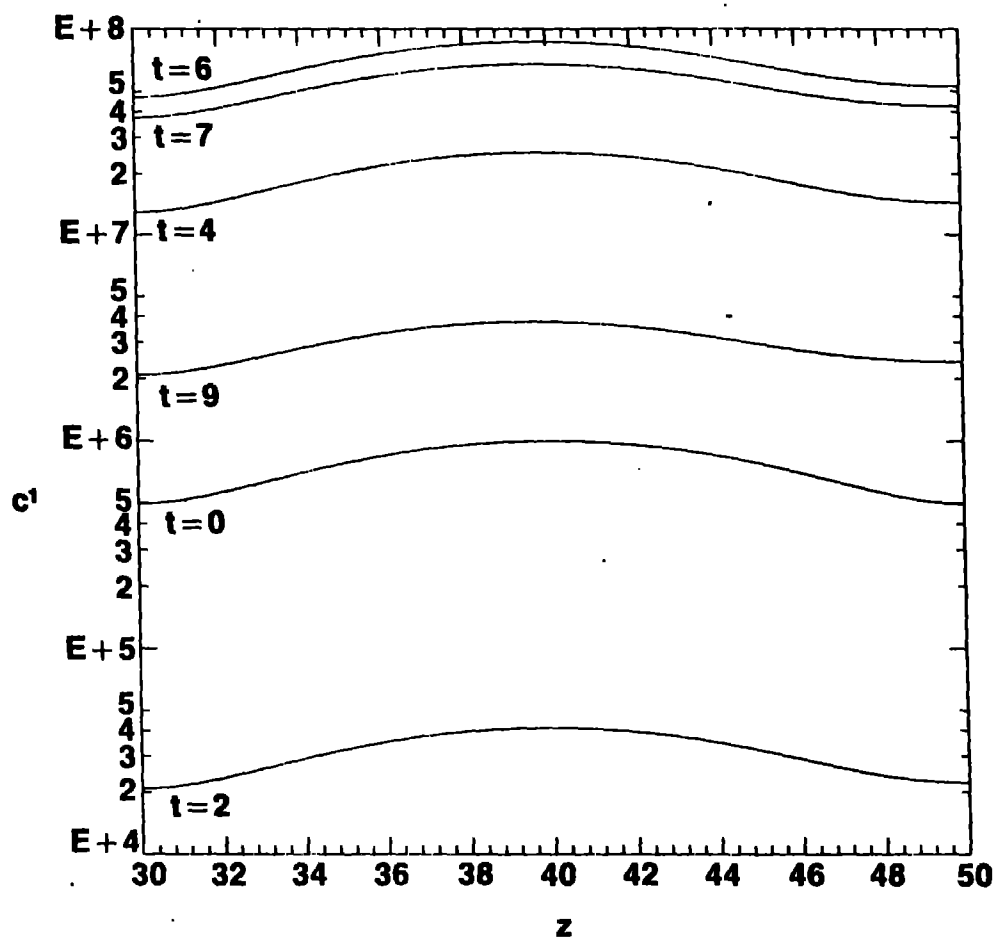


Figure 4.7a. Two Species Diffusion - Diurnal Kinetics - One Dimensional, Logarithmic c^1 Axis, t in Hours, c^1 vs. z at Various t

0186-38C-065

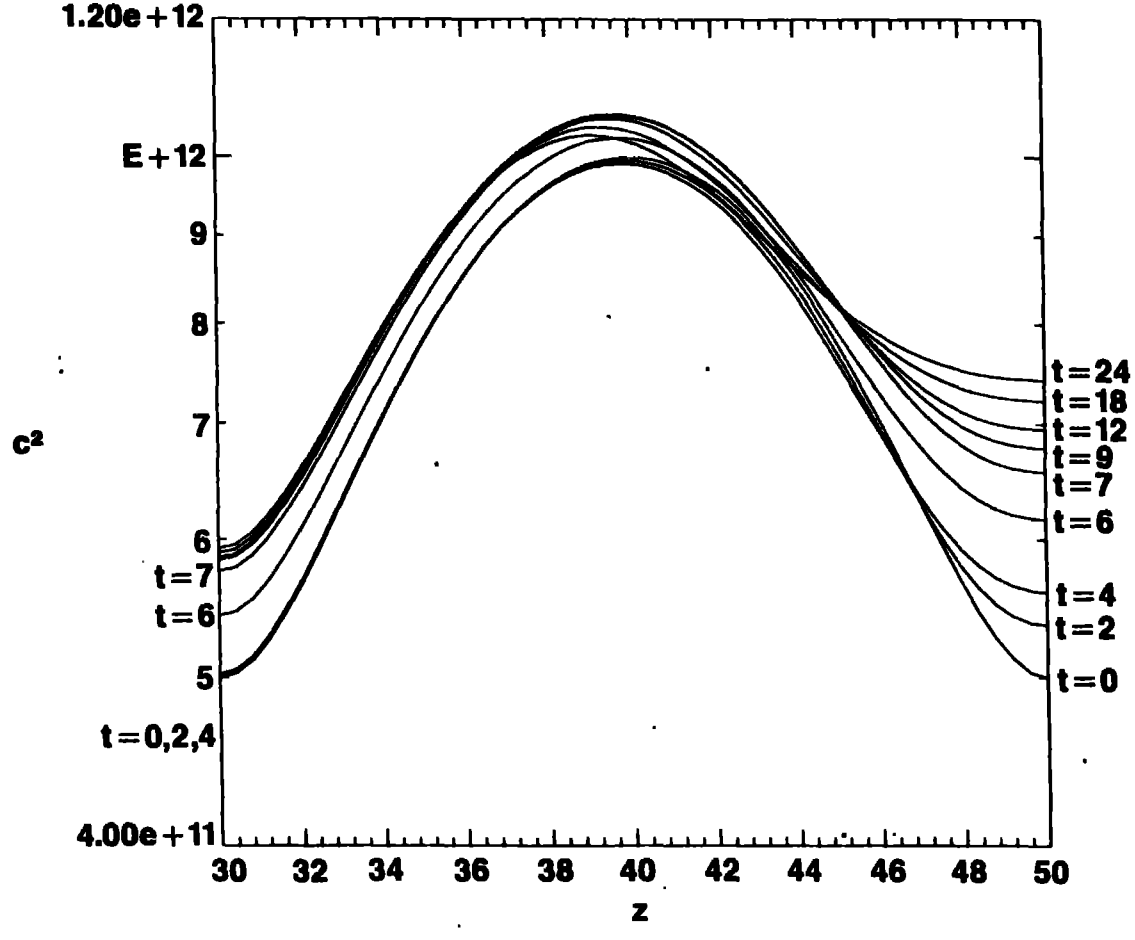


Figure 4.7b. Two Species Diffusion - Diurnal Kinetics - One Dimensional, Logarithmic c^2 Axis, c^2 vs. z at Various t

0186-38C-066

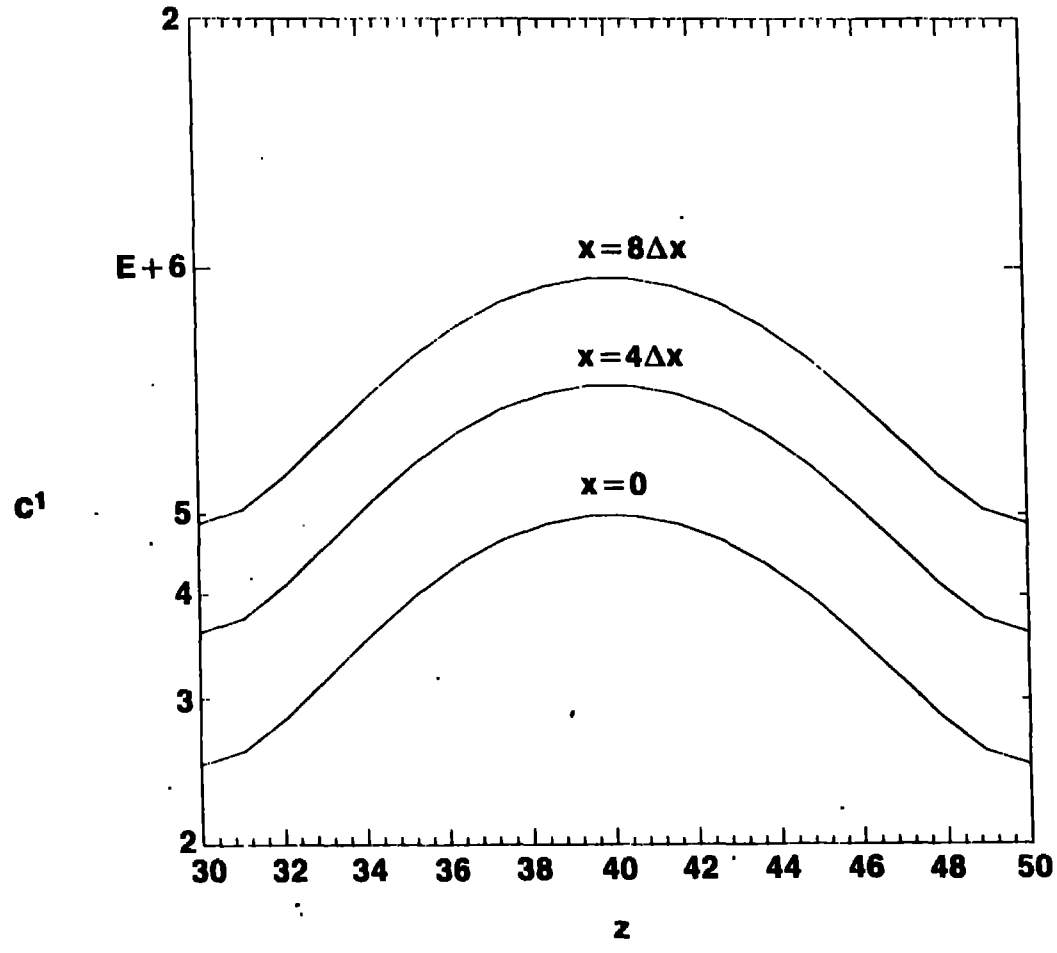


Figure 4.8a. Two Species Diffusion - Diurnal Kinetics - Two Dimensional, Logarithmic c^1 Axis, c^1 vs. z at Various x, t=0 Hrs.

0186-38C-067

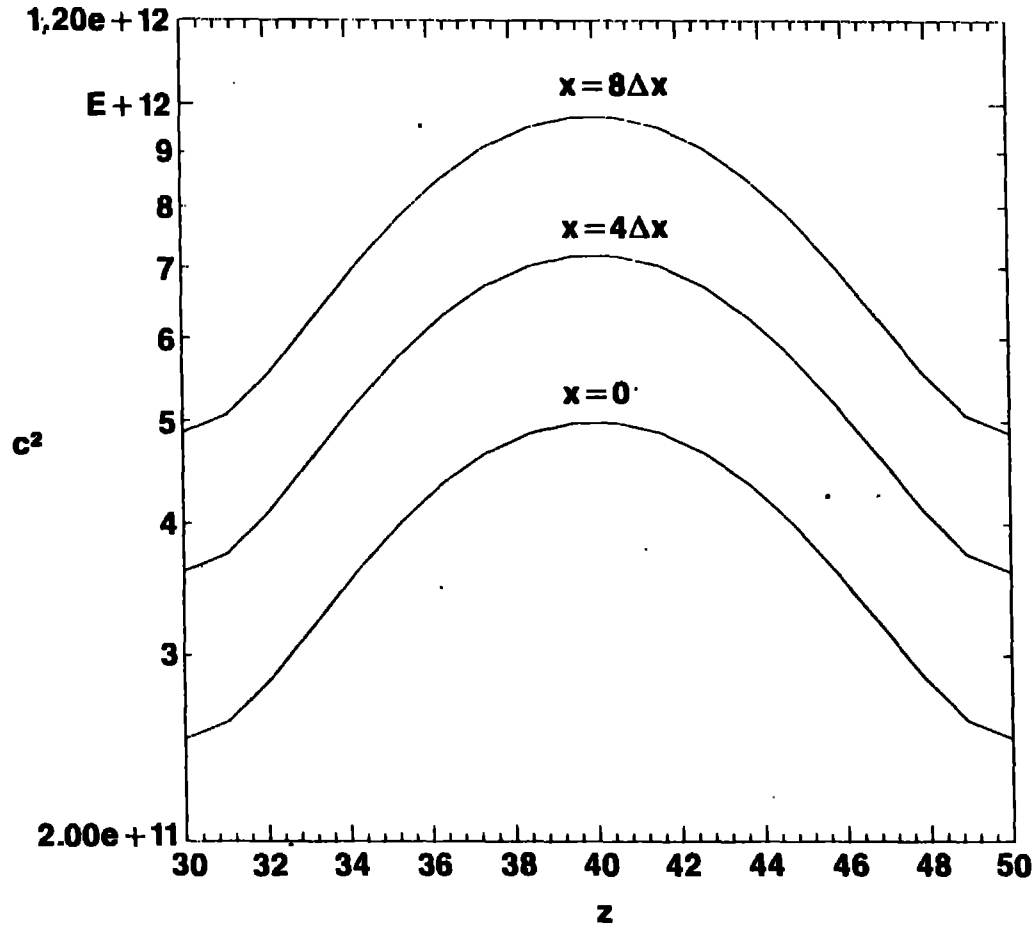


Figure 4.8b. Two Species Diffusion - Diurnal Kinetics - Two Dimensional, Logarithmic c^2 Axis, c^2 vs. z at Various x, t = 0 Hrs.

0186-38C-068

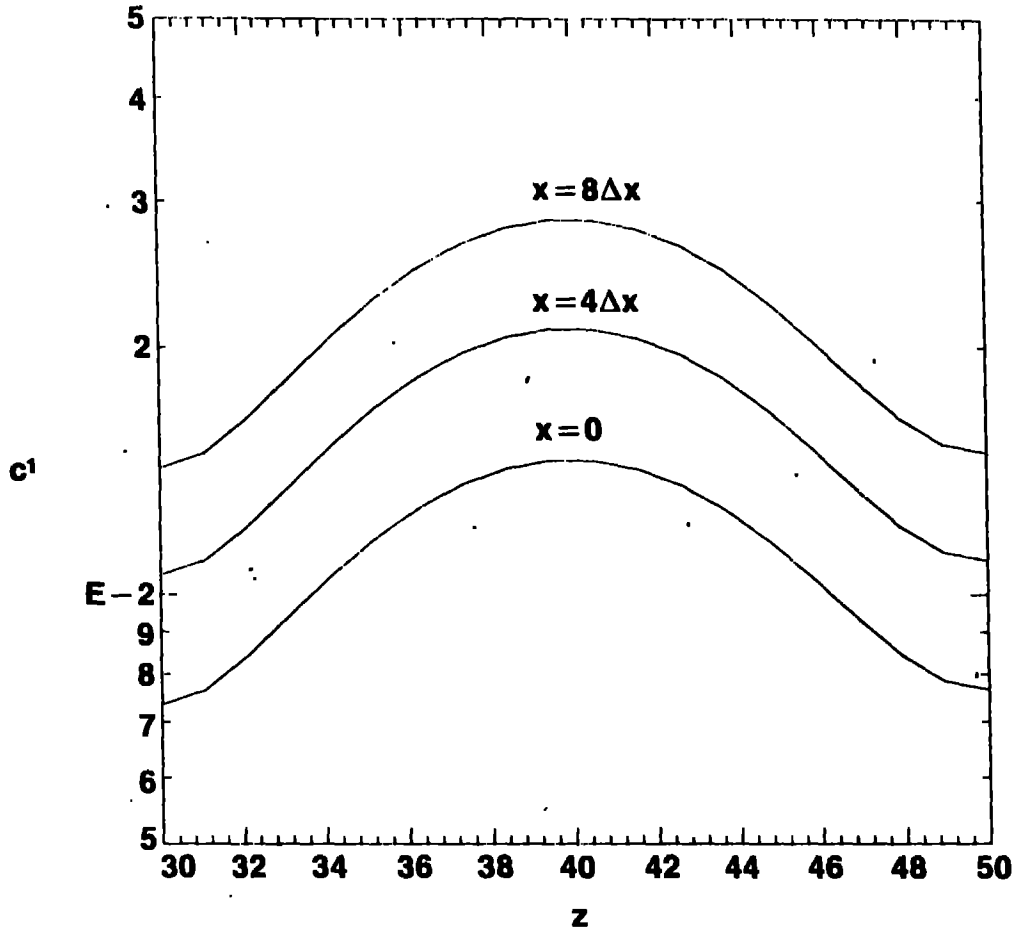


Figure 4.8c. Two Species Diffusion - Diurnal Kinetics - Two Dimensional, Logarithmic c^1 Axis, c^1 vs. z at Various x, t=1 Hr.

0186-38C-069

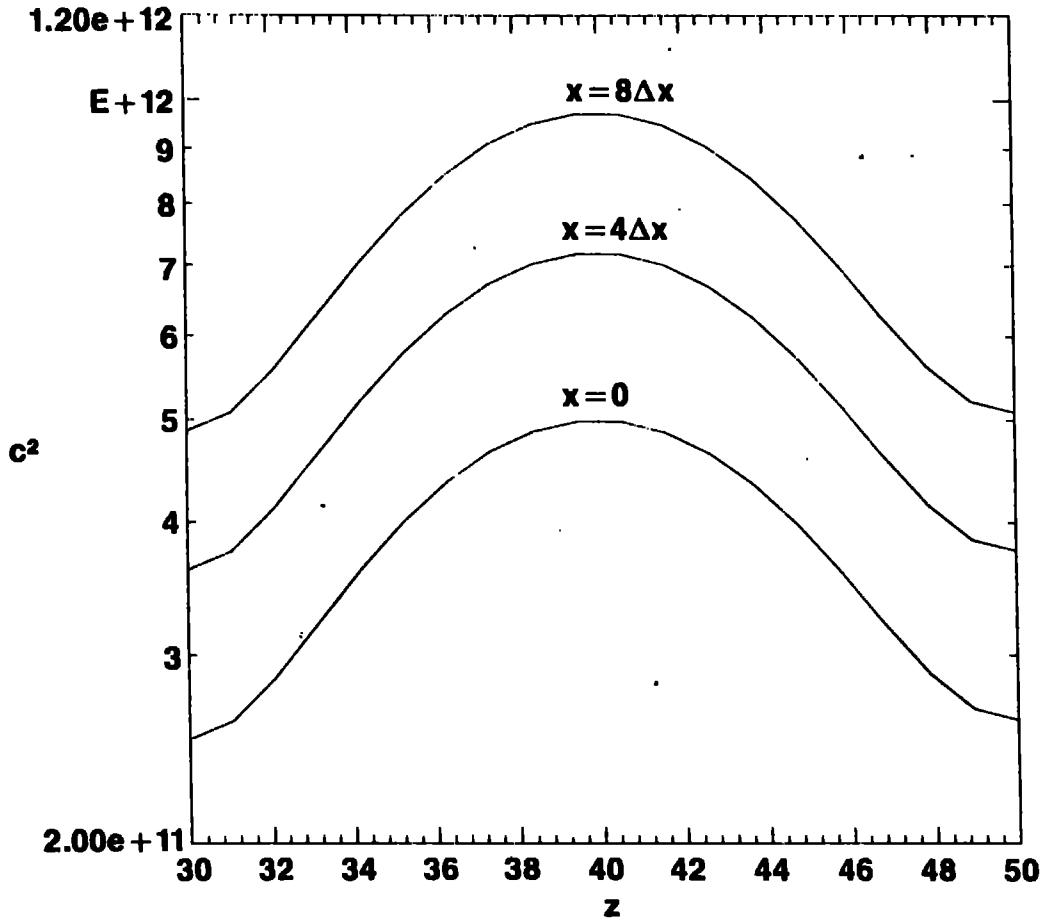


Figure 4.8d. Two Species Diffusion - Diurnal Kinetics - Two Dimensional, Logarithmic c^2 Axis, c^2 vs. z for Various x, t=1 Hr.

0186-38C-070

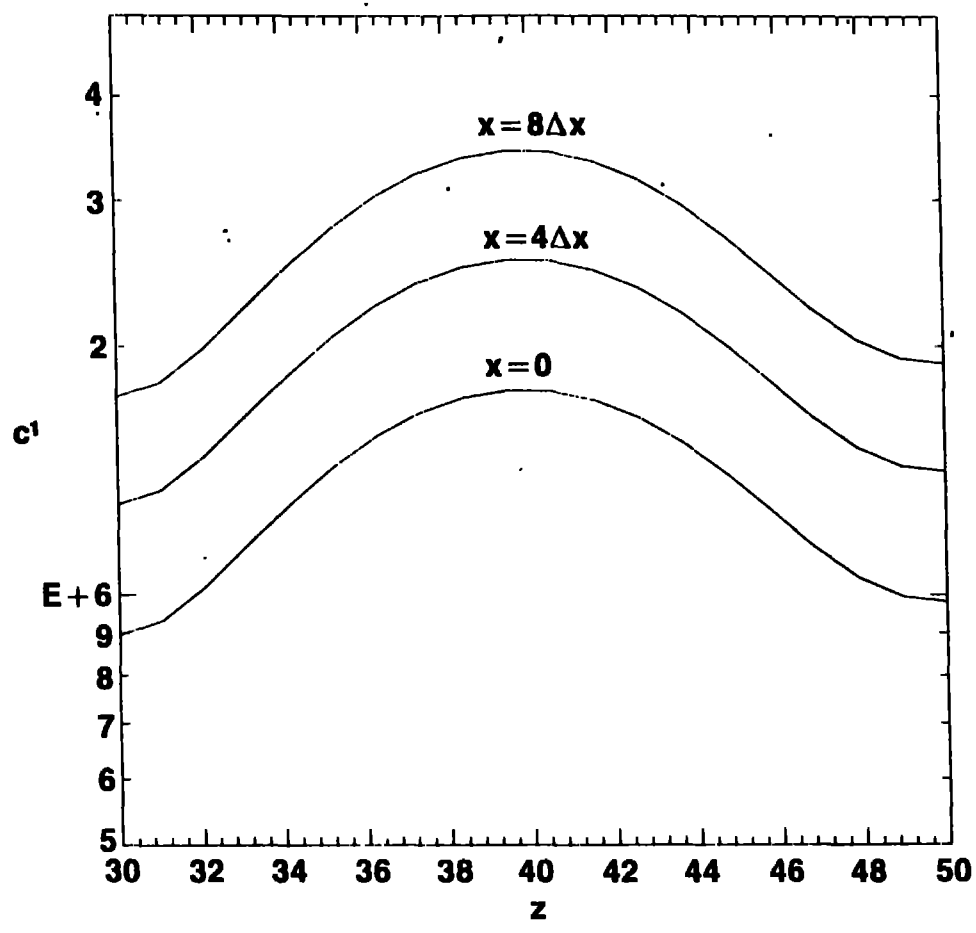


Figure 4.8e. Two Species Diffusion - Diurnal Kinetics - Two Dimensional, Logarithmic c^1 Axis, c^1 vs. z for Various x, t=3 Hrs.

0186-38C-071

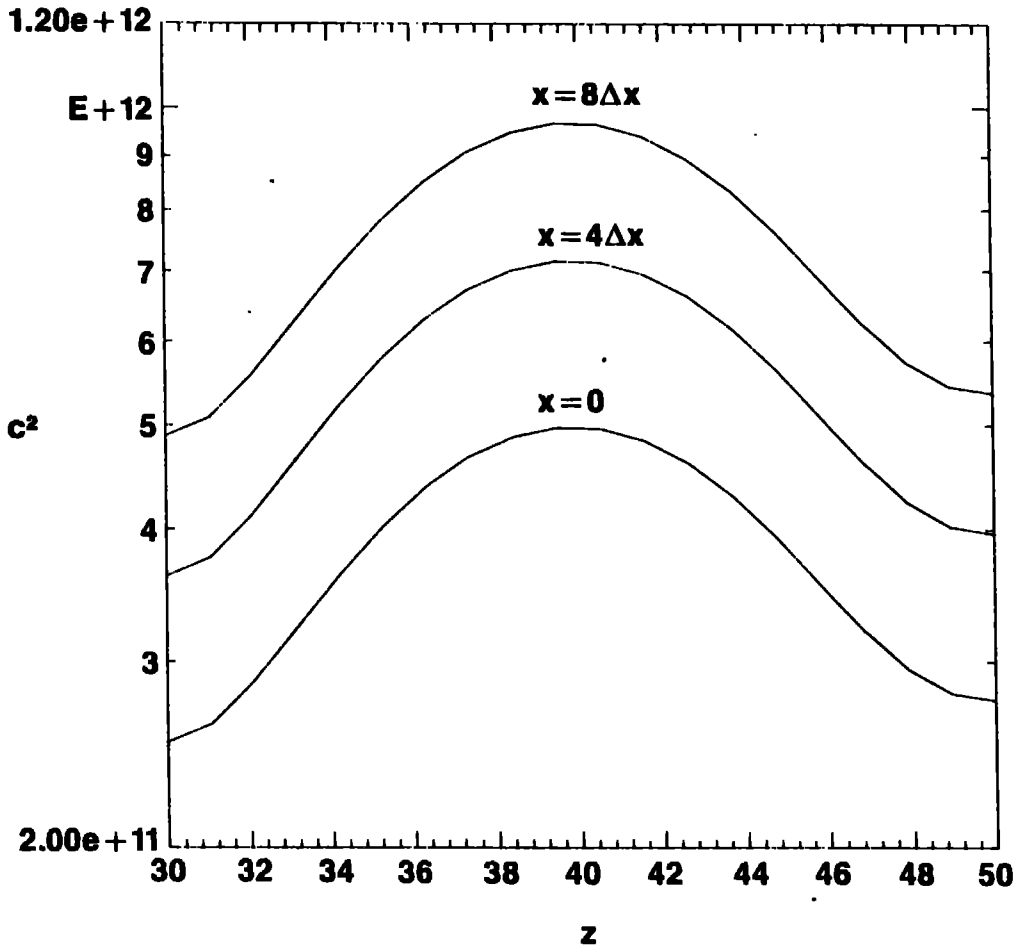


Figure 4.8f. Two Species Diffusion - Diurnal Kinetics - Two Dimensional, Logarithmic c^2 Axis, c^2 vs. z for Various x, t = 3 Hrs.

0186-38C-072

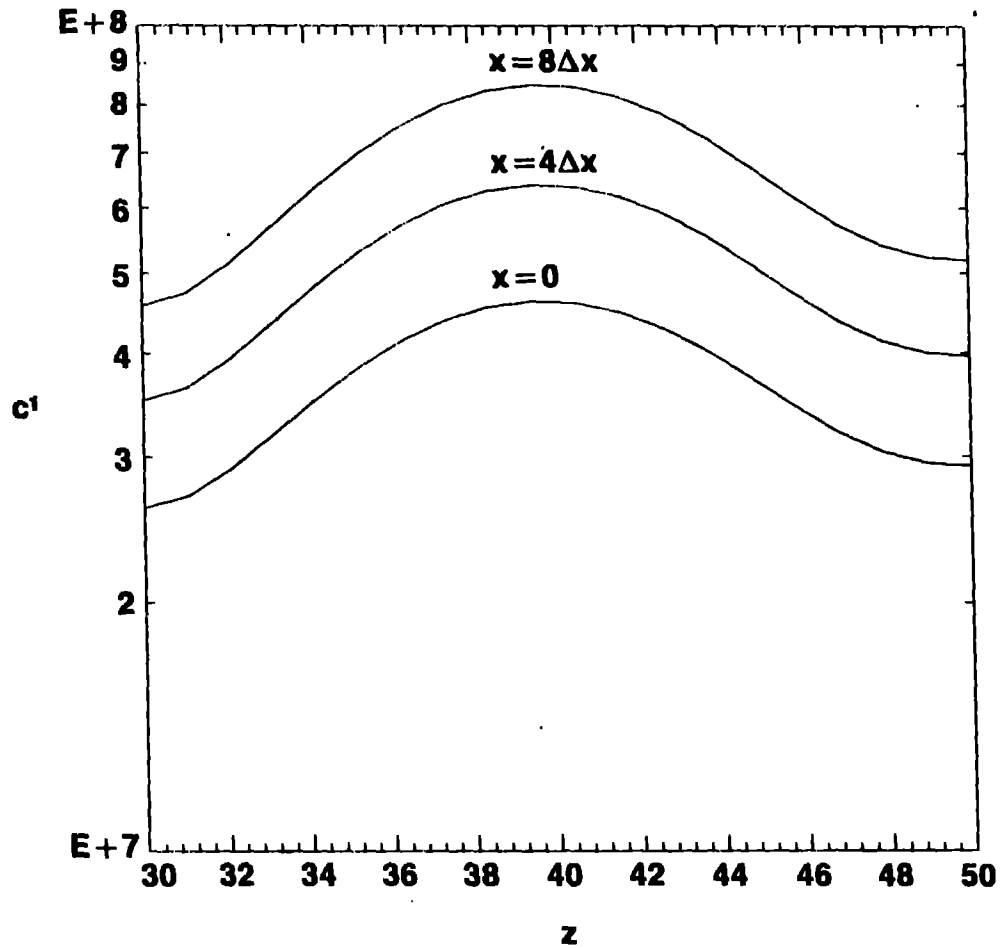


Figure 4.8g. Two Species Diffusion - Diurnal Kinetics - Two Dimensional, Logarithmic c^1 Axis, c^1 vs. z for Various x, t=6 Hrs.

0186-38C-073

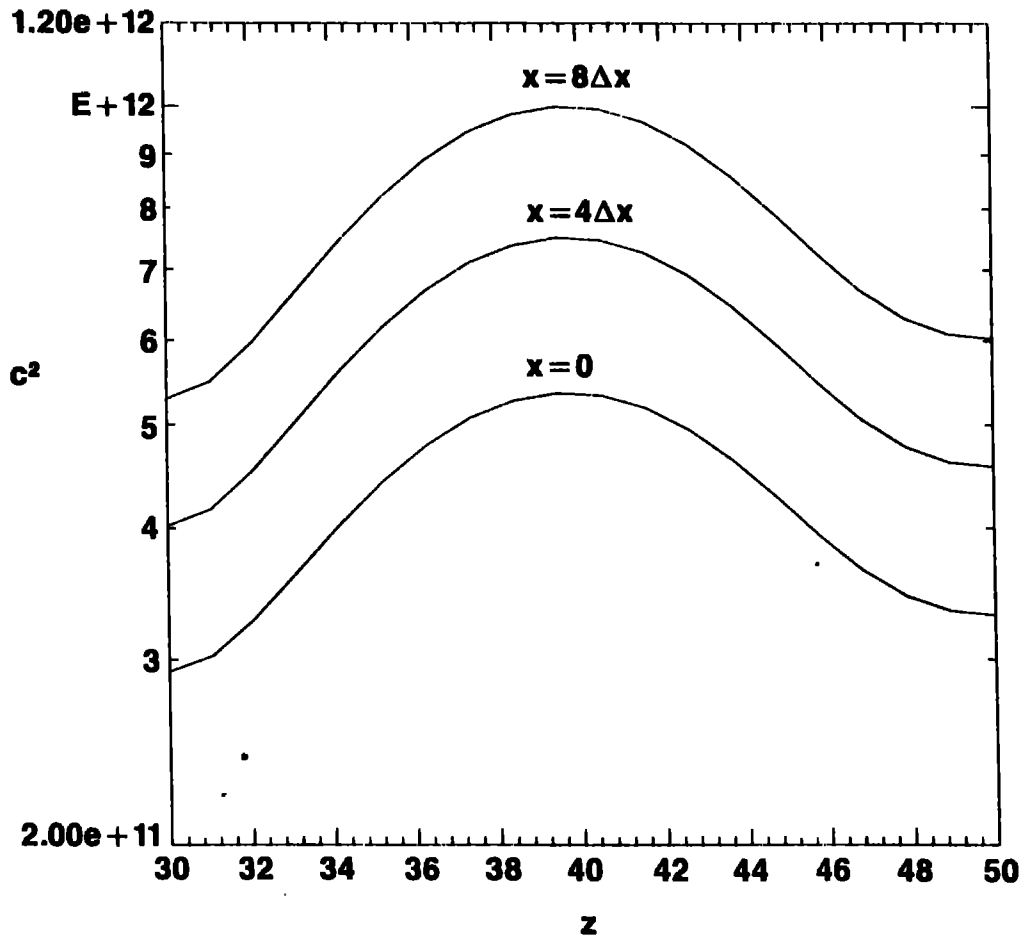


Figure 4.8h. Two Species Diffusion - Diurnal Kinetics - Two Dimensional, Logarithmic c^2 Axis, c^2 vs. z for Various x, t=6 Hrs.

0186-38C-074

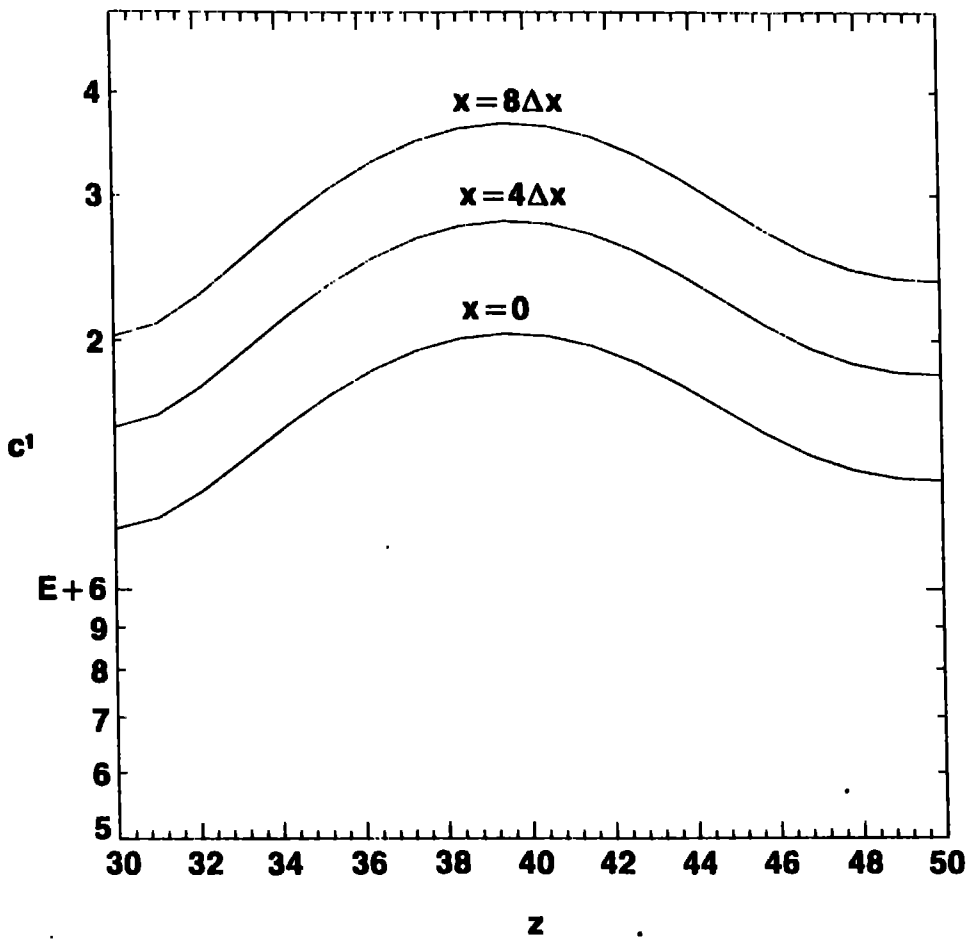


Figure 4.8i. Two Species Diffusion - Diurnal Kinetics - Two Dimensional, Logarithmic c^1 Axis, c^1 vs. z for Various x, t=9 Hrs.

0186-38C-075

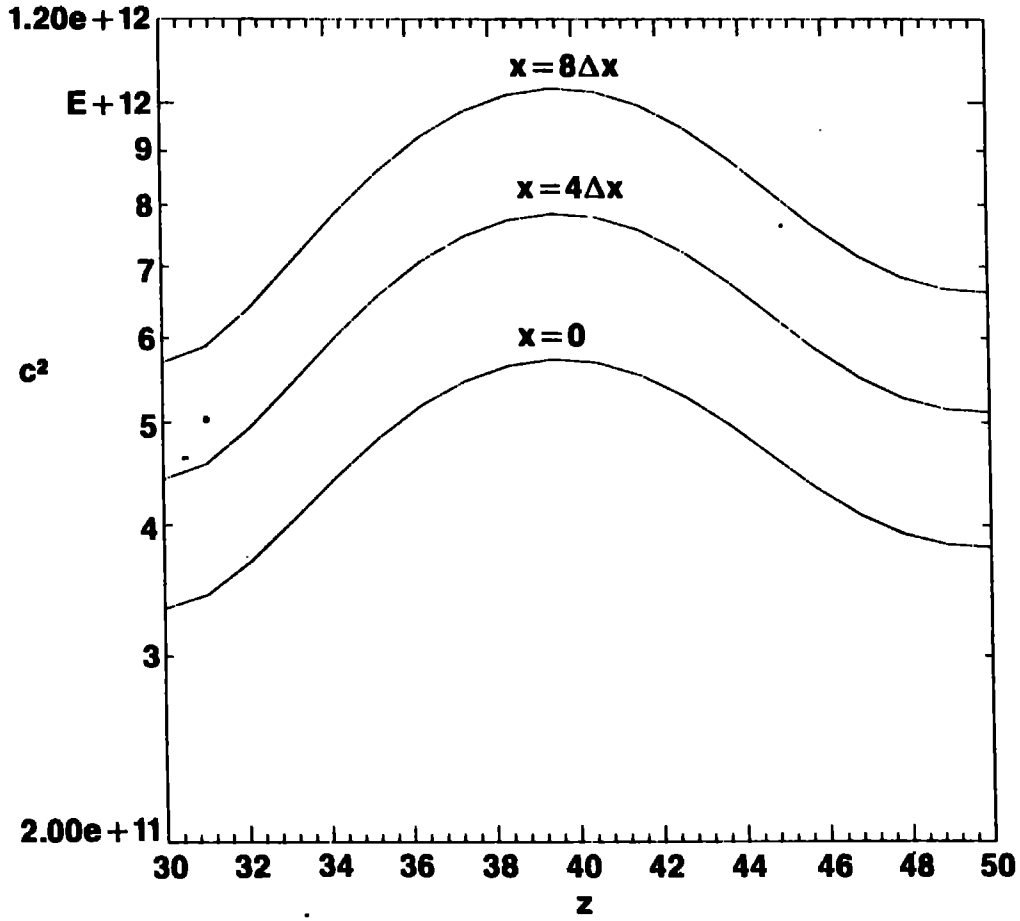


Figure 4.8j. Two Species Diffusion - Diurnal Kinetics - Two Dimensional, Logarithmic c^2 Axis, c^2 vs. z for Various x, t=9 Hrs.

0186-38C-076

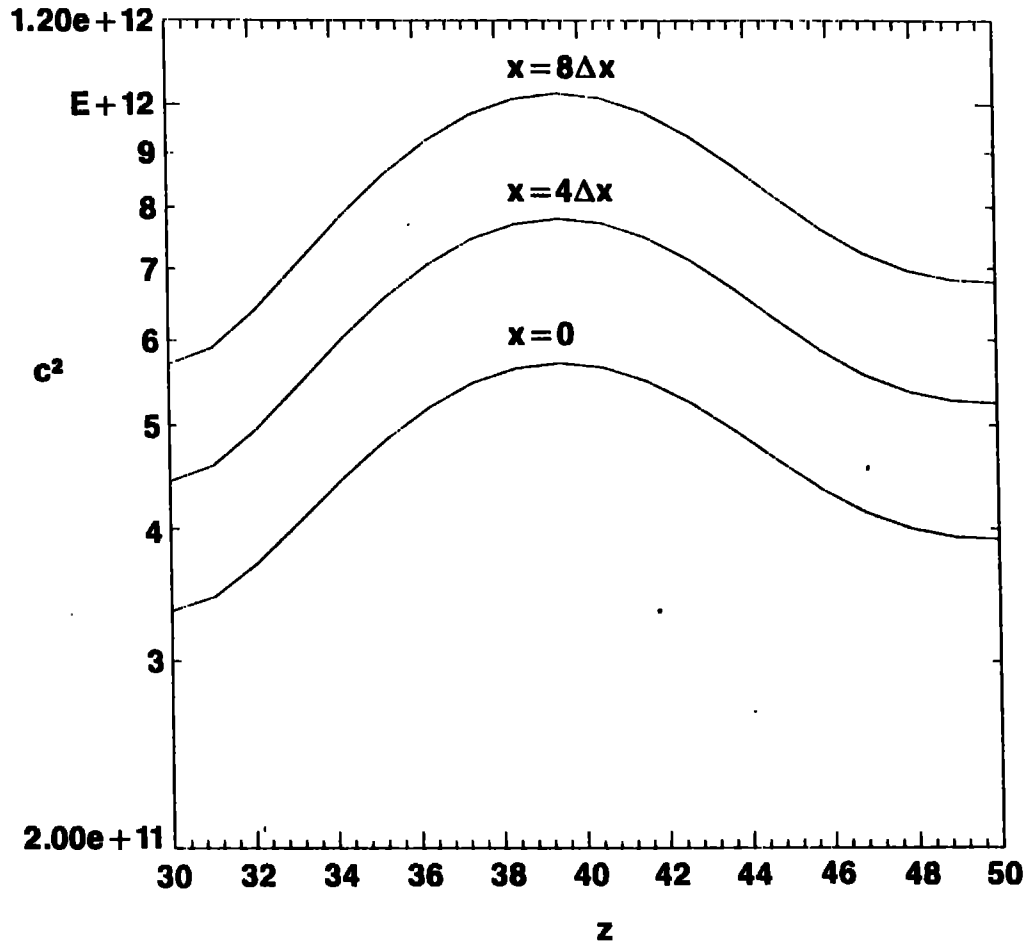


Figure 4.8k: Two Species Diffusion - Diurnal Kinetics - Two Dimensional, Logarithmic c^2 Axis, c^2 vs. z for Various x, t=12 Hrs.

0186-38C-077

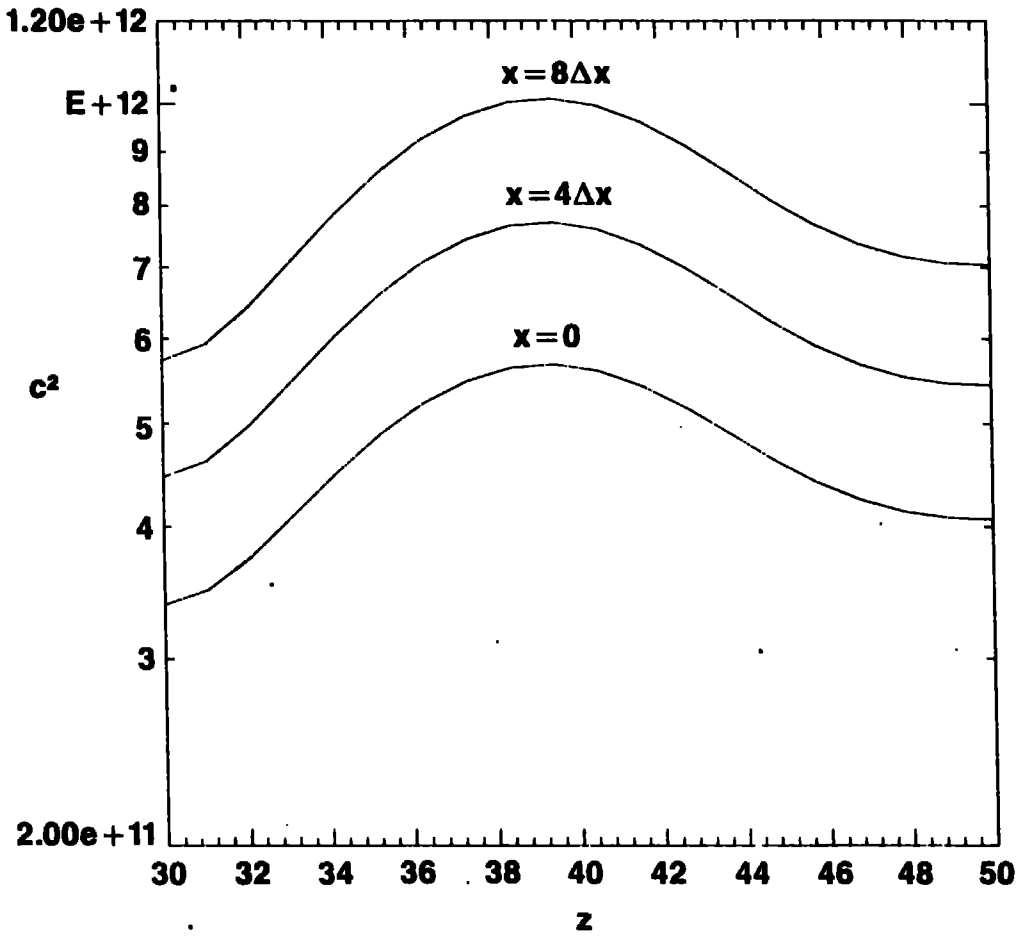


Figure 4,8l. Two Species Diffusion - Diurnal Kinetics - Two Dimensional, Logarithmic c^2 Axis, c^2 vs. z for Various x, t=18 Hrs.

0186-38C-078

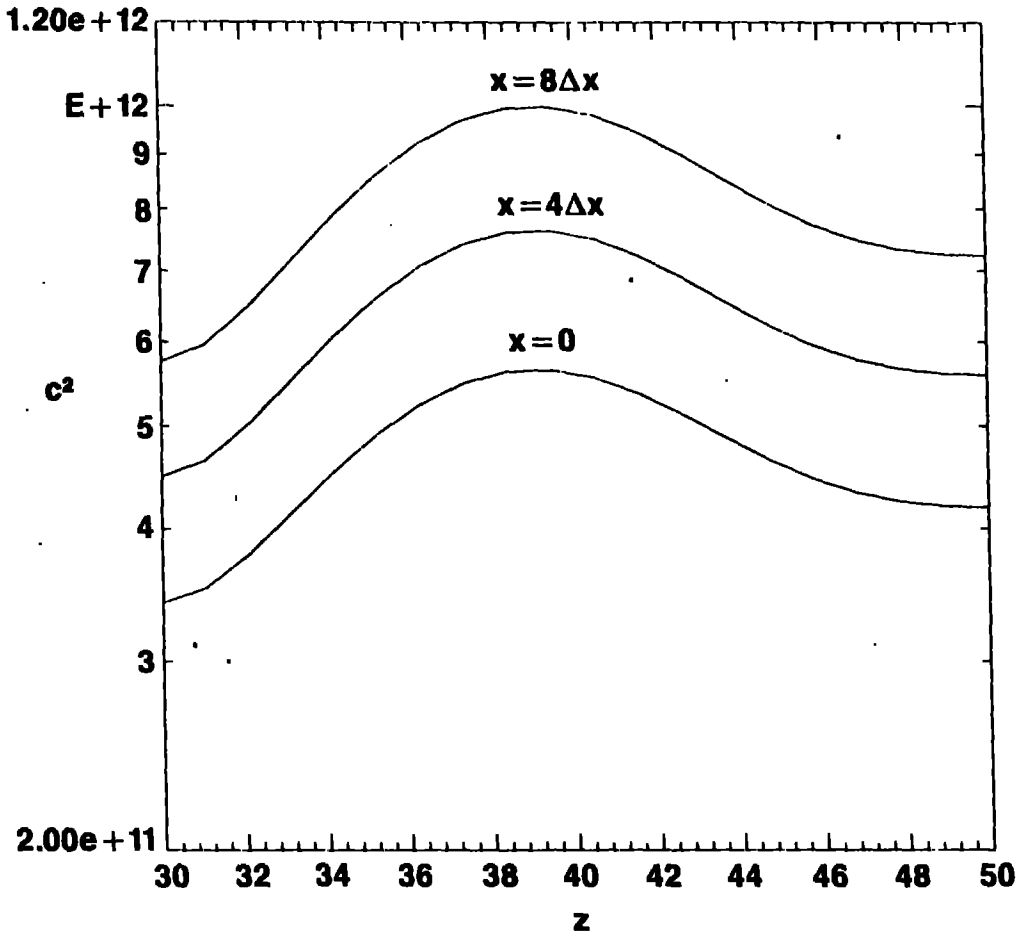


Figure 4.8m. Two Species Diffusion - Diurnal Kinetics - Two Dimensional, Logarithmic c^2 Axis, c^2 vs. z for Various x, t=24 Hrs.

0186-38C-079

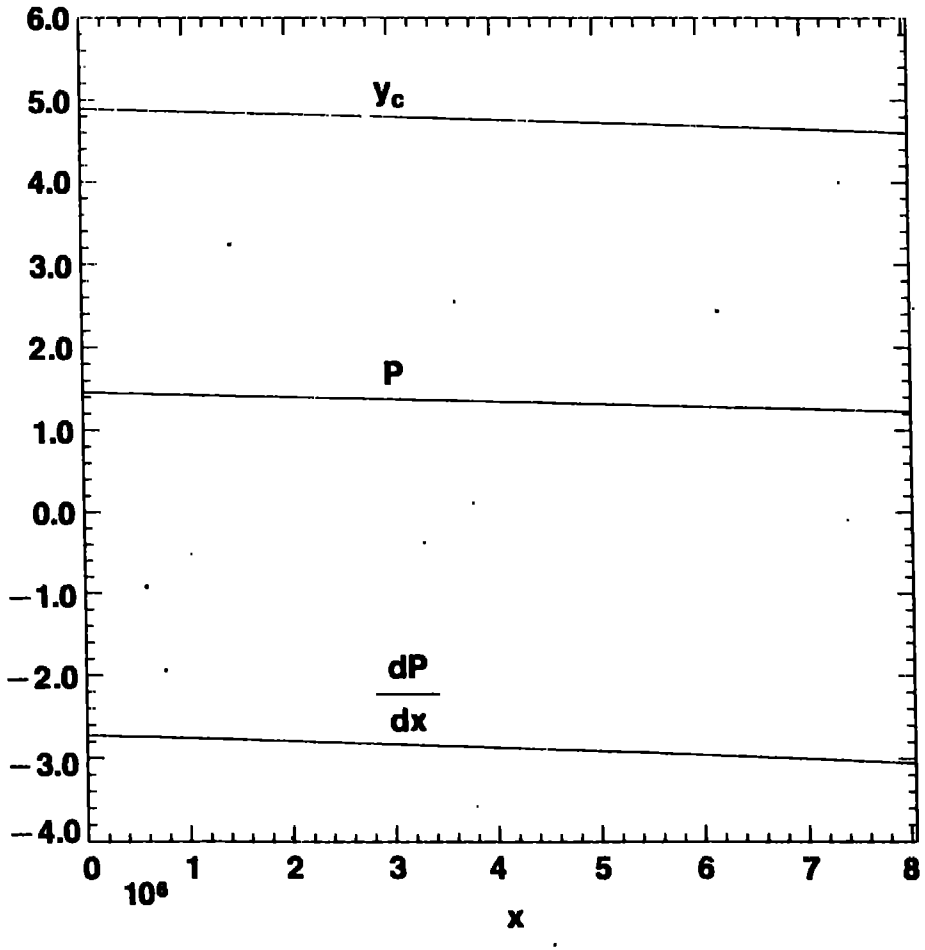


Figure 4.9a. A Two Phase Plug Flow Problem, Normal Flow

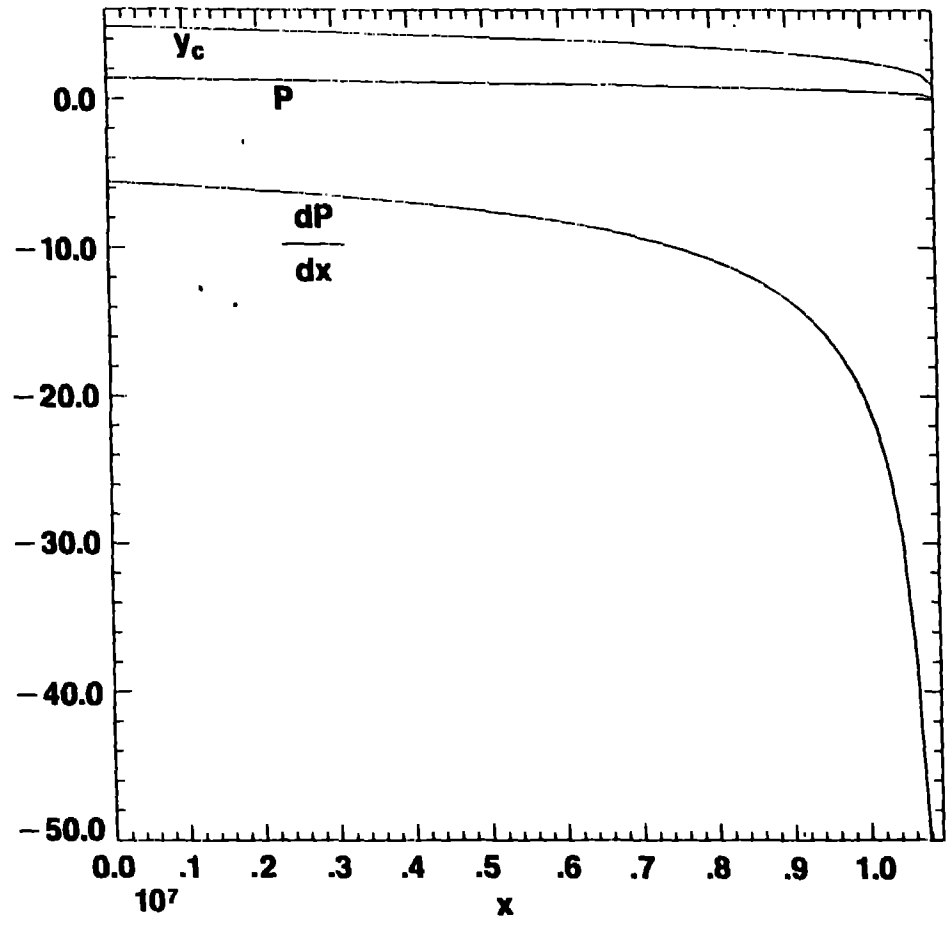


Figure 4.9b. A Two Phase Plug Flow Problem, Choking

0186-38C-081

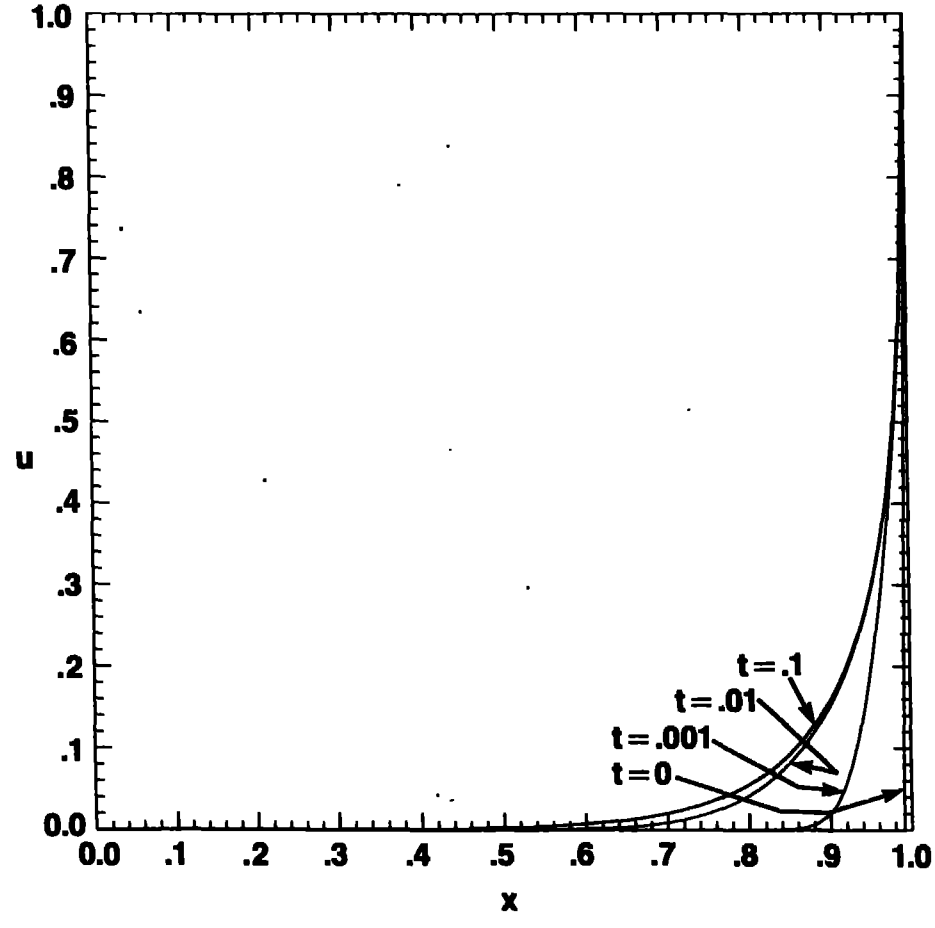


Figure 4.10a. Troesch's Two Point Boundary Value Problem

0186-38C-082

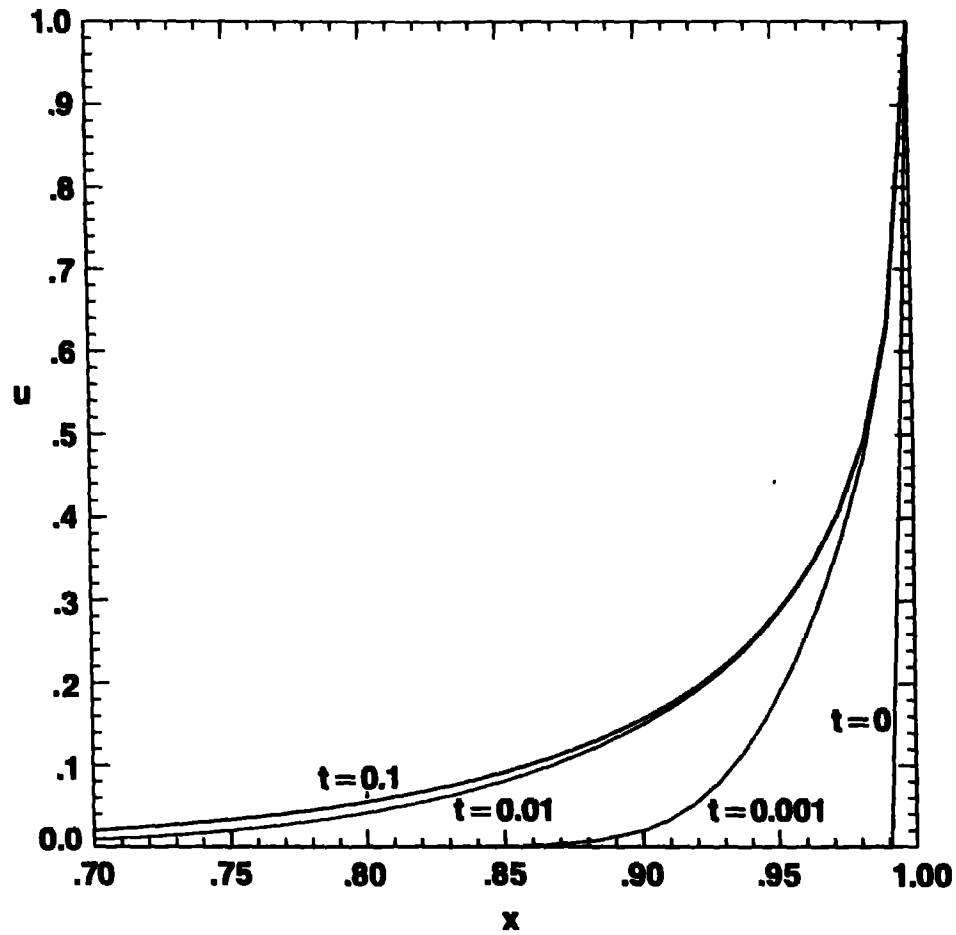


Figure 4.10b. Troesch's Two Point Boundary Value Problem

0186-38C-083