UCRL--87406

DE82 009393

UCRL-87406
PREPRINT

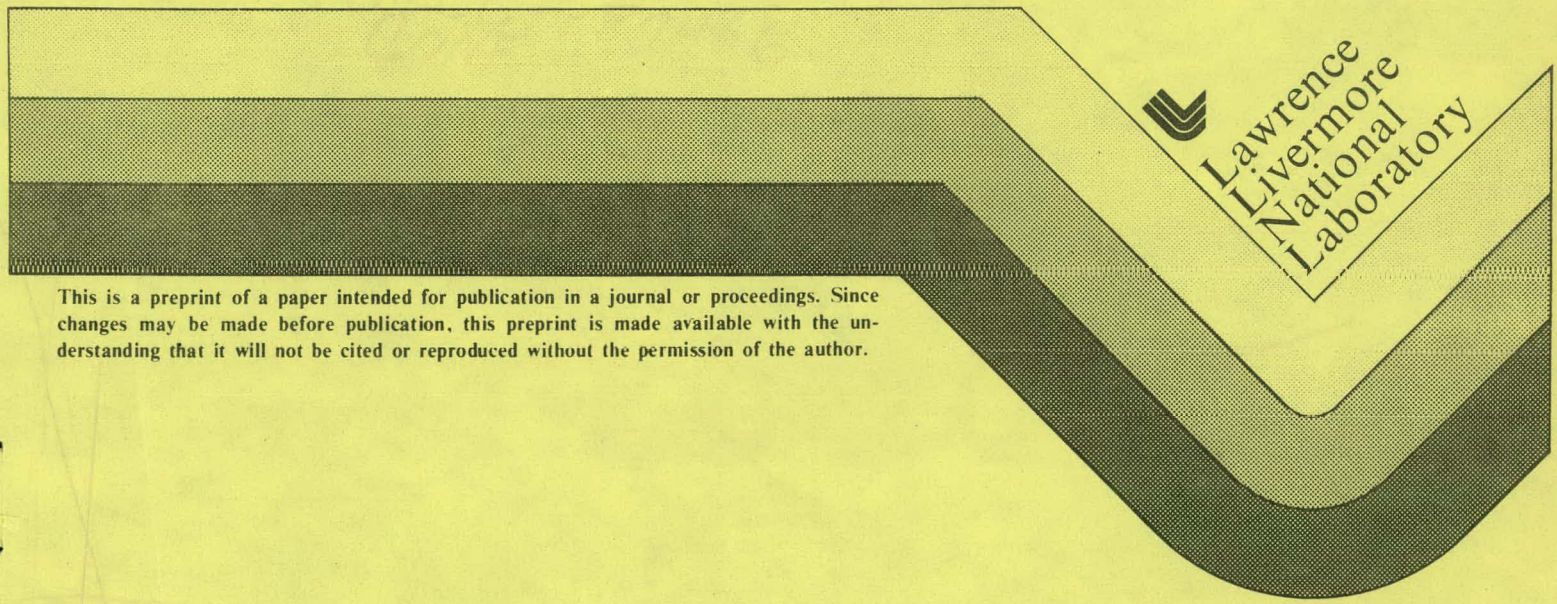
Conf-820425--2

STIFF-SYSTEM PROBLEMS AND SOLUTIONS AT LLNL

Alan C. Hindmarsh

This paper was prepared for presentation at the
International Conference on Stiff Computation,
Park City, Utah, April 12-14, 1982.

March 1982



DISCLAIMER

Stiff-System Problems and Solutions At LLNL*

Alan C. Hindmarsh
Mathematics and Statistics Division, L-316
Lawrence Livermore National Laboratory
Livermore, California 94550

ABSTRACT

Difficult stiff system problems encountered at LLNL are typified by those arising from various atmospheric kinetics models, which include reaction kinetics and transport in up to two space dimensions. Approaches devised for these problems resulted in several general purpose stiff system solvers. These have since evolved into a new systematized collection of solvers, called ODEPACK, based on backward differentiation formulas in the stiff case. A model kinetics-transport problem is used to illustrate the various solvers.

*This work was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore Laboratory under contract No. W-7405-Eng-48, and supported in large part by the DOE Office of Basic Energy Sciences, Mathematical Sciences Branch.

MGW

Stiff System Problems and Solutions At LLNL

1. Introduction

Initial value problems for systems of ordinary differential equations (ODE's) have long been a topic of great interest at LLNL. Stiff systems are particularly prevalent and are, of course, much more challenging. Applications giving rise to stiff ODE systems vary widely. But one area that typifies the difficulties encountered is that of atmospheric computer models, on which a great deal of effort has been spent at LLNL since about 1971. These problems, in most cases, take the form of systems of partial differential equations (PDE's) in space and time, involving chemical kinetics and transport processes. A discretization process leads to large stiff ODE systems. In Section 2, this class of problems and the various approaches pursued for their solution are described.

The problem features discussed here are not at all unique to this particular application, nor to problems at LLNL, and the software developed for their solution was designed with full awareness of that fact. Thus the ODE solvers used were designed to be as much general purpose as possible. However, a great deal has been learned in the intervening years about methods, algorithms, and software design for general ODE solvers. As a result, a new collection of initial value solvers has recently evolved at Livermore--the ODEPACK collection. There are currently five solvers in the collection. They are based on Adams methods (nonstiff case) and on the backward differention formula (BDF) methods (stiff case), and also on an inter-laboratory effort to set user interface standards for initial value solvers. These are described in Section 3.

In Section 4, a model problem on kinetics-transport type is used to illustrate the capabilities and relative merits of various solvers, including both those in ODEPACK and older codes.

2. Atmospheric Model Problems

The computer modelling of various atmospheric chemistry and transport process has been of great interest in the context of (a) ozone depletion from supersonic transport exhausts in the stratosphere [1,2], (b) stratospheric ozone depletion from terrestial fluorocarbon sources [3], (c) regional air pollution in the lower atmosphere [4], among others. In all cases, the mathematical model can be put in the form of a set of time-dependent PDE's in space (or, as a special case, as ODE's without space effects), in the concentrations c^i of the various chemical species of interest. These PDE's can be written

$$\partial c^i/\partial t = \nabla \cdot (D\nabla c^i + Vc^i) + K^i + S^i \quad (i=1,2,\ldots,p) \ , \tag{1}$$

where D is a matrix of diffusion coefficients, V is the vector of mean atmospheric motion, K^i is the kinetics rate for species i, and S^i is its external source rate. All of these quantities can in general depend on time t, on the point in space, and on the dependent variable vector $c = (c^1,\ldots,c^p)^T$. However, in the applications of interest, D and V and the S^i depend only on space and possibly time. The number of space dimensions is usually 2, but sometimes 1 or 0. The kinetics rates K^i usually involve diurnally varying rate coefficients, corresponding to photochemical reactions in the system.

The particular kinetics system involved varies from one application to another, but in all cases there is a mixture of fast and slow reactions (large and small rate coefficients), and the rates are nonlinear. The fast reactions correspond to strong damping effects with short time constants, i.e. they cause stiffness. This means that the ODE problem will be stiff regardless of the transport processes in the model, although they can also contribute to stiffness. The small time constants from the kinetics are usually in the microsecond range or smaller, while the time range of interest is usually measured in years. Thus stiffness ratios exceeding 10^{12} are common in these problems.

Historically, the spatial operator in (1) has usually been treated by finite difference approximations, because of their simplicity and their long history of use. In 2-D, a rectangular mesh is used. Actually, however, there is nothing inherent in the problem to prevent the use of a finite element, collocation, or Galerkin treatment, although in the latter case the local integrals pose some difficulty.

To illustrate the spatial differencing process, consider the one-dimensional operator

$$Lc = \frac{\partial}{\partial x}\left[D(x)\frac{\partial c}{\partial x} + V(x)c\right] . \tag{2}$$

On a mesh $x_1 < x_2 < \ldots < x_M$, the standard central differencing of (2), in terms of discrete values $c_i \approx c(x_i)$, is

$$Lc\Big|_{x_i} \approx \frac{D(x_{i+1/2})\, c_x(x_{i+1/2}) - D(x_{i-1/2})c_x(x_{i-1/2})}{(x_{i+1} - x_{i-1})/2} + \frac{V(x_{i+1})c_{i+1} - V(x_{i-1})c_{i-1}}{x_{i+1} - x_{i-1}} , \tag{3}$$

where, by definition,

$$x_{i+1/2} = (x_i + x_{i+1})/2,$$

$$c_x(x_{i+1/2}) = (c_{i+1} - c_i)/(x_{i+1} - x_i) ,$$

and similarly at $x_{i-1/2}$. The differencing process is handled in a similar way for two dimensional operators. Boundary conditions are similarly approximated in discrete form. For example, a zero normal derivative boundary condition, say

$$\frac{\partial c}{\partial x}\Big|_{x_1} = 0 ,$$

is approximated by setting $c_0 = c_2$ in the ODE's corresponding to $x = x_1$.

In this way, a set of p species PDE's becomes a coupled stiff system of ODE's whose size is

$$N = p \qquad \text{in 0-D (space-independent),}$$

$$N = pM \qquad \text{in 1-D (mesh size M),}$$

$$N = pM_x M_y \qquad \text{in 2-D (mesh size } M_x \text{ by } M_y\text{)}$$

It is clear that for a reasonably realistic model in 2-D, the system size can easily exceed 10,000.

In 1971, the only reliable stiff system solver available (to us) was C. W. Gear's DIFSUB. This routine was installed, modified and improved, and used as the GEAR package [5]. For 0-D problems or small 1-D problems, it worked well, and produced accurate answers with efficiencies that were quite impressive when compared with anything else tried (including a number of nonstiff methods, before the nature of stiffness was recognized). The GEAR package (like DIFSUB), also had a nonstiff method option (using Adams methods), so that it was widely usable (and used) as a general purpose ODE solver. Still, the GEAR package gave little hope of solving the full atmospheric models, because it had to construct, and perform LU factorizations on, full N x N matrices, followed by backsolve operations for solutions of linear systems. If the ODE system is written

$$\dot{y} = dy/dt = f(t,y)\,, \tag{4}$$

and has a system Jacobian matrix

$$J = \partial f/\partial y\,, \tag{5}$$

then the linear systems take the form

$$Px = (\text{BDF residual vector}), \tag{6}$$

where x is a correction vector and P is an approximation to $I-h\beta_0 J$. Here I denotes the identity matrix, h is a time step size, and β_0 is a scalar depending only on the current method order. The N x N system (6) occurs within a modified Newton iteration to solve the implicit BDF relation.

The next step was the realization that our problems had a very sparse Jacobian matrix, and that the LU method was extendable to sparse matrices. This was clearest if

J was thought of as banded, and so a variant solver, with J treated as banded, was written, and called GEARB [6]. This approach could be applied to the atmospheric problems if the dependent variables in the ODE system were correctly ordered. The appropriate ordering is to group together all p values c^i at one node, then all the c^i at the next node, and so on, with a natural linear (in 1-D) or rectangular (in 2-D) ordering of the nodes. Ordering in the reverse manner (by nodes, then by species) may seem natural, but produces much larger bandwidths, as long as p is small compared to either of the mesh dimensions. In two dimensions, the bandwidth is also minimized by numbering the nodes in the shorter direction first (if there is one). GEARB was used successfully in solving the 1-D models of interest, and small 2-D models.

The full 2-D models still seemed out of reach. Combining GEARB with reordering algorithms to reduce bandwidth failed because of the regular structure of the problem. A general sparse LU approach seemed inappropriate (though it was not thoroughly pursued) because matrix fill-in would result in much the same storage costs as for the banded treatment.

Inasmuch as the matrix elements in (6) are rather easily generated, the idea of iterative linear system methods was a natural next choice. For example, successive overrelaxation (SOR), with careful attention to the choice of relaxation parameter, was known, both theoretically and experimentally, to do well on linear systems based on transport PDE's of the type (1), with a single species and no kinetics. The implementation of SOR for (6) requires that P be written as

$$P = L + D + U \text{ with} \qquad (7)$$

L strictly lower triangular,

D diagonal and nonsingular,

U strictly upper triangular,

that one can generate (or access from memory) the elements of D, and that one can easily generate matrix-vector products $(L+U)x$. Specifically, for a given relaxation parameter ω, the ν^{th} SOR iteration in the solution of $Px = r$ is given by

$$(D + \omega L)x^{\nu+1} = \omega r + (1-\omega)Dx^{\nu} - \omega U x^{\nu} \quad . \tag{8}$$

This can be rephrased as

$$\begin{aligned} x^{\nu+1} &= (1-\omega)x^{\nu} + \omega z, \\ Dz &= r - Ux^{\nu} - Lx^{\nu+1} \quad . \end{aligned} \tag{9}$$

This pair of equations appears circular, but is not; z is computed and x^{ν} updated to $x^{\nu+1}$ one component at a time according to (9).

For the atmospheric problems, the coupling induced by the kinetics does not even come close to satisfying the conditions needed for ordinary SOR. Thus, to have any hope for convergence, it is necessary to treat the c vector (of length p) as a unit, and use a block-SOR algorithm. This results from replacing scalars in SOR by matrix or vector blocks of size p. Thus one must write

$$P = L + D + U \text{ with} \tag{10}$$

L strictly lower block-triangular,

D block diagonal and nonsingular,

U strictly upper block-triangular.

One must again generate products $(L + U)x$, and one must generate D and be able to solve $p \times p$ systems with the diagonal blocks in D. This can be easily done by forming and using LU factorizations of those blocks. Equations (9) also provide an algorithm for the block-SOR iteration, by updating x^{ν} in blocks (from first to last).

For a 2-D atmospheric model, a special block-SOR variant of GEAR was written, and later made into a general-purpose solver, called GEARBI [7] (BI denoting Block-Iterative). This was soon modified to take advantage of Large Core Memory (LCM) storage on the CDC-7600 computer, resulting in a package called GEARBIL. The latter stores most of the large data arrays in LCM, the largest being the one containing the matrix D in (10) (and

later its LU decomposition), of size $p^2 M_x M_y$. These two solvers, as they stand, are still somewhat specialized to the atmospheric models, in that the L and U matrices in (10) have a regular block structure (in p x p blocks), with each block being a scalar multiple of the p x p identity matrix. (The latter is due to the fact that D and V are independent of c in (1).) However, it is a straightforward matter to modify the codes to accommodate a more general coupling, and this has been done in at least two instances.

The GEARBIL package was used for a variety of studies of kinetics in the stratosphere, including ozone depletion from SST's [1,2] and from fluorocarbons [3], with the two dimensions being altitude and latitude. In most cases, the mesh sizes were 37 x 44 (1628 spatial nodes), and the number of chemical species p was 9, giving an ODE system size of N = 14,652. GEARBIL was subsequently used in several similar models developed for regional air quality calculations [4], with the third dimension (altitude) accounted for by assuming uniformity between the ground and the temperature inversion layer. One of these, LIRAQ2, is currently used to model the San Francisco Bay Area air layer, as an ongoing pollution control tool of regional government.

In the stratosphere studies, one of the lessons learned concerns the tolerance on the part of the ODE solver for errors in the Jacobian matrix, when supplied by the user. For some time, routines to supply J were written by hand, and therefore subject to error, especially for these complicated systems. When plots of step size history were generated, and in one case published [1,p.58], they often showed great irregularities, as if frequent instabilities or problem discontinuities were forcing drastic reductions in step size periodically. Subsequently, with the aid of automated Jacobian generators, errors in J were found and corrected, and reruns of these problems showed a remarkably smooth, nearly monotone, growth in step size.

Independently of the development of complex transport models in 1-D and 2-D, the kinetics mechanisms of the lower and upper atmosphere were studied. In particular, the detailed kinetics processes of the stratosphere, with diurnal effects included, required

much effort in both the model-building and the numerical solution phases, even in the absence of spatial transport, i.e. in 0-D. The stiff ODE systems that arise are further complicated by the fact that rate coefficients for photochemical reactions follow a nearly square-wave pattern in response to sunlight [8].

In many cases, these diurnal kinetics problems were found to cause great difficulty for the GEAR package, sometimes causing it to crash irrecoverably. It was found that this is due to the buildup of errors associated with the use of fixed-step BDF's with interpolatory step changing. As a result, variable-step forms of the BDF's were developed, jointly with G. Byrne [9]. These were implemented in a general solver called EPISODE [10,11], along with variable-step Adams methods for non-stiff problems. EPISODE resembles GEAR externally, but differs internally in all details associated with integration coefficients, error estimation, and step selection. EPISODE was found to be able to handle the diurnal problems quite reliably, although it was usually somewhat less efficient than GEAR on problems with smooth solutions [12]. A banded Jacobian variant EPISODEB [13] was also written, to accommodate diurnal kinetics-transport problems.

After most of the model building for these atmospheric problems was completed, the question of alternative space discretizations was nevertheless studied to some extent. N. Madsen and R. Sincovec showed that collocation methods could be used quite effectively on 1-D problems of this type, and developed a general purpose PDE package, called PDECOL, from that idea [14]. In the process, it was clear that a different type of ODE solver was needed, namely one which would treat linearly implicit systems,

$$A\dot{y} = g(t,y) \qquad (11)$$

(A a square matrix), in a direct and efficient manner. To this end, another GEAR variant, GEARIB, was written [15] (IB for Implicit systems, Banded matrix treatment), and a modified form of this is used in PDECOL. Later, an analogous EPISODE variant, EPISODEIB, was also written [16]. These two ODE solvers are intended mainly for the case of a nonsingular A matrix in (11), which is the most common situation in PDE-based problems, but can also be used in the singular case, if used with caution.

3. The ODEPACK Solvers

The DIFSUB, GEAR, and EPISODE packages were added to a list of available general purpose initial value solvers that was growing quite sizable by 1975. The length and diversity of this list caused some concern to users and software developers alike. There was much duplication of capabilities offered, but at the same time there was very little in common among the solvers in terms of either their external appearance or their internal structure. This situation was in sharp contrast to that in other areas in which "systematized collections" of Fortran routines were being developed. The earliest examples were EISPACK [17], for computing matrix eigensystems, LINPACK [18], for solving linear systems, and FUNPACK, for certain special functions.

3.1 The ODEPACK Concept

The idea of a systematized collection of initial value ODE solvers, tentatively called ODEPACK, was discussed informally as early as 1974, in workshops attended by people from all over the world [19]. However, it was quickly realized that the task was much larger in the ODE case than in other areas, partly because of the complexity of the subject, and partly because of widely divergent views of what ODEPACK should look like. Starting in 1976, attempts were made to reduce the problem by involving only people at U.S. Department of Energy laboratories, and LLNL received funding to study the feasibility of ODEPACK from the Applied Mathematical Sciences Research Program under the Office of Basic Energy Sciences in DOE.

The natural first step, and a necessary preliminary to any actual development of an ODEPACK, is the setting of standards for the interface between the user and the ODE solvers. The user interface to a solver consists mainly of the call sequence of the routine the user must call, together with definitions of the one or more user-supplied routines called by the solver. To the extent that solvers for various problem types and using various methods must all communicate certain specific things to and from the user, it is

possible to formulate a loose set of standards for the user interface. An early proposal is given in [20]. A sequence of workshops and discussions on user interface standards for ODE solvers succeeded in producing a reasonable consensus in 1978 [21,22]. The resulting tentative interface standard was achieved only through considerable compromise by the various participants, which included ODE software authors and users at various DOE laboratories.

At that time, it was agreed that several of the more popular ODE solvers, including GEAR, GEARB, DE/STEP [23] and RKF45 [24], would be rewritten to conform with the tentative standard interface, resulting in a small collection that was at least systematized in its external appearance. The first result of that agreement was a package based on the GEAR and GEARB packages, called LSODE (Livermore Solver for ODE's) [25]. The LSODE solver and variants of it written subsequently (all in accordance with the tentative standard interface [21], with minor modifications) are briefly described in the following subsections. In the meantime, unfortunately, the other software authors involved withdrew from the agreement, and so this collection does not yet have analogous rewritten versions of their codes.

3.2 LSODE

LSODE combines the capabilities of GEAR and GEARB. Thus it solves explicitly given stiff and nonstiff systems $\dot{y} = f(t,y)$, and in the stiff case it treats the Jacobian as either full or banded, and as either user-supplied or internally approximated by difference quotients. By comparison with GEAR and GEARB, LSODE offers a number of new features that make it more convenient, more flexible, more portable and easier to install in software libraries. Some of these are the following:

(a) Through the redesigned user interface, many new options and capabilities are available, and others are much more convenient than before. Some examples are--more flexible error tolerance parameters, independent flags for starting and stopping options, internally computed initial step size, two work arrays in the call sequence for all internal dynamic work space, user names for f and J in the call sequence, easy changing of input parameters in mid-problem, convenient optional inputs (such as maximum method order), convenient optional outputs (such as step and function evaluation counts), optional provision of derivatives of the

solution (of various orders) at any point, and real and integer user data space (of dynamic length) available in the f and J routines (with no extra burden on the casual user).

(b) The user documentation, which is contained in the initial comment cards of the source, is given in a two-level form. A short and simple set of instructions, with a short example program, is given first, for the casual user. Then detailed instructions are given for users with special problem features or a desire for nonstandard options. The latter is also organized so as to allow selective reading by a user who wants only a fraction of the nonstandard capabilities.

(c) When stiff options are selected, linear systems are solved with routines from LINPACK [18], which is becoming a widely accepted standard collection of linear system solvers.

(d) Some retuning of various heuristics was done so that performance should be more reliable than for GEAR/GEARB.

(e) The core routine which takes a single step, called STODE, is independent of the way in which the Jacobian matrix (if used) is treated. Thus as variant versions of LSODE are written for other matrix structures (such as LSODES), these will share the same subroutine STODE.

(f) The writing of all error messages is done in a small isolated general-purpose message handler called XERRWV. Two other small subroutines are user-callable for optional changing of the output unit number and optional suppression of messages. This trio of routines is compatible with a much larger error package (the SLATEC Error Handling Package) written elsewhere [26].

(g) LSODE easily allows a user to interrupt a problem and restart it later (e.g. in switching between two or more ODE problems). Also, using LSODE in overlay mode is very easy, with no loss of needed local variables.

(h) The various lists of constants needed for the integration, formerly appearing in a subroutine called COSET, are now computed (once per problem). This adds to the portability of LSODE.

3.3 LSODI

The LSODI solver [25], written jointly with J. F. Painter, (LLNL) treats systems in the linearly implicit form $A(t,y)\dot{y} = g(t,y)$, where A is a square matrix. Many problems, including PDE's treated by finite elements and the like, result in such systems, and it is almost always more economical to treat the system in the given form than to convert it to an explicit form $\dot{y} = f$. LSODI allows A to be singular, but the user must then input consistent initial values of both y and $\dot{y}$. In the singular case, the system is a

differential-algebraic system, and then the user must be much more cautious about formulating a well-posed problem, as well as in using LSODI, which was not designed to be robust in this case. LSODI is based on (and supersedes) the GEARIB package, but corrects a number of deficiencies, as follows:

(a) The matrices involved can be treated as either full or banded, by use of the method flag.

(b) The dependence of A on y is automatically and inexpensively accounted for, whether partial derivatives are supplied by the user or computed internally by difference quotients.

(c) When A is singular, the user needs to supply only the initial value of dy/dt , and this array (along with the initial y) is passed through the call sequence, rather than computed in a user-replaceable package routine. (Admittedly, correct initial data can be difficult to obtain for some types of problems.) When the initial dy/dt is not being supplied, an input flag instructs LSODI to compute it on the assumption that A is initially nonsingular. Thereafter, no such assumption is made, but ill-conditioning can be a problem when A is singular.

(d) The user-supplied residual routine includes a flag which allows the user to signal either an error condition or an interrupt condition.

(e) To the maximum extent possible, LSODI shares the same user interface as LSODE, and so reflects all the advantages over GEARIB that LSODE has over GEAR and GEARB.

The differences between the LSODI and LSODE user interfaces occur primarily in the user-supplied subroutines. With LSODI, one must supply a routine to compute the residual function $r = g(t,y) - A(t,y)s$ for a given t , y , and s , and another routine to add the matrix A to a given array. Optionally, the user can supply a routine to compute the Jacobian matrix $\partial r/\partial y$.

By virtue of the modular and systematized organization of LSODE and LSODI, the two packages share most of their routines with each other.

Some examples of the use of LSODE and LSODI on systems arising from PDE problems can be found in [27] and [28]. In the latter, experiments by Painter on incompressible Navier-Stokes problems shed some light on the difficulties involved with differential-algebraic systems.

3.4 LSODES

The LSODES package solves explicit systems $\dot{y} = f$, but treats the Jacobian matrix J as a general sparse matrix in the stiff case. LSODES was written jointly with A. H. Sherman (Exxon Production Res. Company), and supersedes a sparse variant of GEAR called GEARS [29,30]. In LSODES, the linear systems (6) are solved using parts of the Yale Sparse Matrix Package (YSMP) [31,32]. This involves several phases:

(a) Determination of sparsity structure. This is either inferred from calls to the f routine, inferred from calls to a J routine (if one is supplied), or supplied directly by the user. A user input flag determines which is done.

(b) Determination of pivot order. Diagonal pivot locations are chosen, and the choice is based on maximizing sparsity. This is done by YSMP.

(c) Symbolic LU factorization of the matrix P. This is based only on sparsity and the pivot order, and uses the module in YSMP designed for nonsymmetric matrices with compressed pointer storage.

(d) Construction of J. This can be done internally by difference quotients, or with a user-supplied routine. In the difference quotient case, the number of f evaluations needed is kept to a minimum by a column grouping technique due to Curtis, Powell, and Reid [33]. In the other case, the user-supplied routine provides one column of J at a time, in the form of a vector of length N (although only non-zero elements need be computed and stored), so that users need never deal with the internal data structure for J and P. In any case, J is stored internally in an appropriate packed form. Evaluations of J are done only occasionally, as explained below.

(e) Construction of $P = I - h\beta_0 J$. In contrast to GEARS, LSODES does not force a re-evaluation of J whenever the existing P is deemed unsuitable for the corrector iterations. Instead, when the value of J contained in the stored value of P is likely to be usable (and P is not, only because $h\beta_0$ has changed significantly), then a new matrix P is constructed from the old one, with careful attention to roundoff error. This cuts down greatly on the number of J evaluations necessary.

(f) Numerical LU factorization of P. This is done by YSMP in sparse form, and the array containing P is saved in the process (this allows for updating P as described above). Because of the absence of partial pivoting for numerical stability, this operation can conceivably fail. However, this has only rarely been observed in practice, and if it does occur (with a current value of J), the step size h gets reduced and the problem disappears.

(g) Solution of $Px = r$. This is done by YSMP using the existing sparse factorization of P. Because a modified Newton iteration is used, many values of r (i.e., many linear systems) can arise for the same P, and the separation of the various phases takes advantage of that fact.

The first three phases, and part of the fourth (column grouping for difference quotients), are normally done only at the start of the problem. However, the user can specify that the sparsity structure is to be redetermined in the middle of the problem, and then these operations are repeated.

Actually, the matrix operated on by YSMP is $A = P^T$, not P, because P is generated in column order while YSMP requires the matrix to be described and stored in row order. This causes no difficulty, however, because YSMP includes a routine for solving the transpose problem $x^T A = r^T$ (which is equivalent to $Px = r$) as well as for the direct problem $Ax = b$.

A package called LSODIS, similar to LSODI (for the $A\dot{y} = g$ problem) but using YSMP for general sparse treatment of matrices as in LSODES, is in the process of being written.

3.5 LSODA

LSODA is a variant of LSODE of yet another kind. It was written jointly with L. R. Petzold (Sandia-Livermore), and switches automatically between nonstiff (Adams) and stiff (BDF) methods, by an algorithm developed by Petzold [34]. (The suffix A is for Automatic.) Thus it is more convenient than LSODE for users who would rather not be bothered with the issue of stiffness. Also, it is potentially more efficient than LSODE (when used with a fixed method option), when the nature of the problem changes between stiff and nonstiff in the course of the solution. In place of the method flag parameter of LSODE, the user of LSODA supplies only a Jacobian type flag. The storage space supplied to the solver can be either static (and thus allow for either the stiff or nonstiff method), or dynamic (and altered each time there is a method switch, to an amount specified by the solver).

3.6 LSODAR

LSODAR combines the capabilities of LSODA with a rootfinder. It allows one to find the roots of a set of functions $g_i(t,y)$ of the independent and dependent variables in the ODE system. Thus, for example, it could be used in a particle tracking problem to determine when a particle path reaches any of the walls of a container. LSODAR was also written jointly with L. R. Petzold based on an algorithm [35] developed by K. Hiebert and L. F. Shampine (Sandia-Albuquerque). The user must supply, in addition to the LSODA inputs, a subroutine that computes a vector-valued function $g(t,y) = (g_i,\ i=1,2,\ldots,NG)$ such that a root of any of the NG functions g_i is desired. Of course there may be several such roots in a given interval, and LSODAR returns them one at a time, in the order in which they occur along the solution, with an integer array to tell the user which g_i (if any) were found to have a root on a given return.

4. An Example Problem

In order to illustrate the various solvers (new and old) described above, and to demonstrate their relative merits on a realistic problem, we consider here an example problem. The problem is a simple atmospheric model [30] with two chemical species undergoing diurnal kinetics and transport in two space dimensions. The independent variables in the PDE system are horizontal position x, altitute z (both in kilometers), and time t (in sec), with $0 \le x \le 20$, $30 \le z \le 50$, $0 \le t \le 86400$ (1 day). The dependent variables are $c^1(x,z,t)$ = the concentration of the oxygen singlet [**O**], and $c^2(x,z,t)$ = that of ozone [$\mathbf{O}_3$] (both in moles/cm^3). The concentration of molecular oxygen [$\mathbf{O}_2$] is assumed constant. The equations of the model are:

$$c^i_t = (K_v(z)c^i_z)_z + K_h c^i_{xx} + R^i(c^1,c^2,t) \quad (i=1,2), \tag{12}$$

where R^1 and R^2 represent the chemistry and are given by

$$R^1(c^1,c^2,t) = -(k_1+k_2c^2)c^1 + k_3(t)c^2 + k_4(t)\cdot 7.4\cdot 10^{16}$$

$$R^2(c^1,c^2,t) = (k_1-k_2c^2)c^1 - k_3(t)c^2 \quad .$$

Subscripts t, z, and x denote partial derivatives. The various coefficients are as follows:

$K_v(z) = 10^{-8} \cdot \exp(z/5)$, $K_h = 4 \cdot 10^{-6}$, $k_1 = 6.03$, $k_2 = 4.66 \cdot 10^{-16}$,

$$k_3(t) = \begin{cases} \exp(-7.601/\sin(\pi t/43200)) & \text{for } t < 43200 \\ 0 & \text{for } t \geq 43200 \end{cases},$$

$$k_4(t) = \begin{cases} \exp(-22.62/\sin(\pi t/43200)) & \text{for } t < 43200 \\ 0 & \text{for } t \geq 43200 \end{cases}.$$

The initial conditions are

$$c^i(x,z,0) = 10^{6i}\left[1-\left(\frac{x-10}{10}\right)^2 + \frac{1}{2}\left(\frac{x-10}{10}\right)^4\right]\left[1-\left(\frac{z-40}{40}\right)^2 + \frac{1}{2}\left(\frac{z-40}{40}\right)^4\right]$$

$$(i = 1,2),$$

and both c^1 and c^2 are required to satisfy homogeneous Neumann boundary conditions along all the x and z boundaries.

To solve the system (12) numerically, we apply the method of lines using a regular square mesh with constant mesh spacings

$$\Delta x = 20/(M_x-1), \quad \Delta z = 20/(M_z-1).$$

The spatial derivatives are approximated by standard 5-point central differences, as given by (3) in each direction. The boundary conditions are similarly replaced by difference relations. The resulting ODE system $\dot{y} = f(t,y)$ has size $N = 2M_xM_z$. The initial value vector y_0 is taken from the initial condition functions given above. The system Jacobian J is sparse, with roughly $12M_xM_z = 6N$ nonzero elements. As a band matrix, with component ordering first by species, then by x, and lastly by z, it has a half-bandwidth of $2M_x$, and thus a full bandwidth of $4M_x+1$.

As a nominal case, consider the choice $M_x = M_z = 10$. As to accuracy, a crude model of this type calls for no more than a few significant figures. To be conservative in recognizing that tolerance parameters are applied to local errors, which can accumulate into global error, we might impose a local relative tolerance of 10^{-4}. We must also specify a positive absolute tolerance on the values of c^1 because it decays to negligible values at night. A reasonable absolute tolerance is 10^{-2}. With the ODEPACK solvers, specifying such a mixed relative/ absolute error control is trivial, but with the GEAR and EPISODE families, a slight modification to the driver is necessary.

Of the various solvers mentioned, five are suitable for this particular problem—LSODE, LSODA, LSODES, EPISODEB, and GEARBI. Recall that LSODES uses a general sparse treatment of the Jacobian matrix, GEARBI uses block- SOR, and the others (in this case) treat the Jacobian as banded. The problem was set up for each of these five solvers and run on a CDC-7600 computer. For all but GEARBI, both the user-supplied Jacobian option and the internal difference quotient Jacobian option were tested. (For GEARBI, there is no difference quotient option.) The results of the various runs are given in Table 1. The tabulated quantities are:

R.T. = CPU run time in sec

NST = number of steps

NFE = number of f evaluations

NJE = number of J evaluations

NLU = number of LU decompositions

W.S. = total size of work space arrays

In the table, the notation USJ denotes the user-supplied Jacobian option, and DQJ denotes the internal difference quotient Jacobian option.

Table 1
Results of Kinetics-Transport Test Problem

Solver	R.T.	NST	NFE	NJE	NLU	W.S.
LSODE (USJ)	23.2	344	519	68	68	14,242
LSODE (DQJ)	28.4	337	3338	69	69	14,242
LSODA (USJ)	21.3	339	584	55	55	14,242
LSODA (DQJ)	24.6	339	2795	55	55	14,242
LSODES (USJ)	13.1	364	529	10	70	12,455
LSODES (DQJ)	13.5	369	602	8	72	12,664
EPISODEB (USJ)	18.6	264	461	81	81	14,400
EPISODEB (DQJ)	25.1	264	3782	81	81	14,400
GEARBI (10x10)	6.3	316	526	50	50	3,004
GEARBI (38x38)	199	393	698	96	96	43,324

For the sake of illustration, the GEARBI test was repeated on a 38x38 grid, and the results given in the last line of Table 1. This is the largest square grid that could be accommodated with that solver on the CDC-7600 within its Small Core Memory (about 57,000 words). As noted, the Large Core Memory (about 400,000 words) was used for the larger actual atmospheric models. The Cray-1 computer will accept even larger problem sizes.

Several points of interest can be noted in the table, for the 10x10 problem. First, the number of steps does not vary greatly from solver to solver, because that is determined almost entirely by the accuracy requirement, and the accuracy is much the same for all these runs. The relative merits of the solvers must be judged from other statistics.

The performance characteristics of LSODE, LSODA, and EPISODEB are similar, as expected, since they all use a banded Jacobian. The variations are partly attributable to

differences in the fine tuning, but the fact that EPISODEB has the smallest NST and (with the USJ option) the shortest run time can be attributed to its use of variable-step formulas. (Recall that the diurnal kinetics problems motivated the development of EPISODE.) The use of a difference quotient Jacobian is invariably more expensive here, owing to its cost of 41 evaluations of f for each evaluation of J .

The LSODES results show that a general sparse matrix treatment gives a significant speedup over the band treatment. This results partly from the matrix software itself, and partly from the algorithm of effectively saving old values of J for greater reuse. Note that each computed value of J is used for 36 to 46 steps, as opposed to only 3 to 6 steps with the solvers using a banded Jacobian. Also, the cost penalty for a difference quotient Jacobian is much smaller with LSODES, because each J evaluation here costs only 8 f evaluations. The storage requirement is only slightly smaller, reflecting the need for sparsity information arrays and the fact the Newton matrix P is not overwritten with its LU decomposition, as it is in the band case.

The best performance on this problem, however, is that of GEARBI. This should not be a surprise, since the Jacobian has a very regular block structure of which the block-SOR method in GEARBI is taking full advantage, both in storage and computation. The LU decompositions here are only those of the block diagonal part of the Newton matrix (with 2x2 blocks). The total number of block-SOR iterations for the 10x10 grid was 607, or an average of less than 2 per step. For the 38x38 grid this cost rose to 2122 iterations, or an average of 5.4 per step. The latter run also shows that spatial discretization errors in the 10x10 grid answers are as large as 2% . For an earlier comparison test on this problem, see [30].

For large stiff systems with wide-bandwidth coupling, of which this is an example, the single most important criterion for selecting a method or solver usually turns out to be the storage requirement. For this problem, on a m x m grid, the storage for a band-oriented solver is roughly $22m^2 + 12m^3$, while that for GEARBI is $30m^2$. The storage for LSODES is harder to predict, but behaves very roughly like $150m^2$ in the range m = 15 to

25 . Thus LSODES has an increasing storage advantage over LSODE etc. for $m \geq 10$, but GEARBI has a lower storage requirement than any of them. However, if the problem fails to have the regularity needed for such an approach, or if block-SOR is not appropriate for numerical reasons, then the choices seem to be reduced to

(a) solvers using a general sparse direct linear system solver, such as LSODES,

(b) solvers using more suitable (possibly ad hoc) iterative methods (and which economize on matrix storage in some way), and

(c) radically different ad hoc treatments such as operator splitting, or methods that combine full implicitness and splitting ideas.

REFERENCES

[1] J. S. Chang, A. C. Hindmarsh, and N. K. Madsen, Simulation of Chemical Kinetics Transport in the Stratosphere, in Stiff Differential Systems, Plenum Press, New York, 1974, R. A. Willoughby, ed., pp. 51-65.

[2] M. C. MacCracken, DOT-CIAP Final Report, LLNL Report UCRL-51336, May 1975.

[3] J. S. Chang, D. J. Wuebbles, and W. H. Duewer, Sensitivity to Parameter Uncertainties for Ozone Reduction from Chlorofluoromethanes, LLNL Report UCRL-77432, January 1976.

[4] M. C. MacCracken, D. J. Wuebbles, J. J. Walton, W. H. Duewer, and K. E. Grant, Livermore Regional Air Quality Model: I. Concept and Development, J. of Appl. Meterology, 17 (1978), 254-272.

[5] A. C. Hindmarsh, GEAR: Ordinary Differential Equation System Solver, LLNL Report UCID-30001, Rev. 3, December 1974.

[6] A. C. Hindmarsh, GEARB: Solution of Ordinary Differential Equations Having Banded Jacobian, LLNL Report UCID-30059, Rev. 2, June 1977.

[7] A. C. Hindmarsh, Preliminary Documentation of GEARBI: Solution of ODE Systems with Block-Iterative Treatment of the Jacobian, LLNL Report UCID-30149, December 1976.

[8] R. J. Gelinas, Diurnal Kinetic Modeling, LLNL Report UCRL-75373, January 1974; published in Proc. Intern. Conf. Structure, Composition and General Circulation of the Upper and Lower Atmospheres and Possible Anthropogenic Perturbations, Melbourne, Australia, 1974, N. J. Derco and E. H. Truhlar, Eds. (Atmospheric Environment Service, Downsview, Ontario, 1974).

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

[10] A. C. Hindmarsh and G. D. Byrne, EPISODE: An Effective Package for the Integration of Systems of Ordinary Differential Equations, LLNL Report UCID-30112, Rev. 1, April 1977.

[11] A. C. Hindmarsh and G. D. Byrne, Applications of EPISODE: An Effective Package for the Integration of Systems of Ordinary Differential Equations, in Numerical Methods for Differential Systems, Academic Press, 1976, L. Lapidus and W. E. Schiesser, eds., pp. 147-166.

[12] G. D. Byrne, A. C. Hindmarsh, K. R. Jackson, and H. G. Brown, A Comparison of Two ODE Codes: GEAR and EPISODE, Computers & Chem. Eng., 1 (1977), 133-147.

[13] G. D. Byrne and A. C. Hindmarsh, EPISODEB: An Experimental Package for the Integration of Systems of Ordinary Differential Equations with Banded Jacobians, LLNL Report UCID-30132, April 1976.

[14] N. K. Madsen and R. F. Sincovec, Algorithm 540. PDECOL, General Collocation Software for Partial Differential Equations, ACM-Trans. Math. Software, 5 (1979), 326-351.

[15] A. C. Hindmarsh, Preliminary Documentation of GEARIB: Solution of Implicit Systems of Ordinary Differential Equations with Banded Jacobian, LLNL Report UCID-30130, February 1976.

[16] G. D. Byrne, The ODE Solver EPISODE, Its Variants, and Their Use, in the Proceedings of the ANS Topical Meeting on Computational Methods in Nuclear Engineering, Williamsburg, VA, April 23-25, 1979.

[17] B. T. Smith, J. M. Boyle, B. S. Garbow, Y. Ikebe, V. C. Klema and C. B. Moler, Matrix Eigensystem Routines-EISPACK Guide, Lecture Notes in Computer Science, Vol. 6, Edition 2, Springer-Verlag (New York, 1976).

[18] J. J. Dongarra, J. R. Bunch, C. B. Moler, and G. . Stewart, LINPACK User's Guide, SIAM, Philadelphia, 1979.

[19] G. D. Byrne, A Report on the ODE Workshop, Held at San Antonio, Texas, January 26-28, 1976, in the ACM-SIGNUM Newsletter, Vol. 11, No. 1, pp. 27-28, May 1976.

[20] A. C. Hindmarsh and G. D. Byrne, A Proposed ODEPACK Calling Sequence, LLNL Report UCID-30134, May 1976.

[21] A. C. Hindmarsh, A Tentative User Interface Standard for ODEPACK, LLNL Report UCID-17954, October 1978.

[22] A. C. Hindmarsh, A User Interface Standard for ODE Solvers, in the Proceedings of the 1979 SIGNUM Meeting on Numerical Ordinary Differential Equations, April 1979, Univ. of Illinois (Dept. of Computer Science) Report 79-1710, R. D. Skeel, ed., 1979; also in the ACM SIGNUM Newsletter, Vol. 14, No. 2 (June 1979), p. 11.

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

[24] G. E. Forsythe, M. A. Malcolm, and C. B. Moler, Computer Methods for Mathematical Computations, Prentice-Hall (1977) pp. 129-147.

[25] A. C. Hindmarsh, LSODE and LSODI, Two New Initial Value Ordinary Differential Equations Solvers, in the ACM-SIGNUM Newsletter, Vol. 15, No. 4, pp. 10-11, December 1980.

[26] R. E. Jones, SLATEC Common Mathematical Library Error Handling Package, Sandia National Laboratories Report SAND78-1189, September 1978.

[27] 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, New Brunswick, N.J., 1981.

[28] J. F. Painter, Solving the Navier-Stokes Equations with LSODI and the Method of Lines, LLNL Report UCID-19262, December 1981.

[29] J. W. Spellmann and A. C. Hindmarsh, GEARS: Solution of Ordinary Differential Equations Having a Sparse Jacobian Matrix, LLNL Report UCID-30116, August 1975.

[30] 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.

[31] S. C. Eisenstat, M. C. Gursky, M. H.Schultz, and A. H. Sherman, Yale Sparse Matrix Package: I. The Symmetric Codes, Research Report No. 112, Dept. of Computer Sciences, Yale University, 1977.

[32] S. C. Eisenstat, M. C. Gursky, M. H. Schultz, and A. H. Sherman, Yale Sparse Matrix Package: II. The Nonsymmetric Codes, Research Report No. 114, Dept. of Computer Sciences, Yale Universtiy, 1977.

[33] A. R. Curtis, M. J. D. Powell, and J. K. Reid, On the Estimation of Sparse Jacobian Matrices, J. Inst. Math. Applic. 13, (1974), pp. 117-119.

[34] L. R. Petzold, Automatic Selection of Methods for Solving Stiff and Nonstiff Systems of Ordinary Differential Equations, Sandia National Laboratories Report SAND80-8230, September 1980.

[35] K. L. Hiebert and L. F. Shampine, Implicitly Defined Output Points for Solutions of ODE's, Sandia National Laboratories Report SAND80-0180, February 1980.

Technical Information Department · Lawrence Livermore Laboratory
University of California · Livermore, California 94550