

AD-A122 171

PROCEEDINGS OF THE INTERNATIONAL CONFERENCE ON STIFF
COMPUTATION APRIL 12... (U) UTAH UNIV SALT LAKE CITY DEPT
OF CHEMICAL ENGINEERING R C AIKEN 1982

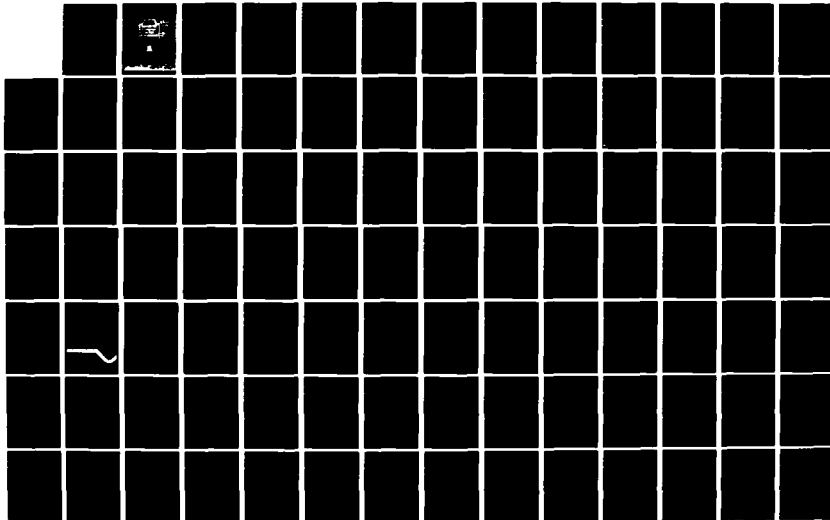
1/4

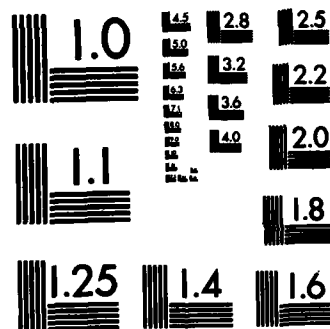
UNCLASSIFIED

AFOSR-TR-82-1036-VOL-3 AFOSR-82-0038

F/G 12/1

NL

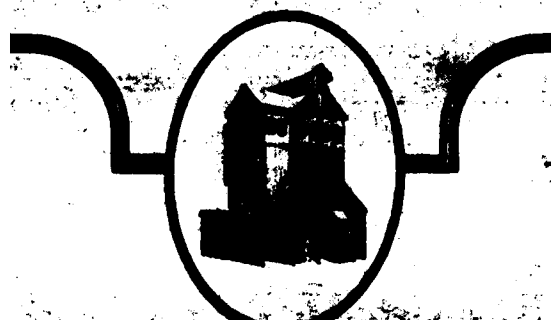




MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

INTERNATIONAL Conference on Soil Compaction

April 12, 13, 14, 1982
at Park City, Utah



UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER AFOSR-TR- 82-1036	2. GOVT ACCESSION NO. AD-A122 171	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) PROCEEDINGS, INTERNATIONAL CONFERENCE ON STIFF COMPUTATION, VOLUMES I, II, AND III		5. TYPE OF REPORT & PERIOD COVERED TECHNICAL
		6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) Richard C. Aiken		8. CONTRACT OR GRANT NUMBER(s) AFOSR-82-0038
9. PERFORMING ORGANIZATION NAME AND ADDRESS Department of Chemical Engineering University of Utah Salt Lake City UT 84112		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS PE61102F; 2304/A3
11. CONTROLLING OFFICE NAME AND ADDRESS Directorate of Mathematical & Information Sciences Air Force Office of Scientific Research Bolling AFB DC 20332		12. REPORT DATE 12-14 April 1982
		13. NUMBER OF PAGES 385
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office)		15. SECURITY CLASS. (of this report) UNCLASSIFIED
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE
16. DISTRIBUTION STATEMENT (of this Report) Approved for public release; distribution unlimited.		
17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)		
18. SUPPLEMENTARY NOTES Proceedings, International Conference on Stiff Computation, April 12-14, 1982 Park City, Utah.		
19. KEY WORDS (Continue on reverse side if necessary and identify by block number)		
20. ABSTRACT (Continue on reverse side if necessary and identify by block number) These three volumes constitute the written contributions of speakers at the International Conference on Stiff Computation, held April 12-14, 1982, at Park City, Utah. As this collection was prepared in advance of the meeting, a few contributions were too late to be included here. The purpose of this meeting was to bring together theorists, software developers, and users on common ground to consider the state of the art - and practice - of stiff computation. Most of the papers in these proceedings will appear formally in the form of a monograph.		

DD FORM 1 JAN 73 1473

EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

**PROCEEDINGS OF THE INTERNATIONAL CONFERENCE
ON STIFF COMPUTATION**

**April 12-14, 1982
Park City, Utah**

Volume I&I

**AIR FORCE OFFICE OF SCIENTIFIC RESEARCH (AFSC)
NOTICE OF TRANSMITTAL TO DTIC
This technical report has been reviewed and is
approved for public release IAW AFR 190-12.
Distribution is unlimited.
MATTHEW J. KEMPER
Chief, Technical Information Division**

> PARTIAL CONTENTS INCLUDE:

VOLUME III

APRIL 14, 1982

C.W. GEAR, University of Illinois at Urbana-Champaign:

STIFF SOFTWARE: WHAT DO WE HAVE AND WHAT DO WE NEED?

W. H. ENRIGHT, University of Toronto:

(1) PITFALLS IN THE COMPARISON OF NUMERICAL METHODS FOR STIFF ORDINARY DIFFERENTIAL EQUATIONS.

A.C. HINDMARSH, Lawrence Livermore National Laboratory:

STIFF SYSTEM PROBLEMS AND SOLUTIONS AT LLNL

G.K. GUPTA, Monash University:

DESCRIPTION AND EVALUATION OF A STIFF ODE CODE DSTIFF

P. DEUFLHARD (speaker), G. BADER, U. NOWAK, Universitat Heidelberg:

(2) AN ADVANCED SIMULATION PACKAGE FOR LARGE CHEMICAL REACTION SYSTEMS.

L. EDSBERG, The Royal Institute of Technology:

(3) CHEMICAL KINETICS - AN OVERVIEW FROM THE POINT OF VIEW OF NUMERICAL ANALYSIS AND SOFTWARE IMPLEMENTATION,

J. DEVOOGHT, Universite Libre de Bruxelles:

(4) AN OVERVIEW OF STIFFNESS PROBLEMS IN NUCLEAR REACTOR KINETICS;



A

21147

S. THOMPSON (speaker),
P.G. TUTTLE, Babcock and Wilcox:

15 THE SOLUTION OF SEVERAL REPRESENTATIVE
STIFF PROBLEMS IN AN INDUSTRIAL ENVIRONMENT:
THE EVOLUTION OF AN O.D.E. SOLVER;

P.G. BAILEY, E.P.R.I. (speaker), P.V.
GIRIJASHANKAR, D.L. HETRICK, W.N. KEEPIN and
O.A. PALUSINSKI, University of Arizona:

16 MULTIRATE INTEGRATION ALGORITHMS APPLIED
TO STIFF SYSTEMS IN NUCLEAR POWER PLANT
SIMULATION,

S.K. DEY, NASA-Ames Research Center:

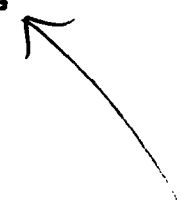
17 APPLICATIONS OF PERTURBED FUNCTIONAL
ITERATIONS TO NONLINEAR STIFF INITIAL
VALUE CHEMICAL KINETIC PROBLEMS.

J.-T. HWANG, National Tsing Hua University,
Taiwan:

18 NONLINEAR SENSITIVITY ANALYSIS IN
CHEMICAL KINETICS--THE SCALED GREEN'S
FUNCTION METHOD,

K.E. CHEN, Bethlehem Steel Corp. (speaker),
W.E. SCHIESSER, Lehigh University:

And (9) SOME EXPERIENCES IN THE SELECTION OF
INTEGRATORS FOR LARGE-SCALE ODE PROBLEMS
IN CHEMICAL ENGINEERING



4

STIFF SOFTWARE: What Do We Have and What Do We Need?*

C. W. Gear
Department of Computer Science
University of Illinois at Urbana-Champaign

International Conference on Stiff Computation
Salt Lake City, Utah
April 12-14, 1982

Abstract

Software for stiff differential equations is surveyed. The classical stiff problem is well-served by existing software and some codes are appearing for some of the other areas, particularly problems with eigenvalues near the imaginary axis. A number of other difficulties are identified and the state of and possibilities for software development in these areas are discussed.

*Supported in part by the U.S. Department of Energy, Grant DOE DEAC0276ERO2383.

1. Introduction

In addition to the large amount of fine theoretical work that has been done on stiff equations in the last decade, there have been a great number of new methods developed for solving stiff equations in the last fifteen years, and a great many improvements to existing methods. This much is very obvious from the literature. In preparing to write this paper I surveyed a few people in major laboratories and looked over some of the literature concerned with libraries and codes, and was surprised to find that in spite of this large amount of work, relatively little of it has found its way into the hands of the users. In fact, except for those fortunate enough to have direct access to some of the people active in ODE code development, virtually all users work with codes based on BDF methods from the NAG or IMSL libraries, from Hindmarsh's LSODE set, or derivatives of earlier codes on which these were based. This is probably no surprise to users on the other side of the fence, as they have long said that until we numerical analysts take time to write good software and get it out to the users, our ideas will not be put into action. However, here in 1982 many of "we numerical analysts" feel that we have seen the light and now think of ourselves as "numericiciens," people who are concerned about the real numerical aspects of problems and produce mathematical software to present our results. Yet, in spite of this, very few of the new methods are actually in use and most of the highly used software is based on early methods.

Does this mean that the user has no needs that are not being met by existing codes? I don't think so. The main development that has affected the average user over the last decade is that earlier software has been greatly improved in two ways: fine tuning and user interfaces. Although there have been some codes developed in that period which embody new methods, they have not undergone the careful modification to tune them for user use such as, for example, Hindmarsh has applied in developing the LSODE set of codes from earlier codes. There are a number of user needs that are not met by the currently highly used codes. For some of these needs there are methods but the software is in various states of development from experimental (a euphemism for badly written) to what we might call preproduction versions. For other needs there currently does not exist software, or in some cases even reasonable methods.

This paper will first list a number of areas that are part of or impact on the solution of stiff equations. Then, each of these areas will be examined to see some of what is available, both in methods and software, and where some of the currently recognized difficulties lie.

2. Classification of problems

The phenomenon of stiffness that first caused numerical difficulty was due to eigenvalues with large negative real parts. The methods developed to solve those are generally effective for problems which are such that all eigenvalues, $\lambda(J)$, of J satisfy $\text{Arg}(-\lambda(J)) < 60^\circ$. For

example, the BDF formulas are $A(73.23^\circ)$ stable for orders up to four, while the fifth order formula is $A(51.83^\circ)$ stable (see [Gaff82], Table 5.1). We will refer to this angle as the argument of stability. A sequence of codes have been written based on these methods, and the current versions of these codes are the ones in common use today. I suspect the reason that newer methods are having very little impact in this area is that the underlying method in these older codes is within a small factor of being as good as one can hope to do, and the codes based on these methods have evolved over a decade to be so finely tuned that recent codes based on slightly better methods show very little improvement. Furthermore, the fact that these older codes have been developed in a "user-active" environment has caused them to be well-tailored to user needs as far as interfaces go.

However, there are other types of problems that are frequently classified as "stiff" and additional features of stiff problems that cause difficulties for many existing codes. There are also a number of features that are needed in codes but are not yet commonly available. Among these are:

- (i) Lightly damped oscillatory problems--they have large eigenvalues near the imaginary axis and for which the higher-order BDF methods are unstable for a large range of stepsizes.
- (ii) Linear, constant-coefficient problems for which more efficient techniques are possible.

- (iii) Large systems of equations in which the matrix manipulation becomes a large part of the cost.
- (iv) Implicit differential equations which, if large, should be handled directly and not converted to explicit form because of the loss of sparsity.
- (v) Differential-algebraic equations which are an extension of implicit equations in which the number of differential equations is less than the number of equations and variables so that fewer initial values can be specified.
- (vi) Differential equations with discontinuities--these cause many codes to be very inefficient.
- (vii) Problems whose solution contains very high frequency components of little interest but which cannot be ignored without introducing error because of non-linearities.
- (viii) Systems of equations which can be partitioned into sets which change at very different speeds and can then be handled with different stepsizes and/or different methods.
- (ix) Equations with delayed terms present.
- (x) The need to estimate global error in codes--most current codes attempt to control local error only.

- (xi) The need to switch between different methods according to the nature of the problem as it changes over the interval.
- (xii) The need to have codes whose output is a "smooth" function of its inputs--for example, if the integrator provides data for another code such as an optimization program, it is important that the integrator appear to implement a smooth function or the optimizer may get stuck in a local minimum that represents noise, or will have great difficulty making realistic estimates of the derivatives of the output of the integrator.

The sections below discuss each of these problem areas and the current state of software development.

3. Current software for the standard stiff problem

In addition to the codes in the IMSL and NAG libraries and the codes of Hindmarsh [Hind80], [Hind79b], and [Hind79a], all of which are widely available, there are a number of newer codes in various stages of development which may have an impact in the near future. Some of these are discussed in the sections below because they appear to be particularly suitable for some of the special problems. Among the other codes are two recent "entries to the market" based on techniques not used in common stiff codes. Bader and Deuflhard [BaDe81] have developed an extrapolation code based on a semi-implicit midpoint rule. This midpoint rule has the advantage that it does not require iteration of

nonlinear equations involving the Jacobian and yet has a quadratic error expansion so that each extrapolation step adds two orders of accuracy. Their program, METAN1, is still experimental but is a promising candidate for future general use. A second program, DSTIFF, is based on multistep formulas chosen to provide a least-squares fit to a number of accuracy requirements as well as stability requirements. The argument of stability reported by Gupta in [Gupt81] exceeds 62° all the way to order ten, so these methods may also be useful for the problems described in the next section.

Mention should also be made of many codes designed primarily for initial value problems in PDEs and of codes designed for particular user groups such as people solving chemical kinetic problems, network simulation problems, etc. PDE codes often make use of ODE codes via the method of lines. In some cases they use standard ODE codes. (See, for example, [Hyma79], [MaSi79], [MeSi81a], [MeSi81b], and [SiMa75]. For a survey, see [MaSw80].) Other codes are applicable to PDEs or ODEs, or combinations of the two. For example, the FORSIM package [Carv73]. There are also many other codes which have been tailored for specific users, e.g. FACSIMILE [CCJK77], and there are undoubtedly many other experimental codes of which I am unaware. I apologize to those whose codes I have overlooked below; I would like to hear about them.

4. Lightly damped oscillatory problems

When the Jacobian has large eigenvalues near to but to the left of

the imaginary axis, the solution to the problem does not exhibit oscillation (except possibly in a transient region) provided the driving terms are not adding energy at the corresponding frequency, but many numerical methods will have difficulty because they will fail to damp the oscillation or even amplify it. The BDF methods are particularly bad because they are not A-stable above second order and are strongly amplifying for eigenvalues close to the imaginary axis at moderate distances from the origin. These problems are very similar to stiff problems in that common numerical methods tend to amplify components that are not present in the true solution. Miranker [Mira81] has also pointed out that they are similar in that both can be viewed from the singular perturbation point of view. However, they are not similar when one considers that the classical stiff problem is "superstable" and any perturbation is rapidly damped in the actual system, whereas the lightly damped oscillating problem is not superstable and a perturbation can cause an effect in the solution for some time. If the eigenvalue is very close to the imaginary axis, an oscillatory component could take a long time to damp out, and, until that happens, almost all integrators would be forced to take a small step to follow the high frequency component. (It is, of course, possible to use methods which are exact for certain oscillatory functions. This is a form of exponential fitting and requires knowledge of the eigenvalues of the Jacobian. Such methods are not usually too useful for general purpose software.) It is tempting to consider methods that damp the oscillations rapidly so that

they do not hinder the numerical method. This is a satisfactory approach in those problems in which the oscillating term does not have a strongly nonlinear effect, so that damping, which is equivalent to a time averaging, does not change the slow components. A problem such as

$$\begin{aligned}y' &= iwy & w \gg 0 \\z' &= y^2 + y'^2\end{aligned}$$

cannot be damped because the value of $z(t)$ is strongly dependent on the amplitude of the oscillation. Such problems will be discussed in a later section. In this section we examine those problems in which the oscillation is either absent or can be damped without harm. For these problems we need methods that are "more stable" than the underlying problem near the imaginary axis to damp the unwanted components.

Gaffney has examined several codes for these types of problems having found codes based on the BDF formulas to be unsatisfactory. Four codes are compared in [Gaff81]. They are STRIDE [BuBC79], STINT [TeBP78], DIRK [Alex77], and BLEND [SkKo77]. Of these, BLEND and DIRK are experimental codes which in no sense are in a form for the general user, having been developed as a test bed for the underlying ideas in the papers cited. STRIDE is an implementation of implicit Runge Kutta methods available as a Fortran or Algol code. STRIDE uses singly-implicit RK methods [Burr78] which give stability out to an argument of at least 83° . (STRIDE can also be used for nonstiff equations by using fixed-point iteration instead of Newton's methods to solve the nonlinear

equations at each step, but it appears that its strength is for stiff equations with eigenvalues near the imaginary axis.) STINT is an implementation of cyclic composite methods. These achieve larger regions of absolute stability than is possible with simple linear multistep methods by applying a sequence of different multistep methods. If its order is limited to four, it is stable out to arguments of almost 81° , far superior to the BDF methods. BLEND uses a combination of Adams and BDF methods in a way that can give the stability of second derivative methods. The basic idea is that if the Adams method is written as $A = 0$, the BDF method as $B = 0$, and J is the Jacobian, the linear combination

$$A + J.B = 0$$

becomes the Adams method when J is small, and the BDF method when J is large. However, it also involves a term of the form Jy' in the $J.B$ component. For the linear ODE $y' = Jy$ we see that $y'' = Jy'$, so this term is equivalent to the second derivative. Consequently it can achieve the stability of second derivative methods--for example, it can be A-stable up to the fourth order. Furthermore, its accuracy is obtained from the accuracy of the underlying Adams and BDF methods, so it does not depend on the accuracy of the Jacobian for integration accuracy: an error in the Jacobian only affects the region of stability. Consequently, it is not necessary to re-evaluate the Jacobian frequently, as would be required in second derivative methods to be

discussed in a later section. Table 5.1 of [Gaff82] compares the maximum argument of stability of STINT, BLEND and the BDF methods. Even through order 7, BLEND is competitive with the 83^0 of the STRIDE implementation. Unfortunately BLEND has never been developed into a production code. (We do have a version embedded in LSODE in addition to the original modification of DIFSUB, but it is inadequately documented.) DIRK is an experimental implementation of diagonally implicit Runge-Kutta methods with a fixed order. The diagonal coefficients are chosen to be the same so that although at each step of an R-stage method it is necessary to solve R systems of non-linear equations versus the one system of multistep methods, the same LU factorization can be used for all equations. [Gaff82] presents a table of recommendations based on his tests on page 80 of the cited document. Since the method of choice varies with accuracy requirements, argument of stability needs, and user demands on the interface to the code, I refer the reader who needs to solve such problems to that document for further details. (Second derivative methods may also be competitive for these problems because they have a large argument of stability, although they tend to be inefficient for large problems with a non-constant Jacobian. See the remarks in the next section.)

5. Linear, constant-coefficient problems

Although there are probably not many problems in this category, considerable savings are possible when they occur. Clearly it is not

necessary to re-evaluate a Jacobian matrix, and techniques to avoid refactoring can be exploited. To my knowledge, there has been very little code development in this area except for the work by Enright [Enri77]. He modified two codes, one of Hindmarsh's GEAR codes and one of his second derivative codes SDBASIC [Enri74] and [Addi79]. A typical stiff code has to solve a nonlinear equation at each step and this is done with a Newton iteration which requires solving a linear system involving the matrix $[I - h\beta J]$ where J is the Jacobian. Refactorization is necessary for two reasons: the Jacobian changes or the stepsize h and/or the coefficient β changes. If the system is linear, the Newton iteration converges in one step, which is one saving. Furthermore, the Jacobian doesn't change so it is necessary to refactor only when h or β change. This is only a shift to the matrix, but unfortunately there are no known fast techniques for updating the LU decomposition after a shift of a matrix--indeed it appears that it cannot be done any faster than the original decomposition in general--so Enright uses a different decomposition, LHL^{-1} where L is unit lower triangular and H is upper Hessenberg. A shift of this is obtained by a shift of the H , while H can be factored in $O(n^2)$ steps. A further advantage of a constant Jacobian accrues in the case of second derivative methods, namely that the calculation of the second derivative, done by forming Jy' , does not require re-evaluation of the Jacobian at each step. (It is that re-evaluation that makes the second derivative methods less competitive than they would otherwise be when compared to methods that only loose

stability but not accuracy when the Jacobian is somewhat inaccurate.)

6. Large systems of equations

The major difficulty with large problems is matrix manipulation due to the need to solve linear equations a little more than once per time step. As the size of the problem increase, the percentage of time in this part of the program increases and it becomes important to look for fast methods. When the problem gets very large, the storage for the Jacobian and its decomposition may also be a problem. Fortunately, the Jacobian for large problems is almost always sparse because at least part of the problem has arisen from the partial discretization of a PDE. The use of direct sparse techniques has received a fair amount of attention and one version of LSODE, namely LSODES which uses the Yale sparse matrix package, is available. Less attention has been paid to iterative methods which will be needed for the really large problems, although much of the work done on iterative methods for symmetric problems is applicable when the system of ODEs arise from a method of lines approach to many PDEs. Miranker discusses the use of conjugate gradient methods for symmetric Jacobians in his book [Mira81]. An earlier code of Hindmarsh, GEARBI, (see [Hind76]) uses a block SOR method for problems arising from the method of lines applied to two space dimension PDEs. Hindmarsh has indicated that this will be upgraded to an LSODE code. However, in problems in which the Jacobian is non-symmetric and even has an irregular non-zero structure, most of

the currently used methods for solving equations iteratively are not too satisfactory. There is some work continuing on this problem, for example, [GeSa81], and this has been put in a version of LSODE, but it has not been distributed because it still represents an initial attempt.

If the problem is not so large that indirect methods spill out of primary storage, they can be used but can be very expensive. In that case, there is a lot of interest in techniques which minimize the number of times the Jacobian must be recomputed and/or the matrix refactored. There are few codes generally available based on these techniques but some of them are likely to appear in future codes. For example, Krogh and Stewart are working on a new code, STRUT, based on BDFs for stiff equations which goes to considerable effort to detect whether the Jacobian need be re-evaluated or the matrix refactored. See [KrSt81a], [KrSt81b], and this conference. The "constant leading coefficient" technique [JaSa80] has been used to reduce matrix refactorizations by holding the β coefficient constant. This changes as the stepsizes change in variable step methods. By sacrificing a small amount of accuracy it can be held constant.

7. Implicit differential equations

An implicit differential equation is written in the form

$$F(y, y', t) = 0$$

and can be handled directly by a number of methods including BDF and

certain implicit Runge-Kutta methods. A few codes have been written using the BDF methods but until recently they have not been widely distributed. Some of them have been embedded in proprietary network analysis software [HaBG71] so for a long time the only code available was a modified version of DIFSUB which was "experimental" in the worst meaning of the word (see [Gear71] and [Brow74]). Recently, two better codes have appeared, LSODAI [Pain81], and DASSL [Petz82]. Both of these are based on the BDF methods which replace the derivative in the implicit equation with a difference approximation. For example, the first order method is the Backward Euler method and is obtained by substituting $(y_n - y_{n-1})/h$ for y'_n to get the nonlinear equation

$$F(y_n, (y_n - y_{n-1})/h, t_n) = 0$$

to solve for y_n at each step. To my knowledge, no codes are available based on implicit Runge-Kutta methods, although such methods would work provided that the final stage is evaluated at the end point of the step and is used to provide the output value.

8. Differential-algebraic equations

These problems arise when an implicit system represents fewer differential equations than there are variables and equations because the Jacobian $\partial F/\partial y'$ is singular. For some time it has been known that difficulties can be encountered with some such systems, but only recently has the extent of some of these difficulties been comprehended.

The current situation is that we do not know how to solve the general problem, but we do know that all conventional ODE methods break down. Fortunately the difficulties do not arise very frequently, so large classes of problems can be solved by the software mentioned in the last section, but it is possible to construct examples which are physically reasonable but impossible to handle, and there are cases in which examples have arisen in practical simulation studies. The nature of the difficulty can be seen in some very simple model problems. Consider the system of three equations

$$x = g(t)$$

$$y = x'$$

$$z = y'$$

These are clearly not differential equations: their solution is uniquely defined independently of initial values. If a fixed step integrator is applied to these it is equivalent to applying numerical differentiation to $g(t)$. For example, if the Backward Euler method is applied, we calculate

$$x_1 = g_1$$

$$y_1 = (x_1 - x_0)/h$$

$$z_1 = (y_1 - y_0)/h$$

This numerical calculation uses initial values for x and y which are probably incorrect, so the values of y_1 and z_1 will be incorrect. In

fact, they will diverge as h tends to zero. However, it can be shown that, for fixed stepsize methods such as BDF methods, after a number of steps the answers have the required degree of accuracy. However, if the step varies or the problem has time dependent coefficients, we may run into difficulty. Precisely, we must talk about the index of nilpotency of the system. It is the maximum size of a subsystem of the sort above. In the example above it is three. If this index exceeds one, the time dependent problem cannot be solved by known methods. If the index exceeds two, variable stepsize methods also fail for the constant coefficient case. Details can be found in [Petz81] and [GeHP81]. A further report is also in preparation by the same authors.

Although these problems do not arise very often, they are known to arise on occasions. For example, [Pain81] states that the Navier Stokes equations have an index of two. They were integrated successfully by an implicit ODE code, perhaps because the critical parts of that problem are linear. It would be highly desirable for a code to detect the existence of a high index, but at the present we do not know how to do that. The symptom of such a problem in an automatic code is that the step control mechanism gets into trouble and reduces the step endlessly without reducing the error--possibly even increasing it, so that eventually the code reports that the error tolerance cannot be achieved, leaving the user to guess the reason. Furthermore, since it happens only when the Jacobian with respect to y' is singular, there are conventional stiff equations arbitrarily close to these very hard

problems. If such systems are integrated with existing integrators, it is possible that they will fail also.

9. Differential equations with discontinuities

Discontinuities in the derivatives of solutions cause difficulties in codes unless they are handled carefully. The difficulty arises because the code is proceeding under the assumption that the solution is well approximated by a smooth form such as a polynomial. This is clearly not true when there is a discontinuity in a derivative of order lower than the degree of the polynomial. If a one-step method is being used, the problem can be overcome by placing a mesh point at the discontinuity so that separate smooth approximations are used on each side of the discontinuity. (Even then, the code has to be organized carefully. If, for example, a trapezoidal rule is being used to integrate the problem $y' = f_i(t)$ where $i = 0$ for $t < t_c$ and $i = 1$ for $t > t_c$, the step ending at t_c must be integrated using f_0 , even when y' is evaluated at t_c in that step. After that step has been completed, the switch to f_1 should be made. This means that the code to evaluate the derivatives must contain a switch and be told by the integrator when the discontinuity has been passed.) If a multistep method is being used, the order of the multistep method must not exceed the order of the discontinuity in any steps which use information from both sides of the discontinuity, meaning that we also need to know the order of the discontinuity. Alternatively, we can simply restart a multistep method

to avoid this problem.

From this it is apparent that we need to know the location of the discontinuity and possibly its order. If the discontinuities occur at known values of the independent variable t the problem is fairly straightforward and is simply a matter of code organization. If the discontinuities occur at known values of the dependent variables, or given functions of them, it is necessary to first locate a discontinuity by monitoring the values of these functions, which we call switching functions. Carver's FORSIM code provides such options for locating discontinuities. For a discussion see [Carv77]. The location of a particular point by monitoring functions of dependent variables goes back, I think, to an early Illiac I RK code that was probably written by Nordsieck (I don't have any records on that, I just remember using it around 1958). It was also put into one of Fred Krogh's pioneering variable-order, variable-step codes under the name of the GSTOP feature.

In the two cases above, the detection of a discontinuity is possible because the user provides switching functions to indicate their presence. In the worst scenario, we do not even have that information (perhaps because the derivatives are being evaluated by a large section of code that cannot be analyzed reasonably, or depend on tabular data). Then we must also attempt to detect the presence of a discontinuity. Thus we see that there are three problems to handle with discontinuities:

Detection,
Location, and
Restarting.

Detection has been investigated for non-stiff problems in [GeOs81]. It is not clear that the technique used can be extended to stiff problems. Fortunately, we usually have switching functions to take care of this. The second step is equivalent to that of root finding for the equation $g(y,t) = 0$ where $y(t)$ is given by the numerical integrator. Hindmarsh will have a new version of LSODE, called LSODAR, for this problem, but it is not yet ready for distribution.

Restarting is important in multistep codes because they are usually less efficient in the first few steps which are taken at low order and small stepsize. RK codes can commence with a normally sized step immediately. The problem of starting at a modest order (four) with a reasonable stepsize is examined in [Gear80a] where a special form of a RK step is used to generate enough information for a four-step multistep method to continue. A code for the nonstiff method is available in a related report. There is a version of the technique for stiff problems, but as a general rule a problem will not be stiff after a discontinuity because transient components requiring a small stepsize for accuracy will have been stimulated. However, this is not always true. I have encountered one problem in which the problem remained stiff although there were discontinuities fairly frequently. A one-dimensional PDE

describing a heat conducting material which was melting as heat propagated had been converted into a system of ODEs by the method of lines. The underlying heat equation kept the problem stiff at all times. As the solid-liquid interface passed over a line, there was a discontinuity because of the latent heat effects and the jump change in the heat conductivity. This problem was causing difficulty for any of the integrators the user was trying. (As is common, I don't know if it was finally solved. The user did not return after the third visit, either because by suggestion of that meeting worked or he decided that I was wasting his time!)

10. Problems with high-frequency components

These problems are ones in which the high frequency components are an integral part of the problem and must not be damped because such damping would change the solution, or cannot be damped because they arise from non-linear oscillators. Petzold developed a preliminary code as a modification of DIFSUB in her thesis [Petz78]. The underlying method is called a multirevolutionary method because it attempts to step over many cycles (or revolutions, in the case of satellite orbits where the method was first investigated). The basic idea is that a single cycle is investigated numerically by integrating the original differential equation with a conventional integrator. The results from this are used to predict the rate of change of the behavior of the starting point of each cycle. If the behavior of the starting points is

"smooth" it is possible to extrapolate forward over many cycles without the need to integrate over each one. This extrapolation looks like an integration method with modifications proportional to the ratio of the period to the stepsize. We are using generalizations of the Adams and BDF methods for this extrapolation. Complications arise because the period may be changing so must be recalculated from the solution, and because the problem may be very stable giving rise to the same effect as stiffness in the multirevolutionary method. Gallivan and myself have continued that work at Illinois and are writing a code embodying a number of new techniques for detecting the oscillatory behavior, deciding when to use the multirevolutionary method because it might be more efficient than a conventional method, and deciding when to use the generalized BDF rather than the generalized Adams methods. (See [GeGa81].)

Problems with more than one frequency probably cannot be handled by this method, so these remain something of an open problem. Methods employing Fourier expansions are one possibility, but I know of no general software embodying these.

11. Partitioned systems

It may be profitable to split a large system of equations into one or more subsystems in order that each subsystem can be handled by different methods and/or different stepsizes. Different methods could be profitably used if the stiffness of a system is confined to a

subsystem. Different stepsizes could be used if the bandwidth of one subsystem is significantly less than that of another. Even if the same method and stepsize is used in all subsystems, gains are still possible if the matrix arithmetic associated with stiffness is avoided in the non stiff subsystems. There are relatively few codes available embodying these techniques. [Sode80] describes a code, DASP3, which uses a conventional four-stage RK explicit method for one subsystem, and a three-step BDF method for the other. It is necessary for the user to partition the system into the form

$$y' = f(y,z,t)$$

$$z' = g(y,z,t)$$

a priori. The code does not make this determination automatically. Various approaches to using different stepsizes are discussed in [Gear80], and a code which will allow a system to be divided into a number of subsystems, each to be handled by different, automatically chosen, stepsizes and orders, is being written by D. Wells at the University of Illinois as part of his Ph.D. thesis. This code also requires the user to do the partitioning. Of course, one approach to automatic partitioning is to treat each equation as a separate subsystem and leave it free to select its own stepsize, order, and even method (e.g. stiff versus nonstiff). Unfortunately, it appears that the overhead of such an approach will be so large that any savings in the actual integration will be negated.

One difficulty with partitioning is that it is based on the hope that certain properties are confined to subsystems. In general, this is not true. For example, in a linear system in which the Jacobian is not reducible, the dominant eigenvectors (those corresponding to the large eigenvalues) will have components in all dependent variables, so that stiffness cannot be confined to a subsystem. If a transformation to a basis of eigenvectors were to be made, we could confine stiffness to a subsystem, but this is not usually practical. However, it is not necessary to have a basis of eigenvectors, it would suffice to have a basis, part of which spanned the dominant eigenspace and the other part of which spanned the remainder. (This is really what the iterative methods are attempting to do in [GeSa81]).) An approach to this is the technique in [EnKa79] in which the iteration matrix is decomposed into an LHL^{-1} form where H is upper Hessenberg and L is a combination of a lower triangular matrix and permutations chosen to put the large components into the first part of H. In effect, the lower part of H is ignored, corresponding to ignoring components due to the non-dominant eigenspace. In practice, the reduction is terminated after the large components have been handled. Since this matrix is used to solve the linear equations arising in the Newton iteration, the effect of discarding the small part of H is to handle the stiff components by a Newton step, and the nonstiff components by a functional iteration step. In [GeSa81] it is shown how this can be used to treat the stiff components with BDF and the non-stiff components with Adams methods.

However, there are no distributed codes based on these techniques to my knowledge.

12. Delayed equations

Delayed, or retarded, differential equations can also exhibit stiffness. There has been some work recently on codes for these problems. Among these are codes based on STFDIF, a composite, block, one-step method by the group at Syracuse (see the paper by Bickart at this conference), and a code based on DIFSUB by Roth and Watanabe at Illinois ([Roth80] and [WaRo82]).

13. Global error estimation and control

Most codes attempt to control no more than an estimate of the local error, that is, the error introduced in a single step. If "error-per-unit-step" control is used, the stepsize is adjusted so that the error permitted is believed to be less than a tolerance, TOL, scaled by the stepsize used. Then, if there is no error growth due either to the problem or the method, the error after an interval of length L should be bounded by $L \cdot \text{TOL}$. However, the actual error depends on the amplification, or otherwise, of the errors by the differential equations and method. Some codes have been written for the nonstiff problem to estimate global errors, for example, GERK [ShWa78]. When the problem is stiff, errors introduced in the early stages into the stiff components decay due to the super stability of the problem (provided the method

also exhibits the same stability). In that case, the best stepsize strategy is not an error-per-unit step strategy, but something closer to an error-per-step strategy in which the local error estimate is controlled to be less than TOL. Dahlquist [Dahl81] is investigating the control of global error in stiff equations. It is an important problem because the relationship between the user supplied error control parameter TOL and the global error is even less understood by the typical user than in nonstiff problems.

14. Automatic method switching

Until recently, all codes either required the user to specify whether a stiff or nonstiff method should be used, or provided no choice. Since most problems are not stiff everywhere as the behavior in an initial transient is best handled by a nonstiff method, it is highly desirable to switch between nonstiff and stiff methods. A number of Shampine's codes for nonstiff problems have detection mechanisms that warn users of potential problems such as stiffness. Stewart has investigated using these to switch automatically from Adams methods as implemented in Shampine's codes STEP, to BDF methods for stiff equations. The code STRUT being written by her and Krogh is expected to include this mechanism. Petzold has also investigated techniques for this [Petz82b], and has modified a version of LSDOE with Hindmarsh. Although not distributed yet, it is receiving experimental use at Livermore under the name LSODA.

At Illinois we are using similar techniques to switch to and from stiff methods in the code being written for oscillatory methods. The basic idea is to estimate the stepsizes that can be used in the stiff method and in the non-stiff method. In the case of nonstiff methods, the stepsize must be limited to keep an estimate of the largest eigenvalues inside the region of absolute stability. These are then divided into estimates of the cost per step of each method (stiff methods are much more expensive per step because of the matrix operations), and the smallest resulting value is used to select the method. In our codes this is done for each of several orders in Adams and BDF methods so that order selection is handled at the same time. One difficulty that has been encountered (and requires careful tuning of the code) arises if the stepsize in the nonstiff integrator is not controlled strongly enough to keep the method absolutely stable. In that case the numerical solution exhibits small oscillations at the error tolerance level. The integrator calculates stepsize estimates by estimating derivatives of the solution using numerical differentiation. The oscillating errors give rise to artificially large derivative estimates which cause the stepsize estimates to be too small, so that it appears that stability is not a problem. Since the BDF methods have larger error coefficients than Adams methods, a switch is never made to the BDF methods. It has proved necessary to ensure that the stepsize is kept small enough in Adams methods to be certain that the method is well inside the region of absolute stability.

15. Smooth solutions

The output of most integrators is far from being a smooth function of its inputs because if a change is made to one of the inputs, say an initial value, the stepsize and order sequence may change drastically, causing a significant change in the output. In applications in which the user is only concerned with the difference between the numerical and true solutions this is not a problem. However, there are many situations in which the output of the integrator is the input to another program such as an optimizer. An example of this occurs in the estimation of reaction rates in chemical kinetic equations. These are often solved by performing a least-squares fit to observed concentrations by re-integrating the equations with different values of the reaction rates. We have been particularly concerned with this problem because we use an integrator to determine the behavior of one cycle for use in the multirevolutionary methods for oscillating problems.

We would like to see methods developed which obey certain smoothness properties. In particular we would like methods for which we can get not only bounds on the global error of the solution, but we can get bounds on the difference between the numerical derivative of the solution with respect to any input parameter and the corresponding derivative of the true solution. A student, Vu Thu, at Illinois is working on this problem for ODEs. The critical difficulty is that a

code has to avoid making any "absolute" decisions such as step rejection, because these invariably lead to discontinuous behavior. The code cannot include any "if" statements, MAX and MIN functions, or similar functions with discontinuous properties.

16. Conclusions

Although there are a number of good codes for standard stiff problems there are many areas for which code development is still needed, and other areas in which we do not yet have adequate methods. The experience with the early codes suggests that more effort in code development after the first "production" version has been written is needed if a code is to become widely used.

17. Acknowledgements

I would like to thank the organizers of this conference for providing a forum and motivation. I would also like to thank a number of people who have told me about recent developments. Among these are Ted Bickart, Wayne Enright, Alan Hindmarsh, Fred Krogh, Linda Petzold, Larry Shampine, and Bob Skeel.

18. References

- [Addi79] Addison, C. A., Implementing a stiff method based upon the second derivative formulas, M.Sc. thesis, Dept. Computer Sci. Rpt. 130, Univ. Toronto, Canada, 1979.

- [Addi80] Addison, C. A., Numerical methods for a class of second order ODEs arising in structural dynamics, Ph.D. thesis, Dept. Computer Sci. Rpt. 147, Univ. Toronto, Canada, 1980.
- [Alex77] Alexander, R., Diagonally implicit Runge-Kutta methods for stiff ODEs, SIAM J. Numer. Anal. 14, (6), 1977, 1006-1021.
- [BaDe81] Bader, G. and P. Deuflhard, A semi-implicit mid-point rule for stiff systems of ordinary differential equations, Institute for Applied Mathematics Reprint No. 114, Univ. Heidelberg, W. Germany, April 1981.
- [BrGe73] Brown, R. L. and C. W. Gear, Documentation for DFASUB--A program for the solution of simultaneous implicit differential and nonlinear equations, Dept. Computer Sci. Rpt. UIUCDCS-R-73-575, Univ. Illinois, July 1973.
- [Burr78] Burrage, K., A special family of Runge-Kutta methods for solving stiff differential equations, BIT 18, 1978, 22-41.
- [BuBC79] Butcher, J. C., Burrage, K. and F. H. Chipman, STRIDE: Stable Runge-Kutta integrator for differential equations, Dept. Mathematics Rpt. 150, Univ. Auckland, New Zealand, August 1979.
- [Carv73] Carver, M. B., FORSIM: A FORTRAN-oriented simulation package for the automatic solution of partial and ordinary differential equation systems, Atomic Energy of Canada Rpt. AECL-4608, 1973.
- [Carv77] Carver, M. B., Efficient handling of discontinuities and time delays in ordinary differential equation systems, Proc. Intl. Conf. SIMULATION '77, Montreaux, Switzerland, June 22-24, 1977, 153-158.
- [CaMa80] Carver, M. B. and S. R. MacEwan, Automatic Partitioning in ordinary differential equation integration, Progress in Modelling and Simulation, Academic Press, London, 1982 (F. E. Cellier, ed.).
- [Cell82] Cellier, F. E., Stiff computation: where to go?, Swiss Federal Institute of Technology Zurich, Informal Communication.
- [CCJK77] Chance, E. M., Curtis, A. R., Jones, I. P. and C. R. Kirby, FACSIMILE: a computer program for flow and chemistry simulation, and general initial value problems, Harwell Rpt. AERE-R8775, Oxfordshire, England, December 1977.

- [Dahl81] Dahlquist, G., On the control of the global error in stiff initial value problems, Dept. Numerical Analysis and Computing Science Rpt. TRITA-NA-8109, Royal Institute of Technology, Stockholm, Sweden.
- [DeWa81] Dew, P. M. and J. E. Walsh, Library routines for solving parabolic equations, Trans. on Math. Software 7, (3), September 1981, 295-314.
- [DeWe79] Dew, P. M. and M. R. West, Estimating and controlling the global error in Gear's method, BIT 19, 1979, 135-137.
- [Enri74] Enright, W. H., Optimal second derivative methods for stiff systems, in Stiff Differential Systems (R. A. Willoughby ed.), Plenum Press, 1974, 95-109.
- [Enri74] Enright, W. H., Second derivative multistep methods for stiff ordinary differential equations, SIAM J. Numer. Anal. 11, 1974, 321-332.
- [Enri77] Enright, W. H., The efficient solution of linear constant-coefficient systems of ODes, Dept. Computer Sci. Tech. Rpt. 111, Univ. Toronto, October 1977.
- [EnHL75] Enright, W. H., Hull, T. E. and B. Lindberg, Comparing numerical methods for stiff systems of ordinary differential equations, BIT 15, 1975, 10-48.
- [EnKa79] Enright, W. H. and M. S. Kamel, Automatic partitioning of stiff systems and exploiting the resulting structure, ACM Trans. Math. Software 5, (4), December 1979, 374-385.
- [Gaff82] Gaffney, P. W., A survey of FORTRAN subroutines suitable for solving stiff oscillatory ordinary differential equations, Oak Ridge National Laboratory Rept. ORNL/CSD/TM-134, Oak Ridge, Tennessee, February 1982.
- [Gear71] Gear, C. W., Simultaneous numerical solution of differential-algebraic equations, IEEE Trans. Circuit Theory CT-18, (1), January 1971, 89-95.
- [Gear74] Gear, C. W., What do we need in programming languages for mathematical software?, Dept. Computer Sci. Rpt. UIUCDCS-R-74-652, Univ. Illinois, May 1974.
- [Gear80a] Gear, C. W., Runge-Kutta starters for multistep methods, TOMS 6, (3), September 1980, 263-279.

- [Gear80b] Gear, C. W., Automatic multirate methods for ordinary differential equations, Proc. IFIP 80, North-Holland Publishing Company, 1980, 717-722.
- [Gear81] Gear, C. W., Numerical solution of ordinary differential equations: is there anything left to do?, SIAM Review 23, (1), January 1981, 10-24.
- [GeGa81] Gear, C. W. and K. A. Gallivan, Automatic methods for highly oscillatory ordinary differential equations, Proc. Biennial Dundee Conference on Numerical Analysis, Dundee, Scotland, 1981.
- [GeOs81] Gear, C. W. and Østerby, O., Solving ordinary differential equations with discontinuities, Dept. Computer Sci. Rpt. UIUCDCS-R-81-1064, Univ. Illinois, September 1981.
- [GeSa81] Gear, C. W. and Y. Saad, Iterative solution of linear equations in ODE codes, Dept. Computer Sci. Rpt. UIUCDCS-R-81-1054, Univ. Illinois, January 1981.
- [GeHP81] Gear, C. W., Hsu, H. H. and L. Petzold, Differential-algebraic equations revisited, in Numerical Methods for Solving Stiff Initial Value Problems Ed. Dahlquist, G., and Jeltsch, R., Proc. Oberwolfach Meeting, June 18-July 4, 1981.
- [Glad79] Gladwell, I., Initial value routines in the NAG library, ACM Trans. Math. Software 5, (4), December 1979, 386-400.
- [Gupt79] Gupta, G. K., DSTIFF: a set of subroutines for solving stiff equations, user's guide part 1, Dept. Computer Sci. Rpt. 3, Monash Univ., Clayton, Victoria, Australia.
- [Gupt81] Gupta, G. K., Description and evaluation of a stiff ODE code DSTIFF3, Dept. Computer Sci. Rpt., Monash Univ., Clayton, Victoria, Australia.
- [HaBG71] Hachtel, G. D., Brayton, R. K. and F. G. Gustavson, The sparse tableau approach to network analysis and design, IEEE Trans. on Circuit Theory 18, (1), January 1971, 101-113.
- [Hind76] Hindmarsh, A. C., Preliminary documentation of GEARBI: solution of ODE systems with block iterative treatment of the Jacobian, Lawrence Livermore Laboratories Rpt. UCID-30149, December 1976.

- [Hind79a] Hindmarsh, A. C., On numerical methods for stiff differential equations--getting the power to the people, Lawrence Livermore Laboratories Rpt. UCRL-83259, Univ. California, December 1979.
- [Hind79b] Hindmarsh, A. C., A collection of software for ordinary differential equations, Lawrence Livermore Laboratories Rpt. UCRL-82091, Univ. California, January 1979.
- [Hind80] Hindmarsh, A. C., LSODE and LSODI, two new initial value ordinary differential equation solvers, ACM SIGNUM Newsletter 15, (4), December 1980, 10-11.
- [Hind81] Hindmarsh, A. C., ODE solvers for use with the method of lines, Lawrence Livermore Laboratories Rpt. UCRL-85293, Rev. 1, Univ. California, March 1981.
- [Hyma79] Hyman, J. M., MOLID: A general purpose subroutine package for the numerical solution of partial differential equations, Report, Los Alamos Scientific Laboratory, 1975.
- [JaSa80] Jackson, K. R. and R. Sacks-Davis, An alternative implementation of variable stepsize multistep formulas for stiff ODEs, ACM Trans. on Math. Software 6, (3), March 80, 295-318.
- [KrSt81a] Krogh, F. T. and K. Stewart, Implementation of variable-step BDF methods for stiff ODEs, in Numerical Methods for Solving Stiff Initial Value Problems Ed. Dahlquist, G., and Jeltsch, R., Proc. Oberwolfach Meeting, June 18-July 4, 1981.
- [KrSt81b] Krogh, F. T. and K. Stewart, Asymptotic ($h \rightarrow \infty$) stability for BDF's applied to stiff equations, Jet Propulsion Laboratory Computing Memorandum 478, Pasadena, California, June 1981.
- [MaSw80] Machura, M. and R. A. Sweet, A survey of software for partial differential equations, ACM Trans on Math. Software 6, (4), April 1980, 461-488.
- [MaSi79] Madsen, N. K. and R. F. Sincovec, PDECOL: General collocation software for partial differential equations, ACM Trans. Math. Software 5, (3), September 1979, 326-351.
- [MeSi81] Melgaard, D. K. and R. F. Sincovec, General software for two-dimensional nonlinear partial differential equations, Trans. Math. Software 7, (1), March 1981, 106-125.

- [MeS181] Melgaard, D. K. and R. F. Sincovec, Algorithm 565. PDETWO/PSETM/GEARB: solution of systems of two-dimensional nonlinear partial differential equations [D3], ACM Trans. Math. Software 7, (1), March 1981, 126-135.
- [Mira81] Miranker, W. L., Numerical methods for stiff equations, D. Reidel Pub. Co., Dordrecht, Holland. 1981.
- [MiWa76] Miranker, W. L. and G. Wahba, An averaging method for the stiff highly oscillatory problem, Mathematics of Computation 30, 1976, 383-399.
- [Pain81] Painter, J. F., Solving the Navier-Stokes equations with LSODI and the method of lines, Lawrence Livermore Laboratory Rpt. UCID-19262, Univ. California-Livermore, 1981.
- [Petz78] Petzold, L. R., An efficient numerical method for highly oscillatory ordinary differential equations, Dept. Computer Sci. Rpt. UIUCDCS-R-78-933, Univ. Illinois, Ph.D. Thesis, 1978.
- [Petz81] Petzold, L. R., Differential/algebraic equations are not ODEs, Sandia National Laboratories-Livermore Rpt. SAND81-8668, April 1981.
- [Petz82a] Petzold, L., A Description of DASSL: A differential/algebraic system solver, to appear, Proc. IMACS World Congress, 1982.
- [Petz82b] Petzold, L., Automatic selection of methods for solving stiff and non-stiff systems of ordinary differential equations, to appear, SIAM J. on Scientific and Statistical Computing.
- [Roth80] Roth, M. G., Difference methods for stiff delay differential equations, Dept. Computer Sci. Rpt. UIUCDCS-R-80-1012, Univ. Illinois, Ph.D. Thesis, December 1980.
- [Sham77] Shampine, L. F., Stiffness and nonstiff differential equation solvers, II: detecting stiffness with Runge-Kutta methods, ACM Trans. Math. Software 3 (1), 1977, 44-53.
- [Sham79] Shampine, L. F., Global error estimation for stiff ODEs, Sandia National Laboratories-Albuquerque Rpt. SAND79-1587, October 1979.
- [Sham80a] Shampine, L. F., Type-insensitive ODE codes based on implicit a-stable formulas, Sandia National Laboratories-Albuquerque Rpt. SAND79-1444, September 1980.

- [Sham80b] Shampine, L. F., Implementation of Rosenbrock methods, Sandia National Laboratories-Albuquerque Rpt. SAND80-2367J, November 1980.
- [ShWa78] Shampine, L. F. and H. A. Watts, Algorithm 504 GERK: global error estimation for ordinary differential equations, ACM Trans. Math. Software 2, (2), 1978, 200-203.
- [SiMa75a] Sincovec, R. F. and Madsen, N. K., Software for nonlinear partial differential equations, ACM Trans. Math. Software 1, (3), September 1975, 232-260.
- [SiMa75b] Sincovec, R. F. and N. K. Madsen, Algorithm 494 PDEONE. Solutions of systems of partial differential equations, ACM Trans. Math. Software 1, (3), September 1975, 262-263.
- [SkKo77] Skeel, R. and A. Kong, Blended linear multistep methods, ACM Trans. Math. Software 3, (4), 1977, 326-345.
- [Sode80] Soderlind, G., DASP3--A program for the numerical integration of partitioned stiff ODEs and differential-algebraic equations, TRITA-NA-8008, Royal Institute of Technology, Stockholm, Sweden.
- [TeBP78] Tendler, J. M., Bickart, T. A. and Z. Picel, A stiffly stable integration process using cyclic composite method, ACM Trans. Math. Software 4, (4), 1978, 339-368.
- [WaRo82] Watanabe, D. S. and M. G. Roth, The stability of difference formulas for delay differential equations, submitted to SIAM J. Numerical Analysis.

PITFALLS IN THE COMPARISON OF NUMERICAL METHODS FOR
STIFF ORDINARY DIFFERENTIAL EQUATIONS

by

W.H. Enright

Dept. of Computer Science

University of Toronto

Toronto M5S 1A7 Canada

Abstract

The difficulties inherent in attempts to evaluate methods designed for stiff equations are identified and discussed. It is observed that while evaluations of individual methods are worthwhile and often provide insights into the method's performance attempts to compare the performance of different methods seldom prove fruitful because of the many factors that contribute to the effectiveness of a stiff solver. Some recent progress and applications of the evaluation of stiff methods are reviewed and recommendations are made regarding where evaluations of stiff methods can be most helpful.

1. Introduction

During the past decade several new approaches for the numerical solution of stiff equations have been proposed and many of these have been implemented as software packages intended for

general use. In this investigation we will identify some of the pitfalls inherent in attempts to evaluate the performance of stiff methods and we will review some of the progress that has been made. We will distinguish between the comparison of different methods and the assessment of an individual method. One of our recommendations will be that while the latter is feasible and can be very worthwhile the former is controversial and subject to many difficulties.

It must be emphasized that comparing or assessing stiff methods should not be thought of as a static activity. While it is true that attempts to compare stiff methods has not lead to many definitive statements about the relative merits of various approaches, it has lead to a better understanding of numerical methods and has resulted in improvements and modifications which otherwise may not have been possible.

Before discussing the evaluation and comparison of methods in detail we will review what a stiff problem is; what comprises a stiff method; and what a stiff method is attempting to do. In doing this we will identify those features which cause the most difficulty when one attempts to compare stiff methods. In the next section we will clarify what a stiff problem is and discuss particular properties that a stiff problem can exhibit which can have an influence on the performance of a method. We will then introduce the various components that comprise a stiff method and discuss how each can contribute to the performance of a method. Indeed one must appreciate that when assessing a stiff method one is not only assessing the underlying formula (or class

of formulas) but also the numerous other strategies which collectively determine the method's performance.

In the fourth section we will discuss the objectives of contemporary stiff methods and how these objectives are achieved. We will discuss in particular how the accuracy of a numerical method is controlled and suggest an approach which could lead to a more meaningful and uniform interpretation of accuracy for stiff solvers. Other special features of problems or methods which complicate the evaluation of a method will then be identified. In the last section we will review some of the progress that has been made and indicate where future advances are most likely.

2. Properties of a Stiff Problem

While the term 'stiffness' is widely used it is a difficult concept to define precisely. Shampine and Gear (1979) discuss the notion of stiffness in some detail and identify properties they feel must be present for a problem to be considered stiff. It must be understood that stiffness is a characteristic that depends on the differential equation, the length of integration, the initial conditions and the accuracy requirement and that in addition a problem can be stiff for parts of the integration and nonstiff for other parts. We will consider a problem to be stiff if during its solution by classical methods (such as an explicit Runge Kutta method or an Adams multistep method) the stepsize is severely constrained by the numerical stability of negligible transient components rather than the accuracy of the

significant components. In particular we are assuming that some component of the solution of a stiff problem must become negligible (relative to the accuracy requirement). These components generally will be significant only in the initial or transient region. Problems with high frequency undamped components are a special case and can only be considered stiff if the initial conditions are such that the amplitudes of the high frequency components are small relative to the accuracy requirement. For this class of problems the inherent cost of the integration is extremely sensitive to the accuracy requirement and the specified initial conditions - a situation that is definitely not typical of stiff problems. (For this reason some people would not consider this type of problem to be stiff.)

Stiff problems often involve large systems of equations and when they do it is essential that any special structure be exploited. For example the problem could have a sparse Jacobian, it could possess only a few transient components or it could be linear. In each of these cases methods can exploit the corresponding structure although the extent to which it can be exploited is very method dependent as is the ability to automatically detect the existence of such structure.

3. Components of a Stiff Method

It has long been recognised that a method for initial value problems in ODEs consists of much more than a formula (or a class of formulas) for advancing the solution in a step-by-step fashion. It must also include a local error estimator and a stepsize choosing strategy designed to control the accuracy of the inte-

gration. In methods designed for stiff systems these latter two strategies can have a significant impact on the overall efficiency and reliability of a method as can the iteration control strategy and the linear equation solver that is used. Since the time required to solve a stiff problem is often dominated by the time required to solve the systems of equations that arise on each time step, strategies involved in this iteration can also be critical. While most methods solve these systems of equations using a modified Newton iteration they differ in their respective stopping criterion.

Methods which are not based on A-stable formulas often encounter difficulties when solving problems having eigenvalues of the Jacobian close to the imaginary axis. This difficulty is often compounded when the step-choosing strategy is based only on asymptotic ($h \rightarrow 0$) properties of the local error estimator. Methods that include an heuristic for detecting when numerical stability is influencing the error estimates and which choose the stepsize (and possibly the order) accordingly can deal with this difficulty in a more robust way.

4. Objectives of a Stiff Method

The methods that have been developed for stiff problems over the past decade have tended to have similar calling sequences, but the interpretation of some of the parameters can be quite different for different methods. This is particularly true of the error control strategy and the way in which the error is measured.

Methods inevitably attempt to control some measure of the global error (relative to the requested accuracy, TOL) by monitoring and controlling some measure of the local error on each step of the integration. While the relationship between local and global errors for stiff systems is not well understood, most methods, primarily on the basis of numerical evidence, attempt to ensure that the magnitude of the local error per step is bounded by TOL on each step. When this type of error control is used the relationship between the global error and TOL is inherently method-dependent as the bound on the global error depends on the stepsizes used. Contrary to popular belief it is not true that control of the magnitude of the local error per step alone will guarantee control of the global error. A method which chooses a sufficiently small constant stepsize $h > 0$, and generates the sequence of approximation (x_i, y_i) $i=1,2,\dots,N$ with $x_i = x_0 + ih$, $y_i \approx y_0$ when applied to the stiff problem

$$y' = f(x,y), \quad y(x_0) = y_0 \quad (1)$$

will satisfy the local error per step criterion but will produce a numerical solution of dubious value. When interpreting the accuracy of a solution produced by a method which claims only to control the magnitude of the local error on each step, a user must know more information (such as the average stepsize) before he is justified in having any confidence in the accuracy of the solution. Since most methods do use this type of error control they produce solutions whose global accuracy is related to the

specified tolerance in a very method-dependent fashion. If one is only interested in endpoint accuracy this difficulty is not usually as acute, but if one wishes accuracy throughout the interval this difficulty can make comparison especially awkward.

One way to overcome this difficulty would be to ensure that the next generation of codes are based on a comparable interpretation of the parameter TOL. Direct control of the magnitude of the local error per unit step would result in such a method-independent relationship between a global error bound and TOL but this is known to be inefficient for stiff problems. An alternative approach would be to control the magnitude of the local error per step but to combine this with local extrapolation. Since extrapolation would alter the stability properties of the method, one would have to develop new formulas or use a lower order predictor in the local error estimator. A more promising approach which could also lend to a method independent interpretation of tolerance is based on a variable local error per unit step designed to control the magnitude of the defect of the numerical solution. If a method, when applied to the stiff initial value problem (1), produces the discrete numerical solution (x_i, y_i) $i=1,2,\dots,N$ as an approximation to $(x_i, y(x_i))$ $i=1,2,\dots,N$ and there exists a continuous piecewise differentiable function $z(x)$ defined on $[x_0, x_N]$ interpolating (x_i, y_i) $i=1,2,\dots,N$ such that

$$z' = f(x, z) + \delta(x),$$

then $\delta(x)$ is the defect associated with the approximate solution $z(x)$. This point of view has been applied successfully in the analysis of non-stiff methods (see for example Hanson and Enright (1981)) and it is hoped that it will prove fruitful in the analysis of stiff methods as well. This analysis will be complicated by the fact that, to account for the very stable global error control observed in practice one will have to consider the direction of the local error (and hence the defect) as well as its magnitude. We will return to this point in the next section.

Most codes provide options which dictate how the error should be measured and this complicates the comparison of methods as the options provided by different methods are often not compatible. Pure absolute error control is the simplest but it can be inappropriate for stiff problems as they often have solutions with components of widely different magnitude. Componentwise relative error has been suggested for stiff problems, but control of local error in a relative sense does not imply that the relative global error is being controlled and indeed it is very difficult to say what effect this has on the global error. On the other hand, if one interprets global accuracy from the defect point of view then componentwise control of the local error relative to the magnitude of the derivative would lead to a defect that is small relative to $f(x,z)$. Shampine has recently argued (Shampine (1982)) that for most problems a componentwise weighted absolute error control is appropriate with the weights depending on the solution. As he points out this makes it awkward for users to choose the appropriate weights without a few preliminary integrations, but for test problems this is probably the best error measure to use.

5. Other complications

In addition to the difficulties discussed above that arise because of inherent properties of stiff problems, stiff methods or the error control strategy there are other difficulties that can arise which complicate any attempt to compare stiff methods some stiff problems have an unstable manifold of solutions close to the solution that is being sought. For example it is well known that in stiff problems arising from chemical kinetics the problems are stable and well conditioned provided the concentrations remain nonnegative, but a problem can become mathematically unstable if a negative concentration is introduced. During a numerical solution it is possible that a small negative concentration (relative to the accuracy requirement or the round-off level) will be introduced and this can lead to an unstable trajectory being followed (Edsberg (1976) discusses this difficulty in more detail). Although some methods may be more prone to this difficulty than others it is not clear that it should be considered a fault of the method as long as the specified error control strategy is being respected. Certainly one instance of this difficulty can distort a comparison of methods.

Recently some new techniques for stiff equations have been proposed for which even linear analysis is difficult. The familiar stability definition such as A-stability and $A(\alpha)$ -stability are based on the assumption that analysing the scalar equation $y' = \lambda y$ for each eigenvalue λ of the matrix A will allow one to characterize the stability and propagation of local errors for the stiff system, $y' = Ay$. This assumption is valid for most

stiff methods since the nonsingular similarity transformation which reduces A to its Jordan canonical form will also 'uncouple' most methods and reduce the original problem to an equivalent sequence of scalar problems (of course the equivalent error measures will change but for a given sequence of stepsizes and a fixed formula the solutions will be equivalent). Some recent approaches such as those based on nonlinear combinations of the derivative (such as those suggested by Lambert (1974)) or those that use different formulas for different components of the system (such as those suggested by Hofer (1976) or Soderlind (1981)) do not satisfy this assumption. When a method does satisfy this assumption then it is clear that when one is in the transient region $|h\lambda_i| \ll 1$ for eigenvalues λ_i corresponding to the smooth components and consequently there will be virtually no error (except for round-off) in the direction of these components with the local error almost entirely in the subspace spanned by the transient components. Similarly after the transient region is passed and the method is in the smooth region we should have $|h\lambda_j| \gg 1$ for eigenvalues λ_j corresponding to the negligible transients of the numerical solution. For these components stability is ensured and their magnitude will remain negligible relative to the accuracy requirement and hence the local error must be mostly in the direction of the smooth components. It is not obvious that this will also be the case for methods which don't satisfy this assumption.

Some methods require an analytic Jacobian while others can use either an analytic Jacobian or an approximate Jacobian based on numerical differences. Even when an approximate Jacobian

is allowed the performance of a method can be very sensitive to the stepsize used in determining the approximate Jacobian (see Gaffney (1982)) and this can be very disconcerting.

6. Progress that has been made.

Any comparison of stiff methods will be based on monitoring the performance of methods as they solve a representative collection of stiff problems over a range of tolerances. It is clear that one should include measures of reliability as well as efficiency when describing the performance of a method and it is also clear that there should be several measures of efficiency available. The relative importance of these efficiency measurements will depend, among other things, on the size and complexity of the differential equation. From our discussion of the 'objectives' of a stiff method it should be apparent that the choice of an appropriate reliability measure is a controversial issue and while a few possibilities exist none is entirely satisfactory. In our early comparisons of stiff methods (Enright et al. (1975)) we gathered a collection of test problems; proposed measures of reliability and efficiency; and reported a comparison of several modified versions of existing methods. The modification to existing methods were made necessary because of the variable local error per unit step control we assumed in order to justify our choice of reliability measure. It was clear to us that while this type of testing could be used to identify weaknesses of a particular method or to compare modified versions of the same code it was not possible to identify the 'best' method in any sense. Our primary goal in that study was

to establish a testing methodology which could be used to compare the performance of methods on any class of stiff problems. It is rather unfortunate that the main impact of that investigation seems to have been that the test problems we used have been used as a 'representative set of stiff problems' in several subsequent investigations. This was not our intention as the problem set was proposed more to illustrate and justify the methodology we were introducing. Shampine (1982) has recently made several suggestions and corrections which would certainly be necessary if these test problems are to be used as a standard test set.

After our initial testing it became clear that while comparisons of methods was very difficult and controversial (since it involved modifying methods) the assessment of an individual method was feasible and much less controversial. In Enright (1980) we reported on the use of testing tools we had developed to automatically assess an unmodified stiff method on any class of stiff problems. The measurements of reliability were method-dependent and various levels of detailed statistics could be provided. This package of testing tools has been widely distributed and results obtained using this package for a selection of contemporary stiff methods are summarized in Addison (1982). As Addison notes there are still few methods which operate efficiently at stringent tolerances (essentially only those based on variable-order multistep formulas) while at larger tolerances several good methods are available. It is also clear from Addison's results that a robust implementation of the BDF formulas such as the well known code GEAR (Hindmarsh (1973)) is

still the benchmark that new codes should be compared to. Its main well-known weakness is on problems with eigenvalues of the Jacobian near the imaginary axis and, for this class of problems, several acceptable alternatives are available.

In the future we feel there are three important applications where the detailed evaluation of a stiff method's performance will prove invaluable. The first is when one has a class of problems of a special type and he wishes to choose an appropriate method which will then be adopted for use in his application. An interesting example of such a study is that of Gaffney (1982) where he reports on the performance of a selection of methods on a particular stiff oscillatory problem.

The second application involves the comparison of different versions of the same basic code or methods with very similar strategies and objectives on general stiff problems. The former situation arises frequently when one is developing a method and an example of the latter is the comparison of GEAR and EPISODE (Byrne et al. (1977)). The third application is in connection with quantifying the costs involved and the potential advantages in exploiting structure when solving problems of a special type. As an example of such an application we have used (Kamel and Enright (1980)) the testing tools described above to quantify the advantages possible when one automatically partitions a problem with only a few transient components and subsequently exploits this partitioning in the linear algebra modules.

In summary we feel that while a comparison of stiff methods on general stiff problems is not feasible, one can compare methods on a special type of stiff problems or versions of similar

methods on general stiff problems with far fewer difficulties. Automatic testing tools and the collection of standard test problems can make this a straightforward painless process which could prove essential in many applications.

References

- C. Addison (1982). 'Summary of the Illinois-Toronto OlympiODE Test Results', ACM SIGNUM Newsletter 16, 4, pp. 19-22.
- J.D. Byrne, A.C. Hindmarsh, K.R. Jackson and H.G. Brown (1977), 'A Comparison of Two ODE Codes: GEAR and EPISODE', Computers and Chem. Eng., 1, pp. 133-147.
- W.H. Enright (1979), 'Using a Test Package for the Automatic Assessment of Numerical Methods for ODEs', in L.D. Fosdick, ed., Performance Evaluation of Numerical Software, pp. 199-213, North-Holland, Amsterdam.
- W.H. Enright, T.E. Hull and B. Lindberg (1975), 'Comparing Numerical Methods for Stiff Systems of ODEs', BIT, 15, pp. 10-48.
- W.H. Enright and M.S. Kamel (1979), 'Automatic Partitioning of Stiff Systems and Exploiting the Resulting Structure', ACM Trans. Math. Software, 5, 4, pp. 374-385.
- P.W. Gaffney (1982), 'A Survey of FORTRAN Subroutines Suitable for Solving Stiff Oscillatory Ordinary Differential Equations, Tech. Memo 134, Oak Ridge Nat. Lab., Comp. Sc. Dept., Oak Ridge.

- P.M. Hanson and W.H. Enright (1981), 'Controlling the Defect in Existing Variable-Order Adams Codes for Initial Value Problems', Dept. of Comp. Sc. Tech. Rep. No. 154, University of Toronto, Toronto.
- A.C. Hindmarsh (1973), 'GEAR: Ordinary Differential Equation System Solver, Lawrence Livermore Laboratory Rep. UCID-30001, Rev. 3, Livermore.
- E. Hofer (1976), 'Partially Implicit Method for Large Stiff Systems of ODEs with only a Few Equations Introducing Small Time Constants', SIAM J. Numer. Anal., 13, pp. 645-663.
- J.D. Lambert (1974), 'Two Unconventional Classes of Methods for Stiff Systems', in R.A. Willoughby, ed., Stiff Differential Systems, pp. 171-187, Plenum, New York.
- L.F. Shampine (1982), 'Evaluation of a Test Set for Stiff ODE Solvers' to appear in ACM Trans. Math. Software.
- L.F. Shampine and C.W. Gear (1979), 'A User's View of Solving Stiff Ordinary Differential Equations', SIAM Review 21, 1, pp. 1-17.
- G. Soderlind (1980), 'DASP3 - A Program for the Numerical Integration of Partitioned Stiff ODEs and Differential-Algebraic Systems', Tech. Rep. TRITA-NA-8008, Royal Inst. of Tech., Stockholm.

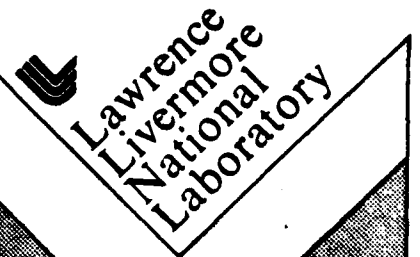
UCRL- 87406
PREPRINT

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



This is a preprint of a paper intended for publication in a journal or proceedings. Since changes may be made before publication, this preprint is made available with the understanding that it will not be cited or reproduced without the permission of the author.

**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.

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 differentiation 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 terrestrial 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

$$\partial c^i / \partial t = \nabla \cdot (D \nabla c^i + V c^i) + K^i + S^i \quad (i=1,2,\dots,p) , \quad (1)$$

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 $c = (c^1, \dots, c^p)^T$. 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

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

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

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

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

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

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

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

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 $N \times N$ matrices, followed by backsolve operations for solutions of linear systems. If the ODE system is written

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

and has a system Jacobian matrix

$$J = \partial f / \partial y, \quad (5)$$

then the linear systems take the form

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

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 $N \times N$ 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

$$P = L + D + U \text{ with} \tag{7}$$

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 v^{th} SOR iteration in the solution of $Px = r$ is given by

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

This can be rephrased as

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

This pair of equations appears circular, but is not; z is computed and x^v updated to x^{v+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

$$P = L + D + U \text{ with} \quad (10)$$

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^v 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 $p \times p$ blocks), with each block being a scalar multiple of the $p \times p$ 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 37×44 (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,

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

(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]. This 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 Research Co.), 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 reetermined 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 $\dot{Ay} = g$ 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 $g(t,y) = (g_i, i=1,2,\dots,NG)$ 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 , altitude z (both in kilometers), and time t (in sec), with $0 \leq x \leq 20$, $30 \leq z \leq 50$, $0 \leq t \leq 86400$ (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 $[O_3]$ (both in moles/cm³). The concentration of molecular oxygen $[O_2]$ is assumed constant. The equations of the model are:

$$c_t^i = (K_V(z)c_z^i)_z + k_h c_{xx}^i + R^i(c^1, c^2, t) \quad (i=1,2), \quad (12)$$

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

$$R^1(c^1, c^2, t) = -(k_1 + k_2 c^2) c^1 + k_3(t) c^2 + k_4(t) \cdot 7.4 \cdot 10^{16}$$

$$R^2(c^1, c^2, t) = (k_1 - k_2 c^2) c^1 - k_3(t) c^2.$$

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

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

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

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

The initial conditions are

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

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 rectangular 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 $N = 2M_x M_z$. The initial value vector y_0 is taken from the initial condition functions given above. The system Jacobian J is sparse, with roughly $12M_x M_z = 6N$ 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

<u>Solver</u>	<u>R.T.</u>	<u>NST</u>	<u>NFE</u>	<u>NJE</u>	<u>NLU</u>	<u>W.S.</u>
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 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 2×2 blocks). The total number of block-SOR iterations for the 10×10 grid was 607, or an average of less than 2 per step. For the 38×38 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 10×10 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 $m \times m$ 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, K. 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: 1. Concept and Development, J. of Appl. Meteorology, 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 GEARB: 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.W. 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. li.
- [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 University, 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.

DISCLAIMER

This document was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor the University of California nor any of their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial products, process, or service by trade name, trademark, manufacturer, or otherwise, does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or the University of California. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government thereof, and shall not be used for advertising or product endorsement purposes.

Reference to a company or product name does not imply approval or recommendation of the product by the University of California or the U.S. Department of Energy to the exclusion of others that may be suitable.

DESCRIPTION AND EVALUATION OF A STIFF ODE CODE DSTIFF

G.K. Gupta,
Department of Computer Science,
Monash University,
Clayton, Victoria, 3168,
Australia.

ABSTRACT

The paper describes and evaluates DSTIFF, a set of subroutines for solving stiff ordinary differential equations. The code is somewhat similar to the well-known packages LSODE, GEAR and DIFSUB but the present set of subroutines are based on least squares multistep formulas rather than BDF. The paper describes the formulas used in the code, the structure of the code and the heuristics used, and evaluates its performance. The code seems to be much more efficient than LSODE in solving stiff equations which have jacobians with eigenvalues having large imaginary parts.

On other problems, DSTIFF is as efficient as LSODE on larger tolerances and somewhat less efficient than LSODE on stringent tolerances.

INTRODUCTION

Several codes are now available for solving stiff ordinary differential equations. Enright, Hull and Lindberg (1975) and Enright and Hull (1976) have tested codes for solving stiff equations and Shampine and Gear (1979) discuss some of the available codes. Shampine and Gear also discuss why there is a need to distinguish a special class of problems termed stiff and describe the common characteristics of methods used for solving stiff equations.

Most of the codes available for solving stiff equations may be divided into the following three classes:

- (a) BDF codes - The first code using the Backward Differentiation Formulas (BDF) was designed by Gear (1971). This code, DIFSUB, uses formulas up to order 6. Several attempts have been made to improve this code and several variants of the code are now available. The best known variants are GEAR by Hindmarsh (1974) and EPISODE by Byrne and Hindmarsh (1975). GEAR and EPISODE themselves have several versions available. For example a version is available for solving problems with banded jacobians and another for linearly implicit equations of the form $Ay' = g$, A a square matrix. A new version of GEAR called LSODE is now available from Hindmarsh (1980). Several other DIFSUB variants are known to exist.

Test results for codes cited above indicate that the BDF codes are the most efficient for solving stiff equations except when the eigenvalues of the jacobian of the differential equations have large imaginary parts. For such oscillatory problems, BDF codes become very inefficient because of the poor stability properties of the 5th and 6th order BDF. To partly overcome this problem, several BDF codes restrict the maximum order of the formulas used to 5.

- (b) Second-Derivative Codes - To overcome the stability problems of BDF, Enright (1972) suggested the use of second-derivative formulas (SDF); linear multistep formulas which include second-derivative terms. Enright designed

the first SDF code, SDBASIC, using formulas of orders up to 9. Other SDF codes have been designed by Addison (1979) and Sacks-Davis (1980). Although the SDF used in the above codes have much better stability than BDF and therefore overcome the problem of BDF codes, they have three serious disadvantages. These are: requirement of analytic jacobians, necessity of more linear algebra at each step than in BDF codes, and difficulty in solving some problems with large coupling between the equations. The last problem arises because the corrector iterations used in the SDF codes assume that $\partial^2 f / \partial y^2$ terms for the given differential equations are negligible. Testing of SDF codes indicates that they are not really very efficient when compared to BDF codes like GEAR except for solving small systems at stringent tolerances or for solving problems with jacobians having large imaginary parts for which BDF codes become very inefficient. As noted above, the biggest problem in using SDF codes is that the analytic jacobian must be provided. However, if the jacobian is available, SDF codes can also solve non-stiff equations efficiently because the truncation error coefficients of the SDF of Enright (1972) are quite small.

- (c) Runge-Kutta Codes - Implicit Runge-Kutta formulas (RKF) suitable for solving stiff equations have been derived and some codes based on implicit RKF were tested by Enright et al (1975) and Enright and Hull (1976). The amount of linear algebra required at each step in using a fully implicit RKF is so large, that these formulas can not be expected to be efficient even for solving small systems of equations. Progress has recently been made in reducing the amount of linear algebra involved in using RKF and a code, STRIDE, has been designed by Butcher, Burrage and Chipman (1979) based on singly-implicit RKF of Burrage (1978). The formulas used in STRIDE are $A(\alpha)$ -stable, $\alpha \geq 83^\circ$ (Widlund, 1967). To solve a system of N equations, STRIDE needs to do about the same amount of $O(N^3)$ operations per step as

a code based on BDF but several additional $O(N^2)$ operations per step must be performed. Kaps and Rentrop (1979) take a different approach in designing two codes GRK4T and GRK4A based on embedded generalized RKF (also called the Rosenbrock methods) of order 4. The formula used in GRK4A is A-stable while that in GRK4T is A (89.3°)-stable. These codes evaluate the jacobian and then solve a linear system of order N with four right hand sides at each step. Preliminary testing by Addison (1980) and results presented by Kaps and Rentrop (1979) indicate that the codes are reliable and efficient in solving problems whose jacobians have eigenvalues near the imaginary axis. We therefore expect these codes to be useful in solving small systems of stiff equations but the additional linear algebra involved at each step would make them inefficient for large systems.

In addition to the above, a code based on cyclic formulas of orders upto 7, STINT, has been designed by Tendler, Bickard and Picel (1978). Another code, TRAPEX, based on extrapolation was tested by Enright et al (1975) and Enright and Hull (1976). Also Skeel and Kong (1977) have shown that blended formulas using Adams formulas and BDF together are suitable for solving stiff equations. The numerical testing has shown that TRAPEX is not an efficient code. Tendler, Bickart and Picel (1978) show that STINT is reliable but not as efficient as GEAR. Only a few test results of a blended formulas code are available. The blended formula approach does overcome one of the problems of the SDF codes since it does not require analytic jacobians. Blended formulas still have the other two disadvantages of the SDF codes regarding additional linear algebra and assumption of small $\partial^2 f / \partial y^2$ in corrector iterations.

The present paper describes a code based on linear multistep formulas derived using least squares. The code, DSTIFF, overcomes the weakness of BDF codes in solving problems which have highly oscillatory solution components and is almost as efficient and reliable as the BDF codes on most other problems.

2. FORMULAS USED

We have noted that the BDF codes like GEAR and LSODE are very efficient for solving many stiff problems but have trouble on systems which have jacobian with eigenvalues close to the imaginary axis. This difficulty arises because of the poor stability of BDF of orders 5 and 6 near the imaginary axis. To overcome this Wallace and Gupta (1973), Gupta and Wallace (1975) and Gupta (1975, 1976) have derived linear formulas having much better stability than BDF. In fact some formulas presented in Gupta (1975, 1976) are very close to being A-stable even at orders 6 and 7. These formulas with good stability however are not very efficient for solving stiff equations because the formulas have large truncation error coefficients and tend to have roots close to unity at $h\lambda = \infty$. This is in contrast to BDF which for orders 5 and 6 have poor stability near the imaginary axis, small truncation error coefficients and zero roots at $h\lambda = \infty$. There are, of course, a large number of formulas somewhere between the two extremes and we have derived several sets in Gupta (1975). We have chosen a set based on least squares to be included in DSTIFF. This set is similar to Adams-Moulton formulas (AMF) but instead of interpolation we use least squares approximation. To clarify this, we first define the predictor corrector method as

$$P_n(x) = P_{n-1}(x) + \delta_n C((x-x_n)/h)$$

where we want to compute a polynomial $P_n(x)$ of degree m at x_n so that $y_n = P_n(x_n)$ may be computed. $P_{n-1}(x)$ is the polynomial approximation at x_{n-1} and C is a fixed polynomial of degree m characteristic of the particular multistep formula. For AMF of order m , C is such that

$$C(-1) = 0, C'(-k) = 0, \quad k = 1, 2, \dots, m-1.$$

For the formulas we use for solving stiff equations, we require that C of order m satisfy the condition $C(-1) = 0$ and C' be a least squares approximation to points $(0,1), (-1,0), (-2,0) \dots (-N,0)$ where $N \geq m-1$ and is so chosen that the

formula is stiffly stable and has small truncation error. These formulas were labelled FLS in Gupta and Wallace (1975) and we retain this name for them although the values of N used for some formulas in the present set is different than those in that paper.

In Table 1, we present some of the characteristics of the formulas up to order 10. Higher order formulas have been derived but their inclusion in the code did not seem to affect the subroutine's performance very much. We will show in the last section that it may be desirable to further restrict the maximum order used to 7.

In Table 1, α is the angle defined by $A(\alpha)$ - stability (Widlund, 1967). D is the most negative $Re(h\lambda)$ value on the stability curve. k_{m+1} is the truncation error coefficient, and N is the parameter used in the least squares approximation above.

Order	α	D	k_{m+1}	Modulus of Largest Root at $h\lambda = \infty$	Value of N
1	90.00	0.0	0.500	0.0	-
2	90.00	0.0	0.083	1.0	1
3	86.46	-0.075	0.242	0.32	3
4	80.13	-0.282	0.374	0.43	5
5	73.58	-0.606	0.529	0.567	7
6	67.77	-1.218	0.724	0.878	9
7	65.53	-1.376	1.886	0.898	12
8	64.96	-1.149	7.686	0.790	16
9	62.78	-2.086	16.737	0.989	19
10	63.74	-1.223	133.955	0.878	26

Table 1: Truncation Error Coefficients and
Stability of Formulas FLS

3. ORDER AND STEP-SIZE CHANGING

It is assumed that the reader is familiar with the Nordsieck representation of the predictor-corrector methods. At any point x_n , let a_n be the vector of scaled derivatives of the approximating polynomial $P_n(x)$ when a step-size of h_n is being used. We have (for example, refer to Gear (1971, p 216))

$$a_{n+1} = Aa_n + Zw$$

where A is a pascal triangle, Z is a corrector vector depending on the multi-step formula being used and w is the correction necessary.

In DIFSUB, Gear (1971, Section 9.3) uses the following techniques in step-size and order changing. Let the present order be q and the present step-size be h_n

- i) a_n has $q+1$ elements. The last element $a_{n,q+1} = h_n^q y^{(q)}/q!$
Therefore the local truncation error, which is equal to

$$k_{q+1} h_n^{q+1} y^{(q+1)} + O(h_n^{q+2}), \text{ can be estimated by } k_{q+1} \cdot q! \cdot \nabla a_{n,q+1}.$$

($\nabla a_{n,q+1}$ being the correction applied to the last element).

Similarly, estimate of the local truncation error for order $q-1$ is given by

$$k_q \cdot a_{n,q+1} \cdot q!$$

and at order $q+1$ by

$$k_{q+1} \cdot q! \cdot (\nabla^2 a_{n,q+1})$$

Let us call these three estimates E_q , E_{q-1} and E_{q+1} respectively.

- (ii) Let $h_{n+1} = \alpha_k h_n$ where $k = q-1, q, q+1$. New step-sizes for orders $q, q-1$ and $q+1$ are now calculated. If the user specified error tolerance is ϵ , the ratios, α_k , are

$$\alpha_q = \frac{1}{1.2} \left[\left\| \frac{\frac{\epsilon}{E_q}}{w} \right\| \right]^{1/q+1}$$

$$\alpha_{q-1} = \frac{1}{1.3} \left[\left\| \frac{\frac{\epsilon}{E_{q-1}}}{w} \right\| \right]^{1/q}$$

$$\alpha_{q+1} = \frac{1}{1.4} \left[\left\| \frac{\frac{\epsilon}{E_{q+1}}}{w} \right\| \right]^{1/q+2}$$

where for each member of the system there is a weight w and $\|.\|$ is the L_2 -norm. The order to be used on the next step is selected corresponding to the largest of these α 's.

- (iii) The following heuristics are also used.

- (a) If the largest α value is less than 1.1, the step-size is not increased.
- (b) Step-size is not changed for $q+1$ steps after the last change except if a step fails.
- (c) Order can not be increased if a step fails.
- (d) If the step-size is not increased after computing α 's, the step-size and the order is not changed for 10 steps except if a step fails.
- (e) If a step fails three times, the order is set to one.
- (f) If the corrector iterations do not converge then the step-size is reduced by a factor of 4.

In the main integrator of DSTIFF E_q , E_{q-1} and E_{q+1} are computed as above but the α 's are computed as follows:

i) If $\epsilon > E_q$, that is, after a successful step we have

$$\alpha_q = \frac{1}{1.05} \left[\frac{\epsilon_q}{\left\| \frac{E_q}{w} \right\|} \right]^{1/q+1}$$

$$\alpha_{q-1} = \frac{1}{1.05} \left[\frac{\epsilon_{q-1}}{\left\| \frac{E_{q-1}}{w} \right\|} \right]^{1/q}$$

$$\alpha_{q+1} = \frac{1}{1.05} \left[\frac{\epsilon_{q-2}}{\left\| \frac{E_{q+1}}{w} \right\|} \right]^{1/q+2}$$

where $\epsilon_k = \epsilon/k^{\frac{1}{2}}$, $k = q-1, q, q+1$.

Instead of the L_2 -norm, we use the L_∞ -norm. The order is then selected corresponding to the largest α . The main reason for changing the heuristics is that we want to encourage the code to go to as high an order as possible. In BDF codes, order 5 or 6 is reached quickly because the truncation error coefficients of higher order formulas are smaller. For the FLS formulas used in DSTIFF, higher order formulas have larger truncation error coefficients as shown in Table 1.

ii) If $\epsilon > E_q$, then the unsuccessful step must be repeated and the new step-size is based on

$$\alpha_q = \frac{1}{1.05} \left[\frac{\epsilon_q}{\left\| \frac{E_q}{w} \right\|} \right]^{2/q+1}$$

$$\alpha_{q-1} = \frac{1}{1.05} \left[\frac{\epsilon_{q-1}}{\left\| \frac{E_q}{w} \right\|} \right]^{2/q}$$

This change is desirable because the need for reducing the step-size indicates that the polynomials approximating the solutions are not very good approximations at the present point. Therefore the step-size needs to be reduced by a larger factor than would be reduced by formulas in (i) above.

(iii) and the following heuristics are used:

- (a) If the largest α value is less than 1.025 then the step-size is not increased.
- (b) Step-size is not changed for $q+1$ steps after the last change except if a step fails.
- (c) Order cannot be increased if a step fails.
- (d) If a step fails twice and the order being used is more than 3, the order is reduced by one.
- (e) If a step fails three times, then the order is set to one.
- (f) If the corrector iterations do not converge then step-size is reduced by a factor of 2 and the step-size and the order is not changed for the next 10 steps (except if a step fails).
- (g) DSTIFF, GEAR and LSODE use fixed-step interpolation when step-size is varied. This means that the coefficients of formulas do not change when the step-size is changed. In DSTIFF also we use fixed-step interpolation most of the time, but we use a technique called "average-step interpolation" discussed by Gupta and Wallace (1979) whenever the step-size is reduced. New coefficients of formulas are computed for $(q+2)/3$ steps after a step-size reduction takes place.

4. CORRECTOR ITERATIONS

To compute the correction necessary so that the approximating polynomial at the new point x_{n+1} satisfies the differential equation, an iterative procedure must be used. For stiff equations, the simple iterations do not converge and the modified Newton's iterations need to be used. Using the Nordsieck notation, the iterations

may be expressed as follows (Gear, 1971, p 207)

$$a_{n+1,(m+1)} = a_{n+1,(m)} - \mathcal{L}[-I + h\mathcal{L}_0 J]^{-1} F(a_{n+1,(m)})$$

$a_{n+1,(m)}$ is the m th approximation to the vector a_{n+1} , $\mathcal{L}$ is the corrector vector with first coefficient $\mathcal{L}_0$, J is the jacobian and F is a function such that for a vector a , $F(a) = hf(a_0) - a_1$. $F(a) = 0$ if vector a satisfies the differential equation.

As common with other codes, DSTIFF evaluates J only when necessary and the matrix $-I + h\mathcal{L}_0 J$ is stored as its LU-decomposition. The present version of DSTIFF does not have any sparse matrix facilities to solve large problems more efficiently.

An important part of corrector iterations technique is the convergence test used in terminating the iterations. In DIFSUB, the iterations are terminated if the following test is satisfied.

$$\left\| \frac{y_{n+1,(m+1)} - y_{n+1,(m)}}{w} \right\| \leq \frac{\epsilon \mathcal{L}_0}{2N(q+2)}$$

w is a weight and L_∞ -norm is used. ϵ is the user-specified error tolerance and N is the number of equations. The constant $\mathcal{L}_0/2N(q+2)$ is empirical and used to ensure that the implicit equations are solved somewhat more accurately than ϵ .

Hindmarsh (1974) uses a slightly different convergence test in GEAR in an attempt to reduce the number of iterations. The test is based on the assumption that corrector iterations converge linearly and the rate of convergence is almost constant.

Let

$$d_m = \frac{y_{n+1,(m+1)} - y_{n+1,(m)}}{w}$$

$$c_m = \frac{d_m}{d_{m-1}}$$

w is a weight depending on the error criterion and L_2 -norm is used in computing d_m . c_m is the rate of convergence. The following convergence test is used

$$2 c_m d_m \leq \frac{\epsilon l_0}{2(q+2)}$$

$2 c_m d_m$ is an estimate of the correction at the next iteration. The rate of convergence, c_m , is not available at the first iteration of each new step and the last value of c_m from the previous step is used. DSTIFF also uses this test of corrector convergence. In LSODE, the convergence test has been modified further. d_m is computed as above but the RMS-norm is used instead of the L_2 -norm. The convergence test seems to be the following

$$1.5 c_m (d_m q + k_{m+1} l_m) \leq \frac{\epsilon}{2(q+2)}$$

where k_{m+1} is the truncation error coefficient of the formula being used and l_m is the last coefficient of the formula vector l . In addition, if the rate of convergence c_m becomes greater than 2 on the second or the third iteration, the code assumes that the iterations are diverging.

Recently Shampine (1979, 1980) has discussed the problem of testing for corrector convergence and pointed out some weaknesses of the tests discussed above.

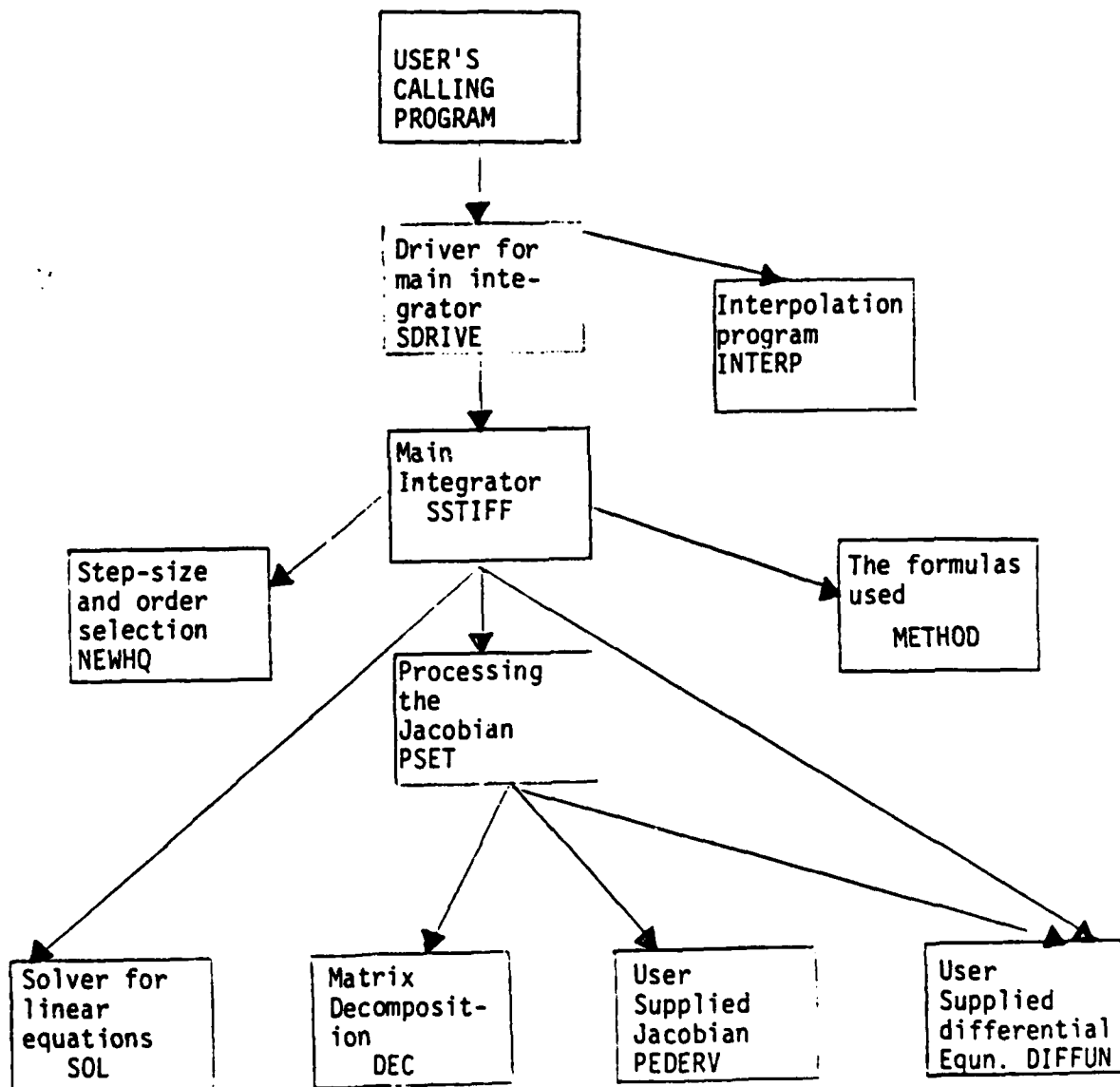
5. STRUCTURE OF DSTIFF

DSTIFF consists of eight subroutines and the user provides two subroutines and the calling program as discussed in Gupta (1979). Out of the eight subroutines in DSTIFF, five subroutines (SDRIVE, DEC, SOL, PSET, INTERP) are very similar to those used by Hindmarsh and Byrne in GEAR and EPISODE. The remaining three subroutines (SSTIFF, METHOD, NEWHQ) are quite different. We discuss the subroutines briefly.

- (1) SDRIVE is the driver subroutine for the main integrator SSTIFF. SDRIVE checks the parameters passed to it, and calls SSTIFF repeatedly till the integration reaches the end-point specified by the user. The calling sequence of DSTIFF is the same as EPISODE and GEAR.
- (2) DEC and SOL are subroutines for solving linear equations.
- (3) SSTIFF is the main integrator. It takes one step and returns control to the calling routine. On the first call, the subroutine sets the order to one and computes a suitable initial step-size. At the end of one step of integration, the subroutine computes, if necessary, new step-size and order to be used for the next step.
- (4) METHOD specifies the coefficients of the formulas being used and the truncation error coefficients of the formulas. The three dimensional array PERTST used in DIFSUB and GEAR has not been used so that the subroutine is easier to understand.
- (5) NEWHQ is called by SSTIFF when new step-size and order needs to be computed. Separating the step-size and order changing techniques into a subroutine makes it easier to understand and modify the package.

- (6) PSET is called by SSTIFF to process the jacobian. PSET numerically evaluates the jacobian if necessary and calls DEC to obtain a LU decomposition of the linear equations.
- (7) INTERP is an interpolation routine used when the user wants the solution at intermediate points.

The overall structure of DSTIFF, therefore, looks like the following:



6. PERFORMANCE EVALUATION OF DSTIFF

Enright, Hull and Lindberg (1975), Enright and Hull (1976), and Addison (1981) have tested several subroutines for solving stiff equations. As we have noted, BDF codes have been found to be most efficient. We therefore compare the performance of DSTIFF with LSODE, the most recent BDF code. Tables 2 and 6 compare the performance of LSODE and DSTIFF in solving the test problems in the testing package described in Enright (1979). The testing package New Detest differs from the package used in Enright et al (1975) in that the code being tested does not need to be modified and normalised statistics if required are provided. New Detest includes a set of 30 test problems consisting of 25 test problems in five classes A, B, C, D and E used in Enright et al (1975) and 5 additional chemical kinetics problems in class F. We have presented the overall results for each of the six problem classes. (New Detest was run on an IBM 3031/6 at the Asian Institute of Technology, Bangkok.) Absolute error criterion was used in the testing. Detailed results of DSTIFF are presented in Gupta (1982). We note that LSODE and DSTIFF are similar codes in many ways. Both codes use Nordsieck representation of the linear multistep formulas and both carry out about the same amount of work per step. Therefore the statistics provided by New Detest are a good basis for comparing the two codes.

PROBLEM CLASS		A	B	C	D	E	F	OVERALL
DSTIFF	10**-2	346	1001	804	378	534	2135	5198
	10**-4	957	2462	1715	962	1207	2927	10230
	10**-6	1970	4157	2891	1686	2235	4633	17572
LSODE	10**-2	250	3613	528	300	381	1198	5820
	10**-4	535	3672	1028	623	712	2086	8656
	10**-6	1007	4622	1711	1267	1375	3634	13616

Table 2: The Total Number of Function Evaluations
(Overall results of each Problem Class)

AD-A122 171

PROCEEDINGS OF THE INTERNATIONAL CONFERENCE ON STIFF
COMPUTATION APRIL 12. (U) UTAH UNIV SALT LAKE CITY DEPT
OF CHEMICAL ENGINEERING R C AIKEN 1982

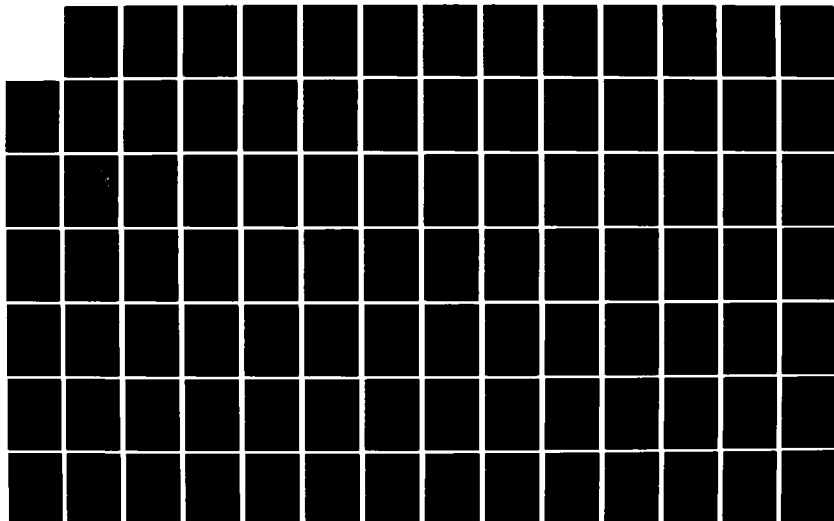
2/4

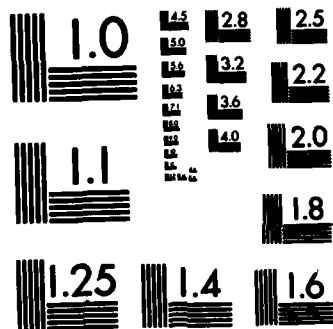
UNCLASSIFIED

AFOSR-TR-82-1036-VOL-3 AFOSR-82-0038

F/G 12/1.

NL





MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

PROBLEM CLASS		A	B	C	D	E	F	OVERALL
DSTIFF	10**-2	61	72	78	60	59	159	489
	10**-4	88	90	104	96	82	145	605
	10**-6	106	98	118	130	101	217	770
LSODE	10**-2	59	206	86	67	75	183	676
	10**-4	86	248	127	112	100	229	902
	10**-6	119	289	160	166	144	346	1224

Table 3: The Total Number of Jacobian Evaluations and LU
Decompositions (Overall results of each Problem Class)

PROBLEM CLASS		A	B	C	D	E	F	OVERALL
DSTIFF	10**-2	1.741	3.335	2.560	0.742	1.130	5.131	14.640
	10**-4	6.511	10.434	6.375	2.099	3.191	9.123	37.733
	10**-6	14.118	22.416	12.376	4.515	6.993	16.789	77.207
LSODE	10**-2	1.664	18.288	2.159	0.930	1.261	3.969	28.271
	10**-4	3.765	21.490	4.355	1.958	2.499	7.032	41.100
	10**-6	7.100	26.282	7.331	4.169	5.049	12.678	62.609

Table 4: The Total Time in Seconds
(Overall results of each Problem Class)

PROBLEM CLASS		A	B	C	D	E	F	OVERALL
DSTIFF	10**-2	204	484	390	173	250	918	2419
	10**-4	620	1176	866	467	607	1349	5085
	10**-6	1189	2139	1526	869	1180	2364	9267
LSODE	10**-2	187	2804	395	184	255	803	4628
	10**-4	444	3123	805	406	542	1473	6793
	10**-6	832	3989	1392	922	1100	2767	11002

Table 5: The Total Number of Steps Taken
(Overall results of each Problem Class)

Fraction of Steps on which Local Error > Tolerance								
PROBLEM CLASS		A	B	C	D	E	F	OVERALL
DSTIFF	10** ⁻²	0.000	0.004	0.010	0.006	0.020	0.024	0.014
	10** ⁻⁴	0.000	0.006	0.016	0.004	0.013	0.020	0.011
	10** ⁻⁶	0.004	0.008	0.006	0.000	0.011	0.019	0.010
LSODE	10** ⁻²	0.059	0.339	0.096	0.098	0.141	0.173	0.258
	10** ⁻⁴	0.027	0.443	0.063	0.059	0.110	0.145	0.256
	10** ⁻⁶	0.053	0.359	0.093	0.060	0.043	0.145	0.192

Table 6: The Local Error Deception
(Overall results of each Problem Class)

We note that DSTIFF is very efficient, specially at larger tolerances. The main reason for the much better overall performance of DSTIFF is the problem B5 which is solved by DSTIFF very efficiently.

We note that both DSTIFF and LSODE failed to solve problem F3 at tolerances of 10^{-2} and 10^{-4} , and problem F5 at 10^{-2} . In addition, DSTIFF failed to successfully integrate problem E5 at tolerances of 10^{-2} and 10^{-4} . Problem E5 is a semi-stable chemical kinetics problem which can become unstable if a computed solution becomes negative. Shampine (1981) and Enright and Hull (1976) discuss the difficulty of solving such problems.

On the basis of the above testing we can say that:

- (1) Code DSTIFF seems to be more reliable than LSODE and the error deception of DSTIFF is much smaller than for LSODE.
- (2) The number of jacobian evaluations are smaller for DSTIFF than for LSODE because LSODE reevaluates jacobians regularly even if there is no need for it.

- (3) DSTIFF solved the problems in Class B, much more efficiently than LSODE. This is because the problem B5 was solved by DSTIFF in 134, 358 and 615 steps at the three tolerances respectively while LSODE took 2351, 2397 and 2620 steps respectively. Problem B5 has eigenvalues very close to the imaginary axis ($-10 \pm 100i$).
- (4) On problems other than those which have large imaginary eigenvalues, LSODE is usually more efficient than DSTIFF but is usually less reliable in keeping the local error below the user specified tolerance.
- (5) Somewhat surprisingly, in spite of the higher order formulas used by DSTIFF, LSODE seems to be the more efficient code at stringent tolerances. Partly this is due to the high truncation error coefficients and roots of the order of 0.9 at infinity of the higher order formulas in DSTIFF.
- (6) The average number of function evaluations per step is close to 2.0 for DSTIFF while for LSODE, it is only about 1.25. The smaller number of function evaluations per step by LSODE seem to be partly responsible for larger local error deceptions. Also if the corrector convergence tests of both codes were to be changed to the test recommended by Shampine (1979) to improve reliability, we expect both codes to take about two function evaluations per step. Therefore the probably will have much more impact on the performance of LSODE than on the performance of DSTIFF.
- (7) The CPU time taken per step on the average is close to 0.006 seconds for LSODE but for DSTIFF, the times are 0.0060, 0.0074 and 0.0083 seconds for the three tolerances 10^{-2} , 10^{-4} and 10^{-6} respectively. Time per step is larger for DSTIFF partly because of the larger number of function evaluations per step and, at stringent tolerances, due to the higher order formulas used.

Another set of tests were run to evaluate the performance of DSTIFF in solving stiff equations when analytic jacobian is not available and the code must approximate the jacobian by finite differencing. Table 7 gives the overall results for DSTIFF and LSODE. DSTIFF failed to complete the integration for Problem F4 at TOL = 10^{-6} and therefore results for the tolerance 10^{-6} for LSODE also do not include the cost of solving F4.

	TOL	TIME	FCN CALLS	MAT FACT	NO OF STEPS
DSTIFF	10^{-2}	14.884	7244	488	2417
	10^{-4}	37.833	12804	605	5085
	10^{-6}	64.714	16884	648	7522 (*)
LSODE	10^{-2}	27.797	8506	652	4510
	10^{-4}	40.942	12394	882	6723
	10^{-6}	54.887	15700	1028	9007 (*)

Table 7: Overall Results with Numerically Approximated Jacobian
(* excluding cost of solving problem F4)

To evaluate the effect of high order formulas on the efficiency of DSTIFF, we conducted another set of tests. We used DSTIFF with the maximum order of formulas restricted to 5 and then with maximum orders of 7 and 10 to solve the New Detest stiff problems. The results obtained are summarised in Table 8.

For better comparison of results we have excluded class F problems from the summary in Table 8. This has been done because for maximum order of 5 we had three failures in Class F while maximum order 7 and 10 had only two failures. We note that the results at $TOL = 10^{-2}$ are not affected very much by setting maximum order to as low as 5. For more stringent tolerances, maximum order of 7 seems to be better than maximum order of 5 or 10 since the number of function evaluations and the number of steps for maximum order 7 are the lowest.

MAX ORDER	TOL	TIME	FCN CALLS	MAT FACT	NO. OF STEPS
5	10^{-2}	9.308	3210	331	1551
	10^{-4}	24.257	7042	473	3857
	10^{-6}	50.244	13769	520	7631
	OVERALL	83.809	24021	1324	13039
7	10^{-2}	9.208	3081	331	1499
	10^{-4}	26.375	7134	448	3694
	10^{-6}	53.204	12743	565	6841
	OVERALL	88.787	22958	1344	12034
10	10^{-2}	9.509	3063	330	1501
	10^{-4}	28.610	7303	460	3736
	10^{-6}	60.418	12939	553	6903
	OVERALL	98.537	23305	1343	12140

Table 8: Overall results when maximum order in DSTIFF is set to 5, 7 and 10 (excluding Class F)

7. CONCLUDING REMARKS

The code DSTIFF has been shown to be a reliable and efficient code. It is much more efficient than LSODE in solving problems which have jacobians with eigenvalues having large imaginary parts. On other problems, DSTIFF is almost as efficient as LSODE on larger tolerances and somewhat less efficient than LSODE on stringent tolerances.

The evaluation of DSTIFF using New Detest is only one indicator of its performance. For example, the Jacobians in New Detest are so cheap to evaluate that the fact that one code takes smaller numbers of jacobian evaluations is not really reflected in the CPU time statistics provided by New Detest. We further note that if a code fails on one or more of the problems in New Detest, comparison of codes becomes very difficult. Also it is difficult to say how serious a failure should be considered when some of the problems in the set are very difficult problems. Recently Shampine (1981) has critically looked at the set of problems in New Detest (except Class F) and it is hoped that New Detest will be modified in line with Shampine's comments.

We hope to further improve DSTIFF and are studying why DSTIFF doesn't perform better at stringent tolerances. We are looking at the order changing algorithm and are considering reducing the maximum order used in the code.

8. ACKNOWLEDGEMENTS

I would like to thank Dr. R. Sacks-Davis, Mr. P. Tischer, Dr. L.F. Shampine and two anonymous referees for their comments on an earlier version of this paper. Programming assistance of Kao Chung-Cheng is gratefully acknowledged.

9. REFERENCES

- Addison, C.A. (1979), "Implementing a Stiff Method Based Upon the Second Derivative Formulas", Technical Report 130, Department of Computer Science, University of Toronto.
- Addison, C.A. (1980), "Results of the 1979 ODE Olympics", Unpublished manuscript.
- Burrage, K. (1978) "A Special Family of Runge-Kutta Methods for Solving Stiff Differential Equations", BIT, 18, pp 22 - 41.
- Butcher, J., Burrage, K. and Chipman, F. (1979), "STRIDE: Stable Runge-Kutta Integrator for Differential Equations", Computational Mathematics Report No. 19, University of Auckland. (Also published in BIT, 20, pp 326-340, 1980).
- Byrne, G.D. and Hindmarsh, A.C. (1975), "A Polyalgorithm for the Numerical Solution of Ordinary Differential Equations", ACM-TOMS, 1, pp 71 - 96.
- Enright, W.H. (1972), "Studies in the Numerical Solution of Stiff Ordinary Differential Equations", Technical Report No. 46, Department of Computer Science, University of Toronto, Toronto.
- Enright, W.H., Hull, T.E. and LINDBERG, B. (1975), "Comparing Numerical Methods for Stiff Systems of ODE's ", BIT, 15, pp 10-48.
- Enright, W.H. and HULL, T.E. (1976), "Comparing Numerical Methods for the Solution of Stiff Systems of ODE's Arising in Chemistry", in "Numerical Methods for Differential Systems", L. Lapidus and W.E. Schiesser (Eds.), Academic Press.
- Enright, W.H. (1979), "Using a Test Package for the Automatic Assessment of Methods for ODE's" in "performance Evaluation of Numerical Software", L.D. Fosdick (Ed.), North-Holland, pp 199-213.
- Gear, C.W. (1971), "Numerical Initial Value Problems in Ordinary Differential Equations", Prentice-Hall, N.J.
- Gupta, G.K. (1975), "New Multistep Methods for the solution of Ordinary Differential Equations", Ph.D. Thesis, Dept. of Computer Science, Monash University.

References (contd.)

- Gupta, G.K. (1976), "Some New High-Order Multistep Formulae for Solving Stiff Equations", Math. Comp., 30, pp 417-432.
- Gupta, G.K. (1979), "DSTIFF: A Set of Subroutines for Solving Stiff Equations - User's Guide Part 1", Technical Report No. 3, Dept. of Computer Science, Monash University.
- Gupta, G.K. (1982), "Performance Evaluation of a Stiff ODE Code", Technical Report No. 25, Dept. of Computer Science, Monash University.
- Gupta, G.K. and Wallace, C.S. (1975), "Some New Multistep Methods for Solving Ordinary Differential Equations", Math. Comp., 29, pp 489-500.
- Gupta, G.K. and Wallace, C.S. (1979), "A New-Step-Size Changing Technique for Multistep Methods", Math. Comp., 33, pp 125-138.
- Hindmarsh, A.C. (1974), "GEAR: Ordinary Differential Equation System Solver", Tech. Report UCID-30001, Rev. 3, Lawrence Livermore Laboratory, Calif.
- Hindmarsh, A.C. (1980), "LSODE and LSODI, Two New Initial Value Ordinary Differential Equation Solvers" ACM SIGNUM Newsletter, 15, 4, pp 10 - 11.
- Kaps, P. and Rentrop, P. (1979) "Generalized Runge-Kutta Methods of Order Four with Step-size Control for Stiff Ordinary Differential Equations", Numer. Math., 33, pp 55-68.
- Sacks-Davis, R. (1980), "Fixed Leading Coefficient Implementation of SD-Formulas for Stiff ODE's", ACM-TOMS, 6, pp 540-562.
- Shampine, L.F. (1981), "Evaluation of a Test Set for Stiff ODE Solvers", ACM-TOMS, 7, pp 409 - 420.
- Shampine, L.F. and Gear, C.W. (1979), "A User's View of Solving Stiff Ordinary Differential Equations", SIAM Review, 21, pp 1 - 17.
- Shampine, L.F. (1979), "Evaluation of Implicit Formulas for the Solution of ODEs", BIT, 19, pp 495-502.

References (contd.)

Shampine, L.F. (1980), "Implementation of Implicit Formulas for the Solution of ODEs", SIAM J. Sci. Stat. Comput., 1, pp 119 - 130.

Skeel, R.D. and Kong, A.K. (1977), "Blended Linear Multistep Methods", ACM-TOMS, 3, pp 326 - 345.

Tendler, J.M., Bickart, T.A. and Picel, Z. (1978), "A Stiffly Stable Integration Process", ACM-TOMS, 4, pp 399-403.

Wallace, C.S. and Gupta, G.K. (1973), "General Linear Multistep Methods to Solve Ordinary Differential Equations", Aust. Comp. J., 5, pp 62-69.

Widlund, O.B. (1967), "A Note on Unconditionally Stable Linear Multistep Methods", BIT, 7, pp 65-70.

**An Advanced Simulation Package
for Large Chemical Reaction Systems**

G. Bader, U. Nowak, P. Deuflhard

**UNIVERSITÄT HEIDELBERG
INSTITUT FÜR ANGEWANDTE MATHEMATIK**

*Die ich rief, die Geister,
Werd ich nun nicht los.*

Goethe, Der Zauberlehrling

The authors wish to thank Mrs. G. Bahne for her careful typing of the manuscript.

0.	Introduction	1
1.	General Description of the Package	3
2.	Input/Output Facilities	6
3.	Chemical Compiler System	9
4.	Numerical Simulation	12
References		17

This paper will be part of an invited survey talk to be given at the International Conference on Stiff Computation at Park City, Utah, USA, April 12 - 14, 1982.

0. Introduction

The numerical integration of "stiff" ordinary differential equations (ODE's) is one of the active fields of research in numerical analysis. Among the possible applications of any stiff integration method, the class of problems arising from *mass action chemical kinetics* is of still increasing interest. It is therefore only natural that the development of a new stiff integrator on an extrapolation basis by Bader/Deuflhard [1] has also led to the development of a software package designed to handle *large* chemical kinetics problems. Details about this package LARKIN have been presented by Deuflhard/Bader/Nowak [4] in the proceedings of a recent workshop on modelling of chemical reaction systems (ed. by Ebert/Deuflhard/Jäger [6]). This early version of LARKIN, however, was portable only on IBM computers. In addition, part of that code was written in PL/1, (whereas most of it was written in FORTRAN). These disadvantages of the previous code were certainly prohibitive for a wider distribution.

It is the purpose of the present article to report on progress made in the development of the package LARKIN since then. The present code which is exclusively written in FORTRAN IV, should be portable on a wide range of computers. Moreover, it permits a user to attack much larger problems. This feature has been essentially achieved by the implementation of a new "chemical compiler system" and a new sparse solver (for details see Duff/Nowak [5]).

A general outline of the structure and some special properties of LARKIN are given in section 1. Input/output facilities are described in section 2. Section 3 contains

details of the "chemical compiler system" that automatically generates the ODE right-hand side together with the associated Jacobian matrix and its sparse pattern from the input in terms of chemical reaction equations (and rate constants). A short presentation of the selected stiff integrator of extrapolation type follows in section 4, where also a rough and short comparison with the competing BDF-type integrator is made.

The philosophy behind the development of this code has been to allow any chemical user to concentrate on the *modelling aspects* of his chemical reaction system under consideration. In view of this aim, the package has been developed in direct and steady contact with a local group of physical chemists that have been using the system successfully for quite a while (see e.g. [12]). Therefore, the software package LARKIN, as it stands now, should be a useful and reliable tool also in other chemical research environments.

1. General Description of the Package

In developing LARKIN, the main emphasis has been put onto the implementation of an *interactive dialogue system*. In order to be able to use this system, a user is only required to know details about his chemical model (and certainly not details about the computer system or about numerical analysis!). In particular, the question-answer dialogue contains only either self-explaining or chemical terms. There is *no* command language to be memorized by the user.

The package LARKIN as a whole is written in a "reasonably portable" FORTRAN. The FORTRAN compiler needs to be called essentially once, when the load module is produced from the source program (typical service of a computing center). Further calls of the FORTRAN compiler are only necessary, if changes in the size of the allocated storage are made. All model changes (say, changes of chemical reactions, initial values, rate constants, ...) can be easily and quickly performed within the frame of the dialogue system.

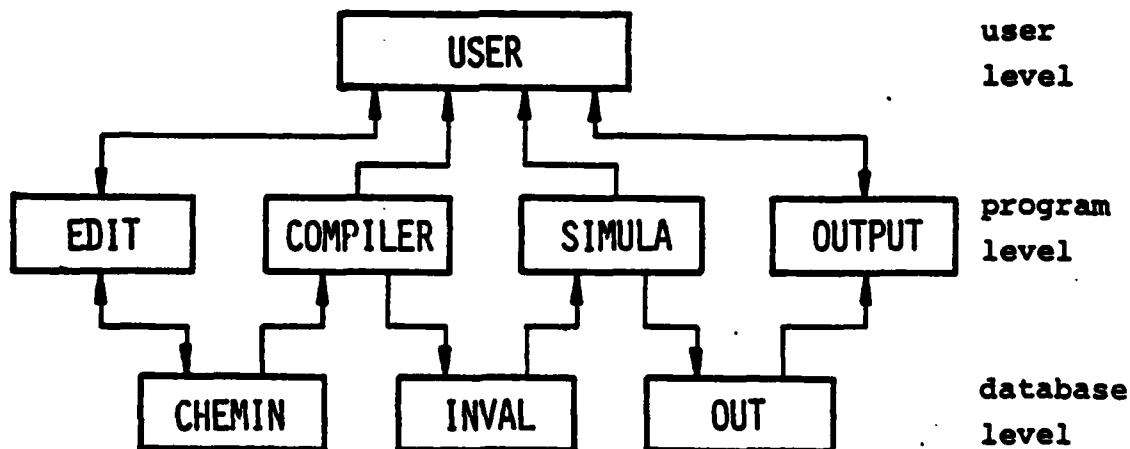


Fig. 1: structural diagram of LARKIN

A structural diagram of the package LARKIN is represented in Fig. 1. Details concerning the individual main blocks are as follows:

EDIT - input dialogue block: accepts chemical input and transfers it to the database CHEMIN; the contents of CHEMIN are available in EDIT(for possible modifications). For more details see section 2.

COMPILER - "chemical compiler" block: reads contents of database CHEMIN; checks syntax of chemical input; produces all pertinent information, which is necessary for the numerical simulation; stores this information in binary database INVAL. For more details see section 3.

SIMULA - numerical integrator block: performs numerical simulation of chemical reaction system using information from database INVAL. In the course of the integration, data of chemical interest (usually not only chemical species concentrations) are updated in database OUT. For details concerning the stiff integrator see section 4.

OUTPUT - output dialogue block: gives (possibly selected) output information in either graphical or numerical form as requested by the user; requires interface to plot software of each individual computing center. For more details see section 2.

The whole package LARKIN, as it stands now, applies to the numerical solution of *mass action kinetics* problems, which involves the numerical integration of large stiff ODE systems with *polynomial* right-hand sides. The standard formulation is in terms of molar concentrations (changes to mass or mole fractions are possible, compare section 2). The additional

treatment of *diffusion* or the handling of *differential - algebraic* systems is presently not possible (in preparation). Generally speaking, the *modularised* form of the package facilitates changes of individual blocks.

In view of the possible solution of very large systems, *storage economy* has had a high priority in the development of the package. With the notation

NSP: number of species,

NCEQ: number of chemical equations
(reversible reactions count as 2),

NZE: number of non-zero elements in Jacobian matrix,

NLU: maximum number of non-zero elements in LU-decomposition,

the total workspace required can be estimated as follows:

integer workspace:

$$I := NLU + NZE + 6 \cdot NCEQ + 15 \cdot NSP$$

real workspace:

$$R := NLU + NZE + 3 \cdot NCEQ + 15 \cdot NSP$$

Upon referring to the rough considerations of [4], the following rules of thumb are obtained (with $r_{\max} := 7$, $\kappa := 2$):

$$I_{\text{est}} := 6 \cdot NCEQ + 36 \cdot NSP$$

$$R_{\text{est}} := 3 \cdot NCEQ + 36 \cdot NSP$$

For the largest example (RNA-polymerization) presented in this paper, these numbers lead to 34K from I_{est} (in integer *2) and 60 K from R_{est} in single precision or 120 K from R_{est} in double precision. For comparison, the actually required array storage

in this example turned out to be 32 K from I, 55 K from R in single precision, and 110 K in double precision. These numbers may be compared with about 100 K for the object code of the total package. Considerable savings, however, can be made by using OVERLAY techniques.

As a consequence, the present version of LARKIN should be helpful in simulating comparatively large chemical reaction systems on moderate size computers.

2. Input/Output Facilities

This section deals with details of the two dialogue blocks EDIT and OUTPUT, as far as they are relevant from the user's point of view. A complete description of the syntax of the chemical input would be beyond the scope of this paper. Rather, part of the documentation of problem HEXAN 1 of section 4 is inserted for illustration purposes. First, the block EDIT will be described in a sequential order. Note, however, that the actual user input may be given in arbitrary order.

Head (optional). This sub-block may contain a short documentation of the problem (up to 5 lines).

ISBARN, EDERER, EBERT:
THE THERMAL DECOMPOSITION OF N-HEXANE,
KINETICS, MECHANISM, AND SIMULATION.
(DATE OF LATEST CHANGE: JAN. 21, '82)

Elements (optional) - list of chemical element names, each of which may be followed by its atomic weight. The names are used in the subsequent stoichiometric test. The atomic weights are necessary for a possible change from molar concentrations to mass or mole fractions in the output.

H 1.008
C 12.011

Species (optional) - list of chemical species names, each of which may be followed by the element composition (for stoichiometric test).

H	H 1	
CH3	C 1	H 3
3-M-C4H8		C 5 H 10
E-BENZOL	C 8	H 10
STYROL	H 8	C 8

The order of appearance of the names defines the order in the output. Further species names appearing only in the block *reaction* are automatically added. However, it is strongly advised to use the optional stoichiometric test for the possible detection of input flaws.

Reaction - complete list of irreversible or reversible chemical reactions, each of which is followed by the associated kinetic parameters (or (ln A, ΔE) for Arrhenius law).

CH4	=>	H	+	CH3	(15.3,104000)
C2H6	=>	H	+	C2H5	(16.0, 98000)
C2H6	<=>	CH3	+	CH3	(16.6, 88D3) (10.7,0)
C2H5 + 1-M-AL	=>	3-M-C5H10	(9.0, 0.0)		
C2H5 + 1-E-AL	=>		(9 , 0)		

O2-	+	CS+	=>	CS	+	O2	(0.5D-7)		
CS+	+	E-	=>	CS			(0.1D-11)		
CS	=>	CS+	+	E-			(0.324D-2)		
O2-	=>	O2	+	E-			(0.4D0)		
O2	+	CS	+	M	=>	CSO2	+	M	(0.1D-30)
O2	+	O2	+	E-	=>	O2-	+	O2	(0.124D-29)
O2	+	E-	+	N2	=>	O2-	+	N2	(0.1D-30)

As can be seen from the above special example, the equations may be written in a chemical standard nomenclature. (The above insertions have deliberately been presented in a rather irregular shape to demonstrate the robustness of the package.)

Initial concentrations - list of non-zero initial concentrations in the form

C6H14 0.0021

Temperature (optional, fixed value) - only required, if either Arrhenius or modified Arrhenius law is used for the rate constants; may be skipped, if rate constants are prescribed as numbers.

Numerical Input - list of additional information, which is necessary in the simulation, such as: required relative accuracy, user prescribed output points (including initial and final point; output points, which are automatically selected by the integrator, are independently stored in database OUT for print or plot purposes), and print option parameter.

The above sub-blocks comprise the block EDIT. Every (even line-wise) change of the input is automatically updated in database CHEMIN.

All output information is available from database OUT, for further use either inside or outside the package LARKIN. The dialogue block OUTPUT offers possibly selected results of the simulation either in the form of numerical data or in graphic form. The numerical data in database OUT contain at least the concentrations at all intermediate points (both integrator selected and possibly user prescribed), if not suppressed by the above print option; from the experience of the authors, additional numerical information like reaction rates, integrated reaction rates, pressure etc. are helpful in the process of modelling and are therefore available from OUT. The graphic output contains several options including either single or double logarithmic scale and phase diagrams. Only minor

emphasis has been put into the development of the plot software - the adaptation of the graphic part of the package anyway requires a special interface to the local plotting facilities.

In view of any adaptations, the database OUT has been especially well-structured to facilitate the access.

3. Chemical Compiler System

The above input in terms of chemical reactions, rate constants, and initial concentrations uniquely defines an associated ODE initial value problem

$$(3.1) \quad y' = f(y) , \quad y(0) = y_0$$

with time t as independent variable. The actual generation of this ODE system is usually done by a so-called "chemical compiler" or "interpreter", - see [2, 7, 13, 17]. There is the following principal alternative: (a) generation of explicit FORTRAN subroutines f and f_y (required by standard stiff integrators) with the consequence of a necessary FORTRAN compilation step after each model change, (b) symbolic generation of f and f_y , which are evaluated by means of formal routines contained in the chemical kinetics package.

Advantages of (a) are

- lower evaluation time for f and f_y
- slightly more transparent program structure

Advantages of (b) are

- model changes do not require additional calls of the FORTRAN compiler
- smaller storage for object module

This implies that (b) is preferable for large problems and is certainly more convenient from the user's point of view even for smaller problems. In the standard version of LARKIN, ver-

sion (b) is realized; version (a) can be produced by a few minor changes. The limitations of version (a) may be clearly seen from its performance in the following

Example: RNA polymerization (NCEQ = 770, NSP = 352)
(due to [16])

Version (a): The listing of the source code of f and f_y is about 70 printed pages. After about 10 minutes compilation time, the FORTRAN compiler gives in with the message: "source program too large".

Version (b): A listing of the symbolic representation of f and f_y would be about 1 printed page. For comparison, note that about 2 sec are needed to generate f and f_y and less than 1 min. is needed to simulate the whole chemical system (see also section 4).

The symbolic generation of f uses partly ideas that have been suggested by Stabler/Chesick [17] in their realization of the package HAVCHM. Unlike HAVCHM, however, but similar to the package CHEMKIN due to Kee/Miller/Jefferson [13], LARKIN accepts a familiar chemical language formulation with only few format restrictions. This user-oriented structure of the input leads to the necessary implementation of a *scanning* procedure. All species names appearing in the optional species list or in the chemical reactions are identified and internally listed in an integer representation. The integer position in this list (called NAME) represents the internal coding. This coding is then used to update the structure of the reaction scheme line by line in two integer vectors. In the course of this scanning process, the rate constants (following each reaction) are simultaneously read and analyzed. The evaluation of f is done in a standard way as in comparable packages, say HAVCHM. The build-up of the list NAME needs a large number ($\sim 4 \cdot \text{NCEQ}$) of table look-ups. In order to avoid a blow-up of this look-up time in very large systems, a special *hash* technique is used (for details see Knuth [14]).

The symbolic generation of the Jacobian f_y would then be comparatively trivial, if a full mode updating scheme were applied (where each entry of f_y is characterized by its row and column index). However, in view of large systems, a sparse storage scheme seems to be crucial, which means that in addition to the non-zero elements of f_y the associated *sparse pattern* of $(I-hf_y)$ has to be provided. The pattern representation and the storage scheme in LARKIN are on one hand rather concise and on the other hand general enough to permit the use of any of the presently available sparse solvers. For more details see [4]. This side condition implies that the evaluation of f_y is technically more complicated than in packages that just use full mode representation.

By means of scanning routines similar to the one described above further information from database CHEMIN is picked up, analyzed, and transferred to the binary file INVAL. In the course of the compilation, a series of *error checks* are made, e.g.

- syntax monitor for each line,
- check on input inconsistencies,
- optional stoichiometric test.

If any of these checks is not passed, self-explanatory warning or error messages are issued.

The priorities in the construction and implementation of the chemical compiler system have been *storage economy*, *portability*, and *ease of handling* from the user's point of view. These priorities lead to a lot of rather technical but important details, which are not suited for a presentation within the scope of this paper. These details are anyway best documented by the code itself.

4. Numerical Simulation

The actual numerical simulation (= integration of the ODE system) is performed in the block SIMULA, which uses information only from database INVAL. Part of the information, which is not used, is just passed over to database OUT. Another part of the information is preprocessed before the call of the stiff integrator (e.g. computation of the rate constants by virtue of the Arrhenius law). In the course of the numerical integration, the intermediate results are transferred to database OUT by an extra routine. Once more note, that this interface structure facilitates a possible exchange of sub-blocks (such as the integrator).

The stiff integrator used in LARKIN is METASC, a special version of METAN 1 (due to [1]) adapted to large chemical kinetics problems. The code is based on the semi-implicit mid-point rule with polynomial extrapolation - for details see [1,4]

The special adaptation of METASC refers to

- a scaling technique, which uses further information from mass action kinetics (see [4], p. 45 - 48),
- the additional computation of quantities of chemical interest,
- the sparse solver MA 30 in the version developed by Duff/Nowak [5].

Extrapolation integrator versus BDF integrator. Nearly all chemical kinetics packages use a stiff integrator of BDF-type for the numerical simulation. The question to be discussed now is whether the new stiff integrator of extrapolation type is competitive. A number of numerical comparisons have already been documented in [1,4] including standard test problems and chemical kinetics examples. In order to give some impression about the performance of the present version of LARKIN, a few further chemical kinetics examples are presented here - using

either the standard extrapolation integrator or the BDF-integrator due to Gear [8] and Hindmarsh [10]. Of course, both integrators have been endowed with common scaling and sparse techniques. All comparison runs have been made on the IBM 370/168 of the University of Heidelberg using the FORTRAN optimizing compiler with OPT = 3. The following 3 problems may stand for a larger set of problems that have been solved meanwhile:

HEXAN1: thermal decomposition of n-hexane (due to Isbarn, Ederer, Ebert [12]) NCEQ = 240, NSP = 59, NZE = 523

HEXAN2: extended model of n-hexane (due to Isbarn [11])
NCEQ = 705, NSP = 166, NZE = 1270

RNA: polymerization model of RNA (due to Schneider, Heinrichs, Poll [16])
NCEQ = 770, NSP = 352, NZE = 2642

The comparative results for the extrapolation code METASC and the BDF-code GEARS for single precision (tolerance 1.E-3 with a mantissa of about 6 decimal digits) and double precision (tolerance 1.D-3 and 1.D-6 with a mantissa of about 16 decimal digits) are listed in Table 1. The column GEARS with maxord = 3 has been added, because in this case the $A(\alpha)$ -stability properties of the two underlying discretizations are comparable - see Table 2. From Table 1, the following observations are worth discussing:

Table 1. Comparison of simulation times [sec].

	METASC	GEARS maxord = 5	GEARS maxord = 3
HEXAN1			
1.E-3	3.98	Fail*)	Fail*)
1.D-3	3.46	4.09	3.66
1.D-6	7.50	7.14	8.83
HEXAN2			
1.E-3	10.63	12.05	12.02
1.D-3	12.30	11.14	11.10
1.D-6	22.62	21.15	24.22
RNA			
1.E-3	48.28	38.70	41.86
1.D-3	55.68	44.46	48.20
1.D-6	114.36	106.59	147.23

*) fail run also with maxord = 2

- (I) In the largest example (RNA), the code GEARS with maxord = 5 is about 20% faster than METASC. A possible explanation of this fact might be that the dynamic behavior of that chemical system turns out to be rather regular - compare Fig. 2. This property favors multistep methods of this kind. In the other successful runs, the computing times are nearly the same.

(II) In the single precision version, METASC is more robust than GEARS. This property seems to be typical over a wider class of examples.

Table 2. Comparison of $A(\alpha)$ -stability properties [9]
(for discretization of order k)

BDF discretization:

k	1	2	3	4	5	6
α	90°	90°	86°	76°	50°	16°

semi-implicit mid-point discretization:

k	1	3	5	7	9	11
α	90°	90°	90°	87°	86°	86°

In view of *storage economy* (compare section 2) and the fact that *low accuracy* requirements are often sufficient for chemical applications, the single precision version of LARKIN seems to be preferable. For low accuracies, the BDF code usually supplies enough intermediate output points to permit the use of a simple plot routine, whereas the extrapolation code tends to supply comparatively few output points. However, the few output points obtained by the new order and stepsize control (due to Deuflhard [3]) reflect the dynamic behavior of the chemical system so that a more sophisticated plot routine (such as [15]) can be used instead.

The above comparative remarks seem to justify the decision to use the stiff integrator of extrapolation type in the simulation package LARKIN.

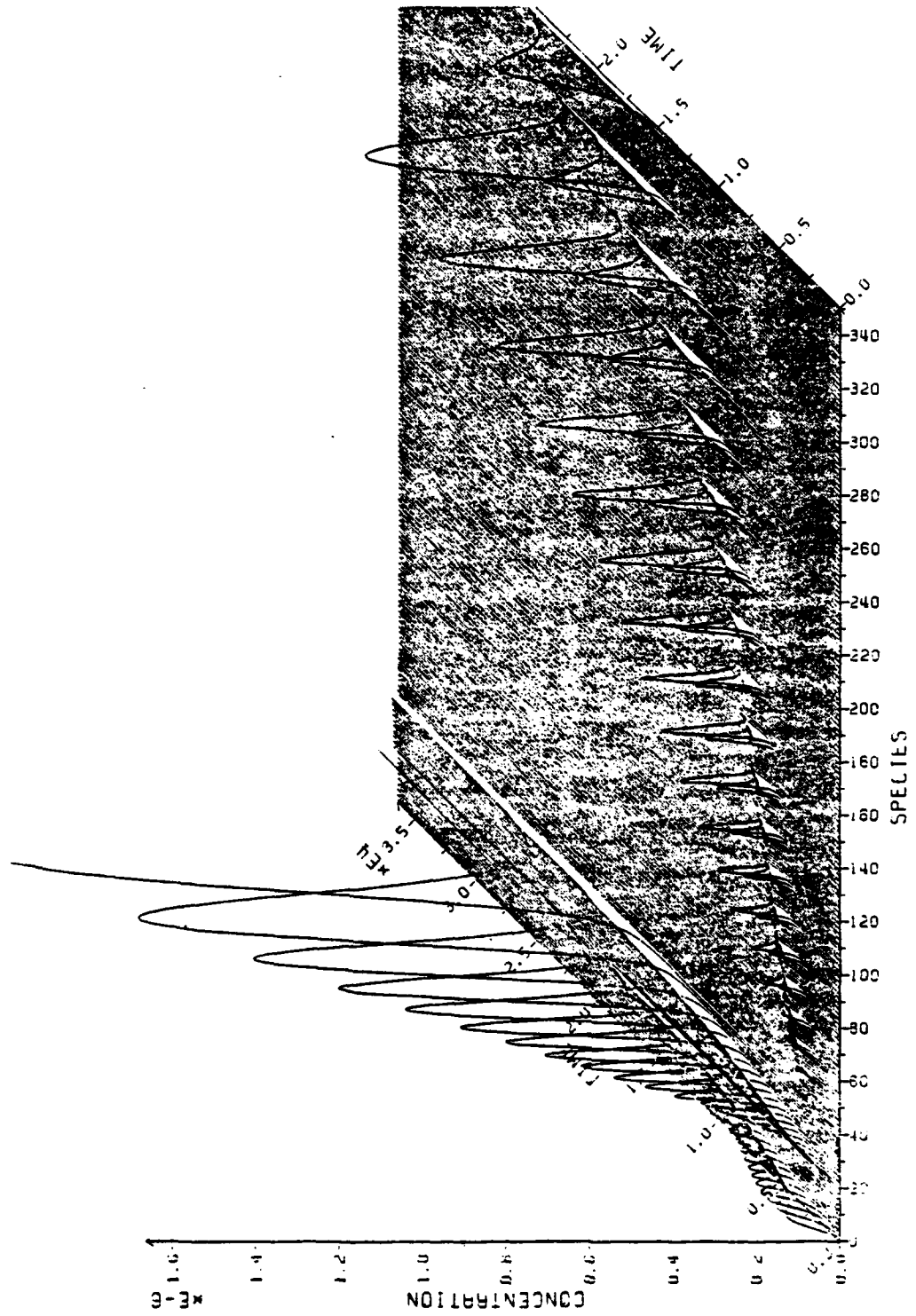


Fig. 2: RNA problem: graphical output for 349 (out of 352) species

Acknowledgments. The authors wish to thank H.J. Ederer and G. Isbarn for extensive helpful discussions concerning the chemist's point of view. They are indebted to F.W. Schneider and M. Heinrichs for ceding the RNA polymerization example to them. Further, they are grateful to P. Kunkel for his programming assistance. The work of the authors has been supported by the Deutsche Forschungsgemeinschaft.

References

- [1] G. Bader, P. Deuflhard:
A Semi-Implicit Mid-Point Rule for Stiff Systems
of ODEs.
Univ. Heidelberg, SFB 123: Tech. Rep. 114 (1981).
- [2] A.R. Curtis:
The FACSIMILE numerical integrator for stiff in-
itial value problems.
AERE Harwell, Tech. Rep. AERE-R 9352 (1979).
- [3] P. Deuflhard:
Order and Stepsize Control in Extrapolation Methods.
Univ. Heidelberg, SFB 123: Tech. Rep. 93 (1980).
- [4] P. Deuflhard, G. Bader, U. Nowak:
LARKIN - a software package for the numerical sim-
ulation of LARGE systems arising in chemical reaction
KINetics.
In[6], p. 38 - 55 (1981).
- [5] I.S. Duff, U. Nowak:
On sparse matrix techniques in a stiff integrator
of extrapolation type.
Univ. Heidelberg, SFB 123: Techn. Rep. (1982).
- [6] K.H. Ebert, P. Deuflhard, W. Jäger (ed.):
Modelling of Chemical Reaction Systems.
Springer Series Chem. Phys. 18 (1981).
- [7] D. Edelson:
A Simulation Language and Compiler to Aid Computer
Solution of Chemical Kinetics Problems.
Computers and Chemistry 1, 29 - 33 (1976).
- [8] C.W. Gear:
Numerical Initial Value Problems in Ordinary Dif-
ferential Equations.
New York: Prentice Hall (1971).

- [9] E. Hairer, G. Bader, C. Lubich:
On the Stability of Semi-Implicit Methods for ODE's.
Univ. Heidelberg, SFB 123: Tech. Rep. 122 (1981).
- [10] A.C. Hindmarsh:
Gear - Ordinary Differential Equation Solver.
Lawrence Livermore Laboratory:
Tech. Rep. UCID - 30 001, Rev. 3 (Dec. 1974).
- [11] G. Isbarn:
Private communication (1981).
- [12] G. Isbarn, H.J. Ederer, K.H. Ebert:
The Thermal Decomposition of n-Hexane: Kinetics,
Mechanism, and Simulation.
In [6], p. 235 - 248 (1981).
- [13] R.J. Kee, J.A. Miller, T.H. Jefferson:
CHEMKIN: A General-Purpose, Problem-Independent,
Transportable, Fortran Chemical Kinetics Code Package.
Sandia National Laboratories, Livermore:
Tech. Rep. SAND 80 - 8003 (1980).
- [14] D.E. Knuth:
The Art of Computer Programming. Vol. III: Sorting
and Searching
Addison - Wesley (1973).
- [15] P. Rentrop:
An Algorithm for the Computation of the Exponential
Spline.
Num. Math., Handbook Series Approx. 35, 81 - 93 (1980).
- [16] F.W. Schneider, M. Heinrichs, T. Poll:
Private communication (1980).
- [17] R.N. Stabler, J.R. Chesick:
A Program System for Computer Integration of Multi-
step Reaction Rate Equations Using the Gear Integra-
tion Method.
Int. J. Chem. Kin. 10, 461 - 469 (1978).

CHEMICAL KINETICS - AN OVERVIEW FROM THE POINT OF VIEW OF
NUMERICAL ANALYSIS AND SOFTWARE IMPLEMENTATION

Lennart Edsberg

Department of Numerical Analysis and
Computing Sciences

The Royal Institute of Technology
Stockholm SWEDEN

Contents

1. Introduction
2. Some mathematical properties of chemical kinetics
3. The development of software for chemical kinetics
4. The use of scaling of the variables in the set of ordinary differential equations
5. Conclusions

Chemical kinetics - an overview from the point of view of Numerical Analysis and Software Implementation

Lennart Edsberg

1. Introduction

Numerical simulation of chemical kinetics is one important application area of stiff computation. During the last decade, when software for stiff problems has been available, even bigger systems of chemical reactions has been simulated on computers. Since most chemical kinetics problems are stiff, the numerical simulation using non-stiff methods such as Runge-Kutta is in general not successful, so before the existence of stiff numerical software it was common to use a combination of analytical as well as numerical tools in order to be able to obtain approximate solutions of the set of ordinary differential equations describing the kinetics of the system. In the analysis of such a system the chemist is often guided by some a priori knowledge about the magnitudes of the concentrations of the components. This information has been utilized especially in small systems in a way that sometimes makes it possible to partition the system into stiff and non-stiff variables. A good example of such an analysis can be found in Lin and Segel [1], pp 304-307, where a biochemical enzyme catalysis reaction system



is treated. Setting up the system of ordinary differential equations (ODE's) for this system and scaling the variables with respect to their magnitudes leads to the following set of ODE's for the scaled concentrations of S and C:

$$\begin{aligned} \dot{s} &= -s + (s + \kappa - \lambda)c & s(0) &= 1 \\ \epsilon \dot{c} &= s - (s + \kappa)c & c(0) &= 0 \end{aligned} \quad \dots(2)$$

where κ and λ are parameters being functions of the reaction rates of the system and ϵ is a small quantity. - After this analytical preprocessing leading to a singular perturbation formulation of the problem a familiar approximation method known as the steady-state approximation (SSA) could be used. In the example given above this method consists of letting $\epsilon=0$, hence giving a system of differential-algebraic equations. This approximation of the dynamics of ... (1) is referred to as the Michaelis-Menten kinetics.

Although the SSA has turned out to be unreliable for general problems the scaling procedure leading to it gives an important piece of information and can be utilized numerically.

In this paper we give an overview of chemical kinetics problems, some important mathematical properties and how the numerical treatment is performed in existing simulation programs for chemical kinetics. There is also shown some recent results on a scaling procedure proposed by Dahlquist leading to a singular perturbation formulation of the ODE's and how this information can be used from numerical point of view.

2. Some mathematical properties of chemical kinetics

Consider a closed system where several chemical reactions are taking place simultaneously. The system is held at constant temperature and the volume does not change. It is also assumed that the system is operating under homogeneous conditions, i.e. no spatial gradients occur. - Under these very idealized conditions we have a reactor, in chemistry usually called the isothermal batch reactor. Although operating under these restricted conditions this reactor is found in many important applications.

Assume that n chemical species A_i , $i=1,2,\dots,n$ simultaneously interact in m reactions



where p_{ij} and q_{ij} are integers (stoichiometric coefficients) and k_j are parameters called the rate-constants of the reactions. Under the physical conditions mentioned above and the assumption that the mass-action law is applied for the rate-functions, the following mathematical model consisting of a set of ordinary differential equations can be set up:

$$\dot{c}_i = \sum_{j=1}^m (q_{ij} - p_{ij}) k_j \prod_{k=1}^n c_k^{p_{kj}} \quad c_i(0) = c_{i0}, \quad \dots(4)$$

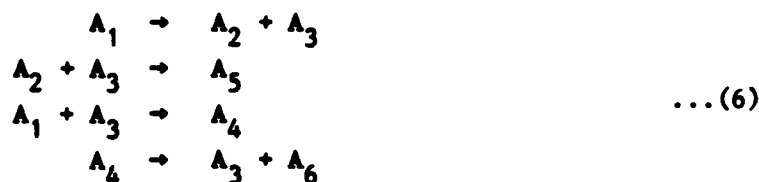
where c_i are the concentrations of A_i and c_{i0} are the concentrations of A_i at time $t = 0$. The system $\dots(4)$ can be written in compact form:

$$\dot{\bar{c}} = (Q - P)K\bar{g}(\bar{c}), \quad \bar{c}(0) \text{ given} \quad \dots(5)$$

where Q and P are $n \times m$ matrices, K a diagonal matrix $= \text{diag}(k_j)$ and $\bar{g}$ is a vector with m components containing the polynomial expressions in $\dots(4)$. The vector $K\bar{g}$ is denoted by $\bar{r}$ and is a vector of m components containing the rate-functions (formed by the mass action law). The matrix $S = Q - P$ is referred to as the stoichiometric matrix.

Thus a chemical kinetics problem is specified by the two matrices S and P (usually being very sparse with integers as entries), the diagonal matrix K (with only positive entries) and the initial concentration vector $\bar{c}(0)$ (with nonnegative entries).

Example (chemical pyrolysis, originating from Datta 1967, the set of ODE's being one of the testproblems in the stiff integrator comparison by Enright, Hull and Lindberg [2].



The corresponding set of ODE's are according to ... (4)

$$\begin{aligned}
 \dot{c}_1 &= -k_1 c_1 - k_3 c_1 c_3 \\
 \dot{c}_2 &= k_1 c_1 - k_2 c_2 c_3 \\
 \dot{c}_3 &= k_1 c_1 - k_2 c_2 c_3 - k_3 c_1 c_3 + k_4 c_4 \\
 \dot{c}_4 &= k_3 c_1 c_3 - k_4 c_4 \\
 \dot{c}_5 &= k_2 c_2 c_3 \\
 \dot{c}_6 &= k_4 c_4
 \end{aligned}
 \quad \dots(7)$$

Only the first four equations were used in the testproblem in [2]

$$S = \begin{bmatrix} -1 & 0 & -1 & 0 \\ 1 & -1 & 0 & 0 \\ 1 & -1 & -1 & 1 \\ 0 & 0 & 1 & -1 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad P = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

For this system $n = 6$ and $m = 4$. The rate constants used in the test problem were $k_1 = 7.89 \cdot 10^{-10}$, $k_2 = 1.13 \cdot 10^9$, $k_3 = 1.1 \cdot 10^7$, $k_4 = 1.13 \cdot 10^3$ and the initial values $c_1(0) = 1.76 \cdot 10^{-3}$, the others were set to zero.

Note that $\dot{c}_2 = \dot{c}_3 + \dot{c}_4$ and $2\dot{c}_1 + \dot{c}_2 + \dot{c}_3 + 3\dot{c}_4 + 2\dot{c}_5 + 2\dot{c}_6 = 0$ i.e. there are linear relations between the variables due to rank deficiency in the matrix S . In fact, for this example, $\text{rank}(S) = 3$. The wide spread of size in the rate constants is the reason of stiffness.

In real world problems we cannot of course neglect the contributions from transport- and energyprocesses such as diffusion, convection, changes in the temperature etc. This seems to imply that problems that can be formulated as ... (5) are academic and has little to do with real problems. However, the model is used in practise and can be found in application areas such as atmospheric chemistry, physical chemistry, biochemistry, metallurgy etc. The size of such problems can vary from a few reactions and species in e.g. physical chemistry to several hundred in biochemistry. The systems can also exhibit all kinds of numerical problems such as different degrees of stiffness and instability phenomena in e.g. oscillating systems.

The model ... (5), which we have discussed so far, is thus applicable to the isothermal batch reactor. Examples of reactors where transport and energy-processes must be taken into account is the continuous-flow stirred tank reactor (CSTR) and the plug-flow tubular reactor (PFTR), which however can be modelled by sets of ODE's obtained by making small modifications of ... (3), see also Edsberg [3] for further discussion. For the CSTR the equations are

$$\dot{\bar{c}} = \frac{1}{\theta}(\bar{c}_f - \bar{c}) + (Q - P)K_g(\bar{c}) \quad \dots(8)$$

where θ is a constant and $\bar{c}_f$ is a constant vector.

Another type of generalisation occurs in systems where spatial gradient cannot be neglected. This type of model is often used in atmospheric chemistry and consists of coupled partial differential equations of parabolic type:

$$\dot{c}_i = -\text{div}(c_i \bar{v}) - \text{div}(D_i \text{grad} c_i) + r_i \quad \dots(9)$$

where r_i are the components of a vector of molar rates, which for mass action kinetics, is the right hand side of ... (5).

For systems of ODE's coming from chemical kinetics it is obvious that the variation of the variables c_i is restricted to certain bounds. Returning to the model ... (5) which is valid under closed conditions the total mass must be preserved. This can be expressed by

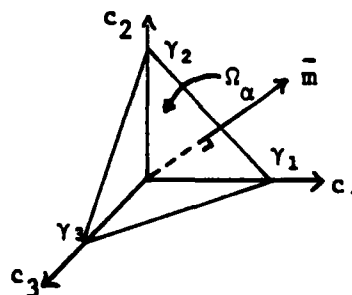
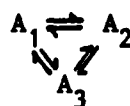
$$\bar{m}^T \dot{\bar{c}} = \bar{m}^T \bar{c}(0) = \alpha \quad \dots(10)$$

where $\bar{m}$ is a vector of molecular weights of the species A_i . Thus

$$\bar{m}^T \dot{\bar{c}} = 0 \quad \dots(11)$$

which means that $\dot{\bar{c}}$ belongs to a subspace L being orthogonal to the vector $\bar{m}$, which is directed into the positive orthant R_n^+ . The concentration vector $\bar{c}$ then belongs to a hyperplane L_α being parallel with L and intersecting the coordinate axis at intercepts $\gamma_i > 0$.

Example A system of three reversible reactions involving three species

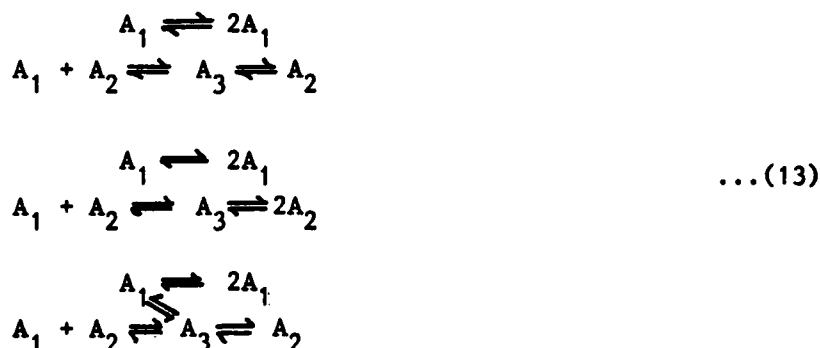


The area $\Omega_\alpha = R_n^+ \cap L_\alpha$ is called the reaction simplex and is the domain in R_n^+ in which a solution of ... (5) resides. This of course demands that $c_i(t) \geq 0$, $i = 1, 2, \dots, n$, which however is true for mass action kinetics if $c_i(0) \geq 0$. Hence from ... (10) we see that

$$0 \leq c_i(t) \leq \alpha/m_i \quad \dots(12)$$

which is an a priori bound for every solution of ... (5)

Moreover, since the ODE's describing the kinetics can be set up from the reaction network ... (3) there is a lot of information of the reaction structure contained in the right hand sides of ... (5). This information has been used by Feinberg, Horn and Jackson in an extensive qualitative analysis of the solution manifolds of ... (5). An overview of their work can be found in Feinberg [4]. The results of their theory consist of statements of the number of steady states and their stability properties. Their theory is restricted to specified classes of chemical reaction networks, namely those having positive steady states belonging to the interior of Ω_α . This excludes networks containing such irreversible steps that lead to a steady state on $\partial\Omega_\alpha$. In other cases, however, the theory assures a single positive steady state which is asymptotically stable for networks which among other conditions have a generalised reversibility character called weak reversibility. It would lead too far to give an overview of their results in this paper, but the statements about the steady state that can be said a priori is certainly of importance when you want to know what can be expected of the solution trajectories from a certain kinetics problem. To illustrate this, consider the following networks, given by Feinberg [4]:



Of these very similar three reaction schemes only the second has a single stable positive steady state for all positive values of the six rate constants in each reaction simplex. The first network on the other hand may have multiple positive steady states for certain values of the six rate constants while the third network cannot have multiple steady states, but for certain values of the eight rate constants an unstable steady state and a stable limit cycle.

3. The development of software for chemical kinetics

All since the beginning of use of digital computers for numerical simulation of dynamical systems, the problem class of chemical kinetics has attracted much interest both among people developing numerical software and among the presumptive users, i.e. the chemists. Several program packages have been implemented during the last two decades. Apart from the practical importance and interest of the problem itself, one reason for this development is probably the appealing structure of the problem of chemical kinetics which among other things makes it possible to program a translator which transforms the chemical network ... (3) into the set of ODE's ... (5). Another reason is perhaps that numerical analysts in this problem class have found an appropriate test-batch for stiff problems making it possible to try an ODE-method on problems of varying size, stiffness and even instability phenomena.

Since the first implementations of implicit methods, such as Gear's method, the list of software for chemical kinetics has increased and here we give just a choice of software that has been published: BELLCHEM by Edelson [5], HAVCHM by Stabler and Chesick [6], KINRATE by Edsberg and Uhlén [7], FACSIMILE by Curtis [8], MACKSIM by Carver and Boyd [9], KISS by Gottwald [10] and LARKIN by Deuflhard, Bader and Nowak [11].

The common features about chemical kinetics programs are that the user can work interactively with the computer using an alpha-numerical or graphical terminal and/or some plotter. The purpose for writing such programs is also that the user need not be acquainted with the numerical methods, but should be able to concentrate upon the chemistry and have visualized how the kinetics of a certain reaction system behaves for different values of initial concentrations $\bar{c}(0)$, rate constants K or the reaction structure itself i.e. Q, P .

Up to now we have discussed only the simulation problem, i.e. the behaviour of the concentrations as function of time. One should, however, be aware of the fact that this is just one important computational aspect of chemical kinetics. A rough partitioning of this and other numerical problems is:

1. Computation of the steady state, i.e. determining $\bar{c}(\infty)$ in ... (5)
2. Simulation of the dynamical behaviour, i.e. solution of the set of ODE's ... (5)
3. Determination of unknown rate constants k_i in ... (5) from experimental measurements of the system.
4. Identification of the mechanism, i.e. determining the matrices Q and P in ... (5) from experiments.

The problems are probably given in increasing order of difficulty, but they cannot be separated from each other, e.g. for some problems $\bar{c}(\infty)$ cannot be solved without solving the ODE's and the determination of rate constants must

be based on an efficient algorithm for simulation.

Returning to the simulation problem: when implemented on a computer it is natural to split a chemical kinetics program into a number of segments, i.e. the program package consists of several parts which communicate through data-files. This is the case for e.g. the programs KINRATE, FACSIMILE and LARKIN, where you can distinguish between the following parts:

1. TRANSLATOR, which transforms the reactions given as input by the user into either a code for the right hand sides $\bar{f}(\bar{c})$ in ..(5) and the corresponding Jacobian $\partial \bar{f} / \partial \bar{c}$ or some sparse representation of the matrices Q and P (or S and P), from which it is possible in a later phase of the program to compute $\bar{f}(\bar{c})$ and the Jacobian when they are needed.
2. PARAMETER-ASSIGNER, which obtains as input the rate constants k_i and the initial concentrations $c_i(0)$. Here the user may also specify certain numerical parameters, such as the starting and final point of integration, the tolerance for the ODE-solver, scaling factors for the components, possibly also initial stepsize and the largest allowed stepsize in the integration interval.
3. SIMULATOR, which performs the numerical integration of ...(5). The user may here prescribe the times t_i where the solution $\bar{c}(t_i)$ should be printed out. In this part it may also be possible for the user to decide how to act if the simulation for some reason goes wrong.
4. POSTPROCESSOR, with the help of which the user may look at the terminal or plot on a plotter the results from the simulation parts in certain interesting intervals or examine more closely just a few of the components c_i of the reaction system.

Most programs for chemical kinetics are implemented in FORTRAN, but the program package KISS by Gottwald is written in SIMULA the better structuring-possibilities of which has made it possible to write the whole program in one module.

We have already mentioned the possibility of translating a chemical network into the right hand sides $\bar{f}(\bar{c})$ of ...(5) with the help of a program. Since the components of $\bar{f}(\bar{c})$ are built up by polynomial expressions, the Jacobian can be formed analytically also with the help of a program. The formal expression for the Jacobian is

$$\partial \bar{f} / \partial \bar{c} = SK \text{diag}(g_i) P^T \text{diag}(1/c_i) \quad \dots(14)$$

if $c_i \neq 0$. The Jacobian is then used in the Newton-iterations when solving numerically the set of nonlinear equations

$$\bar{c} - h\beta \bar{f}(\bar{c}) + \bar{d} = 0 \quad \dots(15)$$

in each time-step. Here $\bar{c}$ is the approximate solution at time $t+h$, t the former time at which the solution was computed, β is a method-dependant

constant and $\bar{d}$ contains information from the solution at time t . Hence a set of linear equations

$$A\bar{x} = (I - h\beta\bar{f}/\partial\bar{c})\bar{x} = \bar{b} \quad \dots(16)$$

has to be solved for each Newton-iteration. This is done by using the LU-decomposition of A , L and U being retained as long as convergence properties are satisfactory. Each time $h\beta$ is changed or some elements of the Jacobian have changed so much that convergence is too slow, the LU-decomposition is updated by first recomputing the Jacobian then computing the new LU-decomposition of A . Comments on the calculation of the Jacobian numerically or analytically is found in Curtis [12].

The set of linear equations $\dots(16)$ should be solved by some sparse technique at least if the number of chemical species is large. Sparse techniques are used in e.g. FACSIMILE and LARKIN, while the full matrices are treated in e.g. KINRATE and KISS. This sparse technique is a general purpose method that can be used for any sparse system, thus no extra information based on the special structure of chemical kinetics problems is used. An up to date overview of the solution of the nonlinear equations $\dots(15)$ is found in Söderlind [13].

So far, no attempt has been made to compare different programs for chemical kinetics with each other. However, comparisons of stiff ODE-solvers applied to chemical problems have been performed and results can be found in e.g. Enright and Hull [14] and in Gottwald [15].

4. The use of scaling of the variables in the set of ordinary differential equations of chemical kinetics.

The possibility of giving scaling factors for the variables c_i in $\dots(5)$ to be used in the local error estimate of the stiff ODE-solver has already been pointed out in section 3. Different ways of performing automatic scaling of the local error estimate is discussed by Curtis in [12] and by Deuflhard in [11]. In some numerical experiments reported by Edsberg [3] it was found that for certain chemical systems some scaling device seemed to be necessary in order to obtain reliable approximations of the solution trajectories at least when the machine accuracy and/or the tolerance used in the numerical integrator was not small enough. In [3] the scaling factors were chosen from a priori knowledge of the size of the variables c_i and scaling was tested both by transforming the set of ODE's $\dots(5)$ to a set of ODE's for the scaled variables and by using different weights in the norm of the local error estimate used to control the stepsize strategy. In both cases the results were satisfactory when tried on some stiff ODE-solvers if scaling was used, but unreliable if no scaling was used (corresponding to using an absolute error tolerance). Similar results are reported by Curtis[12].

A more systematic approach to the scaling problem is given by Dahlquist,

Edsberg, Skölleremo and Söderlind [16], who present a semi-automatic algorithm for scaling the variables. The algorithm is tested on some chemical kinetics problems and the scaling procedure emerges in a singular perturbation formulation of the set of ODE's ... (5)

$$\begin{aligned}\dot{\bar{x}}_1 &= S_1 \bar{f}_1(\bar{x}_1, \bar{x}_2, \epsilon) & \bar{x}_1(0) \\ \epsilon \dot{\bar{x}}_2 &= S_2 \bar{f}_2(\bar{x}_1, \bar{x}_2, \epsilon) & \bar{x}_2(0)\end{aligned} \quad \dots (17)$$

where $\bar{x} = \begin{bmatrix} \bar{x}_1 \\ \bar{x}_2 \end{bmatrix}$, $S = \begin{bmatrix} S_1 \\ S_2 \end{bmatrix}$ and $\bar{x}$ is a vector of the variables c_i scaled by certain scaling factors. $\bar{x}$ and S are then partitioned according to the non-stiff variables $\bar{x}_1$ and the stiff variables $\bar{x}_2$. ϵ is a small parameter. The system ... (17) is then treated with a stiff ODE-solver.

The scaling procedure is simply based on a diagonal scaling of the dependant variables c_i and the time t :

$$\begin{aligned}\bar{c} &= E\bar{x} \\ t &= e_0\tau\end{aligned} \quad \dots (18)$$

E is a diagonal matrix with e_i in the diagonal, $i=1,2,\dots,n$. With this transformation inserted into ... (5) we obtain

$$E\bar{x}' = S\bar{K}\bar{g}(\bar{x}) \quad , \quad \bar{x}(0) = E^{-1}\bar{c}(0) \quad \dots (19)$$

where $\bar{K} = e_0 K G(\bar{e})$, $G(\bar{e}) = \text{diag}(g_i(\bar{e}))$... (20)

where $\bar{e}$ is a vector of the e_i , $i=1,2,\dots,n$, and ' denotes differentiation with respect to τ . Thus only the rate constants k_i and the initial values $c_i(0)$ are affected by this transformation; neither S nor $\bar{g}$ is changed (provided E is kept on the left hand side in front of the derivatives). In this transformed system ... (19) we now try to use the scaling factors to identify which variables are stiff and which are non-stiff.

The way of finding the scaling factors e_i , $i=0,1,\dots,n$ is outlined by Dahlquist and Mao-Zu-Fan [17] and is based on the following assumptions:

1. In a time-interval $[\tau_1, \tau_2]$ where the scaling is performed the scaled variables x_i and their derivatives x_i' are of the same order of magnitude $O(1)$.
2. The vector $\bar{x}$ is partitioned into two vectors $\bar{x}_1$ and $\bar{x}_2$. For the vector $\bar{x}_1$ (the non-stiff variables) the derivatives x_i' balance at least one term on the right hand side. For the vector $\bar{x}_2$ (the stiff variables) there is balance between terms of opposite sign in the left hand side. (Balance here means that two terms are of the same order of magnitude)

If the scaling procedure succeeds in balancing terms the system will be transformed into singular perturbation form. In [17] there is outlined an algorithm which gives the scaling factors e_i provided the "user" gives suggestions of which terms balance each other in all equations. The problem can then be formulated as a linear program which tries to find the e_i

that match the balance conditions as well as possible. Work is now going on at our department which aims at designing an automatic scaling-procedure.

We illustrate the scaling procedure here on the problem ... (7) and the advantages we get of finding the transformed system ... (17)

With rate-constants and initial values inserted and after using one of the linear relations between the c_i 's the system can be written:

$$\begin{aligned} \dot{c}_1 &= -7.89 \cdot 10^{-10} c_1 - 1.1 \cdot 10^7 c_1 c_3, & c_1(0) &= 1.76 \cdot 10^{-3} \\ \dot{c}_2 &= 7.89 \cdot 10^{-10} c_1 - 1.13 \cdot 10^9 c_2 c_3, & c_2(0) &= 0 \\ \dot{c}_4 &= 1.1 \cdot 10^7 c_1 c_3 - 1.13 \cdot 10^3 c_4, & c_4(0) &= 0 \\ c_2 &= c_3 + c_4 \end{aligned} \quad \dots (21)$$

Let the scaling factors e_i be expressed in powers of ϵ , being a small positive given number

$$e_i = \epsilon^{\alpha_i} \quad \dots (22)$$

The problem is now to make a hypothesis in each equation about the leading terms which balance each other. In the first equation, since both terms on the right hand side have the same sign one of them (or both) balance the derivative, let's assume the second term does. Assume also that c_2 and c_4 are stiff variables hence the terms on the right hand sides balance each other in the last two ODE's. In the algebraic equation assume that all terms are of the same magnitude, i.e. both terms on the right hand side balance the left hand side. Finally, guided by the magnitude of the initial value let $\epsilon = 10^{-3}$.

The balance assumptions now give the following relations:

$$\begin{aligned} \epsilon^{\alpha_1 - \alpha_0} &= k_3 \epsilon^{\alpha_1 + \alpha_3} \\ k_1 \epsilon^{\alpha_1} &= k_2 \epsilon^{\alpha_2 + \alpha_3} \\ k_3 \epsilon^{\alpha_1 + \alpha_3} &= k_4 \epsilon^{\alpha_4} \\ \alpha_2 &= \alpha_3 = \alpha_4 \end{aligned} \quad \dots (23)$$

With the known numerical values inserted into ... (23) and rounding every rate konstant to the closest power of ten we get the following set of linear equations:

$$\begin{aligned} 3\alpha_0 &= 7 - 3\alpha_3 \\ -10 - 3\alpha_1 &= 9 - 3\alpha_2 - 3\alpha_3 \\ 7 - 3\alpha_1 - 3\alpha_3 &= 3 - 3\alpha_4 \\ \alpha_2 &= \alpha_3 = \alpha_4 \end{aligned} \quad \dots (24)$$

The closest integer solution of ... (24) is $\alpha_0 = -1$, $\alpha_1 = 1$, $\alpha_2 = \alpha_3 = \alpha_4 = 4$ which gives the following set of scaled ODE's

$$\begin{aligned} x_1' &= 0.789 \epsilon^2 x_1 - 0.011 x_1 x_2 & x_1(0) &= 1.76 \\ \epsilon x_2' &= 0.787 x_1 - 0.011 x_2 x_3 & x_2(0) &= 0 \\ \epsilon^2 x_4' &= 11 x_1 x_3 - 1.13 x_4 & x_4(0) &= 0 \\ x_2 &= x_3 + x_4 \end{aligned} \quad \dots (25)$$

where $t = 10^3 \tau$, $c_1 = 10^{-3} x_1$, $c_i = 10^{-12} x_i$, $i=2,3,4$

From the equations ...(25) we now have good information of the qualitative behaviour of the solution and we have also obtained a natural scaling of each variable where 1 is a proper measure of each variable in a time interval where the variables x_2 and x_4 have reached a nearly steady-state. We see that the transient phase consists of one very quick phase determined by ϵ^2 and one less fast interval determined by ϵ . Hence we get a priori information of what can be expected of the stepsize-regulation of a stiff ODE-solver.

So after this "analytical preprocessing" of the system the problem is now well suited for being treated by a stiff method, and with very moderate value given to the tolerance parameter we obtain the solution with good relative accuracy.

Thus, if it is possible to find a scaling that transforms the set of ODE's ...(5) into the form ...(17) we have gained the following advantages:

1. We have better insight into the dynamics of the problem
2. The problem is better suited for numerical treatment with a stiff numerical integrator; we have variables that are scaled so that 1 is appropriate measure of the size of all variables.
3. The partitioned form ...(17) makes it possible to use numerical methods that utilize this form, see Söderlind [13].

5. Conclusions

In this paper we have given a brief survey of some mathematical and numerical properties in chemical kinetics. We have also given some examples of existing software for interactive simulation on a computer by using stiff numerical integrators. It is further indicated that the special structure of the set of ordinary differential equations of chemical kinetics makes it possible to transform the equations into a singular perturbation formulation by using a scaling procedure. The transformed equations are then used in the numerical integration, giving more reliable results. At our department this scaling procedure is investigated with the aim to develop an algorithmic description which utilizes all information that can be deduced about the problem before performing the numerical integration. In this paper we also point out that other numerical problems are of great importance for chemical kinetics, i.e. the identification problem

References

- 1 C.C. Lin and L.A.Segel, "Mathematics applied to deterministic problems of natural sciences", Mac Milan, New York, 1974
- 2 W.H.Enright, T.E.Hull and B.Lindberg, "Comparing Numerical Methods for Stiff Systems of ODE's", BIT 15, pp 10-49, 1975
- 3 L.Edsberg, "Numerical Methods for Mass Action Kinetics" in "Numerical Methods for Differential Systems" (L.Lapidus and W.E.Schiesser eds), Academic Press, 1976
- 4 M.Feinberg, "Reaction Network Structure, Multiple Steady States, and Sustained Composition Oscillations: A Review of Some Results" in "Modelling of Chemical Reaction Systems" (K.H.Ebert, P.Deuflhard and W.Jäger eds), Springer, Berlin Heidelberg, 1981
- 5 D.Edelson, "A simulation language and compiler to aid computer solution of chemical kinetic problems", Comput. Chem. 1, pp 29-33, 1976
- 6 R.N.Stabler and J.P.Chesick, "A program system for computer integration of multistep reaction rate equations using the GEAR integration method", Int. J. Chem. Kin. 10, pp 461-469, 1978
- 7 M.Uhlén and L.Edsberg, "KINRATE and KINBOX, two program packages for interactive simulation of chemical systems", TRITA-NA-7912, Dep. of Num. Anal. and Comp. Sci., The Royal Institute of Technology, Stockholm, Sweden, 1979
- 8 A.R.Curtis, "The FACSIMILE Numerical Integrator for Stiff Initial Value Problems", Techn. Rep. AERE - R 9352, Harwell, 1979
- 9 M.B.Carver and A.W.Boyd, "A program package using stiff sparse integration methods for the automatic solution of mass action kinetics equations", Int. J. Chem. Kin. 11, pp 1097-1108, 1979
- 10 B.A.Gottwald, "KISS - a digital simulation system for coupled chemical reactions", Simulation 37, pp 169-173, nov 1981
- 11 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, P.Deuflhard and W.Jäger eds), Springer, Berlin Heidelberg, 1981
- 12 A.R.Curtis, "Solution of large, stiff initial value problems - The state of art" in "Numerical Software - Needs and Availability" (D.A.H.Jacobs ed) Academic Press, 1978
- 13 G.Söderlind, "On the efficient solution of nonlinear equations in numerical methods for stiff differential systems", TRITA-NA-8114, Dep. of Num. Anal. and Comp.Sci., The Royal Institute of Technology, Stockholm, Sweden, 1981

- 15-
- 14 W.H.Enright and T.E.Hull, "Comparing Numerical Methods for the Solution of Stiff Systems of ODE's Arising in Chemistry" in "Numerical Methods for Differential Systems" (L.Lapidus and W.E.Schiesser eds), Academic Press, 1976
 - 15 B.A.Gottwald and G.Wanner, "Comparison of numerical methods for stiff differential equations in biology and chemistry", to appear in Simulation, 1982
 - 16 G.Dahlquist, L.Edsberg, G.Skölleremo and G.Söderlind, "Are the numerical methods and software satisfactory for chemical kinetics?", TRITA-NA-8005, Dep. of Num. Anal. and Comp. Sci., The Royal Institute of Technology, Stockholm, Sweden, 1980
 - 17 G.Dahlquist and Mao Zu-Fan, "Partitioning a stiff initial value problem by scaling technique", to appear in a TRITA-report, Dep. of Num. Anal. and Comp. Sci., The Royal Institute of Technology, Stockholm, Sweden

AN OVERVIEW OF STIFFNESS PROBLEMS IN NUCLEAR REACTOR KINETICS

J. DEVOOGHT (*)

1. INTRODUCTION

The integration of the reactor kinetics equations is a time-consuming operation whenever it is performed in the full space-energy-time frame. Moreover when coupling of the neutronic parameters with thermohydraulic parameters is present, the problem is even more taxing for the computer. A large number of approximate integration schemes have therefore been studied to reduce the numerical problem to a manageable size. However neutron kinetics, despite its large spread of eigenvalues is not typically a difficult stiff problem. Moreover special techniques have been devised to cope with the particular structure of the kinetics equations, owing to two facts :

- (1) Neutron kinetics is a linear evolution problem, which becomes non linear in so far as it becomes a neutron dynamics problem through coupling with other variables.
- (2) Stiffness can more or less be "factored out", i.e. if we write the neutron density

$$N(\vec{r}, \vec{v}, t) = T(t) \psi(\vec{r}, \vec{v}, t) \quad (1)$$

a factorisation used in "quasistatic approximation", a suitable function ψ can be obtained such that the stiff behaviour is more or less confined to $T(t)$.

(*) Université Libre de Bruxelles, Brussels, Belgium.

One way to cope with stiffness problems is to use the singular perturbation method, which has been known in reactor kinetics as the prompt jump approximation. Goldstein and Shotkin ⁽¹⁾⁽²⁾ tried to extend the prompt jump approximation to higher orders but their results based upon intuitive derivation are valid only for the first order. The systematic application of the singular perturbation method to reactor kinetics was first proposed by Hendry and Bell ⁽³⁾ who derived the algorithm including the intermediate expansion by a purely heuristic argument.

Recent developments have been given by Mika, Blenski and Gadomski ^(4 to 7). The method seems attractive for point kinetics and some non linear kinetics problems. So far limited numerical experience is available and we shall not develop this point here.

Various approximations have been devised : at one end of the spectrum we have "point kinetics" (a system of seven linear differential equations) where the reactor is acting as a whole, i.e. a single point ., at the other end of the spectrum we have a full-fledged 3D space and energy problem which could involve 10^5 unknowns. Intermediate options involve coarse mesh methods or flux-synthesis methods which we group under the heading of "multipoint reactor kinetics".

Advanced work in numerical reactor dynamics is mostly related to safety. The need for fast and reliable computing schemes for neutron transients stems from the fact that neutron fluxes yield the thermal power that drives a set of thermohydraulic equations. By far, the latter are the more difficult and more time consuming, and neutron kinetics is not anymore the main problem. However two points remain pressing : how to lengthen the time step; how to bound the error. Generally speaking, development has been outside the main stream of ODE stiff solvers, maybe for lack of familiarity. Some progress has been made for the first point but adequate schemes are sorely lacking for the second point. Benchmark problems allow a common point of comparison, but so far the lack of analytical solution for complicated problems does not allow to distinguish the convergence from a possible bias common to various methods.

It is sometime implied that the need for very accurate solution of reactor kinetics problems is not obvious because of the very large uncertainties in the thermohydraulics. However time step lengthening is of utmost importance, and is limited by the fact that kinetic operators are not constant.

The lack of memory space for large problems makes multistep methods rather unsuitable, and therefore one step methods are almost the rule.

As it may be surmised almost all methods make implicit or explicit use of collocation, Galerkin or interpolatory quadrature, and therefore of an assumed time dependent behaviour of the solution, usually linear combinations of exponentials and polynomials. Polynomial however are not very suitable for fast transients of exponential type and some methods place increasing reliance on spectral matching, i.e. the one-step growth function $W(hA)$ which approximates e^{hA} is chosen with the property that

$$W(h\lambda_1) = e^{h\lambda_1}$$

where the λ_1 's are some eigenvalues (or approximations thereof) of A .

2. THE NUCLEAR REACTOR KINETICS PROBLEM

2.1. Multigroup-multipoint kinetics equations

Boltzmann equation reads

$$\frac{\partial \vec{\zeta}}{\partial t} = A(t) \vec{\zeta}(t) + \vec{S}(t) \quad (2)$$

$$\text{where } \vec{\zeta}(t) \triangleq \left[\frac{\phi(\vec{r}, \vec{v}, t)}{v}, F_1(\vec{r}, v, t) \dots F_I(\vec{r}, v, t) \right]^T \quad (3)$$

with the following notation :

(a) $\vec{r}, \vec{v} = v\vec{\Omega}$, $|\vec{\Omega}| = 1$, t resp. position, velocity vector, and time

(b) $\phi(\vec{r}, \vec{v}, t)$: neutron flux density

(c) $F_i(\vec{r}, v, t) \triangleq \chi_i(v) \frac{v}{4\pi} C_i(\vec{r}, t)$

where $C_i(\vec{r}, t)$ is the precursor's density, i.e. the isotope that yields delayed neutrons

$\chi_i(v)$ the fission spectrum normalized by $\int_0^\infty \chi_i(v) dv = 1$

v the number of fission neutrons by fission

(d) $\vec{S} \triangleq [Q(\vec{r}, \vec{v}, t), 0 \dots 0]^T$

with Q the prompt neutron independent source, usually zero.

Matrix A is defined by

$$A \triangleq \begin{vmatrix} (1-\beta) v J_0 - vK & & \lambda_1 & \dots & \lambda_I \\ & \beta_1 v J_1 & -\lambda_1 & & 0 \\ & & \vdots & \ddots & \\ & & & & 0 & -\lambda_I \end{vmatrix} \quad (4)$$

with J_i the fission operators, and K the destruction operator, respectively defined by

$$J_i = \frac{v}{4\pi} \chi_i(v) \int_0^\infty dv' \int_{4\pi} \Sigma_f(\bar{r}, v') \dots dv' d\bar{\Omega}, \quad i=0,1,\dots,I \quad (5)$$

$$K = \bar{\Omega} \bar{V} + \Sigma_t(\bar{r}, v) - \int_0^\infty dv' \int_{4\pi} \Sigma_s(\bar{r}, \bar{v}' \rightarrow \bar{v}) \dots d\bar{\Omega}' \quad (6)$$

where $\Sigma_f(\bar{r}, v)$, $\Sigma_t(\bar{r}, v)$, $\Sigma_s(\bar{r}, \bar{v}' \rightarrow \bar{v})$ are respectively fission cross-sections, total cross-sections and differential scattering cross-sections.

The fraction of neutrons emitted instantaneously is $1-\beta$, and the fraction of delayed neutrons is β emitted from I (usually $I=6$) precursors. A weighted fission operator is defined by

$$J \triangleq (1-\beta) J_0 + \sum_{i=1}^I \beta_i J_i \quad (7)$$

where subscript 0 is reserved for prompt neutrons.

The reader may find a good survey of fundamentals of reactor kinetics as well as some numerical aspects in references⁽⁸⁻¹²⁾. This overview of stiffness problems is by no means a review of numerical reactor kinetics and references have been therefore limited to the few topics examined below.

Usually kinetics problems are solved in the diffusion approximation frame^{(8) (10)}. Full transport kinetics problems present problems of their own which are not examined here⁽¹³⁾. The energy variable is discretized into G "groups" and the space variable into K "points". Operators J and K become KG x KG matrices and λ_1 becomes λ_1 times a unit matrix. The system of multigroup-multipoint neutron kinetics has the same structure as (2) (4) where $\vec{c}$ is a vector with KG(I+1) components.

A Galerkin type approximation yields

$$\frac{d}{dt} \begin{vmatrix} \vec{T} \\ \vdots \\ \vec{c}_1(t) \\ \vdots \end{vmatrix} = \begin{vmatrix} [\Lambda]^{-1} [\rho - \beta] & \dots & \lambda_1 [1] & \dots \\ \vdots & \ddots & & \\ \vdots & & -\lambda_I [1] & \\ [\Lambda]^{-1} [\beta_1] & & & \ddots \\ \vdots & & & & \ddots \\ \vdots & & & & & 0 \end{vmatrix} \begin{vmatrix} \vec{T}(t) \\ \vdots \\ \vec{c}_1(t) \\ \vdots \end{vmatrix} \quad (8)$$

where $[\Lambda]$, $[\rho]$, $[\beta]$ are KG x KG matrices defined by

$$[\rho]_{kl} \triangleq \langle W_k, (J_0 - K) \psi_l \rangle \quad (9)$$

$$[\beta_i]_{kl} \triangleq \langle W_k, \beta_i J_i \psi_l \rangle \quad (10)$$

$$[\Lambda]_{kl} \triangleq \langle W_k, v^{-1} \psi_l \rangle \quad (11)$$

with $\langle, \rangle$ a scalar product in $\bar{r} \otimes \bar{v}$ space and $\{W_k\}_1^{KG}$, $\{\psi_l\}_1^{KG}$ two sets of linearly independent KG functions such that

$$\langle W_k, \psi_l \rangle = \alpha_k \delta_l^k \quad (12)$$

and

$$\frac{da_k}{dt} = 0 \quad (13)$$

Vectors $\vec{T}(t)$ and $\vec{C}_1(t)$ are defined by their components

$$T_i(t) \triangleq \langle W_i, \phi \rangle \quad (14)$$

$$C_{i,j}(t) \triangleq \langle W_i, F_j \rangle \quad (15)$$

When $KG=1$, we recover the usual "point kinetics" equation.

Although all matrices $[\rho]$, $[\beta_i]$ and $[\Lambda]$ are in principle time dependent, in practice they are kept constant, except the reactivity matrix $[\rho]$. Nonlinearities obtained through coupling with thermohydraulic variables usually come through $[\rho]$. A large number of approximation schemes have been developed through various choices of W_k and ψ_ℓ which are scattered through the literature.

2.2. Spectrum of the multigroup-multipoint diffusion operator with delayed neutrons

The spectrum of the multigroup-multipoint diffusion operator is an essential ingredient of any kinetics problem, the simplest offspring being the so-called "inhour equation"⁽¹⁴⁾. Many integration methods of the space-dependant kinetics equations make implicit or explicit use of spectral matching. However, very few rigorous results are available. The reactor theory lore contains statements such as : "in a K-point, G groups, I delayed neutron precursors model, the eigenvalues occur in G+I clusters of K eigenvalues; I clusters are close to $-\lambda_1$; G have much large absolute values; the eigenvalues are either real or close to the real axis".

Mika ⁽¹⁵⁾ and Porsching ⁽¹⁶⁾ have attacked the problem rigorously for mono-energetic plane transport and multigroup diffusion, respectively. Unfortunately, Porsching's assumptions are unrealistic for $G > 1$.

If we make the assumption of an identical fission spectrum χ for all precursors ⁽¹⁷⁾, the eigenvalue equation is

$$(\omega I + vK)\phi = f(\omega)v vF\phi \quad (16)$$

where v is the neutron velocity, K and $F \equiv J_0 = J_i$ the destruction and fission operators, respectively

$$f(\omega) \triangleq 1 - \sum_{i=1}^I \frac{\beta_i \omega}{\lambda_i + \omega} \quad (17)$$

We define the eigenvalues $v / [v_i(\omega)]$ of $(\omega I + vK)^{-1} vF$

$$\frac{v}{v_i(\omega)} \phi_i(\omega) = (\omega I + vK)^{-1} vF\phi_i(\omega) \quad (18)$$

The values of $[v_i(\omega)]/v$ for $\omega=0$ correspond to the lambda modes ⁽¹⁸⁾, the values of ω for $[v_i(\omega)]/v = 1$ correspond to the omega modes ⁽¹⁸⁾. Let ω_k be the eigenvalues of matrix vK . The fission operator is proportionnal to a projector P :

$$vF = \alpha P, \quad \alpha = \langle v\Sigma_f | v\chi \rangle, \quad P = \frac{|v\chi\rangle \langle v\Sigma_f|}{\langle v\Sigma_f | v\chi \rangle} \quad (19)$$

The characteristic equation reads

$$\det \left| \frac{v}{v_1(\omega)} \delta_k^1 - \sum_j (\omega + vK)_{ij}^{-1} [v_j x_j] [v \Sigma_{jk}] \right|$$

$$\equiv \det \left| \frac{v}{v_1(\omega)} \delta_k^1 - [a_i] [b_k] \right| \quad (20)$$

Each matrix $[a_i]$ or $[b_k]$ is $K \times K$ and $[a_i] [b_k]$ is the (i,k) element of a block-partitioned matrix in G^2 blocks.

Each principal minor of rank $p > K$ of $[[a_i] [b_k]]$ is zero; therefore eq. (20), the characteristic equation of $[[a_i] [b_k]]$ reads

$$\lambda^{KG-K} \prod_{i=1}^K \left[\lambda - \frac{v}{v_1(\omega)} \right] = 0 \quad (21)$$

Eq. (18) has K eigenvalues $[v_1(\omega)]/v$, which are algebraic functions of ω . If $K=1$, the unique eigenvalue $[v_1(\omega)]/v$ is a rational function of ω .

The eigenvalue problem (eq. (16)) is the perturbation of a slowing-down problem by a degenerate operator vF . We can apply the Weinstein-Aronszajn theory⁽¹⁹⁾: let

$$W(\omega) \triangleq \det [I - v v F (\omega I + vK)^{-1}] \quad (22)$$

Then

$$W(\omega) = \prod_{k=1}^{KG} \frac{\omega - \theta_k}{\omega - \omega_k} \quad (23)$$

where ω_k and θ_k are the eigenvalues of vK and $vK - v v F$, respectively. It can be shown that

$$W(\omega) = \prod_{i=1}^K \left[1 - \frac{\nu}{\nu_i(\omega)} \right] \quad (24)$$

and

$$\nu_i(\omega_k) = 0, \quad k=1, \dots, KG \quad (25)$$

If $K=1$ and νF , νK are symmetric matrices (unrealistic assumption for $G > 1$), then the zeros and poles of $W(\omega)$ are real and alternate, and the $\nu_i(\omega)$ are all real. This is the case examined by Porsching⁽¹⁶⁾. We now turn the delayed neutrons in.

The $KG(I+1)$ eigenvalues are divided into two sets :

(a) $K(G+I)$ eigenvalues of eq. (16) are solutions of

$$\frac{\nu_i(\omega)}{\nu} = f(\omega) \quad (26)$$

Some of these roots are real.

P 1 : (a) $[\nu_i(\omega)]/\nu$ is real, simple, bounded, smaller than $[\nu_i(\omega)]/\nu$,

$$i > 1$$

(b) $[\nu_i(\omega)]/\nu$ is monotone increasing for $\omega > \omega_0 > \text{Re} \omega_k$

(c) if $N \geq 1$ eigenvalues $[\nu_i(\omega)]/\nu$ are real, where $p \leq N$ numbers

$$[\nu_i(0)]/\nu < 1,$$

then p roots of eq. (16) are positive and $N-p$ are negative, all in the interval $(-\lambda_1, \infty)$. Moreover, there are N roots in each interval $(-\lambda_{i+1}, -\lambda_i)$ and N in $(\omega_0, -\lambda_1)$, where λ_i is the decay constant of the i 'th delayed-neutron precursor.

In practice we can always take $\omega_0 \ll -\lambda_I$. All statements about the non-real roots of eq. (26) are dependent on generalization of the Gershgorin theorem and will not be dealt with here.

(b) Each $-\lambda_i$, $i=1\dots I$ is degenerate of order $KG-K$.

However their eigenmodes can never be excited. Moreover when $G=1$, $-\lambda_i$ are not eigenvalues.

If we disregard the $-\lambda_i$ eigenvalues, we have $G+I$ clusters of K eigenvalues.

How stiff are reactor kinetics equations? The answer depends on the neutron lifetime Λ in the reactor.

A thermal reactor with $\Lambda = 5 \cdot 10^{-4}$ s with $\rho = +1$ % has its 7 eigenvalues (point kinetics) between -4.4 and 2. A fast reactor with $\Lambda = 10^{-7}$ s with $\rho = +0.5$ % has its 7 eigenvalues between $-2.2 \cdot 10^4$ and +1.5. A glance at figure (1) will show the spread of the eigenvalues for the fast reactor ⁽²⁰⁾. For multigroup-multipoint kinetics the spread can be much higher.

3. INTEGRAL FORMULATION OF REACTOR KINETICS WITH SOME CLASSICAL INTEGRATION SCHEMES

Let the solution of the kinetic system : $\frac{d\vec{y}}{dt} = A(t)\vec{y}(t) + \vec{S}(t)$ be cast into the integral form⁽²¹⁾ :

$$\begin{aligned} \vec{y}(t) = & T(t)\vec{y}(0) + \int_0^t T(t-\tau)\vec{S}(\tau)d\tau + \int_0^t \left[\alpha \frac{dT(t-\tau)}{d\tau} + T(t-\tau)M(\tau) \right] \vec{y}(\tau)d\tau \\ & + \int_0^t \left[(1-\alpha) \frac{dT(t-\tau)}{d\tau} + T(t-\tau)N(\tau) \right] \vec{y}(\tau)d\tau \end{aligned} \quad (27)$$

where $A(t)$ is split into $A(t)=M(t)+N(t)$, α being a number.

The matrix operator $T(t,\tau)=V^{-1}(t)V(\tau)$, $V(0)=I$ (28)

and where $V(t)$ is some matrix operator whose inverse is defined for $0 \leq t \leq h$.

A large number of approximate integration schemes for multipoint kinetics can be derived from the integral equation (27) according to the choice of $V(t)$ (or $T(t,\tau)$, α and $M(t)$ [or $N(t)$], as well as the choice of an interpolant for $\vec{y}(\tau)$ which may be different for each integral on the right hand side of (27) and therefore entails the splitting of $A(t)$.

For instance let $T(t) = e^{\omega t}I$, applied to (27) with $\alpha = 1$, $M(t)=A(t)$.

$$\begin{aligned} \phi(t) = & e^{\omega t} \phi(0) + \int_0^t e^{\omega(t-\tau)} \left[\left(\frac{\rho(\tau)-\beta}{\Lambda} - \omega \right) \phi(\tau) + \sum_1 \lambda_1 C_1(\tau) \right] d\tau \\ \{ & \\ C_1(t) = & e^{-\lambda_1 t} C_1(0) + \int_0^t e^{-\lambda_1(t-\tau)} \left[\frac{\beta_1}{\Lambda} \phi(\tau) \right] d\tau \end{aligned} \quad (29)$$

if we choose

$$\Gamma = \begin{vmatrix} \omega & 0 & 0 \\ 0 & -\lambda_1 & 0 \\ \vdots & \vdots & \vdots \\ 0 & 0 & -\lambda_I \end{vmatrix}$$

An equivalent formulation of (29) is

$$\begin{aligned} \phi(t) = e^{\omega t} \phi(0) + \int_0^t e^{\omega(t-\tau)} \left[\frac{\rho(\tau) - \beta}{\Lambda} - \omega \right] \phi(\tau) d\tau + \frac{1}{\Lambda} \sum_i \frac{\lambda_i \beta_i}{\omega + \lambda_i} \int_0^t \left[e^{\omega(t-\tau)} - e^{-\lambda_i(t-\tau)} \right] \\ \times \phi(\tau) d\tau \\ + \sum_i \frac{\lambda_i}{\lambda_i + \omega} C_i(0) (e^{\omega t} - e^{-\lambda_i t}) \end{aligned} \quad (30)$$

A few well known methods stem from either (29) or (30) if we apply Galerkin, collocation or interpolatory methods.

1°/ Adler's choice⁽²²⁾ is $\omega = -\frac{1}{\Lambda}$ and the interpolants are constants :

$$\begin{aligned} \phi(\tau) = \bar{\phi}, \quad C_i(\tau) = \bar{C}_i. \quad \text{An average reactivity } \bar{\rho} \text{ is defined by} \\ \bar{\rho} = \int_0^h e^{\omega(h-\tau)} \rho(\tau) d\tau / \int_0^h e^{\omega(h-\tau)} d\tau. \end{aligned}$$

2°/ The prompt jump approximation is obtained by Hetrick⁽²³⁾ if $\omega = -\frac{\beta}{\Lambda}$, for one precursor.

3°/ Cohen's choice⁽²⁴⁾ is $\omega = \frac{\rho(0) - \beta}{\Lambda}$ with an interpolant assuming $A(t)y(t)$ constant over $[0, h/2]$ and varying linearly over $[h/2, h]$.

4°/ Brittan's method⁽²⁵⁾ applied to (30) is $\omega = 0$ with a linear interpolant.

5°/ Kaganove's method⁽²⁶⁾ applied to (30) is $\omega = \frac{\rho(0) - \beta}{\Lambda}$, the reactivity being assumed quadratic in time. Another choice incorporates an additional exponential factor for the quadratic interpolant. In the same group, we have Volodka's method⁽²⁷⁾ which assumes parabolic behaviour for ρ/Λ , β/Λ and ϕ .

6°/ Hansen's method⁽²⁸⁾ is obtained if we choose ω as the largest eigenvalue of the inhour equation for a suitable average reactivity, and with interpolants : $\phi(t) = \phi(0)e^{\omega t}$ and $C_i(t) = C_i(0)e^{\omega t}$.

7°/ Kang and Hansen's method⁽²⁹⁾ uses both equations (29) and (30) with $\omega=0$ with Hermite interpolants. For further developments and extensions see § 4.3.

8°/ Taylor series methods are usually inefficient for stiff systems. Vigil⁽³³⁾ used a process of analytic continuation by successive evaluation of the derivatives. In the constant reactivity case for example

$$N(t) = e^{(\rho-\beta)/\Lambda t} N(0) + \int_0^t e^{(\rho-\beta)/\Lambda(t-\tau)} \sum \lambda_i C_i^{(k)}(0) \tau^k d\tau \quad (31)$$

where the $C_i^{(k)}(0)$ are evaluated recursively by means of

$$C_i^{(k+2)}(t) = a_i C_i^{(k+1)}(t) + b_i C_i^{(k)}(t) + d_i \sum_j \lambda_j C_j^{(k)}(t) \quad (32)$$

with $a_i = \Lambda^{-1}(\rho-\beta) - \lambda_i$, $b_i = \Lambda^{-1}(\rho-\beta)\lambda_i$, $d_i = \Lambda^{-1}\beta_i$. Equation (32) is obtained after differentiation of (8) ($KG=1$) and elimination of $T(t)$ and $\frac{dT}{dt}(t)$.

The drawback is usually the cumbersome derivative handling if the reactivity is not constant but at least piecewise analytic. However Vigil's method is one of the rare instances where a step control is offered. Secker⁽³⁴⁾ extended the method by improving the choice of the integrating factor to minimize the remainder or maximizing the integration step size.

A straightforward use of Galerkin's method is proposed by Fuller, Meneley and Hetrick⁽³⁵⁾ for a multipoint model with $\phi_{i,k}(t) = (t-t_i)^k$, $k=1\dots n$. The quadrature is done exactly, the reactivity being assumed quadratic in time. Interesting numerical comparisons between some of these older methods has been given by Szeligowski⁽³⁶⁾.

4. ONE-STEP APPROXIMATE INTEGRATION SCHEMES FOR REACTOR KINETICS

4.1. A-stable rational approximation of the exponential

There is no need to justify here the importance of A-stability for stiff problems fully treated elsewhere in this conference. Some published work^{(37-39) (21)}, on numerical reactor kinetics, specifically use A-stable (A-acceptable) rational approximation of the exponential.

The problem of generating A-acceptable functions has an old history.

Indeed the problem of constructing the totality A-acceptable functions $W(z)$ satisfying interpolatory conditions in $|z| < 1$ at arbitrarily chosen points $\{z_i\}_{i=1}^n$ was essentially solved for more than a half century by Schur and Nevanlinna⁽⁴⁰⁾. Let us define a sequence of functions

$$W_k(z) = \frac{M_k^2}{f_k(z)} \frac{W_{k-1}(z) - W_k^{(k-1)}}{M^2 - \bar{W}_k^{(k-1)} W_{k-1}(z)}, \quad k=1 \dots n \quad (33)$$

with

$$f_k(z) = (z - z_k)(1 - \bar{z}_k z)^{-1}$$

$$W_0(z) = W(z) \quad \} \quad (34)$$

$$W_k^{(v)} \equiv W_v(z_k), \quad k=v+1, v+2, \dots, n$$

The real positive constants M_k are arbitrary except for $M \equiv M_1 \leq M_2 \leq \dots M_n \leq 1$. $W_0(z)$ is the solution of the problem.

P 2 : The SN problem admits at least one solution if one of the two situations occurs :

$$1. |W_k^{(k-1)}| < M_k, k=1 \dots u; |W_{u+1}^{(u)}| = M_{u+1}; W_{u+1}^{(u)} = \dots W_n^{(u)}$$

$$2. |W_k^{(k-1)}| < M_k, k=1 \dots n.$$

In the first case, the solution is unique, with $|W(z)| \leq M$ for $|z| \leq 1$ (35), and $W(z_i) = W_i^{(0)}$, $i=1 \dots n$.

In the second case, the solution is not unique : any initial function $W_n(z)$ with $|W_n(z)| \leq M_n$ on $|z| = 1$ is suitable and yields a solution $W(z)$ satisfying (35). If $W_n(z)$ is rational, then $W(z)$ will be rational.

Property 2 is a slight extension of a property proven by SN⁽⁴⁰⁾.

P 3 : Given $\{z_i\}$ and $\{W_i^{(0)}\}$, $i=1, \dots, n$ $M=M_k$, there is a minimum value of M to which corresponds a unique solution of SN problem. The algebraic equation for the determination of the smallest M is $|W_n^{(n-1)}| = |W_{n-1}(z_n)| = M$.

We report below some unpublished work on this subject. See however ref. (21).

P 4 : A sufficient condition that there exists a function $W_0(z)$ of modulus $\leq M$ on $|z| = 1$ which takes on the values $W_k^{(0)}$ at $z=z_k$ is that $W_k^{(0)}$ are the values taken at z_k by a bounded analytic function $\sigma(z)$, with $|\sigma(z)| \leq N$ along $|z| = 1$ and $|\sigma_n^{(n-1)}| \leq M_n$.

Let $z = u-a/u+a$, a real $a < 0$ be the conformal mapping of the complex left half plane $\text{Re } u \leq 0$ on the closed unit disk $|z| \leq 1$. Our original problem, i.e. to find A-acceptable approximations $W_n(z)$ to $\sigma(z) = e^{-a(z+1/z-1)} = e^u$ at a finite number of arbitrarily given points $\{z_i\}_1^n$. $\sigma(z)$ is a bounded function in $|z| < 1$ but has an essential singularity at $z = 1$ and $\sigma(e^{i\theta}) = 1$ almost everywhere. This is no restriction to P 2. We define functions $W(z)$ with property M if

$$W(z)W(1/\bar{z}) = M^2 \quad (36)$$

Analytic function in $|z| \leq 1$ satisfy property M if they have a constant modulus on $|z| = 1$.

P 5 : Let $W(z)$ be a rational function of z with $|W(e^{i\theta})| \leq M$ which interpolates a n arbitrarily given points $\{z_i\}$, $i=1 \dots n$ inside the unit circle a function $\sigma(z)$ satisfying property N. Then $W(z) - \sigma(z)$ has also n zeros $\{z_k\}_1^n$ in $|z| > 1$ and there is a positive constant independent of M and N such that

$$\left| z_k - \frac{1}{\bar{z}_k} \right| < L |M-N| \quad (37)$$

Returning to the plane u , we see that if $M=N=1$, to each interpolation point u_k in $\text{Re } u \leq 0$ is associated another interpolation v_k with $u_k = -v_k$.

P 6 : If the rational function $W_0(z)$ satisfies the condition of P 4, it can be made to interpolate $\sigma(z)$ at an additional point z_{n+1} provided $W_n(z_{n+1}) = \sigma_n^{(n+1)}$ with $W_n(z_{n+1}) \leq M_n$.

If the equality applies and $M_1=M_2=\dots=M_n$, $W_n(z)$ is constant and $W_0(z)$ is the unique function of least maximum modulus in $\text{Re } u \leq 0$.

This additional point has no counterpart : we have a total of $2n+1$ interpolation points. In practice we have to interpolate e^u with $u=\lambda h$ at the points $u_1=\lambda h_1$, when h is the time step. When $h \rightarrow 0$, the interpolation points move towards the origin and therefore the minimum value of M given by P 3 is a function of h with $\lim_{h \rightarrow 0} M_{\min}(h)=1$. Let now $W(z)$ interpolate $\sigma(z)$ with $|W(z)| < M$ on $|z| = 1$ with $M_{\min}(h) \leq M$, at $2n+1$ points $z_k = (h\lambda_k - a)/(h\lambda_k + a)$.

P 7 : For h sufficiently small, $|W(z) - \sigma(z)| < c h^{2n+1} \prod_{i=1}^{2n+1} |\lambda - \lambda_i|$ with $u=\lambda h$. When $\lambda_i=0$, $W(z(u))$ is the diagonal Padé approximation to e^z .

The general first-order (W_{11} : first degree of numerator and denominator approximant of e^u) is

$$W_{11}(u|u_1, u_2, M^2) = M^2 \cdot \frac{u(e^{u_1} + W_1) + u_1(e^{u_1} - W_1)}{u(M^2 + W_1 e^{u_1}) + u_1(M^2 - W_1 e^{u_1})} \quad (38)$$

$$\text{with } W_1 = M^2 \frac{u_2 + u_1}{u_2 - u_1} \frac{e^{u_1} - e^{u_2}}{e^{u_1+u_2} - M^2} \quad (39)$$

Since $e^u = e^a e^{u-a}$, and since $-u_1$ is another interpolation point, it is easily checked that $\hat{W}_{11}(u|u_1, u_2, u_3) = e^a W_{11}(u-a|u_1-a, u_2-a; 1)$ (40) interpolates e^u at $u_1, u_2, u_3 = 2a - u_1$.

However $\hat{W}_{11}$ may not be A-acceptable for all values of a .

The second order W_{22} can be written readily by means of (33,34) but the reduction of well-known approximations for instance Liniger-Willoughby⁽⁴¹⁾ and Nørsett⁽⁴²⁾ to W_{22} is no easy task⁽²⁰⁾.

4.2. Implicit Runge-Kutta with spectral matching

In principle, spectral matching loses its appeal for variable operators. However numerical experiments show that spectral matching improves accuracy for reactor kinetics problems provided the reactivity is slowly varying. Implicit R-K methods yield growth functions that may be identified for instance to one A-acceptable function of § 4.4. Gear⁽⁴³⁾ has shown how to build a R-K procedure associated to a given growth function. We start from 2-stage procedure :

a_{11}	a_{12}	c_1
a_{21}	a_{22}	c_2
b_1	b_2	

Let $\bar{a} = |a_{ij}|$.

Numerical evaluation of the local truncation error as a function of time step for various (1,1)-type algorithms.

The two ordinates represent respectively :

$$1/p \Rightarrow \log \left| \frac{\phi_E - \phi_A}{\phi_E} \right| = 10^{-p}$$

$2/|\omega_i h|, i=1, \dots, 7$ with ω_i solution of the inhour equation

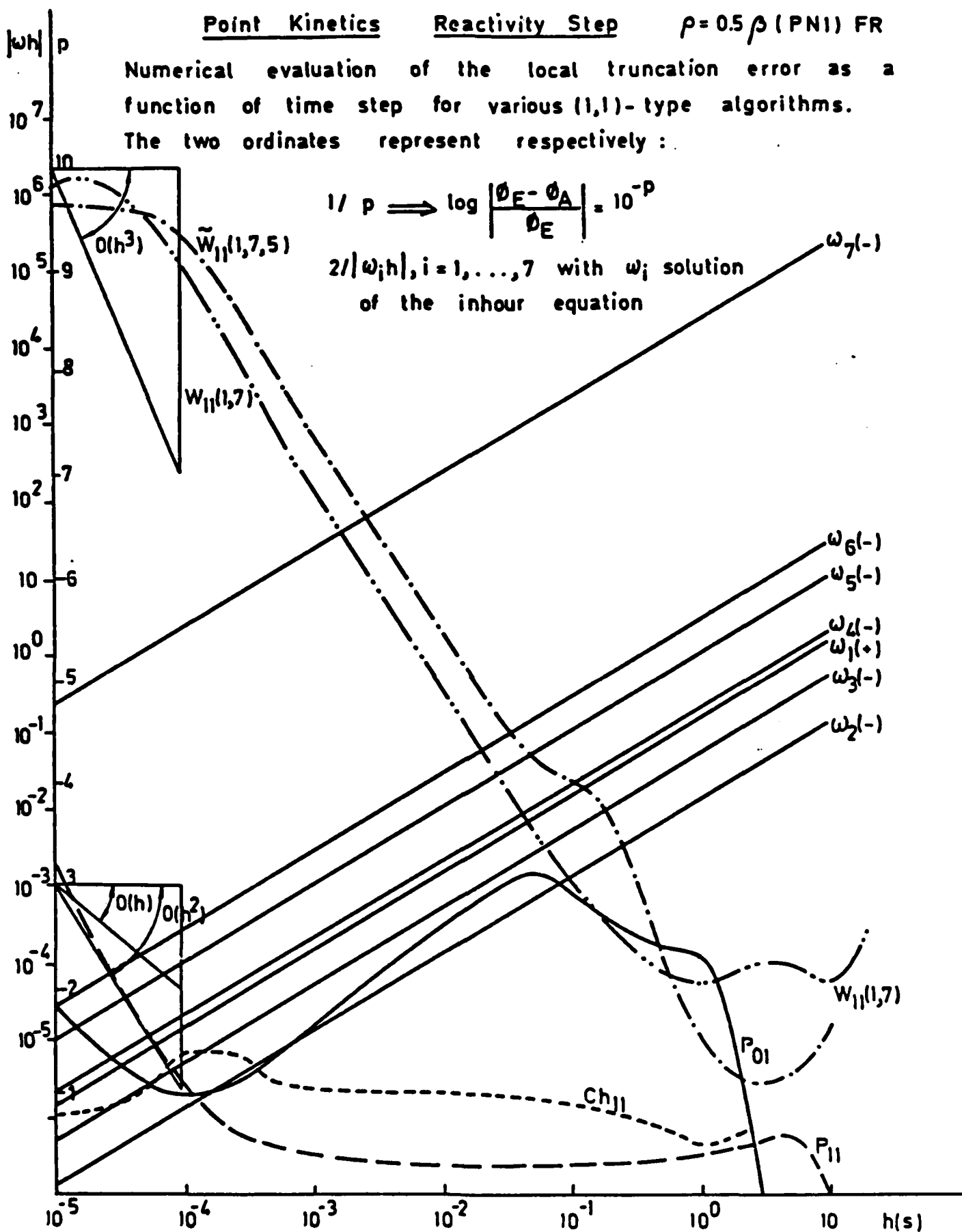


Fig. 1.

Let $\rho = \text{Tr } \tilde{a}$, $\Delta = \det \tilde{a}$. Then the growth function⁽⁴⁴⁾ is

$$W(u=h\lambda) = \frac{1 + u(b_1+b_2-\rho) + u^2(\Delta-\rho(b_1+b_2) + (b_1c_1+b_2c_2))}{1 - \rho u + u^2\Delta} \quad (41)$$

where $\tilde{a}$ is easily expressed in terms of c_1, c_2, Δ, ρ . Identifying $W(u)$ with one of the A-acceptable approximations of e^u of type (2,2), we have 4 equations for 6 unknowns and we are left with a two parameters family of R-K schemes which allows for interpolation at the origin and two arbitrary points λ_1, λ_2 in $\text{Re } \lambda < 0$.

The maximum order $p=4$ will be preserved if

$$\begin{aligned} \sum_i b_i c_i^k &= \frac{1}{k+1} + O(h^{4-k}), \quad k=0,1,2 \\ \sum_i \sum_j b_i a_{ij} c_j &= 1/6 + O(h^2) \end{aligned} \quad (42)$$

as well as an additional condition given in ref.⁽⁴³⁾.

If we choose for $W(u)$, the Nørsett function $T_2(a,b,u)$, a special case of $W_{2,2}$:

$$T_2(a,b,u) = \frac{1 + 1/2(1-a/2)u + 1/8(1+b-a)u^2}{1 - 1/2(1+a/2)u + 1/8(1+b+a)u^2} \quad (43)$$

with

$$a = \frac{4[\alpha(u_2) - \alpha(u_1)]}{u_1 \alpha(u_2) - u_2 \alpha(u_1)}; \quad 1+b = \frac{4(u_2 - u_1)}{u_2 \alpha(u_1) - u_1 \alpha(u_2)}$$

$$\alpha(u) = u^2(1-e^{-u}) [(2+u)e^{-u} + u - 2]^{-1} \quad (44)$$

then the choice of 4th order Butcher parameters :

$$b_1 = b_2 = 1/2 \quad ; \quad c_{1,2} = 1/2 \pm \frac{\sqrt{3}}{6}$$

with

$$\rho = 1/2 + a/4 \quad ; \quad \Delta = (1+a+b)/8$$

yields

$$a = \frac{h}{30} (\lambda_1 + \lambda_2) + O(h^3) \quad ; \quad b = -\frac{1}{3} + \frac{h^2}{30} \lambda_1 \lambda_2 + O(h^3) \quad (45)$$

$$\sum_{ij} b_i a_{ij} c_j = \frac{\rho}{2} - \Delta = \frac{1-b}{8} = 1/6 - \frac{h^2}{240} \lambda_1 \lambda_2 + O(h^3)$$

The last condition (42) is also fulfilled because a_{ij} differ from their optimal values by $O(h)$ terms.

Therefore even for variable matrix $A(t)$, at the cost of two evaluations $A(c_1 h)$ and $A(c_2 h)$ at fixed times, we retain $O(h^5)$ accuracy with spectral matching at $0, \lambda_1, \lambda_2$.

The solution of an m -stage implicit R-K step requires the solution of a linear system of $mKG(I+1)$ unknowns. Precursor equations can be solved one by one and we have in fact mKG unknowns. This may be very expensive even for $m=2$. If the matrix is triangular we have only m systems of KG unknowns : this is in short the principle of factorizable or Rosenbrock type methods. We have a growth function with a denominator factorizable into real factors $\prod_{i=1}^m (1 - a_{ii} h \lambda)$. The remaining degree of freedom can be used for instance to impose L -stability or to minimize the error at infinity.

Table I gives the R-K parameters of various schemes associated with the three interpolation points $0, \lambda_1, \lambda_2$.

TABLE 1 : IMPLICIT R-K METHODS WITH SPECTRAL MATCHING OR FACTORIZABLE

Criterion or interpolation points	ρ	Δ	c_1	c_2	b_1	b_2	Local error
1. $(0, \lambda_1, \lambda_2)$ $q_1 = \lambda_1 h$ $q_2 = \lambda_2 h$	$\frac{1}{2} + \frac{a}{4}$ with $\frac{a}{4} = \frac{\alpha(q_2) - (q_1)}{-q_1 \alpha(q_1) + q_1 \alpha(q_2)}$	$\frac{a}{8} + \frac{1}{2} \frac{(q_2 - q_1)}{[q_2 \alpha(q_1) - q_1 \alpha(q_2)]}$	$\frac{1}{2} - \frac{\sqrt{3}}{6}$	$\frac{1}{2} + \frac{\sqrt{3}}{6}$	$\frac{1}{2}$	$\frac{1}{2}$	Gauss h^5
2. $(0, 0, 0)$	$1/2$	$1/12$	$\frac{1}{2} - \frac{\sqrt{3}}{6}$	$\frac{1}{2} + \frac{\sqrt{3}}{6}$	$\frac{1}{2}$	$\frac{1}{2}$	Gauss (P ₂₂) h^5
3. $(0, \infty, \lambda_1)$	$\frac{2}{3} \frac{q_1}{(1 - \frac{1}{2})} e^{-1} - (1 + q_1)$	$\rho - \frac{1}{2}$	0	$\frac{2}{3}$	$\frac{1}{4}$	$\frac{3}{4}$	Radau I.A h^4
4. $(0, \infty, \lambda_1)$	$\frac{q_1}{q_1(-q_1 e^{q_1} + e^{q_1} - 1)}$		$\frac{1}{3}$	1	$\frac{3}{4}$	$\frac{1}{4}$	Radau II.A h^4
5. $(0, \infty, 0)$	$\frac{2}{3}$	$\frac{1}{6}$	0	$\frac{2}{3}$	$\frac{1}{4}$	$\frac{3}{4}$	Radau IA(P ₂₁) h^4
6. $(0, \infty, 0)$	$\frac{2}{3}$	$\frac{1}{6}$	$\frac{1}{3}$	1	$\frac{3}{4}$	$\frac{1}{4}$	Radau IIA(P ₂₁) h^4
7. $(0, \infty, \infty)$	1	$\frac{1}{2}$	0	1	$\frac{1}{2}$	$\frac{1}{2}$	Lobatto IIIC(P ₂₀) h^3
8. Factorizable and minimization of error at infinity	$1 + \frac{1}{\sqrt{3}}$	$\frac{1}{3} + \frac{1}{2\sqrt{3}}$	$\frac{1}{2}(1 + \frac{1}{\sqrt{3}})$	$\frac{1}{2}(1 - \frac{1}{\sqrt{3}})$	$\frac{1}{2}$	$\frac{1}{2}$	Makinson h^4
9. Factorizable	$1 + \frac{1}{\sqrt{3}}$	$\frac{1}{3} + \frac{1}{2\sqrt{3}}$	$\frac{1}{2}(1 + \frac{1}{\sqrt{3}})$	$\frac{1}{2}$	0	1	Axelsson h^4
10. Factorizable and L-stable	(a) $1 + \frac{1}{\sqrt{3}}$ (b) $2 - \sqrt{2}$	$\frac{1}{3} + \frac{1}{2\sqrt{3}}$ $\frac{3}{2} - \sqrt{2}$	$\frac{1}{2}(1 + \frac{1}{\sqrt{3}})$ $1 - \frac{1}{\sqrt{2}}$	$\frac{1}{2}(1 - \frac{1}{\sqrt{3}})$ $\frac{1}{2}$	$\frac{3}{4}$ 0	$\frac{1}{4}$ 1	Calahan Rosenbrock h^3 h^3

4.3. Collocation methods

Starting from the integral equation with $\vec{S}=0$, we can obtain various integration schemes if we substitute in the right hand side an Hermite interpolation formula. Taking $t=h$, $y(\tau)$ is developed in terms of basis functions which are polynomials and/or exponentials. The only information needed is $y^{(k)}(0)$ and $y^{(l)}(h)$, with $k=0,1,\dots,p-1$; $l=0,1,\dots,q-1$ obtained from the successive derivations of $A(t)$ at $t=0$ and h . For instance for linearly varying operators (ramp reactivities)

$$A(t) = (1 - \frac{t}{h})A(0) + \frac{t}{h} A(h) \quad (46)$$

the following advancement matrices B^{pq} (defined by $y(h) = B^{pq} y(0)$) are obtained by Hennart⁽³¹⁾

$$P_{11M} : [I - \frac{h}{2} A(\frac{2}{3}h)]^{-1} [I + \frac{1}{2} h A(\frac{h}{3})] \quad (47)$$

$$P_{12M} : [I - \frac{2}{3} h A(\frac{5h}{8}) + \frac{h^2}{6} A(\frac{h}{2})A(h)]^{-1} [I + \frac{h}{3} A(\frac{h}{4})]$$

More general collocation methods allowing spectral matching can be found in ref. (30-32) for instance.

4.4. Extrapolation and smoothing

The power of extrapolation and smoothing techniques for stiff problems has been exemplified for instance by Lindberg⁽⁵⁹⁾⁽⁶⁰⁾. It has been used in reactor kinetics by Izumi and Noda⁽⁴⁵⁾, Devooght and Mund⁽²¹⁾, Lawrence and Dorning⁽⁴⁶⁾, Hofer and Werner⁽⁴⁷⁾.

Smoothing vector $\vec{y}$ gives

$$\vec{\hat{y}}(t_n, h) = \frac{1}{4} [\vec{y}(t_{n-1}, h) + 2\vec{y}(t_n, h) + \vec{y}(t_{n+1}, h)] \quad (48)$$

where $\vec{y}(t_n, h)$ is the numerical solution at time t_n computed with time step h .
Then

$$\vec{\hat{y}}(t_n, h) = \vec{y}(t_n, h) + d_1(t_n, h)h^2 + d_2(t_n, h)h^4 + \vec{w}_n(h) \quad (49)$$

where $\vec{w}_n(h)$ is a smoothed error term given in ref. (48). Finally

$$\vec{\tilde{y}}(t_n, h) = \frac{4}{3} \vec{\hat{y}}(t_n, \frac{h}{2}) - \frac{1}{3} \vec{\hat{y}}(t_n, h) = \vec{y}(t_n) + O(h^4) + \vec{\tilde{w}}_n(h) \quad (50)$$

Although there is no guaranty that $\vec{\tilde{w}}_n(h)$ will be small enough, in practice it proved so.

Let us examine for instance the results (46) of a one dimensional two-group benchmark kinetics problem. The reactor has 3 regions, 120 spatial mesh points and a subcritical transient is initiated by an increase the thermal absorption cross-section linearly in one of the outer regions. Results of code WIGLE-40⁽⁶¹⁾ have been smoothed and extrapolated. As shown in figure (2) smoothing yields $O(h^2)$ error, and extrapolation alone failed to give $O(h^4)$. However extrapolation and smoothing combined yield $O(h^4)$ accuracy. The drawback of smoothing for very large space-dependent problems is the need for increased memory storage. Some additional discussion of extrapolation is given in § 5. Other methods are available. For instance the extrapolation method of Bulirsch-Stoer⁽⁴⁹⁾ and its associated time step control has been implemented in the fast reactor kinetics code CASSANDRE⁽⁵⁰⁾.

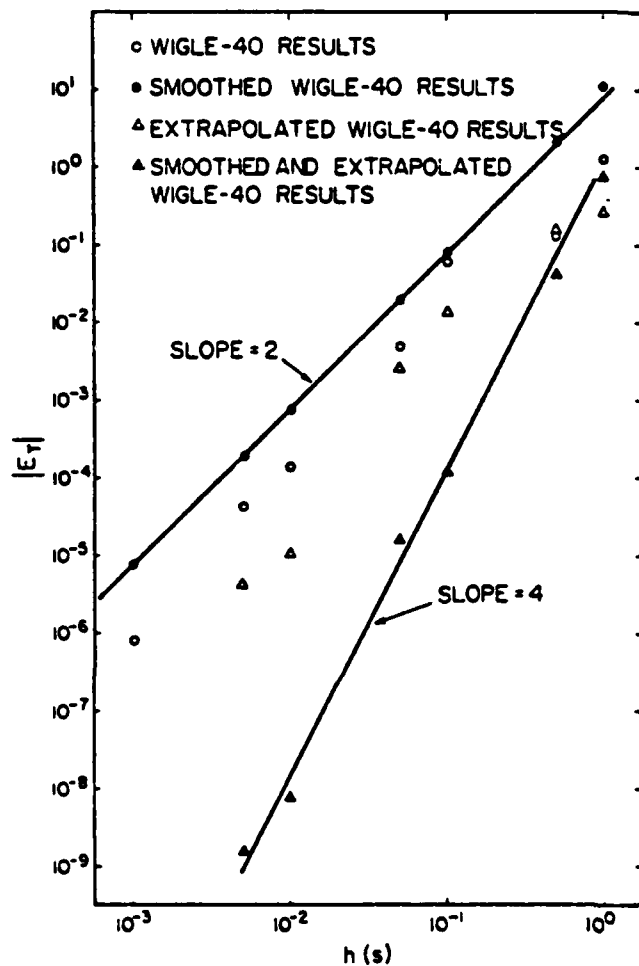


Fig. 2.a. Percent errors for ANL Benchmark ID.6-A1 at $t = 1.0$ s.

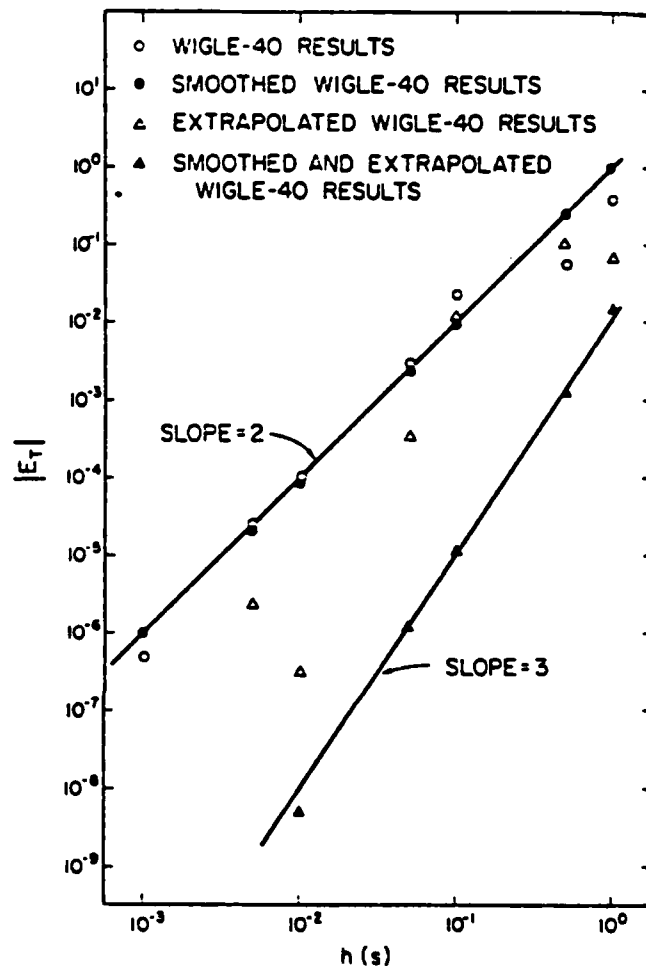


Fig. 2.b. Percent errors for ANL Benchmark ID.6-A1 at $t = 2.0$ s.

5. NUMERICAL TESTS

We shall give in this paragraph a short overview of the effectiveness of various integration schemes devised for reactor kinetics. As we shall find in § 6 it is usually sufficient to examine the point kinetics equation for comparison purposes. The variable parameter is the reactivity $\rho(t)$ and three standard problems are examined : step, linear ramp and periodic reactivities. Because stiffness is noticeably increasing as the neutron life time Λ is decreasing, both thermal and fast reactors must be treated separately. Accuracy is reported in the following way : the "exact" flux $\phi_E(t)$ is obtained with the best integration method applied on the finest mesh compatible with reasonable round-off errors in single precision on a CDC 6600 computer; the approximate flux $\phi_A(t,h)$, the number of exact digits is obtained from $|\phi_E(t) - \phi_A(t,h)| / \phi_E(t) = 10^{-n}$ and reported either as a function of time step h or as a function of arithmetic operations. Spectral matching, when needed, results from interpolation on one, two or three roots of the point kinetics ($KG=1$) characteristic equation (so called "inhour equation") obtained from (16), K and F being scalar :

$$\rho = \omega \Lambda + \omega \sum_i \frac{\beta_i}{\lambda_i + \omega} \quad (51)$$

Figure (1) reports first typical eigenvalues $\omega_i h$ with time step $10^{-5}(s) \leq h \leq 10^{-1}(s)$ for a fast reactor ($\Lambda=10^{-7}(s)$) for a step reactivity $\rho = 0.5 \beta$. The local truncation error is given as a function of h for Padé approximants of the exponential P_{01} , P_{11} , for Chebyshev approximant Ch_{11} , for $W_{11}(1,7)$ and $\hat{W}_{11}(1,7,5)$ defined by eq. (38)(40) and which interpolate respectively at ω_1, ω_7 and $\omega_1, \omega_5, \omega_7$. The effectiveness of spectral matching is apparent; it is essential however to interpolate at the largest eigenvalue $|\omega_7|$, which amounts to impose L-stability.

Four types of approximations are examined : implicit R-K methods, factorizable R-K methods, collocation method P_{pqM} and extrapolation method (without smoothing).

Figure (3) compares best methods for a linear ramp in fast reactor, for the same number of arithmetic operations. The superiority of P_{12M} stems from the fact that for stiff problems L-stable are preferable and that P_{pqM} approximants are devised for linear ramps.

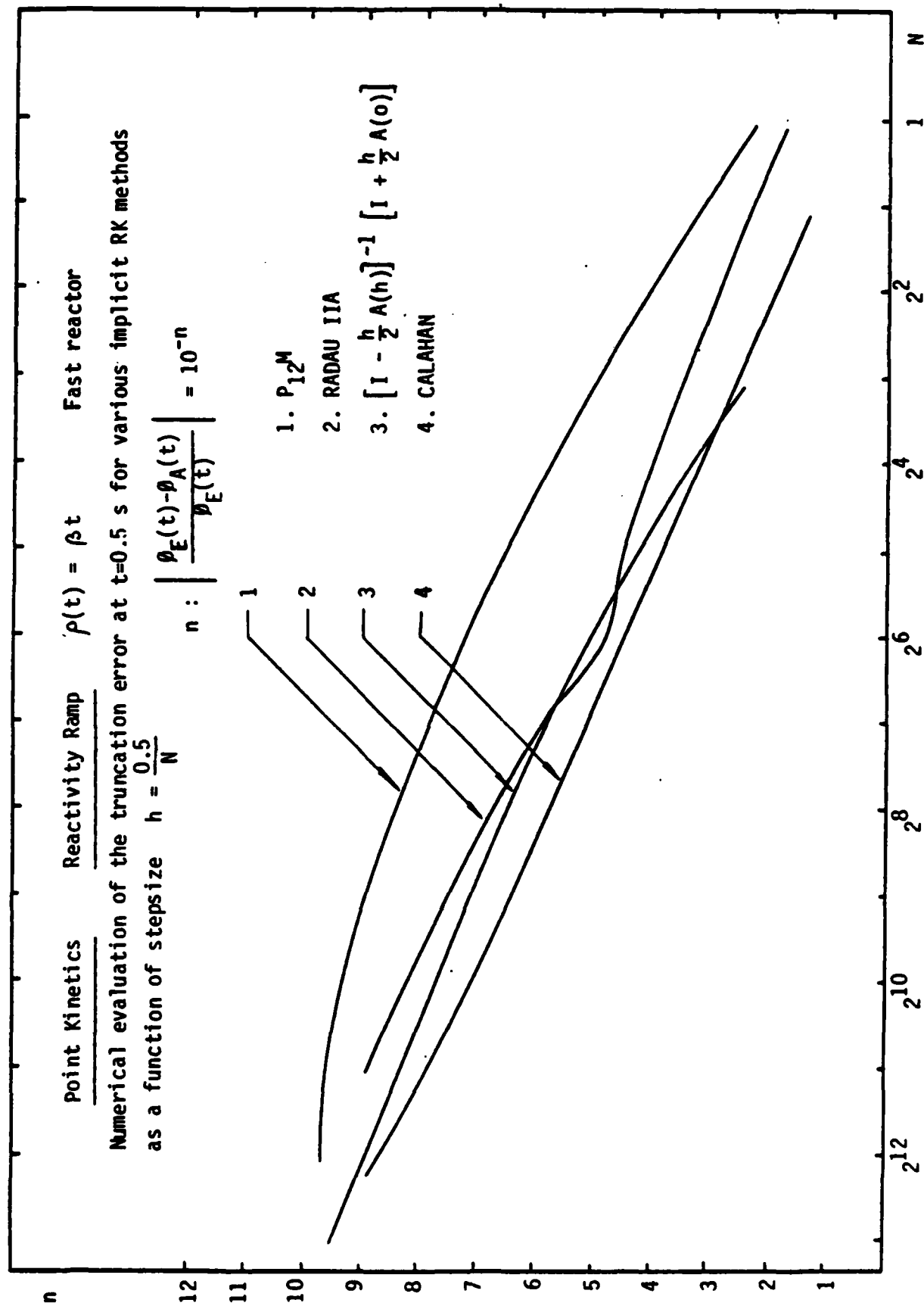


Figure 3.

Figure (4) compares implicit R-K schemes for a reactivity ramp in a thermal reactor. Stiffness is less a problem : L-stability can be exchanged at profit for the higher accuracy of Gauss R-K scheme.

Figure (5) compare best methods in each category for periodic reactivity in a fast reactor : the L-stable factorizable Radau I.A scheme fares better. Gauss scheme accuracy attains its asymptotic $O(h^4)$ only for $h < 5 \cdot 10^{-3}$.

We may draw for experience obtained⁽⁵⁰⁾ (E. Mund, personal communication) of which figures 3,4, 5 are a small sample some general conclusions :

- (1) the theoretical order $O(h^p)$ accuracy is not a good measure of accuracy : the value of p may be obtained in some cases for very small values of h . Sometimes the error has almost a step behaviour as h decreases and assessing accuracy just by halving the time step can be very much misleading.
- (2) L-stability is an important requirement for fast-reactor kinetics.
- (3) The choice of the interpolation points, even the dominant one is not very critical. Moreover it is good enough to choose the dominant root of the point kinetics equation even for a multipoint or fully space dependent problem.
- (4) For multipoint-multigroup problems at least, factorizability is an important requirement.

If the algorithm should be robust with respect to changes in reactivity, Makinson scheme and Radau fare best.

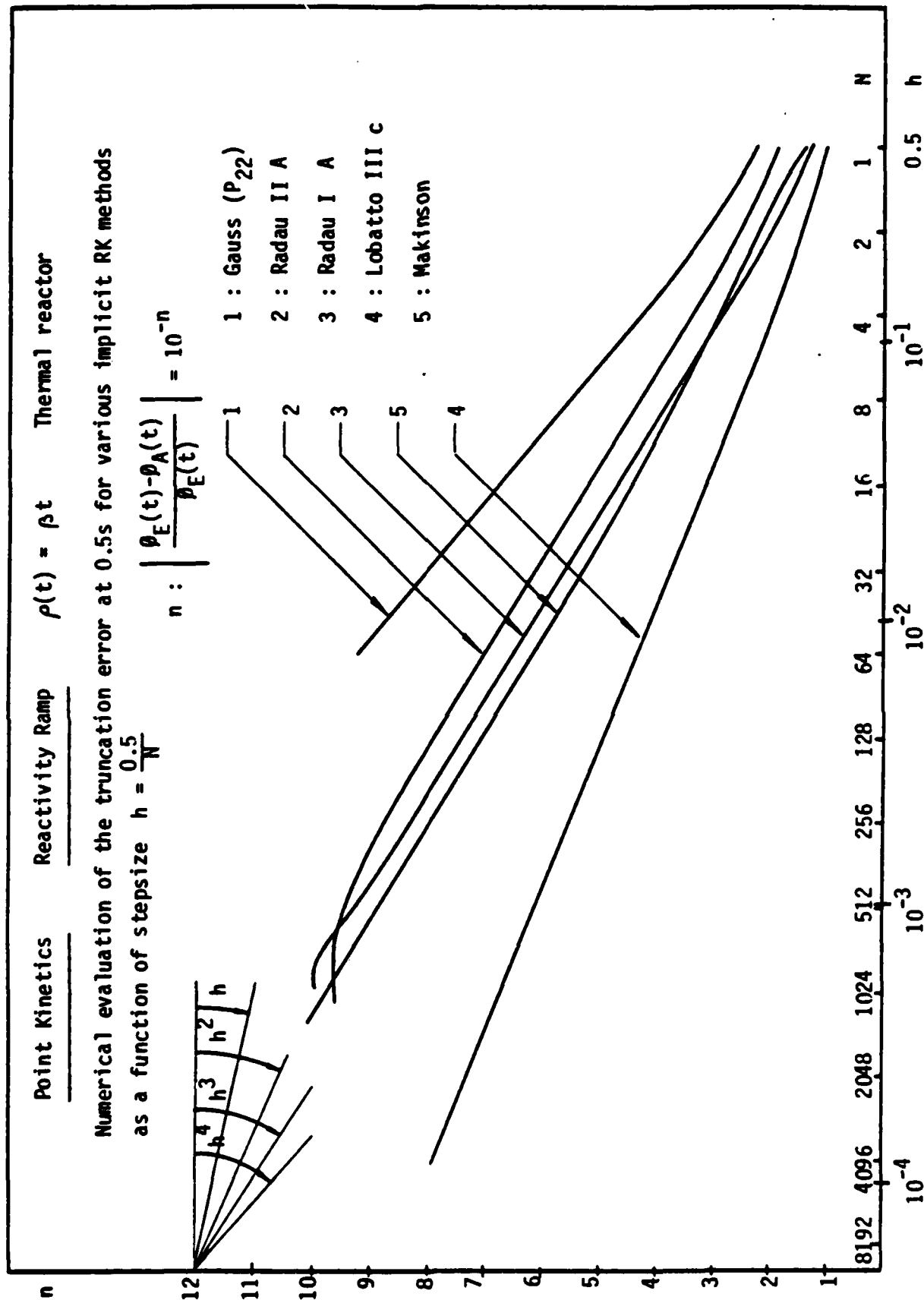


Figure 4.

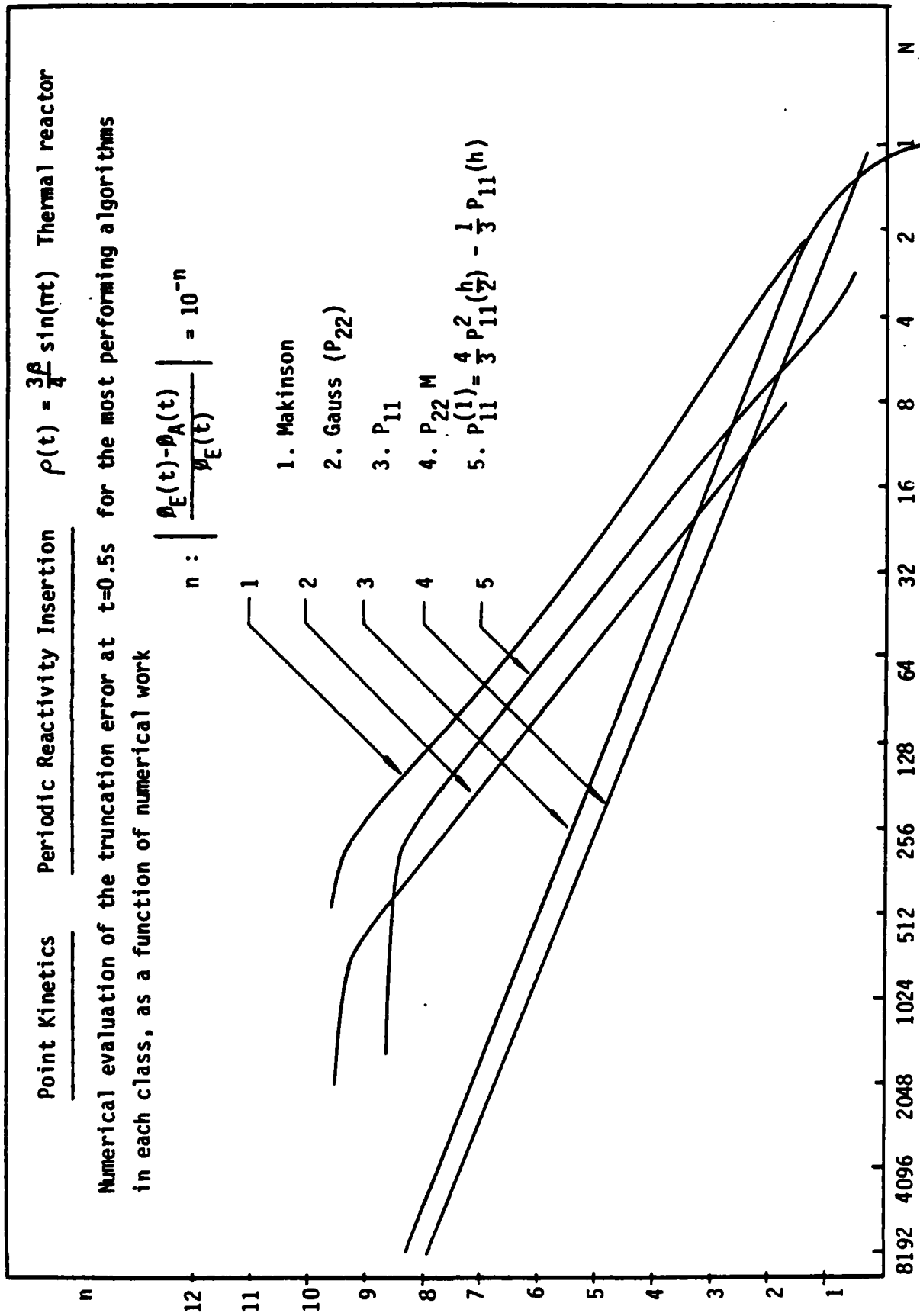


Figure 5.

6. SPACE DEPENDENT KINETICS

We shall not deal here with this important subject but uniquely with some particular aspects where stiffness is involved. Basically, in the diffusion approximation, the techniques are those developed for the integration of parabolic partial differential equations. Let us recall that ADI was first developed for nuclear reactor statics and dynamics. Most problems arise with the direct or iterative solution of the spatial equations, owing to the fact that most methods use Crank-Nicholson algorithms. The reader may find a recent review in ref. (51). Obviously nothing prevents to use the methods devised for multipoint kinetics. However, core storage limitations would automatically restrict useful algorithms to Rosenbrock type with operators evaluated at the same time. Many existing methods can be cast in the form of eq. (27) with very simple interpolants for $\vec{y}(\tau)$. The choice of $T(t) = e^{\omega t}I$ has often been made but a bad choice of ω can easily worsen the results.

A very powerful method can however simplify the space dependent kinetics problem, i.e. the quasistatic method⁽⁵²⁾ and its generalizations⁽⁵³⁾. In its simpler form, the flux density is factored⁽⁵⁴⁾ according to eq. (1). Let the evolutive equation be rewritten as a scalar equation for the neutron density

$$\frac{\partial N}{\partial t} = BN + S \quad (52)$$

where B is an integrodifferential operator which involves the contribution from the delayed neutron precursors. Then

$$\frac{\partial \psi}{\partial t} = T^{-1}(t)B\psi T - \frac{d \ln T}{dt} \psi + T^{-1}S \quad (53)$$

Given initial conditions, we can choose T at will and find ψ from (53). But the reverse is not true. Let

$$\langle f, g \rangle \triangleq \int f g \, d\vec{r} \, d\vec{\Omega} \quad (54)$$

For an arbitrary function $\phi(\vec{r}, v\vec{n}, t)$

$$\frac{dT}{dt} \langle \phi, \psi \rangle + T(t) \langle \phi, \frac{\partial \psi}{\partial t} \rangle = \langle \phi, B\psi \rangle T(t) + \langle \phi, S \rangle \quad (55)$$

i.e. the "point kinetics equation" but in fact an identity which does not bring in any information that is not already in (53). Despite its formal appearance (53) (55) is not a system for unknowns $T(t)$ and $\psi(\vec{r}, v\vec{n}, t)$. We need an additional constraint to make the factorization unique, which is chosen as

$$\langle \phi, \psi \rangle = \alpha(t) \quad (56)$$

where $\alpha(t)$ is a given function usually taken equal to 1. The system to be solved is (53) (56). Then $T(t) = \langle \phi, N \rangle$. But since $T(t)$ can be chosen at will, the solution of eq. (53) being unique, we cannot expect constraint (56) to be satisfied. Therefore we imbed⁽⁵⁵⁾ the problem in a larger problem, by introducing for instance in B a free parameter μ .

We solve the non linear system (53) (56) for ψ and parameter μ . One method⁽⁵⁵⁾ is to use the Anselone-Rail algorithm which is a Newton iterative procedure which determines the iterates ψ_n and μ_n . If we impose $\mu_n = 1$, i.e. its takes its physical value for every n , then $T(t)$ is not any more arbitrary but a solution of the point kinetics equation which becomes a natural offspring of the constraint (56). We have finally an iterative procedure where $\psi_n(\vec{r}, v\vec{n}, t)$ and $T_n(t)$ are determined iteratively with $\langle \phi, \psi_n \rangle = 1$ for every n .

Do we need after all a "point kinetics equation" since we could solve (53) directly for a given $T(t)$? Here comes the stiffness in.

For many practical reactor problems the shape of the flux does not change much, in general in all cases where the perturbation of the critical state is rather homogeneous. It is usually sufficient to start with $\psi_0(\vec{r}, v\vec{n}, t)$ proportional to the fundamental eigenstate at time $t=0$ and stop the iteration with $T_0(t)$! Generalized quasistatic methods have shown in some fast reactor transients the need to iterate a few times.

Constraint (56) imposes that some phase space average of ψ be constant : as long as the shape of the flux does not change too rapidly, the rapid changes of the flux are transferred to $T(t)$ and therefore most of the stiff nature of the problem is concentrated in the simpler point kinetics equation. It means that two different time scales can be used : small time steps for point kinetics, i.e. T and large time steps for the shape function ψ . However for reactor transients that are strongly space dependent like asymmetric channel plugging in fast reactors, quasistatic methods might lose their usefulness.

ACKNOWLEDGMENTS

The author acknowledges fruitful discussions and a long standing collaboration in numerical reactor kinetics with Dr. E. MUND.

REFERENCES

- (1) R. GOLDSTEIN and L.M. SHOLKIN, N.S.E. 38, 94 (1969).
- (2) R. GOLDSTEIN and L.M. SHOLKIN, N.S.E. 40, 498 (1970).
- (3) W.L. HENDRY and G.I. BELL, N.S.E. 35, 240 (1969).
- (4) J. MIKA, Nukleonika 19, 1013 (1975).
- (5) J. MIKA, Nukleonika 21, 573 (1976).
- (6) T. BLENSKI, A. GADOMSKI and J. MIKA, N.S.E. 66, 277 (1978).
- (7) T. BLENSKI, Atomkernenergie 31, 232 (1978).
- (8) A. HENRY "Nuclear Reactor Analysis", The MIT Press (1975).
- (9) D. HETRICK "Dynamics of nuclear reactors", The University of Chicago Press (1971).
- (10) G. BELL and S. GLASSTONE "Nuclear Reactor Theory", Van Nostrand Reinhold Company (1970).
- (11) J. LEWINS "Nuclear Reactor Kinetics and Control", Pergamon Press (1978).
- (12) W. STACEY "Space-time Nuclear Reactor Kinetics", Academic Press, New-York (1969).
- (13) J. DEVOOGHT and E. MUND, International Topical Meeting "Advances in Mathematica' Methods for the Solution of Nuclear Engineering Problems" 2, 103-115 (1981).
- (14) A.F. HENRY, N.S.E. 20, 338 (1964).

- (15) J. MIKA, Report INR 700/XXI, RP (1966).
- (16) T. PORSCING, SIAM J. Appl. Math. 16; 301 (1968).
- (17) J. DEVOOGHT, N.S.E. 67, 147-161 (1979).
- (18) S. KAPLAN, A.F. HENRY, S.G. MARGOLIS, J.J. TAYLOR, P/271 Second Geneva Conference on Peaceful Uses of Atomic Energy.
- (19) T. KATO "Perturbation Theory for Linear Operators", p. 244, Springer (1966).
- (20) J. DEVOOGHT, E. MUND and al. "Developments of New Numerical Methods for Space-Dependent Nuclear Reactor Dynamics", EEC-DGIII-E2/ULB Contract, Vols. 1-3 (1977).
- (21) J. DEVOOGHT, E. MUND, A-stable algorithms for neutron kinetics, in Proc. NEACRP/CSNI Specialist meeting on new developments in 3D-neutron kinetics and review of kinetics benchmark calculations, T.U. München-Garching (1975).
- (22) F.T. ADLER, J. Nucl. En. 15, 81 (1963).
- (23) D.L. HETRICK, (see ref. (9))
- (24) E.R. COHEN, Proc. Gener. Conf. 11, 302 (1958).
- (25) R.O. BRITTAN, ANL-5577 (1956).
- (26) J.J. KAGANOVE, ANL-6132 (1960).
- (27) D.A. MENELEY, K.O. OTT and E.S. WIENER, ANL-7769 (1971).
- (28) K.F. HANSEN, B.V. KOEN, W.W. LITTLE Jr., N.S.E. 22, 51 (1965).
- (29) C.M. KANG, K.F. HANSEN, N.S.E. 51, 456-495 (1973).

- (30) J.P. HENNART, Maths. Comp. 31, 24-36 (1977).
- (31) J.P. HENNART, N.S.E. 64, 875-881 (1977).
- (32) J.P. HENNART, H. GOURGEON, Springer Notes (to appear) (1982).
- (33) J.C. VIGIL, N.S.E. 29, 392 (1967).
- (34) P.A. SECKER Jr., Numerical integration of dynamic nuclear systems equations by optimum integrating factors, Ph. D. Dissertation, University of Arizona, Published as LA-4252 (1969).
- (35) E.L. FULLER, D.A. MENELEY, D.L. HETRICK, N.S.E. 40, 206.
- (36) J. SZELIGOWSKI, Proc. Inter. Conf. on Reactor Math. and Utilization of Res. Reactors, Mexico City (1967).
- (37) J.A.W. de NOBREGA, N.S.A. 46, 366-375 (1971).
- (38) J.C. TURNAGE, N.S.E. 51, 67 (1973).
- (39) J.P. HENNART, B. TORRES BARRIOS, N.S.E. 59, 170-187 (1976).
- (40) J.L. WALSH, Amer. Math. Soc. Coll. Publ., Vol. XX, New York (1935).
- (41) W. LINIGER, R. WILLOUGHBY, SIAM J. Numer. Anal. 7, 47-66 (1970).
- (42) S. NØRSETT, BIT 14, 63-77 (1974).
- (43) C.W. GEAR, Numerical initial value problems in ordinary differential equations, Prentice Hall, Englewood Cliffs, N.J. (1971).
- (44) H.J. STETTER, Analysis of discretization methods for ordinary differential equations, Springer Tracts in natural philosophy, Vol. 23 (1973).
- (45) M. IZUMI, T. NODA, N.S.E. 41, 299 (1970).

- (46) R.D. LAWRENCE, J.J. DORNING, Trans. Am. Nucl. Soc. 24, 199-200 (1976).
- (47) E. HOFER, W. WERNER, CONF-730414, USAEC (1973), p. V-73.
- (48) B. LINDBERG, BIT 11, 29-52 (1971).
- (49) R. BULIRSCH, J. STOER, Num. Math. 8, 1-13 (1966).
- (50) M. BARDIAUX, J. DEVOOGHT, P. DUGNOILLE, M. GOLDWASSER, J. JOLY, E. MUND, I. SARAGOSSI, A. SIEBERTZ, "CASSANDRE : A 2D multigroup diffusion benchmark code for fast reactors power excursions analysis", Proc. of the ENS/ANS Topical Meeting on Nuclear Power Reactor Safety, Brussels, October 1978.
- (51) W. WERNER, Advances in Nuclear Sci. and Tech., Vol. 10, ed. by J. LEWINS, M. BECKER, 1981, Plenum Press.
- See also the biannual Proceedings of the ANS Topical meetings on Computational methods in nuclear engineering.
- (52) K. OTT, D.A. MENELEY, N.S.E. 36, 402 (1969).
- (53) J. DEVOOGHT, E. MUND, N.S.E. 76, 10-17 (1980).
- (54) A.F. HENRY (see ref. (14)).
- (55) J. DEVOOGHT, Annals of Nucl. Energy 7, 47-58 (1980).
- (56) N. B. CARVER, A.P. BAUDOUIN, AECL-5177 (1976).
- (57) J. DORNING, G. SPIGA, Trans. Am. Nucl. Soc. 28, 761-762 (1978).
- (58) J. MIKA, Internal report INR, Swierk, May 1979.
- (59) B. LINDBERG (see ref. 48)).

(60) B. LINDBERG in R. WILLOUGHBY, Int. Symp. Stiff Diff. Systems, Plenum Press (1973).

(61) M.E. RADD, IDO-17125, Idaho Nuclear Corporation (1965).

TRG-82-7
March 1982

THE SOLUTION OF SEVERAL REPRESENTATIVE
STIFF PROBLEMS IN AN INDUSTRIAL ENVIRONMENT:
THE EVOLUTION OF AN O.D.E. SOLVER

by

S. Thompson
P. G. Tuttle
Babcock and Wilcox
Computer Services

A summary of some of the items discussed in this paper will be presented at the 10th IMACS World Congress on Systems Simulation and Scientific Computation.

CONTENTS

1. INTRODUCTION
2. DESCRIPTION OF ENVIRONMENT
3. DESCRIPTION OF PROBLEMS
4. DESCRIPTION OF SOFTWARE
 - 4.1 Initial Selection
 - 4.2 Adams-Type Methods
 - 4.3 Coupled Differential-Algebraic Systems
 - 4.4 Root-Finding
 - 4.5 Diagnostics
 - 4.6 User Support
 - 4.7 Additional Desired Features
5. SUMMARY AND RECOMMENDATIONS

Tables

1	Summary of Representative Problems
2	Summary of Representative Non-Standard Problems
3	Chronology of Software Development

REFERENCES

1. INTRODUCTION

This paper describes the evolution of an automatic ordinary differential equation (O.D.E.) solver in an industrial environment. There are two primary purposes to this paper. The first is to describe the initial selection and subsequent modifications of the software. The second is to show that the selection and modifications were in direct response to the environment, i.e., to the types of problems to be solved, the staff, the hardware, and the interface requirements. Although this paper is devoted to one particular environment, it discusses several generic considerations which are undoubtedly applicable to other industrial installations. Consequently, the issues we discuss should be representative of those encountered by other industrial users of automatic mathematical software.

The software topics discussed are of a general nature. They should be of interest to the developers of other types of complex mathematical software intended for use in a production environment. In particular, software for optimization, the solution of nonlinear equations and, of course, method of lines share many of the same characteristics. For example, both optimization and nonlinear equation codes require evaluation of a user supplied function, possibly formation and processing of a Jacobian that may be either large and sparse or small and dense. Many of the points discussed are also applicable to other types of software (e.g., quadrature or linear algebra routines), that is, all mathematical software should conform to accepted software engineering principles which include adequate documentation and diagnostics for the user.

While we are primarily interested in the software, we must first describe the environment in order to understand the evolution of the software. This description has several facets. It includes a discussion of a typical user -- his general background and training, his specific mathematical and numerical analysis training, and the difficulties he encounters in using automatic software. Also included is a discussion of the manner in which the user interfaces with the mathematicians and scientific applications programmers. Related to this, the computer hardware configuration is briefly described. Following the description of the environment, the types of problems which must be solved numerically on a routine basis are discussed. This discussion highlights problem size, execution time, reliability requirements, and mathematical peculiarities which cause difficulties for automatic software. These problem characteristics demonstrate why software designed to accommodate stiffness can be very effective in this context. Furthermore, they provide useful examples as to why techniques incorporated in specialized software (e.g., root-finding, automatic method switching, coupled differential-algebraic systems, etc.) are required in software intended for serious use in an industrial environment. The material is presented in chronological order, thus emphasizing that the development of the software was in direct response to the requirements of the problems. It clearly demonstrates that the capabilities which we discuss are required to effectively use the software in an industrial environment.

2. DESCRIPTION OF ENVIRONMENT

The Nuclear Power Generation Division of Babcock and Wilcox has been primarily concerned with the design and analysis of nuclear reactors. This emphasis has profoundly affected the structure of the organization, the types of individuals employed, the computer hardware installed and, of course, the types of problems solved. This section discusses those particular aspects of the environment that have had the greatest impact on the evolution of the O.D.E. software. (A detailed discussion may also be found in [1].)

The bulk of the workload on NPGD computers consists of many repetitions of programs which have high execution times. Coupled with the quality assurance requirements imposed by the Nuclear Regulatory Commission, this has brought about a predominantly closed shop environment [1]. This is a closed shop not only in the computer programming, but also in the specification of the engineering problem and of the numerical solution. To develop a computer program in this environment requires the successful interaction of all three groups of people (i.e., the engineers, programmers, and mathematicians). Let us briefly consider then the individuals in each of these three groups.

The Applied Mathematics Unit has a full-time staff of six professionals, five of whom hold Ph.D.'s in applied mathematics. This staff is augmented by graduate and undergraduate cooperative students and by consultants as needed. The Engineering Applications Programming Unit is staffed by twenty-five degreed programmers. Over half of these people hold masters degrees or are actively pursuing graduate work. These degrees are in the areas of pure mathematics, physics, and various engineering disciplines. Only one degree is in applied mathematics. The Technology Units in the Engineering

Department are staffed by forty-five engineers, all of whom hold masters degrees. Approximately half of the people either hold Ph.D.'s or are actively pursuing one. These degrees are in various fields of engineering and physics.

While the programmers and engineers involved in the development of computer programs are extremely well-educated in their respective areas, they lack a formal education in numerical analysis. Only about 10% of the individuals have had a formal course in numerical analysis. Another 25% have had a numerical methods course for engineers or physicists.¹ Thus, it is very important to realize that while these groups are extremely well-educated in other areas, they are, for the most part, numerically unsophisticated. We suspect that this pattern is carried over to most other installations. This background should be kept in mind by developers of mathematical software, particularly in the more complex areas such as O.D.E.'s.

Now let us turn our attention to the hardware that is used by these people. The NPGD Computer Center includes interconnected CDC CYBER 76, CYBER 750, and CYBER 720 computers. The CYBER 720 and CYBER 750 process all of the interactive work and act as a system interface for all of the batch processing. Most of the scientific processing is done on the CYBER 76 and the CYBER 750. The CYBER 76 has a 27.5 nanosecond clock period with 64,000 sixty-bit words of small core memory, having a 220 nanosecond access time per word. It also has available 256,000 sixty-bit words of large core memory, having a 687.5 nanosecond access time per octet of words (i.e., total time to access eight words stored in consecutive storage locations).

¹These percentages are based on the response to a question pertaining to mathematical background that was asked at the beginning of an internal training course on numerical O.D.E.'s taught by R. L. Brown and M. A. Feldstein.

The CYBER 750 has a 25 nanosecond clock period with 198,000 sixty-bit words of memory, having a 575 nanosecond access time per word. Clearly, these systems are relatively large and fast. These facts should be kept in mind during the discussion of the difficulties that arise in terms of required storage and overall execution time.

3. DESCRIPTION OF PROBLEMS

As we have stated, the evolution and development of the software in question was in direct response to requirements of the problems which we must solve. Therefore it is necessary to have an understanding of these problems in order to appreciate the software development. The problems to be solved are primarily related to the design and analysis of nuclear reactors. As has been discussed, the physical model is usually developed by the engineers, although it is becoming more frequent for this to be done in conjunction with the mathematicians. It is not uncommon to encounter problems involving time simulations ranging from a millisecond (Example 3, below) to hundreds of seconds (Example 4), or with solution components ranging in magnitude from 10^{-10} to 10^{10} in the same problem. Furthermore, it is common to encounter problems which, in spite of the introduction of simplifications in the physical model, remain extremely difficult from a numerical point of view. (Indeed the simplifications involved in a crude model often render the simplified model much more difficult numerically.) The software has been used to solve a wide variety of such problems. Table 1 contains a summary of several representative problems. The problems described in Table 1 are relatively straightforward in the sense that they represent initial value problems in the standard form

$$y' = f(t, y)$$

$$y(t_0) = y_0$$

and which do not possess any noteworthy special features (e.g., severe discontinuities in f). They are, therefore, amenable to solution by many of the available high-quality solvers [2]. In fact, several such solvers [3] are routinely used to solve the problems in this first category, both on a production and a non-production basis.

A second category of problems is of primary interest in this paper. These problems include those in non-standard form which possess unusual requirements and which are, therefore, not amenable to solution by previously available standard software. The most frequently encountered subsystems are: the equations of fluid flow, heat diffusion in a cylinder, and neutron kinetics equations. (In each case, one-dimensionality and other simplifying assumptions are frequently introduced.) Although the individual problem types are well-known, several difficulties arise when they are solved simultaneously. The computational difficulties associated with the solution of these coupled systems are primarily related to: the detail and number of models (that is, the number of equations), the complexity of the auxiliary computations involved in the calculation of derivatives (that is, the execution time), and finally, the occurrence of special events which radically alter the character of the equation set (e.g., changing from stiff to non-stiff, changing the number of equations, introducing derivative discontinuities). Although we are well aware of the difficult questions associated with the solution of hyperbolic and mixed systems of partial differential equations, our primary concern here will be with the behavior of the system of O.D.E.'s after some discretization has been imposed. This approach is fairly common [4,60-63] and will allow us to more adequately address the software-related issues in question.

It is this second category of problems which directly influenced the software evolution. Table 2 contains descriptions of four representative problems from this category. The descriptions include the special problem requirements which dictated the software development and other pertinent information for each problem. Each of these problems actually represents the basis for a corresponding large-scale software project. Therefore, the times during which the corresponding project developments took place is also included. This places in perspective the specific O.D.E. software development. (In fact, each of these projects represents an ongoing effort. The entries in the table correspond to the times when the bulk of the work was performed.) Also included are indications of the complexity of the computer coding associated with the calculation of system derivatives. It should be noted that the number of lines of coding does not account for the additional use of a large mathematical software library of auxiliary sub-routines [3]. For example, spline software, linear equation solvers, non-linear equation solvers, and water property tables from this library are extensively used. Use of these routines can effectively add anywhere from a few hundred to several thousand lines of additional coding.

In order to further highlight the special problem requirements associated with the examples described in Table 2, a more detailed description will now be given for each problem. The precise physical and mathematical details of the problems are omitted for the most part, since they are quite lengthy and in each case, they may be found in the corresponding references.

Example 1 (Core Reflood Analysis [22])

The performance of the reactor coolant system during the refill and reflood phases of a postulated loss-of-coolant accident is simulated by this example. At the end of blowdown (i.e., the rupturing of a coolant pipe), the water level in the reactor vessel is usually calculated to have dropped below the elevation of the bottom of the core, leaving the core immersed in saturated or superheated vapor. As the emergency core coolant is injected into the system, the water level in the vessel lower plenum rises and eventually the bottom of the core is recovered. A set of one-dimensional fluid flow equations coupled to a simplified heat equation is used to model this process. What this translates to mathematically is a system of 100-150 problem-dependent O.D.E.'s. During the initial phase of the transient (before the water level rises to the bottom of the core), the problem is both non-stiff and relatively easy to solve. When the water initially comes in contact with the bottom of the core, additional equations are activated, derivative discontinuities occur in several components, and an extremely rapid transient ensues with the accuracy restrictions dominating the stability restrictions. As the core recovery proceeds, a steady-state condition is approached and the problem becomes mildly stiff (stiff, oscillatory in the sense of Gear [23]). Finally, the core flood tanks that injected the emergency core coolant empty and must be explicitly turned off.

The first difficulty that arises for the automatic software is precisely locating the time at which the water strikes the bottom of the core. The difficulty is compounded by the fact that this time is a function of the solution. This point must be properly located in order to generate an accurate solution and also to avoid (in this case, excessive) chattering [43,64] in the integrator. A second difficulty is associated with the

Example 2 (Reactor Pressurizer [24])

Pressurized water reactor systems normally use a steam pressurized surge tank to maintain and control primary system pressure. This surge tank is commonly referred to as a pressurizer. During normal operation, the upper portion of the vessel contains pressurized steam and the lower portion contains liquid. A decrease in the primary system's coolant pressure causes an outsurge of water from the pressurizer to the primary system, which compensates for any cooling or the removal of the primary system coolant. An increase in the primary system's coolant pressure causes an insurge of primary coolant into the pressurizer, which compensates for any heating or the injection of primary coolant. The pressurizer is maintained at a reasonably constant pressure through the use of heaters to raise the pressure, sprays to lower the pressure by condensing steam, and relief valves to rapidly lower the pressure. The fluid model consists of three regions which normally include: superheated steam, saturated mixture, and sub-cooled liquid. Various combinations of these regions are in existence at any given time and a given region can be created or deleted in the course of a transient. The model consists of O.D.E.'s derived from the conservation of mass and energy equations for the liquid, equations defining the rise of the bubbles, and equations describing the behavior of the heaters. This forms a system of fourteen ordinary differential equations. There are also a large number of auxiliary computations related to the interface conditions between the regions and to condensation.

The problem is small, non-stiff and for the most part, relatively easy to solve. It is included here for several reasons. The first is that without an extremely accurate and consistent set of water property tables (e.g., [19]), it is impractical to solve this problem purely as a system of

ordinary differential equations. Rather it is necessary to recast the system into one consisting of a mixed system of ordinary differential and nonlinear algebraic equations. It is interesting to note that several different forms of this system can be written, most of which are analytically unstable. The next important feature is that the sprays, heaters, and valves form a set of nineteen special events that must be correctly accounted for in order to generate an accurate solution (each of these events is a function of the solution). Therefore, there are more special events than O.D.E.'s! Finally, it is necessary to restart the integration following most of these special events in such a way as to avoid missing one of the other events. This model is actually a subsystem that is used in many other computer programs, notably Example 4.

AD-A122 171

PROCEEDINGS OF THE INTERNATIONAL CONFERENCE ON STIFF
COMPUTATION APRIL 12. (U) UTAH UNIV SALT LAKE CITY DEPT
OF CHEMICAL ENGINEERING R C AIKEN 1982

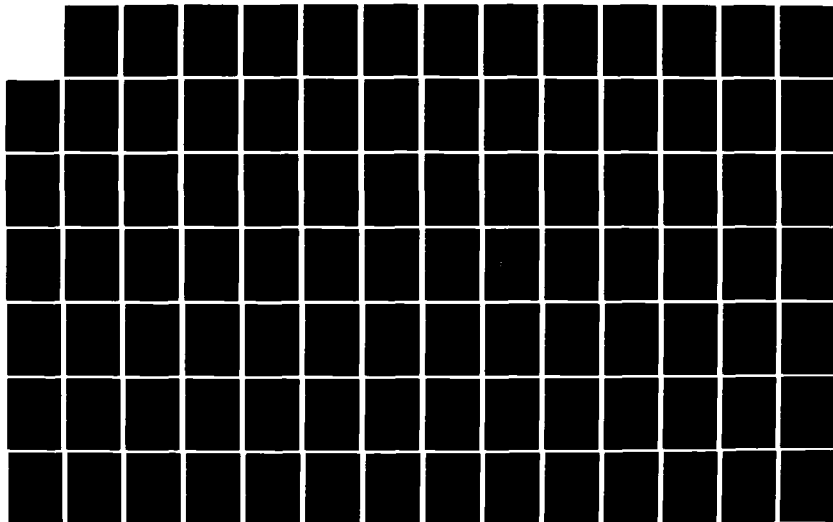
3/4

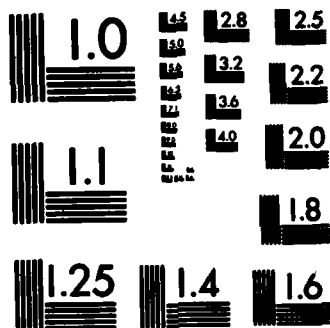
UNCLASSIFIED

AFOSR-TR-82-1036-VOL-3 AFOSR-82-0038

F/G 12/1

NL





MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

Example 3 (Bubble Growth in a Time-Dependent Pressure Field [25])

The problem of homogeneous nucleation (bubble growth) and flashing in a superheated liquid undergoing rapid decompression is illustrated by this example. Since it is conceivable that a reactor coolant undergoing rapid decompression during a loss-of-coolant accident could become highly superheated before flashing, several well-known blowdown experiments have been performed relative to this question. This example represents a general solution technique for analyzing the validity of such experiments and toward the inclusion of nonequilibrium flashing effects in the calculation of the loadings on the reactor's internal components during a loss-of-coolant accident.

The process is modelled by a system of coupled partial and ordinary differential equations consisting of the heat conduction equation (for the thermal layer enveloping a vapor bubble), the equation of motion (to describe the growth of a vapor bubble) and a third subsystem (to describe the state of the surrounding bulk liquid). Since the subsystems are coupled at the bubble surface, a moving boundary problem must be solved. Furthermore, the coupling is algebraic in some cases. This model has been used to correctly analyze the well-known bubble growth measurements in a decreasing pressure field, bubble growth in constant-pressure water, and novel chamber-and-piston experiments. Depending on the problem to be analyzed, the resulting system of O.D.E.'s contains 20-300 equations. Initially the problem is non-stiff. It is necessary to then determine precisely the time at which the bubble forms and begins to grow -- which is again a function of the solution. At this point, the microscopic flashing model is activated -- resulting in a very rapid transient, followed by a gradual approach to steady-state as pressure recovery proceeds.

The first difficulty that arises for the automatic software is the determination of the time corresponding to the onset of bulk nucleation. The next difficulty is related to the algebraic coupling at the bubble surface which results when the microscopic flashing model is activated. A third difficulty is related to the necessity to switch to stiff solution techniques to accommodate the mild stiffness associated with the gradual pressure recovery. Sometimes a final difficulty arises. The problem is solved by a method-of-lines [26] approach in which any one of several transformations is used both to immobilize the moving boundary [26] and to scale the variables [27]. Depending on the various transformations used, root-finding involving the old independent variable and the new dependent variable is required for locating the output points [9].

Example 4 (Simulation of Upset Transients [28])

This problem simulates the performance of the reactor coolant system during transients caused by minor upset conditions (e.g., loss of offsite power, or loss of main heat-sink). It is assumed that all of the transients start from steady-state. Since all of the transients are relatively slow, a simplified mathematical model is used which consists of the following coupled subsystems:

- (i) the one-dimensional cylindrical heat equation
- (ii) a simplified set of one-dimensional flow equations
- (iii) the time-dependent point kinetics equations
- (iv) several auxiliary subsystems (e.g., Example 3)

The total system contains approximately 150 O.D.E.'s and is usually considered stiff due to the presence of subsystem (iii). The system is initialized by holding all boundary conditions constant and integrating until steady-state is achieved. Depending on the transient to be simulated, a time-dependent transient forcing function is then activated. One common simulation involves the step insertion of reactivity into the point kinetics equations. Owing to the system coupling, this effectively introduces a severe jump discontinuity in the time derivative which is quickly transmitted throughout the rest of the system -- resulting in a radical change in the nature of the various subsystems (e.g., changes in the mode of heat transfer from the fuel pin to the reactor coolant). It is also necessary to incorporate approximately a dozen reactor trip functions in the model (e.g., the system pressure too high or too low, the outlet temperature too high, or outlet flow rate too low). Prior to the trip, the problem is stiff, but relatively easy to solve. After trips occur, portions of the system remain stiff. However, other portions undergo rapid transitions. Again it is imperative to

precisely locate the time of the trips in order to generate an accurate solution. Finally as the transient proceeds, it again approaches steady-state. It should also be kept in mind that all of the concerns outlined in Example 3 are also present here. In this problem, it is crucial that the stiff reactivity component be accurately tracked, due to its strong feedback into the remaining system components.

The preceding discussions by necessity are quite terse. Detailed discussions may be found in the indicated references, from which the above discussions were excerpted. However, it is anticipated that the present discussions convey a general understanding of the problem requirements which guided the evolution and development of the software to be discussed in the next section.

4. DESCRIPTION OF SOFTWARE

Ideally, the development of the software should be presented strictly chronologically in order to emphasize its evolutionary nature. Unfortunately, this becomes cumbersome when discussing particular features, since they were actually incorporated and refined over a relatively long period of time. In Table 3 we have indicated the general chronology of the development. This again re-emphasizes the fact that the development of the software was evolutionary and in direct response to problem and user requirements (as may be seen by comparing the dates in Tables 2 and 3). In addition to the items which are included in Table 3, we also wish to discuss several other issues. The material to be discussed below therefore includes:

- (1) the initial assessment and selection of an integrator
- (2) the addition of generalized Adams-Type methods
- (3) the addition of the capability to solve systems of coupled differential and algebraic equations
- (4) the addition of root-finding capabilities
- (5) the addition of diagnostic tools
- (6) user support
- (7) additional software engineering considerations

4.1 Initial Selection

The first step in the development consisted of an assessment of available software and an initial selection from this software. The results of this phase are reported in [2,29-30]. The specific software which was evaluated consisted of Runge-Kutta software [11-12], non-stiff Adams software [9,13], stiff collocation software [7-8], and several variants of GEAR [14,31] and EPISODE [15,17,32]. (Several other commercially available solvers were also evaluated, but were excluded since they were not competitive with the above software.) Fortunately, several previous evaluations had been performed elsewhere, notably [11,33-38]. These previous evaluations significantly influenced our assessment. Subsequently, several of the above solvers were selected for inclusion in our Standard Mathematical Subroutine Library [3]. As indicated by Table 1, these solvers are still used in one or more applications at B&W. Furthermore, the capabilities of [13] have influenced the portion of the development of our software related to the addition of root-finding. However, it was decided that in our environment GEAR and EPISODE were the most appropriate candidates for a general-purpose solver. (On the other hand, any reader familiar with the other software mentioned in this section will perceive our debt to the developers of the other software. Obviously, much of our development consisted of tailoring and expanding their ideas so that the software could be used in our environment. Without their work to rely on, the solution of our problems would not have been possible.)

Due to the sizes of the problems in Table 2, we found it difficult to use GEAR and EPISODE. (Recall that a large amount of available computer memory is usually required to evaluate system derivatives for our problems.) It was also usually not feasible to use their banded counterparts [14-15,17]

— since the coupling of fluid (in a loop), heat, and kinetics equations results in a Jacobian that does not have a banded structure (nor a nearly banded structure in which elements outside a given band may be neglected). Therefore, it was decided to use to sparse solver GEARS [39] as the base for our general-purpose solver. GEARS consisted of GEAR, but with the linear algebra portion replaced by a version of the Yale Sparse Matrix Package [40]. Therefore, the present software evolved specifically from [39]. (More recent versions of this software are available [57], however, they do not address the difficulties encountered in our problems.)

4.2 Adams-Type Methods

It was noted in Example 1 that the problem becomes mildly stiff and oscillatory as the transient proceeds. It was found that the third and fourth order Adams-type methods in [41], when used with a Newton iteration, handle the problem more efficiently than do the standard Adams methods or backward differentiation methods. Because of the manner in which they were constructed (as linear combinations of standard Adams methods), they were also easily implemented into GEARS.

The gain in efficiency over the standard methods is due to the A_0 -stability of the methods (absolute stability on the negative real axis). For problem 1, the solver typically runs at order 3 for these methods whereas it typically runs at order 2 (trapezoidal rule) for the standard Adams methods. Although it also runs at order 3 for the backward differentiation methods, they have larger local truncation error coefficients than those in [41]. The average step size is, therefore, smaller. Overall a reduction in execution time by a factor of 2-4 is obtained for Example 1 using the Adams-type methods. For the associated production computer program, which performs many long repetitive runs, this results in a substantial saving in the course of, say, one year.

4.3 Coupled Differential-Algebraic Systems

Systems of the form

$$(1) \quad y' = f(t, y, w), \quad y(t_0) = y_0$$

$$(2) \quad 0 = r(t, y, w), \quad r(t_0, y_0, w_0) = 0, \quad w(t_0) = w_0$$

arise naturally in many applications. One accepted approach is to differentiate (2) with respect to t to obtain an initial value problem in standard form. As noted in Examples 2 and 4, this may not be possible since an explicit expression for (2) may not be available and the use of finite differencing would be too inaccurate. Another popular approach is to solve (2) as part of the function (1). This frequently proves impractical for several reasons. The first is caused by the interaction of the convergence error in the iteration with error control in the integrator. If the convergence tolerance is too small, the program's execution time is high (due to the iteration), while if the tolerance is too large, the integrator error test fails -- leading to prohibitively small step sizes and resulting in high execution times, as well as possibly erroneous results. The second drawback is that the evaluation of r may require that most of the computations required for the evaluation of f be completed. This again leads to an extremely high execution time for the program. A third drawback that is sometimes overlooked is that questions related to problem stability can be masked by indirect solution techniques (e.g., Example 2). In order to circumvent these difficulties, the software was modified to directly solve these types of systems. This is a relatively straightforward modification -- since a Newton iteration is already available for the solution of the corrector equations. All that is required is to modify the usual iteration matrix [42]

$$P = I - h\beta \left(\frac{\partial f}{\partial y} \right)$$

$$\text{to } P' = \begin{bmatrix} I - h\beta \left(\frac{\partial f}{\partial y} \right) & \left(\frac{\partial f}{\partial w} \right) \\ \left(\frac{\partial r}{\partial y} \right) & \left(\frac{\partial r}{\partial w} \right) \end{bmatrix}$$

The predictor step is performed in the usual fashion [42] for y while an algebraic extrapolation of the same order and based on back-points is used for w .

It is recognized that while there does exist some theory [44-46] pertaining to the existence and uniqueness of solutions to (1) and (2), that there does not, at this time, exist adequate justification for the numerics. However, the actual performance has been very satisfactory. The procedure has been used to accurately solve problems that otherwise could not have been solved. Furthermore, the integrator has failed, with an error message, anytime we have attempted to solve a problem for which there does not exist a solution. Consequently, we have found this capability to be a very useful one. For other approaches to variants of this problem, see [44-46].

For any readers who might be interested in also implementing such a procedure, it is also of interest to note the need to refine the algebraic values at output points. This is necessary since the interpolated output values are based on the predictor. Hence, they do not necessarily satisfy (2).

4.4 Root-Finding

Variants of this capability have been previously used in other software, notably [13,43,47] (see also [10]). Our approach was most influenced by that used in [13], where the procedure is referred to as g-stops. This terminology will also be used throughout this section.

The overall idea is to construct an auxiliary function whose zeroes correspond to the occurrence of the special event. For instance, consider Example 1. When the water level in the lower plenum reaches the bottom of the core, drastic changes in the nature of the system occur. It is necessary at this point to take special action, such as restarting the integration. Thus, the auxiliary function would be the height of the bottom of the core minus the height of the water. Hence, the occurrence of this special event corresponds to a zero of the auxiliary function. When the integrator locates this zero, the integration is then restarted with an appropriately small step size to accommodate the ensuing rapid transient. This same approach can be used in Example 2 for turning heaters on and off, or for opening and closing relief valves.

In general, we wish to locate the zeroes of the equations

$$0 = g_i(t, y, y'), \quad i = 1, \dots, k$$

simultaneously with the integration of the O.D.E. In each case, once an interval is located in which one of the components changes sign, a root-finder is used to locate the corresponding zero. Brent's one-dimensional algorithm [48-49] was modified to perform the root-finding in the present software.

The root-finding is of two types, interpolatory and extrapolatory. Interpolatory g-stops are used when it is possible to easily integrate past the time(s) corresponding to the special event (e.g., when the event does not cause severe derivative discontinuities). For example, this would be the case if the event corresponded to a simple switch with an associated time delay to obtain output, or when it is used to locate times at which a given variable attains a particular value. ([9] discusses several novel uses of this feature.)

The g-stop function requires values for both y and y' . These values are available from the Nordsieck array [23,50] in the present software. Basically, after the completion of each successful step (i.e., following the corrector iteration), the Nordsieck array is extrapolated backwards over the interval corresponding to this step to examine for sign changes and to do the root-finding, if necessary. In this case, it is not necessary to restart the integrator unless, of course, the root corresponds to a severe discontinuity or requires that other special action be taken. Therefore, interpolatory g-stops are generally efficient since they do not interfere with the integrator's order and step size selection procedure.

Extrapolatory g-stops are primarily intended for a totally different purpose. Generally, they are used to cope with severe derivative discontinuities associated with special events which frequently occur in the problems we must solve -- for example, when the water reaches the bottom of the reactor core in Example 1. Experience with these problems indicates that several commonly recommended techniques for dealing with such discontinuities do not work well for us. For example, simply ignoring the discontinuity is usually disastrous. If it is severe enough (as in Example 1), the presence of the discontinuity will, at best, result in

excessive integrator chattering, leading to extremely high execution time. (More often than not, a program abort is encountered.) Furthermore, the special event often necessitates the addition or deletion of portions of the model (e.g., the creation or disappearance of a steam region in Example 2), making it impossible to ignore the event. Another technique which is sometimes recommended is to complete the current step and then allow the changes corresponding to the special event to be made. Depending on the nature of the problem prior to the occurrence of the special event, the step size being used is invariably much too large, resulting in either a program abort or inaccurate results. This technique, therefore, has not worked well for us.

Extrapolatory g-stops correspond to using the predictor to detect sign changes over the next step. In the present software, this is accomplished by extrapolating forward before each integration step is performed (i.e., before the corrector iteration is initiated). Since only the predictor is used in this, the discontinuity can be located without requiring any derivative evaluations for points past the discontinuity. The integration is then restarted with the appropriate changes in order and step size necessary to "transcend" the discontinuity.

Extrapolatory g-stops are generally less accurate than interpolatory g-stops since they are located using only the predictor. In fact, it is possible to miss roots of the auxiliary function if only extrapolatory g-stops are used. This can occur, for example, if the auxiliary function does not change signs following the prediction, but does change signs following a correction. In problems such as Example 4, it is imperative that the times corresponding to all trips and switches be correctly located, since the system response can be drastically altered otherwise. (For a simpler example where this is the case see [43].) Therefore, in our implementation,

extrapolatory root-finding is automatically backed-up by interpolatory root-finding. While the possibility still exists that a g-stop might be missed (the corrector polynomial simply may not reflect the sign change in the auxiliary function), we have not encountered a situation where a missed extrapolatory g-stop is not picked up by the backup interpolatory calculations. Furthermore, the increased reliability of the software more than adequately compensates for the slight increase in overhead.

Related to the use of g-stops, another issue that seems not to have received much attention in some previous software is that of scaling the auxiliary function. It is usually necessary that the function be scaled appropriately in order that the root-finder's error control be compatible with that of the integrator. While it is possible to leave the responsibility for such matters to the user, we consider it preferable to automatically accommodate this matter in the software.

4.5 Diagnostics

Since the evaluation of the derivatives may involve several thousand lines of executable code (e.g., the entire program for the relatively small problem in Example 3 comprises approximately 17,500 lines of coding), it is imperative that effective diagnostic tools be available. Errors do occur in both the program and the model and these errors must be located in a reasonably efficient manner. The diagnostic tools described here evolved as the problems in Tables 1 and 2 were implemented and debugged. They represent devices which were found to be convenient and, in some cases, indispensable. While they probably do not encompass all devices that might be useful, they do give an indication of what is needed. Our philosophy is that the user will introduce errors and that the software should actively cooperate in detecting these errors. This is a departure from the currently accepted approach which is that the software itself need only be correct. Following is a brief description of each of several capabilities which are included as debug options in the software.

(i) Identification of troublesome components -

Frequently, only a few components are causing the error test or the corrector iteration to fail repeatedly. An option is included to automatically identify these components by using a max norm and printing the number of the equation that caused failure.

(ii) Identification of oscillatory components -

Model deficiencies or programming errors are often manifested as oscillatory components. Therefore, an option is included to detect a sign change in the derivative between successive steps and to print the component(s) for which this occurs.

(iii) Checking the function evaluation -

Given the complexity of the function, evaluation of some derivatives may not be defined, particularly during initial debugging. To detect this, the y' vector is preset to negative infinity (that negative number which is largest in magnitude and machine representable) before the evaluation is called. After the function evaluation, y' is checked; and if any components have not been defined, the component is identified and the integration is halted. A related problem is that, from the use of fixed-step, first order techniques used in many extant engineering codes, users have grown accustomed to naively resetting the y -vector if it falls outside of some desired physical range (e.g., resetting of negative flow rates). Needless to say, this usually has a detrimental effect on the integrator. In order to detect this, integer sums of the y vector are formed before and after the function evaluation and on exits and entry from the integrator. If the sums do not agree, an error message is issued and the integration is halted. This procedure requires only four additional storage locations and easily detects an error that otherwise is extremely difficult to locate. (Extreme paranoia resulting from consulting for some particularly difficult program developments has led to also checking all working storage currently in use by the integrator.)

(iv) Halting divergent corrector iterations -

It has been observed that when the Newton iteration diverges, it does so quite rapidly. This can lead to abnormal terminations in the function evaluation (e.g., a pressure out of range in water property tables). To avoid this, a divergent correction iteration is immediately halted.

(v) Printing the Jacobian -

Some programming errors, such as the use of variables from a previous time step, are easily detected if a "picture" of the Jacobian is printed. An auxiliary subroutine is provided which prints 1's for nonzero values of the Jacobian and 0's for the zero values.

(vi) Statistical information related to integrator performance -

A good deal of other information regarding integrator performance is also collected, although it is only indirectly available to the user (to whom the information is usually of little or no interest). This information is often of substantial interest to the numerical analyst, however. For example, counts of the number of order increases and decreases can give an indication of order cycling, which is often associated with model errors or instabilities. Other useful information includes the number of steps taken at each order, the number of integrator crashes, the number of error test failures, and the number of corrector convergence failures.

(vii) Heuristics -

Although this capability is not available to the user, it is possible to alter heuristics that are built into the software, in a very convenient manner. This includes forcing that a given number of corrections be performed at each step, requiring that more steps be taken between changes in order and step size, attempting to steer the integrator toward a particular order, biasing the order selection toward high or low orders, changing the strategies regarding the frequency of updating the Jacobian, changing the definition of weighting vectors, switching to various

norms, and other miscellaneous heuristics. None of these capabilities may be used to override the adaptive nature of the software. Indeed, each item usually degrades integrator efficiency due to the conservative nature of the item. Consequently, this capability is intended strictly for use by the numerical analyst for evaluating relative merits of heuristic procedures during model analysis and related testing.

While most of the above devices are simple and relatively straightforward, they often mean the difference between making or missing a delivery date (and hence, whether or not the software will continue to be used by users who are already suspicious of a black box). They also relieve some of the burden of debugging a complicated program from the numerical analyst by providing meaningful diagnostics to the programmer and the engineer. As a result, they increase the user's confidence in the software by overcoming to some extent his fear of its black box nature, by providing useful information about problems that are present.

4.6 User Support

The typical programmer or engineer described in Section 2 is not equipped to effectively utilize a piece of numerical software as sophisticated and complex as an automatic integration package. To ease this difficulty, simplified driver subroutines that solve initial value problems in standard form have been written. These in turn preset many of the parameters required and then call the standard integrator. This allows the user to learn and grow as the complexity of his problem increases. At any given time, he needs only to access the portions of the integrator that are required by his problem. To increase the level of mathematical sophistication of the typical user, internal training courses in numerical analysis are also offered. Finally, the diagnostic capabilities of the integrator have been expanded to provide meaningful information to the user. In spite of the fact that this does not eliminate the need for the in-depth involvement of an experienced numerical analyst in any large development effort, it does contribute to the usefulness of the software by relieving the numerical analyst from many of the routine tasks that can be performed by engineers and trained programmers.

4.7 Additional Desired Features

There are several additional features which would be quite useful in our environment, but which have not yet been incorporated into the software. These features should receive future attention by software developers. ([56] contains a similar summary.) They include the following.

4.7.1 Global Error Estimation

Global error estimation should be regarded as a necessity rather than as a luxury. A global error estimate should be available in simulations of problems such as those discussed in this paper. Global error estimates would not necessarily be included with the intent that this error be controlled by the integrator, but rather to avoid accepting at face value incorrect results. As an example, consider the following apparently simple problem [51].

$$y' = \begin{bmatrix} -1 - 9 \cos^2 6t + 12 \sin 6t \cos 6t & 12 \cos^2 6t + 9 \sin 6t \cos 6t \\ -12 \sin^2 6t + 9 \sin 6t \cos 6t & -(1 + 9 \sin^2 6t + 12 \sin 6t \cos 6t) \end{bmatrix} \cdot y$$

The function is linear, continuous, Lipschitz and further the eigenvalues of the Jacobian are fixed in the left half plane at -1 and -10. Thus, on the surface this seems to be a very innocent problem. However, the family of solutions is unstable; a fundamental matrix for this problem is

$$\Phi = \begin{bmatrix} e^{2t}(\cos 6t + 2 \sin 6t) & e^{-13t}(\sin 6t - 2 \cos 6t) \\ e^{2t}(2 \cos 6t - \sin 6t) & e^{-13t}(2 \sin 6t + \cos 6t) \end{bmatrix}$$

Then a problem such as

$$y' = A(t)(y-s(t)) + s'(t)$$

$$y(0) = s(0)$$

where $A(t)$ is the 2×2 matrix above and $s(t)$ is any well-behaved vector function has a solution imbedded in an unstable family. Note also that action based on a knowledge of the eigenvalues of the Jacobian matrix for this problem may be disastrous.

If a seemingly innocent problem can exhibit such pathological behavior, any meaningful problem must be treated with care. To our knowledge [12] is the only widely available software which generates an accurate global error estimate for the above problem, although other integrators generate a "solution" to this problem. Further investigation of the ideas in [52-53] could provide a useful means of extending global error estimation techniques to many of the popular linear multi-step O.D.E. solvers.

4.7.2 Jacobian Matrix

The manner in which the Jacobian matrix and related ones are processed should be carefully examined. For example, the ideas related to the use of auxiliary storage in [54] can be invaluable for problems such as Examples 1 and 3, where derivative evaluations are very complex. The simple idea in [54] of saving the Jacobian, J, so that it does not need to be recalculated when a change in order or step size requires that the iteration matrix

$$P = I - h\beta J$$

be reformed and decomposed, would greatly enhance the attractiveness of the software in an environment such as ours.

Alternate forms of the sparse matrix package [40,58-59] that require less storage should be included in the software. Also, on a CDC machine, it is wasteful to use a sixty bit word to store an integer pointer. The use of integer packing techniques could significantly reduce the required storage. Even though many of our problems involve only a few hundred equations, and are, therefore, small by many standards, storage is usually a limiting factor. In our opinion, the benefits of being able to solve a larger problem more than compensate for the machine dependencies introduced. Further, the machine dependencies and subsequent nonconformance with desired software engineering practices could be isolated to a few functions or subroutines.

4.7.3 General

Conformance with accepted software engineering practices would greatly ease the maintenance and modification tasks. In particular, the procedures for error testing, the solution of the nonlinear corrector equations, and the selection of step size and order should be separate modules. This would greatly facilitate the plugging and unplugging of modules to perform these tasks. Also, future upgrades to the software could more easily be incorporated into existing software.

Documentation is a critical area that needs to be addressed in order for the software to be more widely used. Predominantly, the existing documentation is written by numerical analysts -- for other numerical analysts. However, as was noted in Section 2, the majority of the users have not had even a numerical methods course. Thus, it is imperative that documentation be available that is written at any elementary level and which includes a large number of examples with detailed descriptions of their solutions. As an indication of the level of detail which we have in mind for adequate documentation, the following is an outline of the documentation currently in preparation [55] for the present software.

1. Introduction
2. Discussion of software methodology (an elementary discussion of linear multi-step methods and related concepts)
3. Discussion of the software (solution procedure, internal heuristics, and related concepts)
4. Detailed discussion of the solution of several (10) related problems

While it is, of course, not necessary that documentation adhere specifically to an outline such as this one, it does give an indication of the level of detail which is necessary.

5. SUMMARY

This paper discussed the evolution of automatic O.D.E. software in a particular environment, which is representative of many industrial installations. Characteristics of this environment which affected the software development were described. These included descriptions of the typical user, the hardware, and the types of problems to be solved. In addition, several enhancements were discussed. The inclusion of these features in future software would be of great benefit in an industrial environment. Other additional items and capabilities which would be useful in an industrial environment were also identified and discussed.

It was explained why future growth must be toward software which is user friendly. This requires complete and thorough documentation at a relatively elementary level with many examples. Extensive, comprehensible error diagnostics are required. Although the in-depth involvement of an experienced numerical analyst will still be required in large development efforts, the considerations identified would help to remove user suspicion and reduce reluctance to use a black box. For the software to gain widespread acceptance, it is critical that it not present additional hurdles to overcome and that it actively cooperate with the user.

One of the compelling reasons for writing this paper was to help to bring about the broader usage of standard mathematical software, especially in the area of O.D.E.'s. Most industrial installations lack both the resources and the interest to make many of the software enhancements discussed in this paper. Therefore, the responsibility must be assumed by the developers of mathematical software. The future releases of

existing software can gain significantly wider acceptance and usage by including these enhancements. Consequently, this paper should provide valuable feedback to the developers of mathematical software as to what features are needed by industry in numerical software. Hopefully, the approach taken here will help to increase (and in some cases, establish) lines of communication between the software developers on the one hand and the software users on the other, resulting in broader and more effective usage of standard mathematical software.

Acknowledgement. The authors gratefully acknowledge the suggestions of G. D. Byrne and A. C. Hindmarsh whose constructive criticisms of a preliminary version significantly improved the exposition of this paper.

The first author would also like to mention Pseb Vamajoth for many stimulating, albeit unrelated conversations.

TABLE 1. Summary of Representative Standard Problems

<u>Description of Physical Problem</u>	<u>Mathematical Problem Requirements</u>	<u>Applicable Software</u>
Coupled heat exchanger performance [5-6]	Ability to switch from a system of non-stiff equations to a different system of stiff equations.	[7-12]
Interacting species -- point kinetics equations	Integration of small systems of extremely stiff equations.	[7-8]
Structural mechanics modelling of reactor performance during earthquake conditions.	Integration of rapid oscillatory transient followed by a long steady-state integration.	[13]
Finite-element models arising in structural mechanics problems.	Integration of large banded systems of equation (100-2000 equations).	[14-15]
Quench-front type problems.	Integration of large systems of non-stiff equations with mild derivative discontinuities (100-500 equations).	[16-17]
Analytic continuation. Dependence of the solution of systems of non-linear equations depending on a parameter [18-19].	Ability to track solutions of a system of equations containing an imbedded parameter. Simple root-finding requirements.	[10,13,20]
Analysis of reactor vessel internal loads [21].	Integration of an extremely large (500-9000 equations) system of non-stiff equations.	[20]
Analysis of steam generator performance [65-66].	Integration of large banded extremely stiff equations (100-1000 equations). Root-finding to locate phase change boundaries.	[20]

TABLE 2. Summary of Representative Non-Standard Problems

Description of Physical Problem	Description of Special Problem Requirements	Problem Characteristics			Chronological Time in Which the Problem was Solved
		Lines of Active Coding in DFN	Avg. Time Req'd to Evaluate DFN	Typical Execution Time (sec)	
Modelling of reactor coolant system during refill and reflood stages of a postulated loss-of-coolant accident.	Coupled fluid and heat conduction equations. Initially non-stiff. Extremely rapid transient following a severe discontinuity. Mildly stiff approach to steady-state.	2500	.010	200	1977-1978
Modelling of reactor pressurizer system.	Coupled systems of differential algebraic equations. Occurrence of special events.	2300	.040	100	1979
Modelling of the growth of vapor bubbles in a time-dependent pressure field.	Coupled systems of ordinary and partial differential equations. Non-standard coupling at a moving boundary. Rapid transient followed by gradual approach to steady state.	1300	.005	50	1979-1980
Modelling of anticipated reactor response during a system transient resulting from a postulated upset condition in the primary system.	Coupled fluid, heat conduction, and kinetics equations. Steady-state initialization. Severe derivative discontinuities. Special trips & switches.	3000	.050	75	1978-1981

TABLE 3. Chronology of the Software Development

<u>Activity</u>	<u>Time</u>
• Initial assessment and selection.	1975-1976
• Assessment and testing of GEARS.	1976
• Dynamic dimensioning, subroutine standardization, and other related modifications.	1976-1977
• Incorporation of extrapolatory root-finding.	1977-1978
• Incorporation of Adams-Type methods.	1978
• Incorporation of ability to solve systems of coupled differential-algebraic systems.	1978-1979
• Incorporation of extensive diagnostics and user-oriented debugging devices (e.g., auxiliary drivers).	1979-1980
• Incorporation of interpolatory root-finding.	1980-1981
• Incorporation of refinements in the root-finding capabilities.	1981

REFERENCES

1. P. G. Tuttle and L. L. Barinka, Scientific Computing Practices of the Nuclear Power Generation Division of Babcock and Wilcox, Proceedings of the Topical Meeting on Computational Methods in Nuclear Engineering, Vol. 2, Williamsburg, Virginia, April 1979, pp. 5b.1-5b.10.
2. S. Thompson, A Comparison of Available Software for the Numerical Solution of Stiff Ordinary Differential Equations, NPGD-TM-368, Babcock & Wilcox, Lynchburg, Virginia, June 1977.
3. L. L. Barinka, et al., Standard Mathematical Subroutine Library, NPGD-TM-363, Babcock and Wilcox, Lynchburg, Virginia, March 1977.
4. M. B. Carver, et al., The FORSIM VI Simulation Package for the Automated Solution of Arbitrarily Defined Partial and/or Ordinary Differential Equations Systems, AECL-5821, Atomic Energy of Canada, Ltd., February 1978.
5. K. W. Turner, COUP: Decay Heat Removal and Component Cooling Water System Simulation Program, NPGD-TM-447, Babcock and Wilcox, Lynchburg, Virginia, August 1978.
6. K. W. Turner and S. Thompson, Calculation of Coupled Heat Exchanger Performance, NPGD-TRG-78-13, Babcock and Wilcox, Lynchburg, Virginia, June 1978.
7. B. L. Hulme and S. L. Daniel, COLODE: A Collocation Subroutine for Ordinary Differential Equations, SAND 74-0380, Sandia Corporation, Albuquerque, New Mexico, December 1974.
8. B. L. Hulme, Using STFODE/COLODE to Solve Stiff Ordinary Differential Equations, SAND 74-0381, Sandia Corporation, Albuquerque, New Mexico, December 1974.
9. L. F. Shampine and M. K. Gordon, Computer Solution of Ordinary Differential Equations - The Initial Value Problem, W. H. Freeman and Company, San Francisco (1975).
10. M. K. Gordon, Using DEROOT/STEP to Solve Ordinary Differential Equations, SAND-75-0211, Sandia Corporation, Albuquerque, New Mexico, April 1975.
11. L. F. Shampine, H. A. Watts and S. M. Davenport, Solving Non-Stiff Ordinary Differential Equations - The State of the Art, SAND 75-0182 Sandia Corporation, Albuquerque, New Mexico, October 1975.
12. L. F. Shampine and H. A. Watts, Global Error Estimation for Ordinary Differential Equations, ACM TOMS, Vol. 2, No. 2 (1976), pp. 172-186.

REFERENCES CONT'D

13. F. T. Krogh, VODQ/SVDQ/DVDQ, Variable Order Integrators for the Numerical Solution of Ordinary Differential Equations, Tech. Brief NPO-11643, Jet Propulsion Laboratory, California Institute of Technology, Pasadena, California, November 1970.
14. A. C. Hindmarsh, GEARB: Solution of Ordinary Differential Equations Having Banded Jacobian, UCID-30059, Rev. 1, Lawrence Livermore Laboratory, Livermore, California, March 1975 (Errata January 1976).
15. G. D. Byrne and A. C. Hindmarsh, EPISODEB: An Experimental Package for the Integration of Systems of Ordinary Differential Equations with Banded Jacobians, UCID-30132, Lawrence Livermore Laboratory, Livermore, California, April 1976.
16. A. C. Hindmarsh, Preliminary Documentation of GEARIB: Solution of Implicit Systems of Ordinary Differential Equations With Banded Jacobian, UCID-30130, Lawrence Livermore Laboratory, Livermore, California, February 1976.
17. G. D. Byrne, The O.D.E. Solver EPISODE: Its Variants, and Their Use, Proceedings of the Topical Meeting on Computational Methods in Nuclear Engineering, Vol. 3, Williamsburg, Virginia, April 1979, pp. 8.17-8.52.
18. M. Kubicek, Dependence of Solution of Nonlinear Systems on a Parameter, ACM TOMS, Vol. 2, No. 1 (1976), pp. 98-107.
19. F. Aguilar, et al., STP: FORTRAN Library Routines for Steam Table Properties, NPGD-TM-514, Babcock and Wilcox, Lynchburg, Virginia, November 1979.
20. S. Thompson and P. G. Tuttle, DSTPGT: Automatic Software For the Numerical Solution of Differential Equations, NPGD-TM-521, Babcock and Wilcox, Lynchburg, Virginia, January 1980.
21. F. Aguilar and P. G. Tuttle, An R&D Proposal For New LOCA-Load Methods, NPGD-TRG-79-3, Babcock and Wilcox, Lynchburg, Virginia, May 1979.
22. P. A. Treventi, et al, REFLOD3: Multinode Core Reflooding Program, NPGD-TM-519, Rev. 3, Babcock and Wilcox, Lynchburg, Virginia, March 1981.
23. C. W. Gear, Numerical Initial Value Problems in Ordinary Differential Equations, Prentice-Hall, Englewood Cliffs, N.J. (1971).
24. V. Ransom, et al., RELAP4/MOD5: A Computer Program for Transient Thermal-Hydraulic Analysis of Nuclear Reactor and Related Systems, ANCR-Nureg-1335, Idaho National Engineering Laboratory, Idaho Falls, Idaho, September 1976.
25. F. Aguilar and S. Thompson, Non-Equilibrium Flashing Model For Rapid Pressure Transients, ASME Journal of Heat Transfer, to appear, (also ASME paper 81-HT-35).

REFERENCES CONT'D

26. M. B. Carver, Method of Lines Solution of Partial Differential Equations: Some Standard and Devious Applications, Proceedings of the Topical Meeting on Computational Methods in Nuclear Engineering, Vol. 3, Williamsburg, Virginia, April 1979, pp. 8.53 - 8.76.
27. L. F. Shampine, Limiting Precision in Differential Equation Solvers, Math. Comp., Vol. 28 (1974), pp. 141-144.
28. P. W. Ploch, et al., KAPP-VI: Digital Code For the Simulation of Primary System Transients in Pressurized Water Reactors, NPGD-12-1102640-02, Babcock and Wilcox, Lynchburg, Virginia, August 1980.
29. L. B. Bushard, Subroutine Packages for the Numerical Solution of Stiff Ordinary Differential Equations, ARC Report 8098, Babcock & Wilcox, February 1976.
30. K. W. Neves, Subroutine Packages for Numerical Solution of Ordinary Differential Equations - Initial Value Problems - Comparison Study, NPGD-TM-316, Rev. 1, Babcock & Wilcox, Lynchburg, Virginia, June 1975.
31. A. C. Hindmarsh, GEAR: Ordinary Differential Equation System Solver, UCID-30001, Rev. 3, Lawrence Livermore Laboratory, Livermore, California, December 1974.
32. G. D. Byrne and A. C. Hindmarsh, A Polyalgorithm for the Numerical Solution of Ordinary Differential Equations, ACM TOMS, Vol. 1, No. 1 (1975), pp. 71-96.
33. W. H. Enright and T. E. Hull, Test Results on Initial Value Methods for Non-Stiff Ordinary Differential Equations, SIAM J. Numer. Anal., Vol. 13, No. 6 (1976), pp. 944-961.
34. T. E. Hull, et al., Comparing Numerical Methods for Ordinary Differential Equations, SIAM J. Numer. Anal., Vol. 9, No. 4 (1972), pp. 603-639.
35. T. E. Hull, et al., Errata: Comparing Numerical Methods for Ordinary Differential Equations, SIAM J. Numer. Anal., Vol. 11, No. 3 (1974), p. 681.
36. F. T. Krogh, On Testing a Subroutine for the Numerical Integration of Ordinary Differential Equations, Journal of Association for Computing Machinery, Vol. 15, No. 3 (1968), pp. 390-401.
37. W. H. Enright, T. E. Hull, and B. Lindberg, Comparing Numerical Methods for Stiff Systems of ODEs, BIT, Vol. 15 (1975), pp. 10-48.
38. G. D. Byrne, et al., Comparative Test Results for Two O.D.E. Solvers - EPISODE and GEAR, ANL-77-19, Argonne, Illinois, March 1977.
39. J. W. Spellman and A. C. Hindmarsh, GEARS: Solution of Ordinary Differential Equations Having a Sparse Jacobian Matrix, UCID-30116, Lawrence Livermore Laboratory, Livermore, California, August 1975.

REFERENCES CONT'D

40. A. H. Sherman, Yale Sparse Matrix Package User's Guide, UCID-30114, Lawrence Livermore Laboratory, Livermore, California, August 1975.
41. D. J. Rodabaugh and S. Thompson, Low-Order A_0 -Stable Adams-Type Correctors, Journal of Computational and Applied Mathematics, Vol. 5, No. 3 (1979), pp. 225-233.
42. A. C. Hindmarsh, A Collection of Software for Ordinary Differential Equations, Proceedings of the Topical Meeting on Computational Methods in Nuclear Engineering, Vol. 3, Williamsburg, Virginia, April 1979, pp. 8.1 - 8.15.
43. M. B. Carver, "Efficient Handling of Discontinuities and Time Delays in Ordinary Differential Equations," Simulation 77, Montreaux, Switzerland (1977).
44. R. K. Brayton, et al., A New Efficient Algorithm for Solving Differential-Algebraic Systems Using Implicit Backward Differentiation Formulas, Proc. IEEE, Vol. 60, No. 1, January 1972, pp. 98-108.
45. J. W. Starnes, A Numerical Algorithm for the Solution of Implicit Algebraic-Differential Systems of Equations, Ph.D. Dissertation, University of New Mexico, Albuquerque, New Mexico (1976).
46. C. W. Gear, Simultaneous Solution of Differential Algebraic Equations, IEEE Transactions on Circuit Theory, Vol. CT-18, No. 1, January 1971, pp. 85-95.
47. I. Gladwell, Initial Value Routines in the NAG Library, ACM TOMS, Vol. 5, No. 4 (1979), pp. 386-400.
48. R. P. Brent, An Algorithm With Guaranteed Convergence for Finding a Zero of a Function, The Computer Journal, Vol. 14, No. 4 (1971) pp. 422-425.
49. IMSL: International Mathematical and Statistical Libraries, Inc., Edition 5, Houston, Texas (1975).
50. A. C. Hindmarsh, Linear Multistep Methods for Ordinary Differential Equations: Method Formulations, Stability, and the Methods of Nordsieck and Gear, UCRL-51186, Rev. 1, Lawrence Livermore Laboratory, Livermore, California, March 1972.
51. F. Brauer and J. A. Nohel, The Qualitative Theory of Ordinary Differential Equations, W. A. Benjamin, Inc., New York, 1969.
52. H. J. Stetter, Local Estimation of the Global Discretization Error, SIAM J. Numer. Anal., Vol. 8, No. 3, September 1971, pp. 512-523.
53. H. J. Stetter, Global Error Estimation in Adams PC Codes, ACM TOMS, Vol. 5, No. 4 (1979), pp. 415-430.

REFERENCES CONT'D

54. L. F. Shampine, Implementation of Implicit Formulas for the Solution of ODE's, SAND 79-0694, Sandia Corporation, Albuquerque, New Mexico, May 1979.
55. S. Thompson and P. G. Tuttle, User's Guide for Using the Automatic O.D.E. Solver DSTPGT, in preparation.
56. M. B. Carver, "In Search of a Robust Integration Algorithm for General Library Use: Some Tests, Results and Recommendations," Numerical Ordinary Differential Equations, Dept. of Computer Science, Univ. of Illinois at Urbana-Champaign, March 1979.
57. A. H. Sherman, et al., GEARS: A Package for the Solution of Sparse, Stiff Ordinary Differential Equations, UCRL-84102, Lawrence Livermore Laboratory, Livermore, California, March 1980.
58. S. C. Eisenstat, et al., Yale Sparse Matrix Package .. I. The Symmetric Codes, Research Report No. 112, Dept. of Computer Science, Yale University, 1977.
59. S. C. Eisenstat, et al., Yale Sparse Matrix Package .. II. The Non-symmetric Codes, Research Report No. 114, Dept. of Computer Science, Yale University, 1977.
60. R. F. Sincovec and N. K. Madsen, Software for Nonlinear Partial Differential Equations, ACM TOMS, Vol. 1, No. 3 (1975), pp. 232-260.
61. J. M. Hyman, Method of Lines Solution of Partial Differential Equations, Courant Institute of Mathematical Sciences, Report COO-3077-139, October 1976.
62. A. M. Loeb, A New User Oriented Subroutine for the Automatic Solution of Partial Differential Equations by the Method of Lines, Proceedings, 1974 Summer Computer Simulation Conference, SCI, LaJolla.
63. L. A. Larsen, Automatic Solution of Partial Differential Equations, UIU-CDCS-R-72-546, University of Illinois, Urbana, Illinois, 1972.
64. L. F. Shampine, What Everyone Solving Differential Equations Numerically Should Know, SAND 78-0315, Sandia Corporation, Albuquerque, New Mexico, March 1979.
65. K. W. Turner, KAPP-v2 Steam Generator Model Description, NPGD-26-1103941, Babcock & Wilcox, Lynchburg, Virginia, April 1980.
66. M. B. Carver, Pseudo Characteristic Method of Lines Solution of the Conservation Equations, Journal of Computational Physics, Vol. 35 (1980), pp. 57-76.

MULTIRATE INTEGRATION ALGORITHMS APPLIED TO STIFF SYSTEMS

IN NUCLEAR POWER PLANT SIMULATION

**P. V. Girijashankar, D. L. Hetrick,
W. N. Keepin, and O. A. Palusinski**

University of Arizona, Tucson, Arizona

P. G. Bailey

Electric Power Research Institute

The simulation of large engineering systems such as nuclear power plants involves the numerical integration of a large set of differential equations. The conventional numerical methods and solution techniques often used in these simulations can lead to difficulties due to stiffness that can arise in the system as the transient unfolds. A technique for partitioning the system of equations into two separate classifications and methods for integrating the variables in each class are presented. The methods are demonstrated for an auto-pilot model and a pressurized water reactor model of the St. Lucie Unit 2 PWR. The results indicate that these methods are able to yield considerable savings in computer CPU time. Savings up to as high as eight-to-one have been observed over conventionally used methods.

INTRODUCTION

The simulation of large engineering systems such as power plants of both nuclear and fossil type, aircraft systems, etc., has been an important activity for more than a decade. System modeling using the conservation laws of mass, momentum and energy in the case of mechanical and hydraulic systems and Kirchoff's voltage and current laws in the case of electrical systems is a first step in the simulation scheme. These laws when used to model systems such as pressurized water reactors, for example, result in a set of partial differential equations. Using some form of lumping, these are reduced to a set of coupled nonlinear ordinary differential equations. Conventional integration schemes such as Runge-Kutta-Merson,¹ Episode,² Gear,³ etc., while generally satisfactory, are highly unsuitable for some classes of problems. We will direct our attention to one such class of problem and refer to it as two-time-scale and oscillatory stiff.

The above class of problems having highly oscillatory solutions for a small subset of system variables requires small step size for the solution of all system equations when using the above integration techniques. Since the number of system variables that contribute to the oscillatory part of the solution is generally small compared to the total number of system variables, significant advantage could be gained by partitioning the system into (1) those variables which produce highly oscillatory solutions necessitating a small step size for integration and (2) the remaining variables which can be integrated by using a relatively large step size. We define the first subsystem as the fast subsystem, the second as the slow subsystem, the method of identifying and splitting the system into its slow and fast component parts as partitioning, and the scheme of integration of such partitioned system as multirate integration. It must be pointed out that a system which is not stiff at the start of integration could become stiff during integration, and that a system which has been partitioned in one way may need repartitioning during the course of integration.

We present two examples illustrating the above features of partitioning and multirate integration for the cases of (a) a thirteenth order autopilot model and (b) a thirtieth order pressurized water reactor system. Also, we will present a proposed scheme for monitoring the stiffness of the system to be used as an indicator in the partitioning of the system.

PARTITIONING OF SYSTEM EQUATIONS

In this section we will briefly review a scheme for identifying and partitioning a nonlinear system of equations into its fast and slow component parts. The scheme for identifying the slow and fast components of a linear system is due to O'Malley and Anderson⁴ and has been extended to nonlinear systems as will be presented below.

Consider a dynamic system described by

$$\dot{Z} = F(Z, U, t) \quad (1)$$

where Z and U are n and m -dimensional vectors respectively, and F denotes the vector dependence between Z and U . In the control literature, these are referred to as the state and control vector respectively. Around any operating trajectory (Z_0, U_0) , the system equations could be written as

$$\dot{z} = Az + Bu \quad (2)$$

where z and u are perturbations around (Z_0, U_0) and A and B are Jacobians with respect to Z and U respectively. We tacitly assume that the system equations(1) can be adequately represented by the linearized version, equation(2) for the purposes of determining the fast and slow parts of the system. It may be pointed out that the homogeneous part of the solution of equation(2) is governed by A while the forcing part is dictated by B . The total solution, therefore, is a summation of the homogeneous and its forcing part. If u , the forcing function is slowly varying as compared to the fast components of the homogeneous solution, then we can assume that the fast and slow components of the solution can be identified by an analysis of A .

Let us assume that the eigenvalues of the system matrix A can be separated into two sets W and S of order N_1 and N_2 respectively, such that eigenvalues $f_1 \in W$ and eigenvalues $s_j \in S$ and $|s_j| \ll |f_1|$. We refer to such systems as two-time-scale; the set W is the fast set of eigenvalues and the set S is the slow set. Later, we will further assume that the fast set also contains at least one complex conjugate pair with relatively large imaginary parts. This class of problems is referred to as two-time-scale and oscillatory stiff. (The scheme presented below works whether oscillations are present or not.)

We can rewrite equation(2) as

$$\begin{bmatrix} \dot{z}_1 \\ \dot{z}_2 \end{bmatrix} = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} \begin{bmatrix} z_1 \\ z_2 \end{bmatrix} + \begin{bmatrix} B_1 \\ B_2 \end{bmatrix} u \quad (3)$$

where z_1 and z_2 are n_1 and n_2 dimensional subvectors of z and the A 's and B 's are submatrices of A and B , respectively, having proper dimension. We can cast the A matrix into its Jordan form using the eigenvector matrix M and its inverse Q , thus,

$$J = QAM \quad (4)$$

It can be easily seen that if the transformation $P = Mz$ is used then the system equations (2) will be completely decoupled in the new state variable vector P and the homogeneous part of the solution is governed by the eigenvalues in the Jordan form of A . Thus, while accomplishing total decoupling the above transformation requires the evaluation of the complete set of eigenvalues and its eigenvectors. But as Golub and Wilkinson point out the smallest eigenvalues of a large ill-conditioned matrix cannot be calculated accurately and hence their eigenvectors are approximate at best. We use a slightly different approach which achieves partial decoupling and is all that is needed for partitioning. Only a very brief outline of the method is given below and the reader may refer to Reference 4 for details. We will rewrite equation (4) as follows

$$\begin{bmatrix} J_1 & 0 \\ 0 & J_2 \end{bmatrix} = \begin{bmatrix} Q_{11} & Q_{12} \\ Q_{21} & Q_{22} \end{bmatrix} \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} \begin{bmatrix} M_{11} & M_{12} \\ M_{21} & M_{22} \end{bmatrix} \quad (5)$$

where J_1 and J_2 are submatrices of J containing the fast and slow eigenvalues respectively and $(M_{11} M_{21})^T$ corresponds to the eigenvectors of the smaller set of n_1 eigenvalues that has been identified for partial decoupling. Various other submatrices in equation (5) can be easily identified.

Let us define the matrices L and T by the relations

$$L = -M_{21} M_{11}^{-1} \quad (6)$$

and

$$T = \begin{bmatrix} I_1 & 0 \\ L & I_2 \end{bmatrix} \quad (7)$$

We will further define a transformation of the $(z_1, z_2)^T$ state space into $(x, y)^T$ space by

$$\begin{bmatrix} x \\ y \end{bmatrix} = T \begin{bmatrix} z_1 \\ z_2 \end{bmatrix} \quad (8)$$

Using the transformation defined by equation (8), it can be shown⁴ that the system equation (3) will be transformed into

$$\begin{bmatrix} \dot{x} \\ \dot{y} \end{bmatrix} = \begin{bmatrix} \tilde{A}_{11} & A_{12} \\ 0 & \tilde{A}_{22} \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} B_1 \\ LB_1 + B_2 \end{bmatrix} u \quad (9)$$

where

$$\tilde{A}_{11} = A_{11} - A_{12} L \quad (10)$$

$$\tilde{A}_{22} = A_{22} + LA_{12}$$

Note that $\tilde{A}_{11}$ and $\tilde{A}_{22}$ contain the fast and slow eigenvalues respectively. Also, it has been shown⁴ that L satisfies the algebraic Ricatti equation (ARE)

$$LA_{11} + A_{21} - LA_{12}L - A_{22}L = 0 \quad (11)$$

It may be observed by looking at equation (9) that in the (x, y) state space there is partial decoupling between the fast and slow parts of the system. The evaluation of L needs only the eigenvectors corresponding to the fast n_1 eigenvalues. Even if this L is not accurate, the algebraic matrix Ricatti equation (11) could be used to formulate an iterative procedure to improve its accuracy. If order reduction were the main consideration then equation (9) could be used towards this end as has been done by O'Malley and Anderson. However, since our aim is to integrate the original nonlinear system, we will follow a slightly different approach as discussed below.

EXTENSION TO NONLINEAR SYSTEMS

We have assumed in the evaluation of the transformation matrix that the linearized system represents adequately the original nonlinear system. This is true at least in a small neighborhood of the operating point (Z_0, U_0) if the nonlinearities are severe, otherwise will be valid over a wider range. Thus, defining a transformation of the nonlinear variables

$$V = TZ \quad (12)$$

we can write equation (1) as

$$\dot{V} = TF(T^{-1}V, U, t) \quad (13)$$

where $V(t_0) = TZ(t_0)$

Denoting the first n_1 components of V by X and the remaining n_2 components by Y , we have

$$V^T = (X, Y)^T$$

and equation (1), in terms of X and Y will become

$$\begin{bmatrix} \dot{X} \\ \dot{Y} \end{bmatrix} = \begin{bmatrix} G(X, t, Y, U) \\ H(Y, t, X, U) \end{bmatrix} \quad (14)$$

where $X(0) = X_0$, $Y(0) = Y_0$ and G and H correspond to the first n_1 and n_2 right-hand-side components of the nonlinear equation. Using the analogy of the linear system presented earlier, we can denote the X and Y as fast and slow variables respectively. Once the fast and slow subsystems have been identified, these partitioned subsystems can be integrated using a multirate integration scheme as discussed in the next section.

As mentioned earlier, the nonlinearities produce a continually changing Jacobian matrix A as the solution unfolds. Two schemes for monitoring this change have been developed. One is a perturbation technique for estimating a residual in terms of A at a point in time and in terms of the nonlinear functions as they change for subsequent times. When this residual has grown beyond some pre-set value, the new A may be computed and a new partitioning determined as needed. Another scheme monitors the "zero block" in the matrix T . Nonlinearities cause the entries in this block to drift away from zero, and re-partitioning can be instituted when the norm of this sub-matrix exceeds some appropriate value. These results can be used in devising an automated re-partitioning scheme, and this is a subject for continuing research.

MULTIRATE INTEGRATION

In this section we will consider that the original system, equation (1), has already been partitioned into its slow and fast components as given by equation (14) and will present a scheme to integrate the slow system by a relatively large step size and the fast system by small step size. In the discussion below an s -stage explicit Runge-Kutta routine¹ will be used to illustrate the multirate scheme. It may be noted that many other integration routines could be used to implement the multirate schemes.

We have for the slow variable,

$$Y_{n+1} = Y_n + h \sum_{r=1}^s c_r k_r, \quad \sum_{r=1}^s c_r = 1 \quad (15)$$

$$k_1 = H(Y_n, t_n, X_n) \quad (16)$$

$$k_r = H(Y_n + h \sum_{i=1}^{r-1} b_{ri} k_i, t_{n+a_r}, X_{n+a_r}) \quad (17)$$

$$a_r = \sum_{i=1}^{r-1} b_{ri}, \quad r = 2, 3, \dots, s \quad (18)$$

Computing k_r successively requires X_{n+a_r} , hence we need to integrate the fast variable. The Runge-Kutta-Merson scheme has been used in the integration of the fast variable and with interpolation of $Y(t)$ based on values $Y_n, Y_{n+a_2}, \dots, Y_{n+a_{r-1}}$ which are computed successively using imbedded Runge-Kutta schemes of highest possible order⁶. Note that different integration schemes could be used to integrate the two subsystems.

For example, in order to compute

$$k_2 = H(Y_n + h a_2 k_1, t_{n+a_2}, X_{n+a_2}) \quad (19)$$

we have to determine X_{n+a_2} . Since k_1 is known, we can estimate using the Euler formula

$$Y_{n+a_2} = Y_n + a_2 h k_1$$

and this suggests the use of a linear polynomial

$$Y(t) = Y_n + (t - t_n) k_1 \text{ in the interval } t_n \leq t \leq t_n + a_2 h$$

to integrate the fast subsystem. The resulting X_{n+a_2} can now be used to evaluate k_2 by equation (19). Since k_2 is known, we can improve the estimate of Y_{n+a_2} using the improved Euler formula. Estimate Y_{n+a_3} using the linear polynomial between $t_n + a_2$ and $t_n + a_3$ as above. Construct a second order polynomial based on $Y_n, Y_{n+a_2}, Y_{n+a_3}$ to be valid for $t_n + a_2 \leq t \leq t_n + a_3$. Since X_{n+a_2} is known, the fast variable can be integrated from $t_n + a_2$ to $t_n + a_3$. We can use X_{n+a_3} to compute k_3 . The above scheme can be repeated to continue the integration. Thus, the previously evaluated $k_1, k_2, \dots, k_{r-1}$ have been used in updating $Y_{n+a_2}, Y_{n+a_3}, \dots, Y_{n+a_{r-1}}$ and hence the scheme is known as imbedded Runge-Kutta multirate integration. The above scheme can be used continuously to march ahead in time.

APPLICATIONS

We will demonstrate the usefulness of the scheme for two-time-scale and oscillatory stiff problems with the help of two examples. The first is an autopilot model having 13 differential equations while the second is a pressurized water reactor having 30 differential equations. In the first case, the system was stiff at the start of integration itself and hence partitioning was done at time $t=0$. In the second case, the system became stiff at time $t=93$ sec, during the course of integration, and hence the system was partitioned at 93 sec. In both cases, there was considerable savings in computer time as will be seen below.

AUTO-PILOT SYSTEM

The autopilot model is composed of four major subsystems, the vertical sensing unit, the amplifier, the servo-motor along with its pressure stabilizer, and the flight dynamics of the aircraft. Fig. 1 shows the various subsystems and their interconnections. The system equations and other associated details can be found in reference 7. It consists of 13 differential equations and a table look up. The eigenvalues of the system are as shown in Table 1. Since an analytical solution was not available for this system, a bench mark solution was obtained numerically by solving the system with very small values of relative error control parameter. Three different methods, (a) Runge-Kutta-Merson, (b) Gear's stiffly stable, and (c) Hindmarsh's Episode were used to yield agreement in solutions to a four-digit accuracy. For ease of reference we will denote the multirate integration scheme by P_0 and the methods of Runge-Kutta-Merson, Gear and Hindmarsh by P_1 , P_3 and P_{11} respectively. This notation corresponds to M_1 , M_3 , and M_{11} respectively, used in DAREP⁸ simulation software for these three integration methods.

The error criterion for all the schemes was selected so that it was not more than 1.0% during the entire run. The computer times involved for different schemes are shown in Table 2. Two representative fast variables and two representative slow variables from the multirate integration scheme are shown in Fig. 2. From the results of Table 2, it may be concluded that considerable savings in time could be obtained by partitioning and using multirate schemes to integrate such partitioned systems. One more interesting conclusion that can be drawn from Table 2 is that the Runge-Kutta-Merson method yields much shorter solution time as compared to either Gear's stiff or the Episode methods. This may be explained from the fact that stiff routines are good only when the stiffness is caused by a spread of real eigenvalues in a system not having oscillatory component in its fast components.

PRESSURIZED WATER REACTOR SYSTEM

The simulation of nuclear power reactors has gained more attention and importance since the Three Mile Island accident at Harrisburg, Pennsylvania. The PWR models are based on the conservation laws of mass, momentum and energy. These equations, together with the equations of state for water and steam, give a complete set of equations. The integration of such systems may not readily submit to efficient numerical treatment using readily available routines⁹, as we will see for this example.

The PWR model is based on the data for St. Lucie Unit 2, a Combustion Engineering plant in Florida. In building the reactor model, numerous assumptions have been made¹⁰. It may be pointed out that while some of these assumptions limit the use of the model to small transients only, the model should still retain the eigenvalue-eigenvector structure in the neighborhood of any operating point as compared to more detailed models. The plant as modelled is a one-loop representation of the two-loop plant and is shown in Fig. 3. It consists of a core, a pressurizer, a steam generator and a recirculating pump. The core has been modelled by a representative average channel, and the primary coolant is assumed to be under pressure such that no boiling is allowed in the model. The steam generator is simulated by a representative U-tube and the primary flow is divided into three sectors. The first sector extends from the inlet to the level of the boiling interface, the second sector from this interface to the boiling interface on the exit side, and the third sector from this interface on the exit side to the exit itself. Each sector is represented by an average temperature node and an outlet temperature node. No boiling is permitted for the primary liquid. The non-boiling region of the

secondary side of the steam generator is similarly represented by an outlet temperature, while the downcomer exit temperature is determined by treating the downcomer as a tank problem. The steam production rate is calculated by knowing the overall heat transfer coefficient and the difference between the average temperature in the second region and the saturation temperature. The pressurizer has been modelled to represent four states for the water in the bottom portion and steam in the top portion of the pressurizer. These states are (a) top superheated, bottom subcooled, (b) top superheated, bottom boiling, (c) top condensing, bottom subcooled, and (d) top condensing, bottom boiling. The recirculation pump (centrifugal type) has been modelled along the traditional lines of nondimensional homologous curve. This treatment allows the pump to traverse any of the four possible quadrants of pump operation. The final model consists of thirty differential equations and a large number of algebraic equations and table look ups for the steam properties. A digital simulator using these equations has been constructed and maintained as an input file to the DAREP translator. The response of some of the variables for a 10 percent step reduction in steam flow rate is shown in Fig. 4. The response appears to be consistent with plant transient results for similar plants.

Attempts were made to carryout the integration for the above transient initiator up to 100 sec. For all of the three methods of integration mentioned earlier, the first 93 secs of real time needed 1 sec of computed CPU time for each second of real time. At 93 sec, the system became stiff, and nearly 40 sec was needed for each second of real time. The first four largest eigenvalues of the Jacobian at two times (11 sec and 93 sec) are shown in Table 3. It can be seen that the imaginary part of the complex pair has increased by 25 fold in this interval. The equations have been partitioned into fast and slow parts with 3 and 27 differential equations in the fast and slow systems respectively. The three variables that have been selected for treatment as fast variables using the approach given above are neutron power, delayed neutron precursors, and the recirculation flow rate. The simulation was restarted at 93 sec using both the original and partitioned systems. The original system took 80 sec of CPU time for nearly 2 sec of system time, while the partitioned system reached 15 sec of system time in the same CPU time, indicating a 7 fold saving of CPU time, (table 4). The relative error for the most sensitive variable (neutron power) was less than 0.002 percent at 95 sec.

CONCLUSIONS

Certain two-time-scale problems can be advantageously integrated by splitting them into fast and slow component parts and using different step sizes for integrating the fast and slow subsystems. Characteristics of the linear version of the system model were used to identify the partitioned systems. Two nonlinear examples were used to illustrate the scheme. It was pointed out that a nonlinear system which was not stiff to start with could become stiff during integration, and a pressurized water reactor model for which this happened was presented. It was shown that a computer-time savings of 5-8 times over conventional integration schemes could be achieved by this procedure.

ACKNOWLEDGMENT

This research was supported in part by the Electric Power Research Institute as Project RP1381-1.

DOCUMENTATION

The results of this research have previously been documented at an ANS conference,¹¹ and in three formal EPRI research reports.^{10,12} These reports are available from the Research Reports Center, P.O. Box 50490, Palo Alto, California 94303.

REFERENCES

- ¹ Lambert, J. D., Computational Methods in Ordinary Differential Equations, Wiley, 1973.
- ² Hindmarsh, A. C. and Byrne, G. D., "EPISODE: An Experimental Package for Integration of Systems of Ordinary Differential Equations," University of California, Lawrence Livermore Laboratory, 1975.
- ³ Gear, C. W., Numerical Initial Value Problems in Ordinary Differential Equations, Prentice-Hall, 1971.
- ⁴ O'Malley, R. E., Jr., and Anderson, L. R., "Singular Perturbations, Order Reduction and Decoupling of Large Scale Systems," in Numerical Analysis of Singular Perturbation Problems, Henker, P. W. and Miller, T. T. H., Eds., Academic Press, 1979.
- ⁵ Golub, G. H. and Wilkinson, J. H., "Ill-Conditioned Eigensystems and Computation of the Jordan Canonical Form," SIAM Review, Vol. 18, No. 4, Oct., p 578, 1976.
- ⁶ Verner, Y. H., "Families of Imbedded Runge-Kutta Methods," SIAM J. Numerical Analysis, Vol. 16, No. 5, p 857, Oct., 1979.
- ⁷ Girijashankar, P. V., Hetrick, D. L., Keepin, W. N., and Palusinski, O. A., "A Technique for Efficient Simulation of Large Dynamic System," Joint Automatic Control Conference, Vol. 1, WP9-A, 1980.
- ⁸ Korn, G. A. and Wait, J. V., Digital Continuous System Simulation, Prentice Hall, 1977.
- ⁹ Girijashankar, P. V., Hetrick, D. L., Keepin, W. N., Palusinski, O. A., "An Efficient Simulation Technique for PWR Systems" Trans. ANS, Vol. 34, p 321, June 1980.
- ¹⁰ EPRI Research Project RP1381 Interim Report NP-1928, Electric Power Research Institute, July 1981.
- ¹¹ Girijashankar, P. V., Hetrick, D. L., Keepin, W. N., and Palusinski, O. A., "An Efficient Technique for the Simulation of Large Systems," Advances in Mathematical Methods for the Solution of Nuclear Engineering Problems, ANS-ENS International Topical Meeting, April 27-29, 1981, Munich, Germany.
- ¹² Hetrick, D. L., Girijashankar, P. V., Palusinski, O. A., Keepin, W. N., Roten, C. D., "Solution Methods for Simulation of Nuclear Power Systems," EPRI Research Project RP1381 Final Reports NP-2341, Vol. 1, and NP-2341-LD, Vol. 2, Electric Power Research Institute, April 1982.

NOMENCLATURE

A	Jacobian matrix with respect to X
A_{ij}	submatrix of A
a_r	constant in Runge-Kutta algorithm
B	Jacobian matrix with respect to Y
b_{ri}	constant in Runge-Kutta algorithm
e_i	ith column of identity matrix
E_n	local truncation error
ELB	Liquid height in the pressurizer
ELR	steam height in the pressurizer
EMSU	surge mass flow rate in and out of the pressurizer
EMSP	spray water rate into the pressurizer
F	vector describing state variable relationships
f_i	"fast" eigenvalue
I_i	1 x 1 Identity Matrix
J_i	Jordan blocks of order n_1 and n_2 , respectively
k_r	approximation to slope in Runge-Kutta algorithm
L	principal component of transformation matrix T
M	matrix of eigenvectors and generalized eigenvectors for A
M_{ij}	submatrix of M
n	order of the system
n_1	order of "fast" subsystem
n_2	order of "slow" subsystem
P	reactor primary pressure
P_i	notation for integration methods used
PSG	reactor secondary pressure
PWR	reactor core power
Q	matrix of the inverse of the eigenvector matrix M
Q_{ij}	submatrix of Q
r_i	coupling coefficient
s_i	"slow" eigenvalue
T	transformation matrix
t	time
TA1	average temperature in sector 1 of the steam generator
TA2	average temperature in sector 2 of the steam generator
TA3	average temperature in sector 3 of the steam generator
TCA	average temperature in core channel
TCCI	temperature of coolant at core inlet
TCCO	temperature of coolant at core outlet
TCRI	temperature of coolant at inlet of pressure vessel
TCRO	temperature of coolant at exit of pressure vessel
Td	temperature of secondary coolant at downcomer exit
TFA	average temperature of fuel in the core
TNBA	average temperature of water in the nonboiling length
TNBO	temperature of water at boiling boundary
T01	exit temperature of water in sector 1 of the steam generator
T02	exit temperature of water in sector 2 of the steam generator
T03	exit temperature of water in sector 3 of the steam generator
U	control vector
u	displacement of control vector from operating point U_0
W_{ex}	steam exhausted from the pressurizer safety valves
W_{FW}	feed valve flow rate
W_p	primary coolant flow rate
W_{RP}	recirculation flow rate
W_s	steam generation rate in the steam generator
W_{sg}	steam flow rate to the turbine
WSP	spray water rate into the pressurizer
X	transformed vector for the nonlinear system, fast variables

x transformed vector for the linearized system, fast variables
 x_n numerical approximation to solution in Runge-Kutta algorithm
 y transformed vector for the nonlinear system, slow variables
 y transformed vector for the linearized system, slow variable
 y_n numerical approximation to solution in Runge-Kutta algorithm
 z state vector
 z displacement of state vector from operating point z_0

Table 1. Eigenvalues of the Autopilot System

No.	Real part	Imaginary part
1.	-0.1235 E + 05	
2.	-0.4999 E + 04	
3.	-0.8401 E + 03	-0.8998 E + 04
4.	-0.8401 E + 03	+0.8998 E + 04
5.	-0.1000 E + 03	
6.	-0.2437 E + 02	-0.3200 E + 02
7.	-0.2437 E + 02	+0.3200 E + 02
8.	-0.8828 E + 01	
9.	-0.7687 E + 00	-0.4807 E + 01
10.	-0.7687 E + 00	+0.4807 E + 01
11.	-0.1809 E + 00	-0.3299 E + 00
12.	-0.1809 E + 00	+0.3299 E + 00
13.	-0.7640 E - 14	0.

Table 2. Comparison of Computer CPU Times for the Autopilot Model Using Different Integration Schemes (TMAX=0.5 secs)

Method	Error Control Parameters	Cyber 175 Run Time (sec)	Maximum % Errors					TH
			x	y	x ₁	x ₂	x ₃	
P1	EMAX-10 ⁻⁶	15.6	8.49x10 ⁻⁵	2.94x10 ⁻⁶	0.126	0.877	4.37x10 ⁻²	3.28x10 ⁻⁴
P3	EMAX-10 ⁻⁸	33.4	8.79x10 ⁻⁵	2.94x10 ⁻⁶	0.129	0.84	4.53x10 ⁻²	3.26x10 ⁻⁴
P11	EMAX-10 ⁻⁸	44.6	8.79x10 ⁻⁵	3.29x10 ⁻⁶	0.124	0.884	4.54x10 ⁻²	3.25x10 ⁻⁴
P0	EMAX-10 ⁻⁵	5.07	0.90	9.4x10 ⁻³	0.11	0.839	5.03x10 ⁻²	0.476

Table 3

Four Largest Eigenvalues of the Jacobian at 11.0 sec and 93.0 sec.

Time (sec)	Eigenvalue No.	Eigenvalue	
		Real Part	Imaginary Part
11.0	1	-29.12	0.0
	2	-100.68	4.07
	3	-100.68	-4.07
	4	-18.46	0.0
93.0	1	-146.99	0.0
	2	-61.41	108.49
	3	-61.41	-108.49
	4	-60.76	0.0

Table 4

Comparison of Computation Times After T = 93 sec.

Method	T(CPU)/T(REAL)
Episode	40
Multirate	7

Figure 1. Block Diagram Representation of the Autopilot System

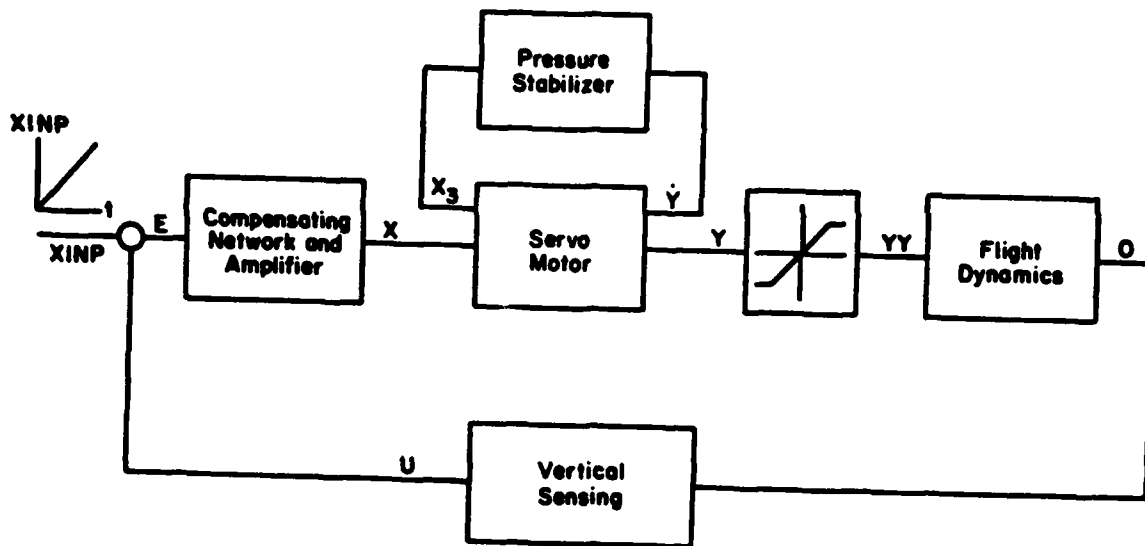


Figure 2. Two Representative Fast and Slow Variables of the Autopilot System

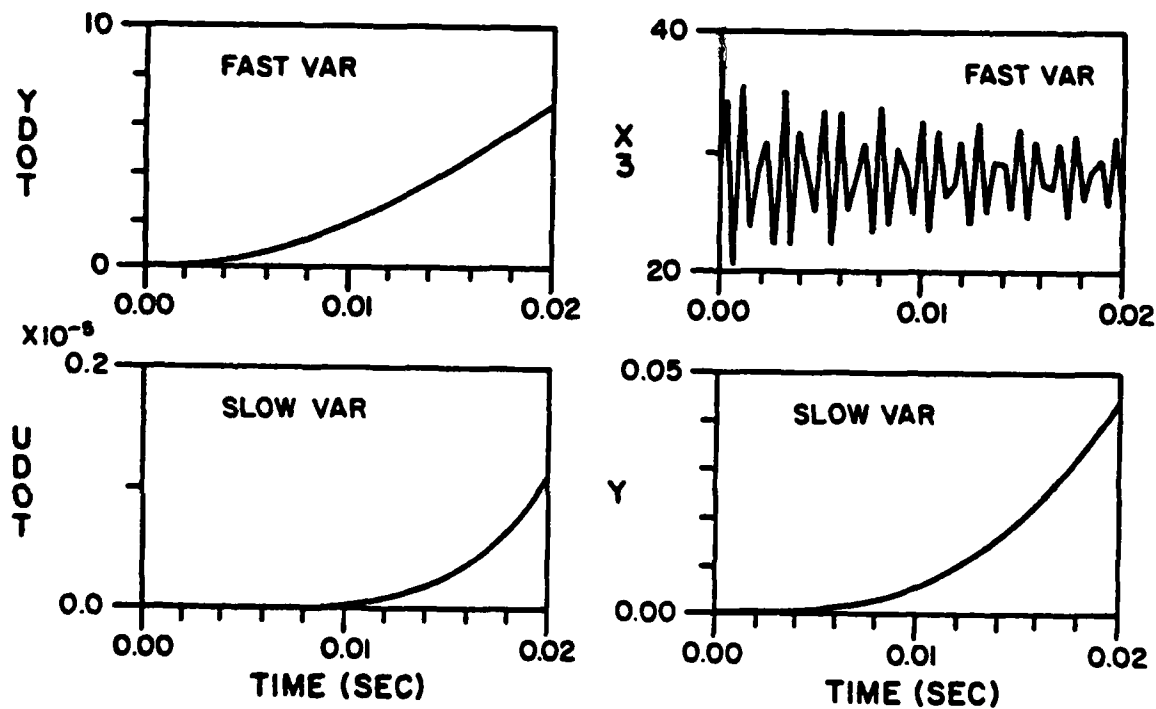


Figure 3. Schematic of the PWR System

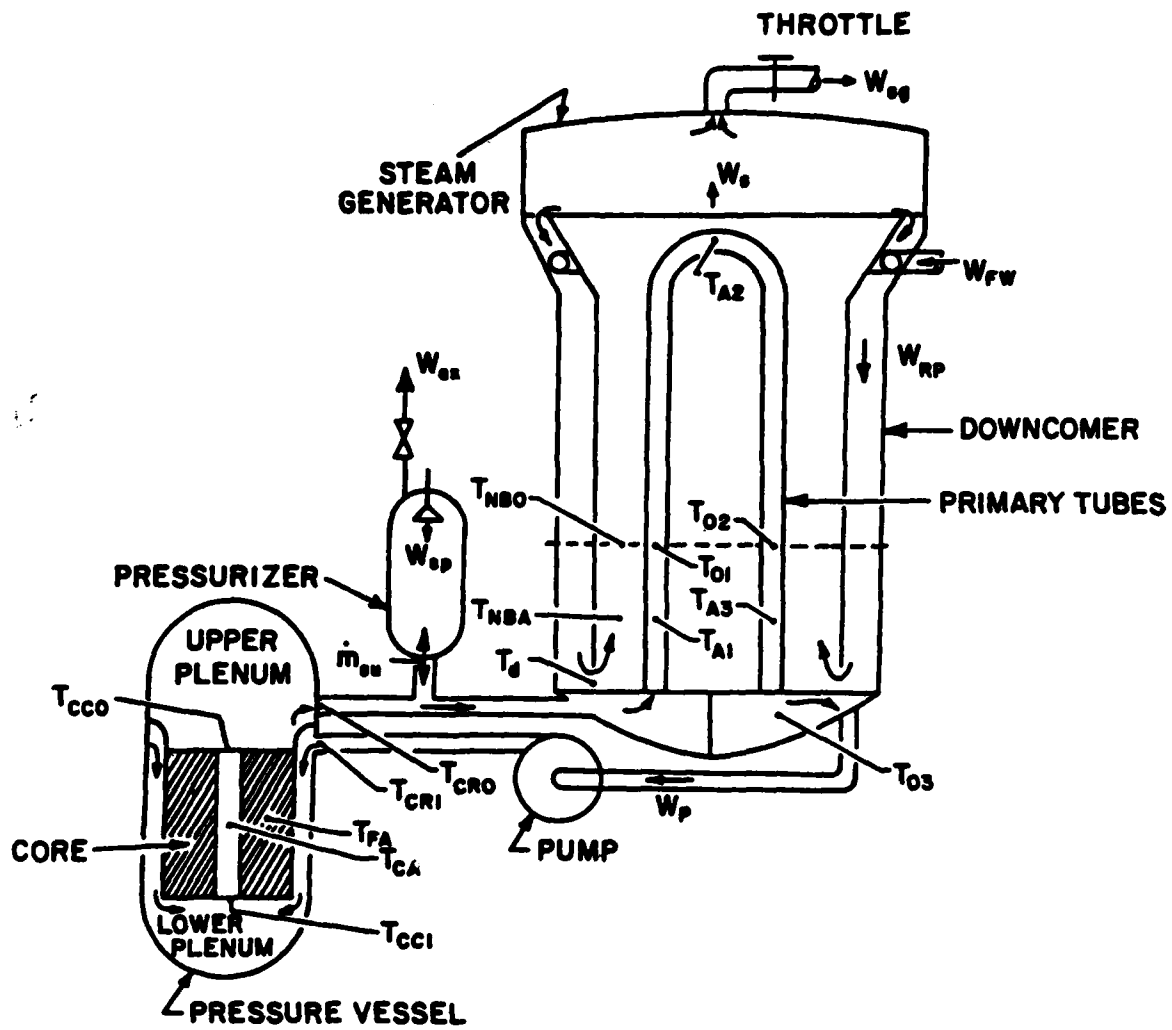
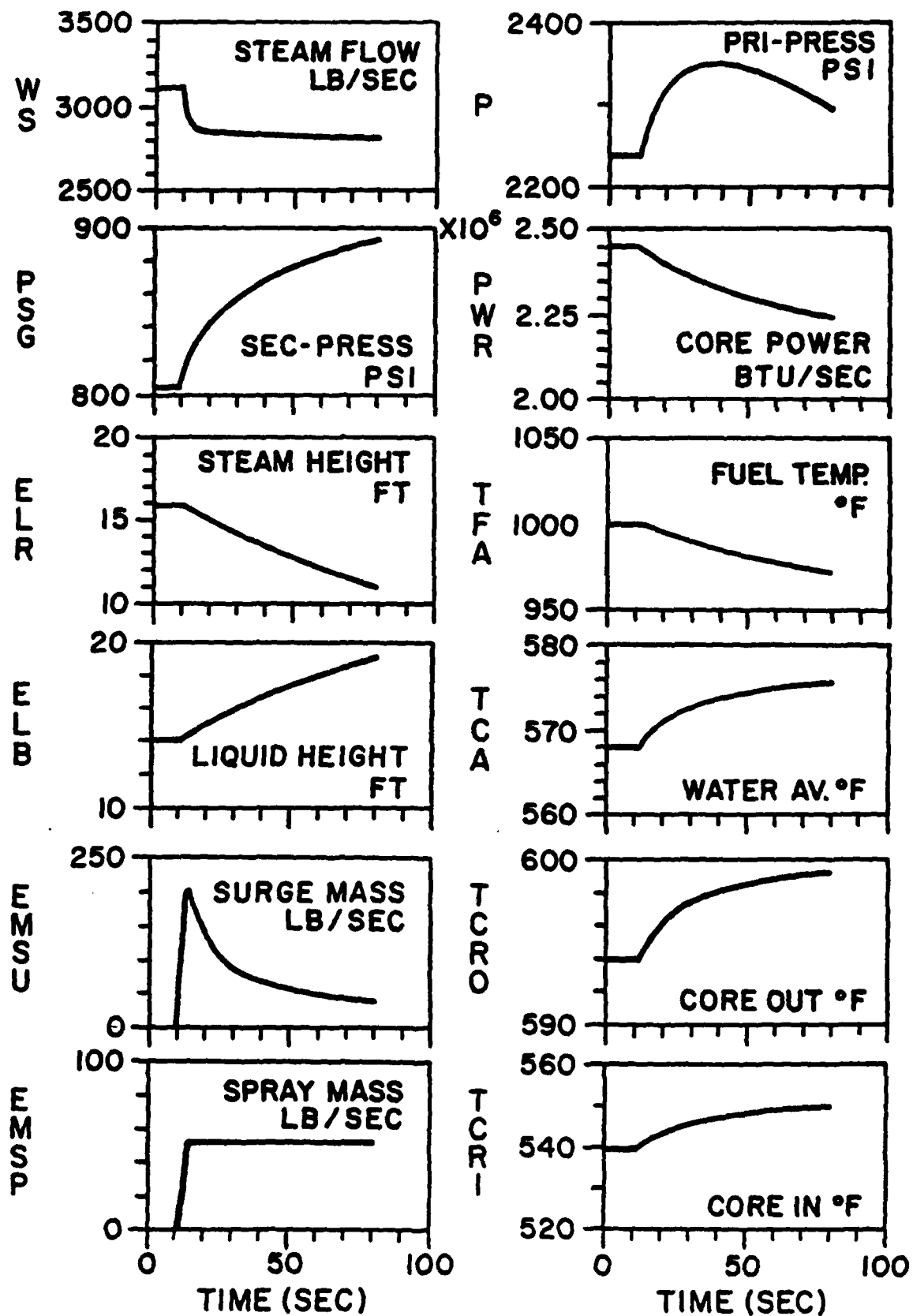


Figure 4. Response of the PWR System for a 10% Step Drop in Steam Flow Rate



APPLICATIONS OF PERTURBED FUNCTIONAL ITERATIONS TO
NONLINEAR STIFF INITIAL VALUE CHEMICAL KINETIC PROBLEMS

S. K. Dey
Senior NRC Research Associate
Computational Fluid Dynamics Branch
NASA-Ames Research Center
Moffett Field, CA 94035

Abstract

Numerical solution of nonlinear initial-value problems by a perturbed functional iterative scheme are discussed. The algorithm does not linearize the system and requires only the diagonal terms of the Jacobian. Some examples are presented.

1. Introduction

Stiff ODE's (ordinary differential equations) are of special significance in chemical kinetics, where decay of one component of the solution could happen much faster than other components. For numerical solution of these problems, it is generally found that some transient components which are negligible in comparison with other components, restrict the step sizes of the explicit methods to be very small in order that stability of the numerical solution may be maintained. Also, in order to understand the proper nature of the solution, the transient effects of the fast decaying terms should be revealed by the numerical process. This poses a very difficult problem. During the last decade, extensive research has been performed by mathematicians, engineers, and chemists to reveal the mechanism of stiff systems both mathematically and computationally.

Most conventional explicit methods such as Euler's method, Runge-Kutta schemes, Adams-Bashforth schemes, etc., require very small step sizes so that the algorithm may remain stable. Although some attempts have been made to extend the stability properties of explicit methods for special types of ODE's [11] by far the most common technique for solving stiff systems numerically is the use of implicit methods which requires the solution of simultaneous equations. In nonlinear cases Newton-type methods also require the evaluation of a Jacobian, a process that has a high arithmetic operation count. These requirements can be costly in both computer time and storage requirements.

Let us now pose the problem, and set up an implicit algorithm for the solution.

$$\frac{du}{dt} = f(u, t) \quad (1)$$

where $u = (u_1 \ u_2 \ \dots \ u_m)^T$; $f = (f_1 \ f_2 \ \dots \ f_m)^T$; and $u(0) = u_0$.

Approximating du/dt by a backward Euler scheme, we find

$$u^{n+1} - u^n = \Delta t f(u^{n+1}, t_{n+1})$$

or

$$u^{n+1} - u^n - \Delta t \cdot f(u^{n+1}, t_{n+1}) = 0 \quad (2)$$

This equation must be solved for u^{n+1} assuming that u^n the value of u at the previous time step is known. Then, we may represent this equation as

$$F(u^{n+1}) = 0 \quad (3)$$

If $U = u^{n+1}$, then it becomes

$$F(U) = 0 \quad (4)$$

which is a nonlinear system. After we have solved this system for U , we update the system by replacing u^n by the computed values of U and

again solve another nonlinear system of the form (4), etc. The process will continue until we reach the steady state which is here assumed to exist.

At present, there are several methods to solve nonlinear systems of the form (4). They may be arranged into two classes:

(i) Methods which require relatively small computer memory storage and operational counts per iteration but have slow rates of convergence (e.g., Point Jacobi, Point Gauss-Seidel, SOR schemes)

(ii) Methods which require relatively large computer memory storage and large operational counts per iteration but demonstrate fast rates of convergence (e.g., Newton's method).

In [1] a functional iterative scheme has been developed to solve nonlinear systems. It is obtained by perturbing nonlinear Gauss-Seidel iterations. The perturbation parameters stabilize the algorithm, control the mode of convergence, and, in some cases, can greatly speed up the rate of convergence. These parameters are essentially corrective factors for the iterates. Thus as the iterations converge, the perturbation parameters are all damped out. The present method has been applied to solve several stiff nonlinear ODE's. The results seemed to be encouraging in comparison with those obtained by conventional explicit schemes. The primary objective in this article is to show how to apply the scheme to nonlinear systems and give some practical demonstrations. A brief discussion regarding the details of the algorithm and its convergence properties is also included.

2. The Algorithm

Given a differential equation

$$h(\dot{x}, x, t) = 0, \quad x(t_0) = x_0$$

where $\dot{x} = dx/dt$ and t is the independent variable, if a difference approximation is made, it is reduced to a nonlinear system of the form

$$F_j(x_1, x_2, \dots, x_n) = 0, \quad j = 1, 2, \dots, n \quad (5)$$

This may be expressed as $F(x) = 0$, $x = (x_1 \ x_2 \ \dots \ x_n)^T$. We assume that a solution exists. Let us express (5) as:

$$x = G_0(x) \quad (6)$$

$G_0: DCR^n \rightarrow D$ (R^n = real n-dimensional space). Nonlinear Gauss-Seidel iterations to solve (6) may be expressed as:

$$x^{k+1} = G(x^{k+1}, x^k) \quad (7)$$

where $x^k = (x_1^k \ x_2^k \ \dots \ x_n^k)^T \in D$; x_j^k = value of x_j at the kth level of iteration; $G: D \times DCR^n \times R^n \rightarrow D$.

Let us perturb (7) and write:

$$x^{k+1} = \omega^k + G(x^{k+1}, x^k) \quad (8)$$

where $\omega^k = (\omega_1^k \ \omega_2^k \ \dots \ \omega_n^k)^T \in R^n$ is the perturbation parameter.

If the method converges, $\lim_{k \rightarrow \infty} x^k = x^*$ and $G(x^*, x^*) = x^*$, which implies:

Theorem: 1 A necessary condition so that (4) may converge is that for some norm

$$\lim_{k \rightarrow \infty} \|\omega^k\| = 0 \quad (9)$$

In the element form (8) may be expressed as:

$$x_j^{k+1} = \omega_j^k + G_j^{k+1,k} \quad (10)$$

where

$$G_j^{k+1,k} = G_j \left(x_1^{k+1}, x_2^{k+1}, \dots, x_{j-1}^{k+1}, x_j^k, \dots, x_n^k \right) \quad (11)$$

The algorithm has been derived in details in [1] and ω_j^k is computed by using the following equation:

$$\omega_j^k = \frac{G_j(G_j^{k+1,k}) - G_j^{k+1,k}}{1 - \partial_j G_j^{k+1,k}} \quad (12)$$

where $G_j^{k+1,k}$ is computed using (11),

$$G_j(G_j^{k+1,k}) = G_j(x_1^{k+1}, x_2^{k+1}, \dots, x_{j-1}^{k+1}, G_j^{k+1,k}, x_{j+1}^k, \dots, x_n^k)$$

and

$$\partial_j G_j^{k+1,k} = \left[\frac{\partial G_j}{\partial x_j} \right]_{x_1^{k+1}, x_2^{k+1}, \dots, x_{j-1}^{k+1}, G_j^{k+1,k}, x_{j+1}^k, \dots, x_n^k}$$

Thus in (12), for x_j^{k+1} , ω_j^k is computed in terms of quantities known a priori. The criterion for convergence is, if at some k ,

$$\max_j |\omega_j^k| < \epsilon \quad (13)$$

where ϵ is positive and arbitrarily small.

3. Convergence Analysis

Given a sequence of scalars $\{a_k\}$, $k = 1, 2, \dots$, a_k is called a D-element iff

$$\lim_{k \rightarrow \infty} a_1 a_2 \dots a_k = 0 \quad (14)$$

It is obvious that if $\forall k > K$, $|a_k| \leq \alpha < 1$ is satisfied.

Given a sequence of square matrices $\{A_k\}$ of the same type with variable elements, A_k is called a D-matrix iff

$$\lim_{k \rightarrow \infty} A_k A_{k-1} \dots A_1 = 0 \quad (15)$$

Obviously if $A_k = A \forall k$, A_k is a D-matrix iff A is a convergent matrix.
 A sufficient condition that A_k is a D-matrix is, for some norm

$$\|A_k\| \leq \alpha < 1 \quad (16)$$

The necessary and sufficient condition so that A_k is a D-matrix is

$$\max_{i,j} \left| a_{i,j}^{k,k-1, \dots 1} \right| < \epsilon \quad (17)$$

where ϵ is positive and arbitrarily small and $a_{i,j}^{k,k-1, \dots 1}$ = an element of the product matrix

$$A_k A_{k-1} \dots A_1$$

An Example: Let $A_1 = \text{diag} (\alpha_1, a_1, b_1)$

$$A_2 = \text{diag} (a_2, \alpha_2, b_2), \quad A_3 = \text{diag} (a_3, b_3, \alpha_3)$$

$$A_4 = \text{diag} (\alpha_4, a_4, b_4), \quad \text{etc.}$$

Let $|\alpha_j| \leq \alpha < 1 \forall j > N$ and a_j 's and b_j 's are bounded and chosen arbitrarily and a finite number of them are greater than 1 in absolute value. Then

$$\lim_{n \rightarrow \infty} A_n A_{n-1} \dots A_1 = 0 \quad (18)$$

Thus each A_j is a D-matrix, however none of these matrices is a convergent matrix.

Let us consider an iterative scheme

$$x^{k+1} = G(x^{k+1}, x^k), \quad k = 1, 2, \dots$$

Let for a given $x^* \in D$,

$$G(x^{k+1}, x^k) - G(x^*, x^*) = A_k(x^{k+1} - x^*) + B_k(x^k - x^*) \quad (19)$$

Both A_k and B_k are matrices with variable elements which change as k changes.

G is called a D-mapping on $D \times D$ iff $(I - A_k)^{-1}$ exists

$\forall k$ and $(I - A_k)^{-1} B_k$ is a D-matrix.

The following theorem may now be proved.

Theorem: 2 In (8) if G is a D-mapping and $x = x^* \in D$, then the scheme of iteration converges to x^* if $\lim_{k \rightarrow \infty} |\omega_k| = 0$. Furthermore, if $\rho \left\{ |(I - A_k)^{-1} B_k| \right\} < 1, \forall k > K, x^*$ is the unique root in D ($\rho(A)$ = spectral radius of a matrix A).

Proof:

$$\begin{aligned} x^{k+1} - x^* &= \omega^k + G(x^{k+1}, x^k) - G(x^*, x^*) \\ &= \omega^k + A_k(x^{k+1} - x^*) + B_k(x^k - x^*) \end{aligned} \quad (20)$$

Let $C_k = (I - A_k)^{-1}$ and $E_k = (I - A_k)^{-1} B_k$. Then from (20)

$$\begin{aligned} x^{k+1} - x^* &= C_k \omega^k + E_k (x^k - x^*) \\ &= \sum E_k E_{k-1} \dots C_j \omega^j + E_k E_{k+1} \dots E_1 (x^0 - x^*) \\ &= (E_k E_{k-1} \dots E_{k_0+1}) \sum_{j=1}^{k_0} (E_{k_0} E_{k_0-1} \dots E_{j+1}) C_j \omega^j \\ &\quad + \sum_{j=k_0+1}^k (E_k E_{k-1} \dots E_{j+1}) C_j \omega^j + E_k E_{k-1} \dots E_1 (x^0 - x^*) \end{aligned}$$

Now $\lim_{k \rightarrow \infty} |\omega^k| = 0$ implies for some $k > k_0 + 1, |\omega^k| < \epsilon$ (ϵ is positive and arbitrarily small). Thus each element ω_j^k is arbitrarily small in absolute value. Also, E_k being a D-matrix

$$\lim_{k \rightarrow \infty} E_k E_{k-1} \dots E_1 = 0$$

and $\lim_{k \rightarrow \infty} E_k E_{k-1} \dots E_{k_0+1} = 0$. Hence, $|x^{k+1} - x^*| < \epsilon$ which establishes convergence.

To prove uniqueness we assume $y^* \in D$ is another root. Then

$$x^* - y^* = G(x^*, y^*) - G(y^*, y^*)$$

This gives

$$x^* - y^* = E_*(x^* - y^*)$$

where $\lim_{k \rightarrow \infty} E_k = E_*$. Thus, $(I - |E_*|)|x^* - y^*| \leq 0$. Since $\rho(|E_k|) < 1 \forall k > K$, $\rho(|E_*|) < 1$. Hence, by Neumann's lemma, $(I - |E_*|)^{-1}$ exists and is non-negative. Hence $|x^* - y^*| \leq 0$ which implies $x^* = y^*$.

More discussions on D-matrices and D-mappings are given in [15].

4. Some Comparison with Other Methods

Implicit methods for solving differential equations usually demonstrate better stability properties than most explicit algorithms. When a system of nonlinear differential equations is reduced to a nonlinear difference system, various iterative schemes are available for numerical solution. The simplest methods involve functional iterative schemes (Jacobi or Gauss-Seidel iterations) which have generally very slow rates of convergence. Newton's method has a quadratic rate of convergence which may be expressed as:

$$\|x^{k+1} - x^*\| = \alpha \|x^k - x^*\|^2$$

This immediately implies that even if $0 < \alpha < 1$, if $\|x^0 - x^*\| > 1$, the method could fail. Thus one basic requirement is that the initial guess x^0 should be sufficiently close to the actual root x^* . This implies $\|x^0 - x^*\| < 1$. For many initial-value problem this condition could be satisfied. However the main disadvantage is that at each k level of iteration and for each iterate x^k a unique Jacobian must be computed which is not quite practical and for large systems such computations are very expensive.

The present method may be expressed as a combination of nonlinear Gauss-Seidel iterations and Lieberstein's method as follows:

$$x_j^{k+1/2} = G_j(x_1^{k+1}, x_2^{k+1}, \dots, x_{j-1}^{k+1}, x_j^k, \dots, x_n^k)$$

$$x_j^{k+1} = x_j^{k+1/2} - \frac{f_j(x_1^{k+1}, x_2^{k+1}, \dots, x_{j-1}^{k+1}, x_j^{k+1/2}, x_{j+1}^k, \dots, x_n^k)}{[\partial f_j / \partial x_j]_{x_1^{k+1}, x_2^{k+1}, \dots, x_{j-1}^{k+1}, x_j^{k+1/2}, x_{j+1}^k, \dots, x_n^k}}$$

$j = 1, 2, \dots, n; k = 0, 1, 2, \dots$

Thus if r_1 is the rate of convergence of Gauss-Seidel iteration and r_2 is the rate of convergence of Lieberstein's method, the rate of convergence of the present scheme is $r_1 r_2$. Also, if at each level Lipschitz condition is satisfied, combining both steps it may be seen:

$$\|x^{k+1} - x^*\| = \beta^2 \|x^k - x^*\|$$

where $0 \leq \beta < 1$. Since β^2 goes to zero at a quadratic speed, the method converges even if $\|x^0 - x^*\| > 1$. For various nonlinear systems this global convergence property has been verified computationally [14].

Furthermore, whereas Newton's method requires $(n^2 + n)$ functionals to be computed at each iteration level, only $3n$ functionals are computed at each iteration level by the present method. Also, to compute a Jacobian $(n \times n)$, n^2 elements must be stored and a total number of N arithmetic operations (addition, subtraction, multiplication) must be done where $N = n![2 + 1/2! + 1/3! + \dots + 1/(n-1)!]$. In comparison with this the present method computes and stores n -diagonal elements of the Jacobian.

There are certain interpolatory methods available to solve stiff ODE's [9]. These algorithms, developed by Certaine and Jain reduce differential equations into integral equations which were integrated by approximating the integrands by interpolation polynomials over the range of integration. Although both of these methods have high accuracy and are A-stable, for large systems they are not practical.

The present method, having a simple algorithm, is applicable to most nonlinear systems in general. It can be applied to nonlinear integral equations or integro-differential equations. Some applications with regard to this are given in [12].

5. Applications

Magee and Chatterjee [4] developed models on chemical kinetics and sought solutions of these problems which should be time-accurate. These models are nonlinear and consist of a sequence of stiff ODE's. Some of these chemical species grow from a concentration of 10^{-30} moles/liter at $t = 10^{-6}$ sec to 10^{-8} moles/liter at $t = 1.0$ sec; whereas by that time certain species which have values of the order 10^{-15} moles/liter at $t = 10^{-6}$ sec grow up to 10^{-8} moles/liter at $t = 1.0$ and stay almost unchanged. Since the equations are stiff, in the transient process numerical effects of some components of the solution which decay much faster in comparison with other components are quite difficult to capture in a computational process. This leads to some inaccuracies in the solution which eventually causes instabilities. For example, if the concentration of a species is found to be 10^{-35} mole/liter but whose true concentration is 10^{-30} moles/liter, this causes severe instabilities. In such computations which are very "sensitive" with regard to small errors, principles of perturbations could be applied in order to stabilize the algorithm. Indeed this was the finding when two distinct chemical kinetics problems were solved by the method. Let us consider them.

Models representing irradiation of water with γ -rays have been considered. Two conditions for water were taken (i) Acid Water and (ii) Pure Water.

Model: 1 Irradiation of Acid Water

The γ -radiation energy is absorbed with the creation of molecular (H_2 , H_2O_2) and radical (H , OH) products which may be treated as if formed

homogeneously in the system. The mechanisms related to formations of these species are described in [5,6]. Seven species created by the radiation participate in thermal reactions, summarized in Table 1. If the irradiation is continuous at the rate of 100 I's electron volts per liter per sec, the differential equations which describe the concentration changes are given in Table 2. These equations are stiff. Here creation terms give concentrations of created species in terms of moles per liter per second.

The system described in Table 2 may be expressed as:

$$\frac{du}{dt} = f(u)$$

where $u = [(H)(OH)(H_2O)(H_2)(H_2O_2)(HO_2)(O_2)]^T$ and $f = [f_1 f_2 \dots f_7]^T$ where f_1 corresponds to the right side of the first equation, f_2 corresponds to the right side of the second equation etc. Applying a backward Euler difference scheme we get

$$u^{n+1} - u^n - hf(u^{n+1}) = 0, \quad h = \Delta t$$

This nonlinear system for u^{n+1} may now be expressed as:

$$F(U) = 0$$

where $U = u^{n+1}$ which could be put in the form as

$$U = G_0(U)$$

where $G_0(U) = U + \alpha \cdot F(U)$ and $\alpha = \text{diag}(\alpha_1, \alpha_2, \dots, \alpha_7)$. In general, $0 < \alpha_j \leq 1$. This is done primarily because, if for any j ($j = 1, 2, \dots, 7$), $|\partial G_{0j} / \partial U_j|$ becomes much larger than 1, the properties of D-Mappings will be violated and thus the method is very likely to fail.

A numerical solution is found for continuous irradiation at constant rate which start with the molecular concentration of H_2 , H_2O_2 and O_2 equal to zero. For this case (irradiated and water) the radical concentrations

approach "stationary values" very quickly and as known from the chemical nature of the system that H_2 and H_2O_2 are destroyed in a chain reaction, these concentrations which build up linearly at first, approach stationary values on a longer time scale. Computational results showed these behaviors of various concentrations of this complicated chemical process. Figures 1, 2, 3 show these results for various values of I . (More results on this project will be published from Lawrence Berkeley Lab.) From "zero" values, H and OH quickly approached 2×10^{-8} and 2.5×10^{-8} , respectively, in 0.2 sec and then slowly decreased attaining "stationary values" at time equal to 1 sec. However, H_2 , H_2O_2 , HO_2 and O_2 attain their stationary values after almost 15 sec.

Chatterjee and Magee [4] applied an implicit second-order Runge-Kutta method to solve this problem and recorded its failure.

Model: 2 Irradiation of Pure Water

The treatment of pure water is somewhat more complicated simply because it involves more species. Here H^+ is so low in concentration that the hydrated electrons are not converted into H atoms before the trade reactions occur and charged species must be treated explicitly. There are eleven equations giving the rates of concentrations of eleven species. In Table 3, the thermal reactions and in Table 4 the differential equations of the system are given. Continuous irradiation starting with zero decompositions for all concentrations of species excepting $H_2O = 55$ moles/liter, $H_3O^+ = 1.0E - 07$ moles/liter and $OH^- = 1.0E - 07$ moles/liter was done. At this stage, the ODE's were represented as integral equations which were approximated by trapezoidal rule. The method virtually failed when the derivatives were approximated by backward Euler's difference formula. In the code a very stringent convergence criterion, namely,

$$WMAX = \max_j |\omega_j^k| < 10^{-20}$$

was used. Since this should give a very high order of accuracy, it was felt that such a criterion is necessary in order to get an almost perfect "mass balance" and "charge balance" both of which should be (theoretically) equal to "zero" at all times. In submicroseconds, the radical species like H, OH, e_{Aq}^- etc. grow much more rapidly than the molecular species O_2 , O_2^- , H_2 , H_2O_2 . These have strong impact upon the mathematical model for the subsequent chemical yields. Thus $\Delta t = 10^{-8}$ was chosen initially. It was noticed that while all other species were growing, O_2 and O_2^- stayed zero up to 5×10^{-8} sec. Then O_2^- grew faster and $O_2^- > O_2$ at all time levels. The special properties of these species which possibly have some strong impact upon the solution as time increases, cannot be detected if a large time step is used. The figure 4 shows the behavior of various species.

6. Discussions

From equation (12) it is clear that if $a_j G_j^{k+1,k} = 1$, the method fails. It has been proved and demonstrated in [14], that the perturbation parameters ω_j^k stabilize the algorithm and speed up the rate of convergence provided the Jacobian of the matrix representation of the system has nonzero diagonal elements (see the Appendix). When diagonal terms are null, artificially they are brought in and are damped out as convergence is approached. This may be called nonlinear scaling. However, if diagonal elements are zeros and nonlinear scaling is not performed the rate of convergence is significantly decreased and the algorithm could even be destabilized. This was found with regard to solution of several problems. The essential strength of the method lies in having functionals with diagonal nonlinearity. Fortunately, the set of equations given in Table 2 and Table 4 fulfill these requirements.

By computer experimentation it has been found that for nonlinear systems having multiple roots, the present scheme is not quite effective.

Although on a trial basis this method can be applied to solve a nonlinear system, it may be useful to apply the properties of D-Mappings to the functionals G_j in order that the method may be applied successfully [8,15]. Such an analysis is generally quite complicated.

7. Concluding Remarks

This is a preliminary report. The objective with regard to the solution of the chemical kinetic problem was primarily to study the transient growths of the species, ^{just} not the final equilibrium values which were somewhat known to the chemists [4] through experimental data. With this regard, the present method served the purpose of the researchers very well.

Acknowledgement

Suggestions received from Dr. Lomax and Dr. Vinokur for betterment of this article are acknowledged with thanks.

My ten-year old son Charlie Dey wrote codes in TR580 color computers and worked with great dedication on the chemical kinetic problem.

The graphs were all drawn at Lawrence Berkeley Lab. Dr. Chatterjee and Dr. Magee of LBL explained the nature of the chemical reaction problems to me.

This research was partially funded by the Faculty Research Council of Eastern Illinois University and the National Research Council, Washington, D. C.

References

1. Dey, S. K.: Numerical Solution of Nonlinear Implicit Finite Difference Equations of Flow Problems by Perturbed Functional Iterations. Computational Fluid Dynamics (Edited by W. Kollmann), Hemisphere Publishing Co., New York, 1980.
2. Dey, S. K.: Numerical Instabilities of Nonlinear Partial Differential Equations, CFT 7800/SKD/IE. von Karman Inst. for Fluid Dynamics, Rhode-St-Gen., Belgium, 1978.
3. Bui, T. D.: Some A-Stable and L-Stable Method for Numerical Integration of Stiff Ordinary Differential Equations. J. Asso. Comp. Mach., Vol. 2, No. 3, 1979.
4. Chatterjee, A. & Magee, J.: Lawrence Berkeley Lab., private communication.
5. Chatterjee, A. & Magee, J.: Theory of the Chemical Effects of High-Energy Electrons. J. Phys. Chem., 82, 1978.
6. Chatterjee, A. & Magee, J.: A Spur Unfolding Model for the Radiolysis of Water. Rad. Phys. and Chem., 15, 1980.
7. Allen, A. O.; Hochanadel, C. J.; Ghormley, J. A. and Davis, T. W.: Decomposition of Water and Aqueous Solutions Under Mixed Fast Neutron and Gamma Radiation. J. Phys. Chem., 56, 1952.
8. Dey, S. K.: Analysis and Applications of a Perturbed Iterative Scheme to Nonlinear Discretized Flow Models. Proc. Intl. Cong. Num. Methods for Eng. (Dunod), Paris, France, Dec. 1980.
9. Miranker, W. L.: Numerical Methods for Stiff Equations. D. Reidel Publ. Co., Boston, 1981.
10. Willoughby, R. A. (Edited): Stiff Differential Systems. Proc. Intl. Conf. Stiff Diff. Wildbad, West Germany, Oct. 1973. Plenum Press, New York, 1974.

11. Lomax, H.: On the Construction of Highly Stable Explicit Numerical Methods for Integrating Coupled Ordinary Differential Equations with Parasitic Eigenvalues. NASA TN D-4547, April 1968.
12. Dey, S. K.: Finite-Difference Solution of Boundary-Layer Equations. Num. Heat Trans., Vol. 3, 1980.
13. Hindmarsh, A. C. and Byrne, G. D.: Applications of Episode: An Experimental Package for the Integration of Systems of Ordinary Differential Equations. Num. Meth. for Diff. Systems. Edited by L. Lapidus and W. E. Schiesser. Academic Press, New York, 1976.
14. Dey, S. K.: Perturbed Iterative Solution of Nonlinear Equations with Applications to Fluid Dynamics. J. Comp. Appl. Math., Vol. 3, No. 1, 1977.
15. Dey, S. K.: Convergence Analysis of a Perturbed Iterative Scheme for Solution of Nonlinear Systems. Intl J. Math. & Math. Sc., Vol. 4, No. 2, 1981.

Table 1: Reactions in Irradiated Water (Acid)

Reactions	Reaction rate constant, l/(mol s)
A. Recombination of Primary Radicals	
1. $H + H \rightarrow H_2$	1×10^{10}
2. $H + OH \rightarrow H_2O$	2.4×10^{10}
3. $OH + OH \rightarrow H_2O_2$	4×10^9
B. Reactions of Radicals and Product Molecules	
4. $H + H_2O_2 \rightarrow H_2O + OH$	1×10^8
5. $OH + H_2O_2 \rightarrow H_2O + HO_2$	5×10^7
6. $OH + H_2 \rightarrow H_2O + H$	6×10^7
7. $HO_2 + H \rightarrow H_2O_2$	1×10^{10}
8. $HO_2 + OH \rightarrow H_2O + O_2$	1×10^{10}
9. $HO_2 + HO_2 \rightarrow H_2O_2 + O_2$	2×10^6
10. $H + O_2 \rightarrow HO_2$	1×10^{10}

Table 2: Differential Equations for Transient Species
and Radiation Products in Irradiated Water (Acid)

$\frac{d}{dt} (H)$	$= 3.71I - 2k_1(H)^2 - k_2(H)(OH) - k_4(H)(H_2O_2) + k_6(OH)(H_2) - k_7(HO_2)(H) - k_{10}(H)(O_2)$
$\frac{d}{dt} (OH)$	$= 2.95I - k_2(H)(OH) - 2k_3(OH)^2 + k_4(H)(H_2O_2) - k_5(OH)(H_2O_2) - k_6(OH)(H_2) - k_8(HO_2)(OH)$
$\frac{d}{dt} (H_2O)$	$= -4.51I + k_2(H)(OH) + k_4(H)(H_2O_2) + k_5(OH)(H_2O_2) + k_6(OH)(H_2) + k_8(HO_2)(OH)$
$\frac{d}{dt} (H_2)$	$= 0.40I + k_1(H)^2 - k_6(OH)(H_2)$
$\frac{d}{dt} (H_2O_2)$	$= 0.78I + k_3(OH)^2 - k_4(H)(H_2O_2) - k_5(OH)(H_2O_2) + k_7(HO_2)(H) + k_9(HO_2)^2$
$\frac{d}{dt} (HO_2)$	$= k_5(OH)(H_2O_2) - k_7(HO_2)(H) - k_8(HO_2)(OH) - 2k_9(HO_2)^2 + k_{10}(H)(O_2)$
$\frac{d}{dt} (O_2)$	$= k_8(HO_2)(OH) + k_9(HO_2)^2 - k_{10}(H)(O_2)$

The range of integration is from 0 sec to 20 sec.

Table 3: Reactions in Irradiated Neutral Water

Reactions	Reaction rate constant $\ell/(\text{mol s})$
A. Recombination of Primary Radicals	
1. $\text{H} + \text{H} \rightarrow \text{H}_2$	1×10^{10}
2. $\text{e}^-_{\text{Aq}} + \text{H} \rightarrow \text{H}_2 + \text{OH}^-$	2.5×10^{10}
3. $\text{e}^-_{\text{Aq}} + \text{e}^-_{\text{Aq}} \rightarrow \text{H}_2 + 2\text{OH}^-$	6×10^9
4. $\text{e}^-_{\text{Aq}} + \text{OH} \rightarrow \text{OH}^-$	3×10^{10}
5. $\text{H} + \text{OH} \rightarrow \text{H}_2\text{O}$	2.4×10^{10}
6. $\text{OH} + \text{OH} \rightarrow \text{H}_2\text{O}_2$	4×10^9
7. $\text{H}_3\text{O}^+ + \text{e}^-_{\text{Aq}} \rightarrow \text{H}$	2.3×10^{10}
8. $\text{H}_3\text{O}^+ + \text{OH}^- \rightarrow \text{H}_2\text{O}$	3×10^{10}
B. Reactions of Radicals with Product Molecules	
9. $\text{H} + \text{H}_2\text{O}_2 \rightarrow \text{H}_2\text{O} + \text{OH}$	1×10^8
10. $\text{e}^-_{\text{Aq}} + \text{H}_2\text{O}_2 \rightarrow \text{OH} + \text{OH}^-$	1.2×10^{10}
11. $\text{OH} + \text{H}_2\text{O}_2 \rightarrow \text{H}_2\text{O} + \text{HO}_2$	5×10^7
12. $\text{OH} + \text{H}_2 \rightarrow \text{H}_2\text{O} + \text{H}$	6×10^7
13. $\text{HO}_2 + \text{H} \rightarrow \text{H}_2\text{O}_2$	1×10^{10}
14. $\text{e}^-_{\text{Aq}} + \text{O}_2 \rightarrow \text{O}_2^-$	1.9×10^{10}

Table 3: Reactions in Irradiated Neutral Water
(cont.)

Reactions	Reaction rate constant l/(mol s)
B. Reactions of Radicals with Product Molecules (cont.)	
15. $\text{HO}_2 + \text{OH} \rightarrow \text{H}_2\text{O} + \text{O}_2$	1×10^{10}
16. $\text{HO}_2 + \text{HO}_2 \rightarrow \text{H}_2\text{O}_2 + \text{O}_2$	2×10^6
17. $\text{H} + \text{O}_2 \rightarrow \text{HO}_2$	1×10^{10}
18. $\text{O}_2^- + \text{H}_3\text{O}^+ \rightarrow \text{HO}_2 + \text{H}_2\text{O}$	3×10^{10}
C. Dissociation Reactions	
19. $\text{H}_2\text{O} \rightarrow \text{H}_3\text{O}^+ + \text{OH}^-$	$5.5 \times 10^{-6*}$
20. $\text{HO}_2 \rightarrow \text{H}_3\text{O}^+ + \text{O}_2^-$	$1 \times 10^6*$

*Rate constant, sec^{-1}

Table 4: Differential Equations for Transient Species and Radiation Products in Irradiated Neutral Water

$\frac{d(H)}{dt}$	$= - 2k_1(H)^2 - k_2(e_{Aq}^{-})(H) - k_5(H)(OH) + k_7(H_3O^+)(e_{Aq}^{-}) - k_9(H)(H_2O_2) + k_{12}(OH)(H_2)$ $- k_{13}(HO_2)(H) - k_{17}(H)(O_2)$	+ 0.55 I
$\frac{d(e_{Aq}^{-})}{dt}$	$= - k_2(e_{Aq}^{-})(H) - 2k_3(e_{Aq}^{-})^2 - k_4(e_{Aq}^{-})(OH) - k_7(H_3O^+)(e_{Aq}^{-}) - k_{10}(e_{Aq}^{-})(H_2O_2) - k_{14}(e_{Aq}^{-})(O_2)$	+ 2.65 I
$\frac{d(OH)}{dt}$	$= - k_4(e_{Aq}^{-})(OH) - k_5(H)(OH) - 2k_6(OH)^2 + k_9(H)(H_2O_2) + k_{10}(e_{Aq}^{-})(H_2O_2) - k_{11}(OH)(H_2O_2)$ $- k_{12}(OH)(H_2) - k_{15}(HO_2)(OH)$	+ 2.70 I
$\frac{d(H_3O^+)}{dt}$	$= - k_7(H_3O^+)(e_{Aq}^{-}) - k_8(H_3O^+)(OH^{-}) - k_{18}(H_3O^+)(O_2^{-}) + k_{19}(H_2O) + k_{20}(HO_2)$	+ 2.65 I
$\frac{d(H_2O)}{dt}$	$= k_5(H)(OH) + k_8(H_3O^+)(OH^{-}) + k_9(H)(H_2O_2) + k_{11}(OH)(H_2O_2) + k_{12}(OH)(H_2) + k_{15}(HO_2)(OH)$ $+ k_{18}(H_3O^+)(O_2^{-}) - k_{19}(H_2O)$	- 4.10 I
$\frac{d(H_2)}{dt}$	$= k_1(H)^2 + k_2(e_{Aq}^{-})(H) + k_3(e_{Aq}^{-})^2 - k_{12}(OH)(H_2)$	+ 0.45 I
$\frac{d(H_2O_2)}{dt}$	$= k_6(OH)^2 - k_9(H)(H_2O_2) - k_{10}(e_{Aq}^{-})(H_2O_2) - k_{11}(OH)(H_2O_2) + k_{13}(HO_2)(H) + k_{16}(HO_2)^2$	+ 0.70 I

Table 4: Differential Equations for Transient Species and
Radiation Products in Irradiated Neutral Water
(cont.)

$$\begin{aligned} \frac{d}{dt} (\text{OH}^-) &= k_2(e^-_{\text{Aq}})(\text{H}) + 2k_3(e^-_{\text{Aq}})^2 + k_4(e^-_{\text{Aq}})(\text{OH}) - k_8(\text{H}_3\text{O}^+)(\text{OH}^-) + k_{10}(e^-_{\text{Aq}})(\text{H}_2\text{O}_2) + k_{19}(\text{H}_2\text{O}) \\ \frac{d}{dt} (\text{HO}_2) &= k_{11}(\text{OH})(\text{H}_2\text{O}_2) - k_{13}(\text{HO}_2)(\text{H}) - k_{15}(\text{HO}_2)(\text{OH}) - 2k_{16}(\text{HO}_2)^2 + k_{17}(\text{H})(\text{O}_2) + k_{18}(\text{H}_3\text{O}^+)(\text{O}_2^-) - k_{20}(\text{HO}_2) \\ \frac{d}{dt} (\text{O}_2) &= -k_{14}(e^-_{\text{Aq}})(\text{O}_2) + k_{15}(\text{HO}_2)(\text{OH}) + k_{16}(\text{HO}_2)^2 - k_{17}(\text{H})(\text{O}_2) \\ \frac{d}{dt} (\text{O}_2^-) &= k_{14}(e^-_{\text{Aq}})(\text{O}_2) - k_{18}(\text{H}_3\text{O}^+)(\text{O}_2^-) + k_{20}(\text{HO}_2) \end{aligned}$$

Appendix

The transient solutions for the chemical kinetic problems were justified by virtue of chemical analysis. When solutions went into the stiff regions, equilibrium values of the concentrations of the species were found and those were somewhat known to the chemists [4].

At this stage, a question may still arise regarding the effectiveness of this method (which could be easily verified mathematically) with regard to the following inquiries: Given a stiff system (i) how good is this method to generate transient solutions which could be verified mathematically and (ii) how fast equilibrium values may be found.

We also intend to write down the algorithm of the method with regard to the solution of a system.

Consider:

$$\left. \begin{aligned} \frac{dx}{dt} &= -10004 x + 10000 y^4 \\ \frac{dy}{dt} &= -y + x - y^4 \\ x(0) &= y(0) = 1 \end{aligned} \right\} \quad (A1)$$

This is a stiff system given in [3]. It has an approximate analytical solution given by

$$y = \left[\frac{10004 e^{-3t}}{10008 - 4e^{-3t}} \right]^{1/3} \quad (A2)$$
$$x = \frac{10000}{10004} y^4$$

Using backward Euler approximation the difference approximations may be expressed as:

$$x_j = \Delta t(-10004 x_j + 10000 y_j^4) + x_{j-1} \quad (A3)$$

$$y_j = \Delta t(-y_j + x_j - y_j^4) + y_{j-1} \quad (A4)$$

$$x_0 = y_0 = 1$$

Now let us see how to apply the present algorithm to solve the difference equations (A3) and (A4). We note that when we are solving for x_j and y_j , x_{j-1} and y_{j-1} are known.

To simplify our notations let us replace x_j by X and y_j by Y , x_{j-1} by X_0 and y_{j-1} by Y_0 . Then the system may be expressed as:

$$X = F(X, Y)$$

$$Y = G(X, Y)$$

$$\text{where } F(X, Y) = \Delta t(-10004 X + 10000 Y^4) + X_0$$

$$G(X, Y) = \Delta t(-Y + X - Y^4) + Y_0$$

$$\text{Thus } F_X = -10004 \Delta t; G_Y = -4 \Delta t Y^3.$$

Step #1 Make guesses for X and Y . Usually the guesses are X_0, Y_0 .

At some $(k + 1)$ iteration and t_n time-level ($X^k =$ value of X at the k th iteration)

$$\text{Step \#2 Compute: } F1 = F(X^k, Y^k)$$

$$\text{Step \#3 } F2 = F(F1, Y^k)$$

$$\text{Step \#4 } DFDX = F_X(F1, Y^k)$$

$$\text{Step \#5 } WX = (F2 - F1)/(1 - DFDX)$$

$$\text{Step \#6 } X^{k+1} = WX + F1$$

$$\text{Step \#7 } G1 = G(X^{k+1}, Y^k)$$

$$\text{Step \#8 } G2 = G(X^{k+1}, G1)$$

$$\text{Step \#9 } DGDY = G_Y(X^{k+1}, G1)$$

$$\text{Step \#10 } WY = (G2 - G1)/(1 - DGDY)$$

Step #11 $Y^{k+1} = WY + G1$

Step #12 $WMAX = \max (|WX|, |WY|)$

Step #13 If $WMAX < \epsilon$, convergence is found and we go to Step #15, otherwise we use Step #14

Step #14 Increase the level of iteration by 1 and start from Step #2.

However if $KMAX$ is the largest number of iterations that could be used and if $k > KMAX$ then the failure of the method is recorded.

Step #15 Print the values of X and Y (and possibly the time-level and the value of k)

Step #16 Now time-level is incremented by 1.
Update $X0$ and $Y0$ by setting $X0 = X$ and $Y0 = Y$
(These are the guesses for X and Y at this new time-level solution.)

Then again at some $(k + 1)$ th level of iteration of this time-step computations start from Step #2.

In general we choose $\epsilon = 10^{-10}$.

Case:1 Transient Solutions

In the Table: A1, the transient-part of the solutions are given. The results are quite close to those obtained analytically. Here we have used a fixed $\Delta t = 0.005$.

Case:2 Steady-State Solutions

If the objective is to compute solutions in the stiff region, we may use variable time steps and Δt may be very large. Using a sequence of $\Delta t = 1, 10, 10^2, 10^3, \dots 10^8$, it has been found that the steady state was found (with no oscillations) after a total of 55 iterations for all time steps using Radio Shack TRS-80 color microcomputer (16K) and the

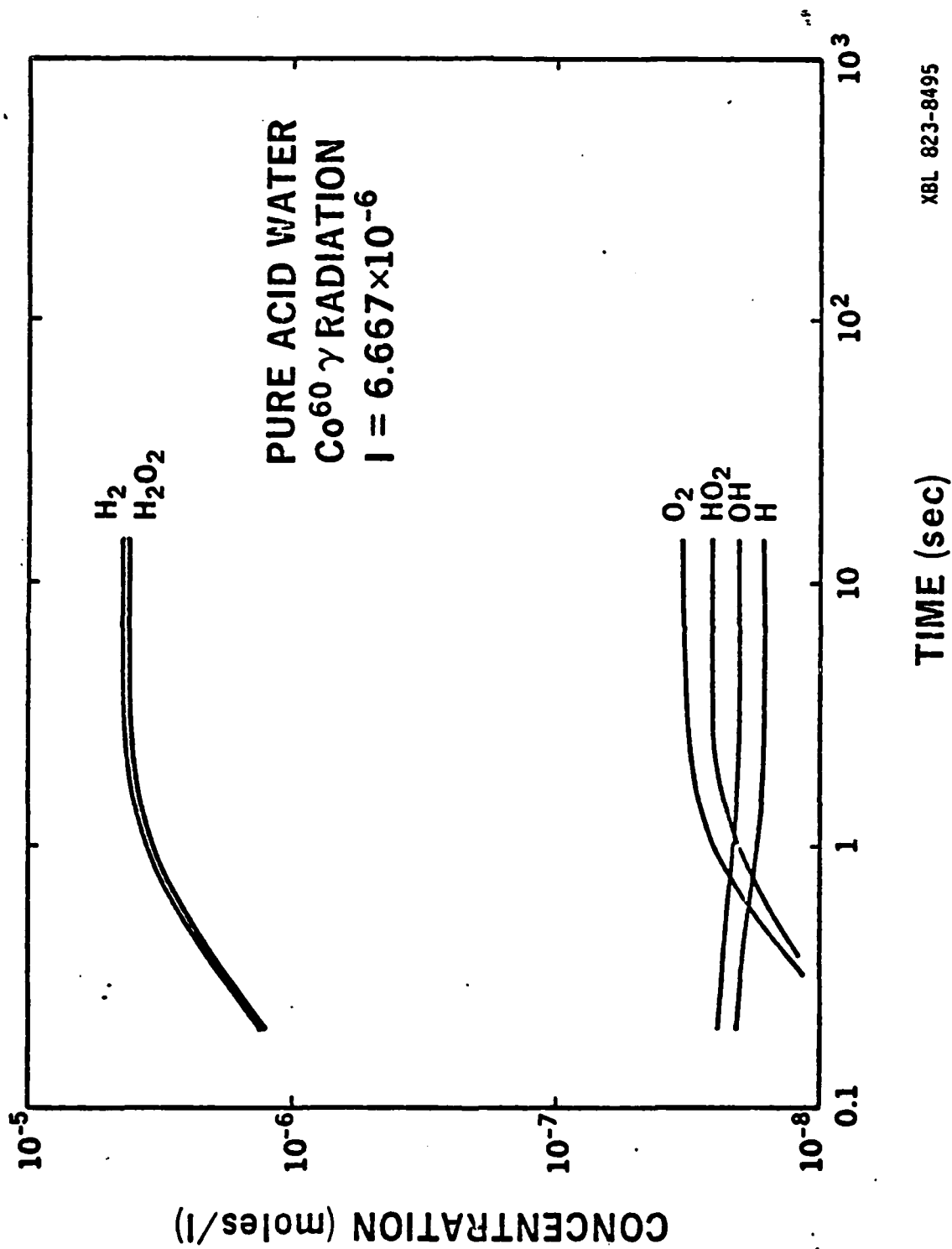
total time for computation was little less than a minute. As expected x_j 's went to zero much faster than y_j 's.

At this stage both Gauss-Seidel iterations and Newton's method were used. Both failed for each case.

Table: A1

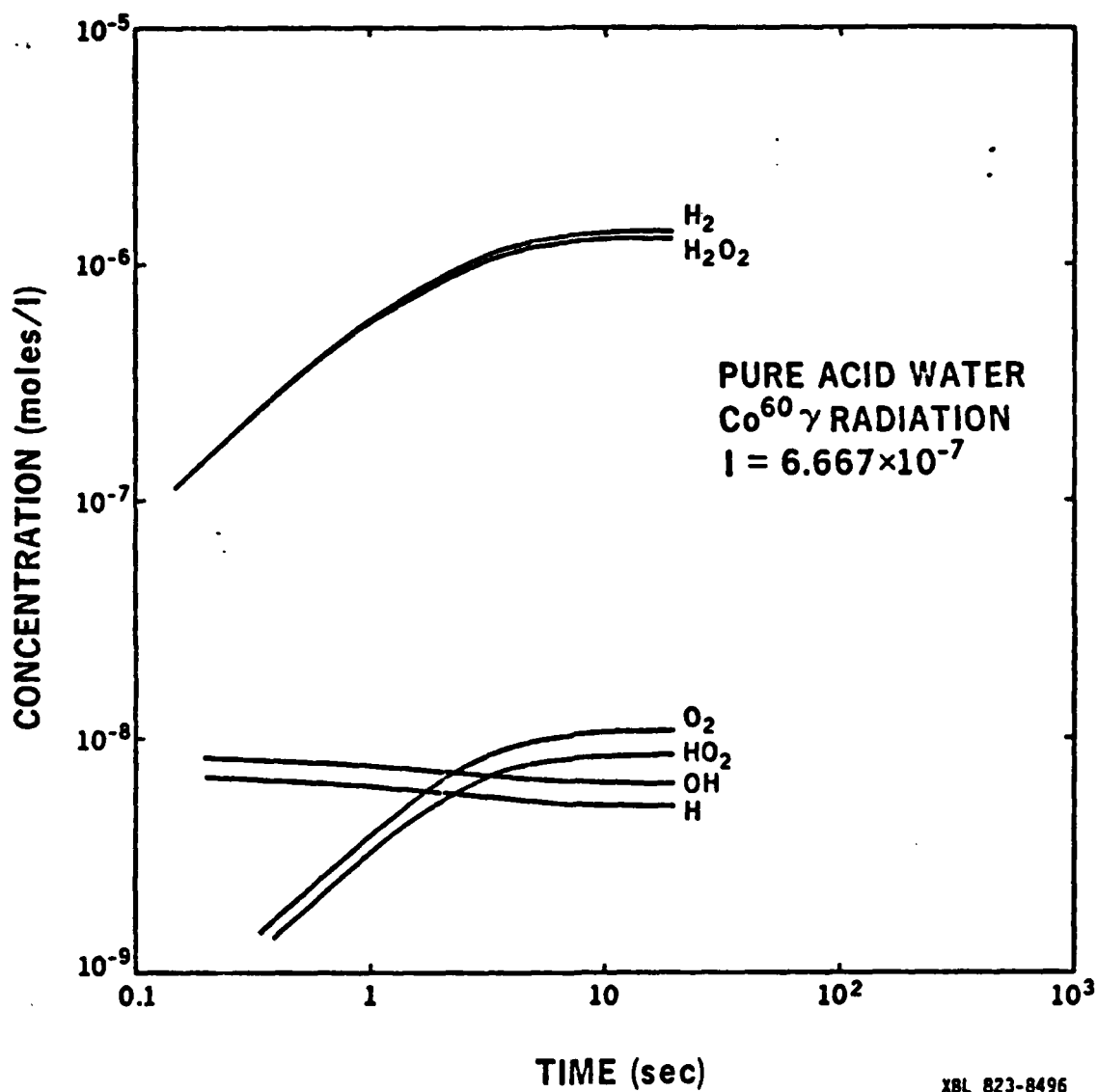
$\Delta t = 0.005$, $k = \text{No. of iterations for convergence at a given time step.}$

t	Exact Solution		Dey		k
	x	y	x	y	
0	1.0	1.0	1.0	1.0	0
0.25	0.368021198	0.778953673	0.368799110	0.779286476	5
0.50	0.135369435	0.606629568	0.136012224	0.607287244	4
0.75	0.049796505	0.472436166	0.050160955	0.473250544	3
1.00	0.0183185671	0.367930928	0.018499225	0.368797576	3
1.25	0.73174825E-03	0.286467519	6.82246374E-03	0.287398821	3
1.50	2.47909732E-03	0.223160237	2.516106E-03	0.223965900	3
1.75	9.12006087E-04	0.173797232	9.27933E-04	0.174533507	2
2.0	3.35507793E-04	0.13535337	3.42219E-04	0.136011530	2
2.25	1.23426334E-04	0.105413292	1.26209E-04	0.105991891	2
2.5	4.54059955E-05	0.0820959478	4.6546E-05	0.082598004	2



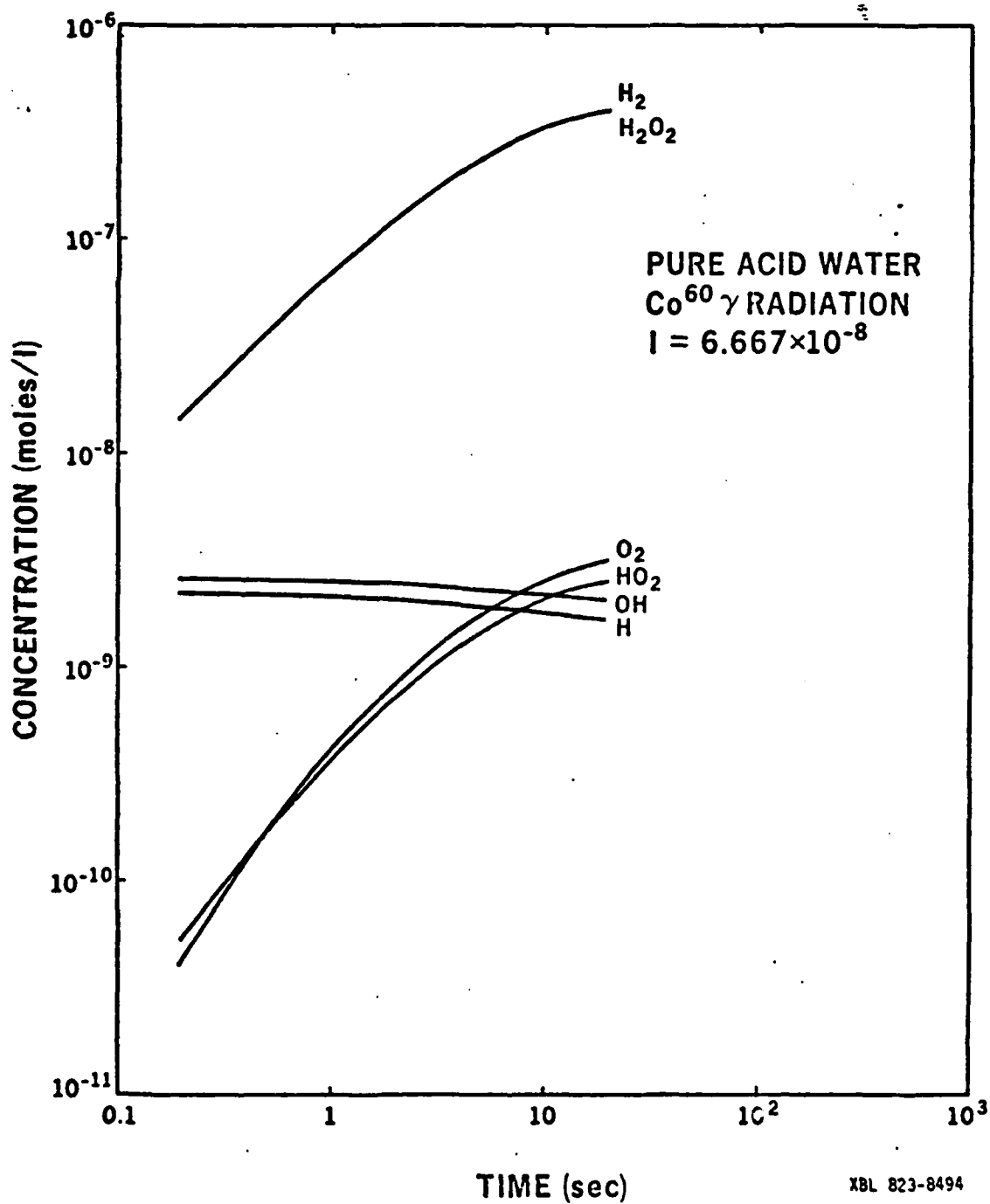
XBL 823-8495

1. Figure 1. Concentrations of Species (Model:1) vs. time for
 I = 6.667 × 10⁻⁶ in the logarithmic scale.

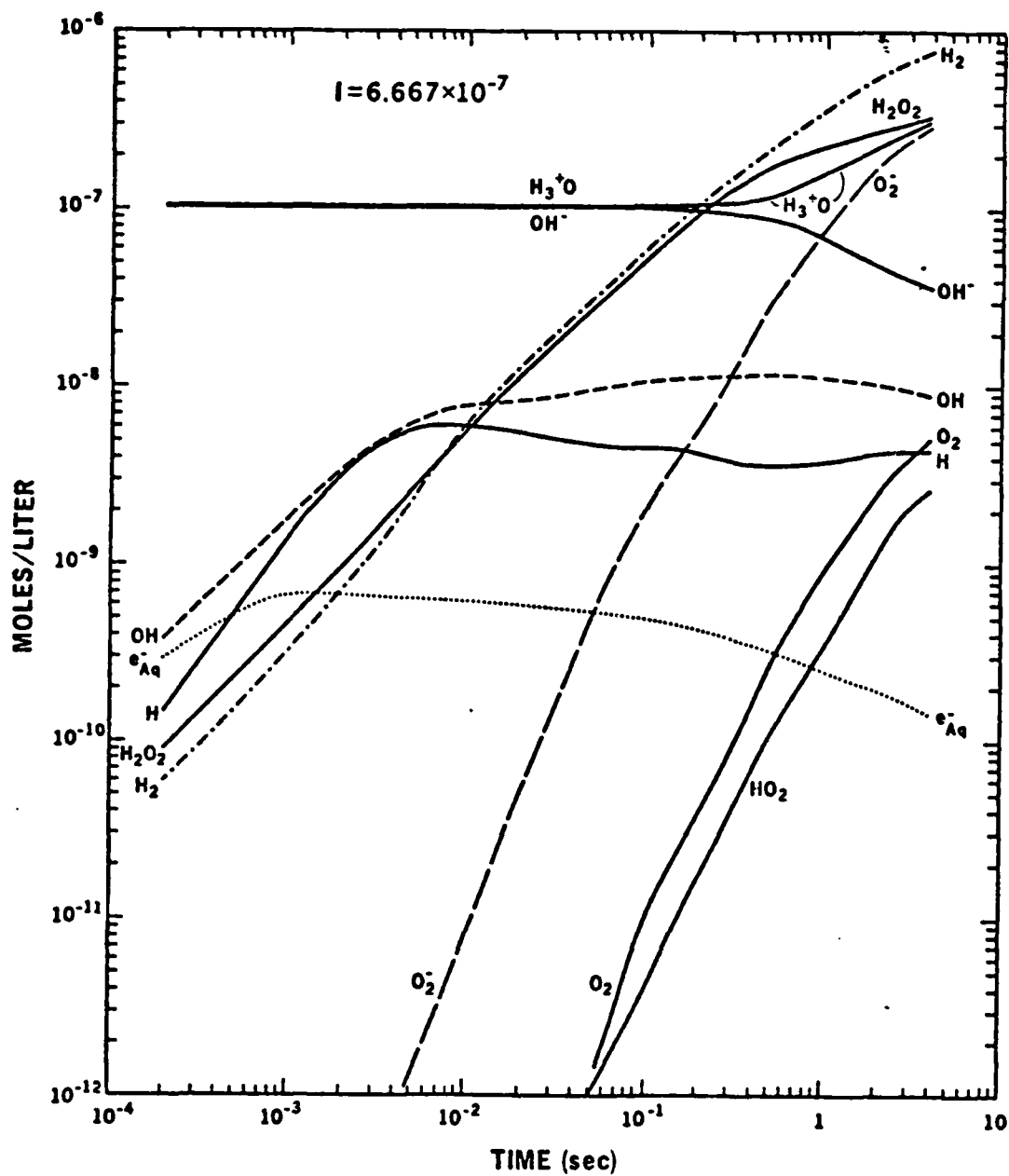


XBL 823-8496

2. Figure 2: Concentrations of Species (Model:1) vs. time
 for $I = 6.667 \times 10^{-7}$ in the logarithmic scale.



3. Figure 3. Concentrations of Species (Model:1) vs. time for
 $I = 6.667 \times 10^{-8}$ in the logarithmic scale.



XBL 823-8767

4. Figure 4. Concentrations of Species (Model:2) vs. time for $I = 6.667 \times 10^{-7}$ in the logarithmic scale.

Nonlinear Sensitivity Analysis in Chemical Kinetics — The
Scaled Green's Function Method

Jenn-Tai Hwang
Department of Chemistry
National Tsing Hua University
Hsinchu, Taiwan 300
Republic of China

Abstract: A new approach to the nonlinear sensitivity problems in chemical kinetics is presented. The method is based on a Green's function method developed earlier by Hwang, et al. In order to calculate higher-order sensitivity coefficients, new ways of evaluating the Green's function matrix and the sensitivity integrals are developed. The Green's function matrix is rescaled to the unity matrix at the beginning of every step in the numerical integration of the rate equations. There is no error control on the Green's function matrix. However, due to the rescaling procedure, this does not entail any stability problem. The start-up difficulty is thus completely eliminated. To solve for the Green's function matrix in each interval, a special exponential method is employed. No matrix inversions are needed to compute the sensitivity coefficients.

This method is compared and contrasted with the direct variation method, and the comparisons indicated it to be a very promising approach to the nonlinear sensitivity problem.

1. Introduction

The need for systematic sensitivity analysis of large computational models has become increasingly apparent in recent years.¹⁻¹⁰ This need is emerging as complex numerical models are more and more frequently employed for problem solving in numerous areas that include atmospheric science, combustion, chemical laser studies, dynamical engineering systems and so on. The objective of a sensitivity analysis is to study the response of a dynamical system to parameter variations. When the complexity of the models makes it difficult to determine the effect errors in the physical and chemical parameters have on the solution vectors, undertaking of a sensitivity analysis is particularly desirable.

Conceptually the simplest approach to a sensitivity analysis is to solve the system equations repeatedly while varying one (or several) parameter(s) at a time. However, this soon becomes impractical as the number of parameters subject to variation increases. Three methods aiming at the reduction of computational efforts thus appeared: they are the Fourier amplitude sensitivity test (FAST)⁵, the direct method (DM)⁷, and the Green's function method (GFM)^{1a}. The applicability regions and disadvantages of the above three methods have recently been discussed.^{1a,1c,1e} Here we merely comment that these three methods are either computationally still too costly or are limited to linear sensitivity

studies only. By a linear sensitivity study we mean a study based on the first-order Taylor series expansion of the solution vector. As a linear analysis is inadequate for a system with large parameter variations and computational models involving parameters of large uncertainties are quite common — especially models in chemical kinetics — it is clearly desirable to have a practical method whereby nonlinear model behavior in parametric space can be assessed.

In this work a new nonlinear sensitivity method based on rescaling of the Green's function matrix in GFM^{1a} is developed. To prevent excessive computation associated with the frequent rescalings, the Green's function matrix equation is solved in a rather novel fashion: its error is not controlled. This seemingly outrageous behavior has its justification in the very act of rescaling. As explained in Sec. 2, rescaling essentially stops the propagation of errors. Therefore, it is only the local truncation error that determines the accuracy of the Green's function matrix; the usual stability problem in the numerical integration of ordinary differential equations never appears.

The basic theory of the scaled Green's function method (SGFM) is presented in the next section. The last section concludes with discussions on possible future improvements and applications to several kinetics systems.

2. The scaled Green's Function Method (SGFM) of Sensitivity Analysis

Consider a general nonlinear first-order N-vector differential equation

$$\underline{y}' = \underline{f}(\underline{y}, \underline{a}, t), \quad (1)$$

with the initial condition

$$\underline{y}(0) = \underline{b}. \quad (2)$$

In Eq. (1) the prime indicates differentiation with respect to t ; the M-vector $\underline{a}$ denotes collectively a set of M parameters $\{a_{\ell}\}_{\ell=1}^M$; $\underline{f}$ is an N-vector function of $\underline{y}$, $\underline{a}$ and t . Equation (1) can be viewed, for example, as N coupled chemical rate equations with $\underline{y}(\underline{a}, t)$ representing the N species concentrations at time t and $\underline{a}$ the rate coefficients associated with M elementary chemical reactions. It should be noted here that due to the (possible) vast difference in time constants of various chemical species in (complex) reaction schemes, Eq. (1) is generally stiff.

In order to study the response of $\underline{y}$ to the variation of $\underline{a}$, partials of $\underline{y}$ with respect to $\underline{a}$ are evaluated. An nth-order partial

$$\underline{y}_{a_{\ell_1} a_{\ell_2} \dots a_{\ell_n}} = \partial^n \underline{y} / \partial a_{\ell_1} \partial a_{\ell_2} \dots \partial a_{\ell_n} \quad (3)$$

is called an nth-order sensitivity coefficient of $\underline{y}$ with respect to $a_{\ell_1}, a_{\ell_2}, \dots, a_{\ell_n}$. The time propagation of a first-order (linear) sensitivity coefficient $\underline{y}_{a_{\ell}}$ is governed by the following first-order sensitivity equation^{1a,2}:

$$Y'_{a_\ell} - \underline{J} Y_{a_\ell} = \underline{f}_{a_\ell}. \quad (4)$$

where

$$\underline{f}_{a_\ell} = \partial \underline{f} / \partial a_\ell, \quad (5)$$

$$J_{ij} = \partial f_i / \partial y_j. \quad (6)$$

The initial condition to Eq. (4) is^{1a,2}

$$Y_{a_\ell}(0) = \underline{0}. \quad (7)$$

A second-order sensitivity coefficient $Y_{a_\ell a_m}$ is governed by the following second-order sensitivity equation^{1a}:

$$Y'_{a_\ell a_m} - \underline{J} Y_{a_\ell a_m} = \underline{f}_{a_\ell a_m} + \underline{J}_{a_\ell} Y_{a_m} + \underline{J}_{a_m} Y_{a_\ell} + \underline{L} Y_{a_\ell} Y_{a_m}, \quad (8)$$

where

$$\underline{f}_{a_\ell a_m} = \partial^2 \underline{f} / \partial a_\ell \partial a_m, \quad (9)$$

$$\underline{J}_{a_\ell} = \partial \underline{J} / \partial a_\ell, \quad (10)$$

$$L_{ijk} = \partial J_{ij} / \partial y_k, \quad (11)$$

and

$$(\underline{L} \underline{\omega} \underline{x})_i = \sum_{j,k=1}^N L_{ijk} \omega_j x_k. \quad (12)$$

Notice that Eq. (4) and (8) share the same homogeneous part — a characteristic among all sensitivity equations.^{1a}

To solve Eqs. (4) and (8), we compute first a Green's function matrix $\underline{G}$ which satisfies

$$\underline{G}' = \underline{J} \underline{G}, \quad (13)$$

$$\underline{G}(0) = \underline{1}, \quad (14)$$

where $\underline{1}$ is the unity matrix^{1a}. Then

$$y_{a_\ell}(t) = \underline{G}(t) \int_0^t dx \underline{G}^{-1}(x) \underline{f}_{a_\ell}(x); \quad (15)$$

a similar equation holds for $y_{a_\ell a_m}(t)$. (In what follows, only $y_{a_\ell}(t)$ will be discussed; the same argument applies to $y_{a_\ell a_m}(t)$ and other higher-order sensitivity coefficients.) The set of $M(M+3)/2$ N-vector differential equations [Eqs. (4) and (8)] has thus been replaced by a set of N N-vector differential equations [Eqs. (13)-(14)] and $M(M+3)/2$ N-vector integrals [Eq. (15) and a similar equation for $y_{a_\ell a_m}$]. This is advantageous when M is large because numerical quadrature of reasonably smooth integrands is much easier than numerical integration of stiff ordinary differential equations — recall that in chemical kinetics M is of the order of N^2 .

Evaluation of y_{a_ℓ} through Eq. (15) requires $\underline{G}^{-1}$. Although in principle $\underline{G}$ is never singular, numerically it could be so for stiff systems^{1a}. One way out of this difficulty is to rescale $\underline{G}$ to $\underline{1}$ at a sufficiently fine grid of points. Let us assume that $\underline{G}$ is rescaled at $t_1, t_2, \dots, t_s$. If $\underline{G}_r$ is the scaled solution to Eq. (13) in $[t_r, t_{r+1}]$ which satisfies

$$\underline{G}_r(t_r) = \underline{1}, \quad (16)$$

the relations between $\underline{G}$ and the $\underline{G}_r$ are easily seen to be^{1e}

$$\begin{aligned} \underline{G}(t) &= \underline{G}_0(t), & t \in [0, t_1] \\ &= \underline{G}_1(t) \underline{G}_0(t_1), & t \in [t_1, t_2] \\ &\vdots \\ &= \underline{G}_s(t) \underline{G}_{s-1}(t_s) \cdots \underline{G}_0(t_1), & t \in [t_s, t]. \end{aligned} \quad (17)$$

Using Eq. (17), y_{a_ℓ} then becomes^{1e}

$$y_{a_\ell}(t) = \underline{G}_r(t) \left[\int_{t_r}^t dx \underline{G}^{-1}(x) \underline{f}_{a_\ell}(x) + y_{a_\ell}(t_r) \right], \quad (18)$$

where $t \in [t_r, t_{r+1}]$. (An equation similar in form to Eq. (18) holds for $y_{a_\ell a_m}$.) Equation (18) demonstrates that sensitivity coefficients can be evaluated in a straightforward and recursive manner without ever worrying about the existence of scaling factors, $\underline{G}_0(t_1), \underline{G}_1(t_2), \dots$. The integrand on the RHS of Eq. (18),

$$g(t, x) = \underline{G}_r(t) \underline{G}_r^{-1}(x) \underline{f}_{a_\ell}(x), \quad (19)$$

has a very interesting and useful property^{1f}: at the two end points t_r and t ,

$$g(t, t) = \underline{f}_{a_\ell}(t), \quad (20)$$

$$g(t, t_r) = \underline{G}_r(t) \underline{f}_{a_\ell}(t_r). \quad (21)$$

Equation (21) follows from the initial condition of the scaled Green's function matrix $\underline{G}_r$, Eq. (16). Notice that there is no involvement of $\underline{G}^{-1}$ in Eqs. (20)-(21).

If the integrand g varies sufficiently slowly in $[t_r, t]$, or if t_r and t are sufficiently close together, the integral can be calculated by means of two end-points only. Thus, using the trapezoidal rule gives

$$y_{a_\ell}(t) = \frac{(t-t_r)}{2} [\underline{f}_{a_\ell}(t) + \underline{G}_r(t) \underline{f}_{a_\ell}(t_r)] + \underline{G}_r(t) y_{a_\ell}(t_r). \quad (22)$$

Equation (22) and other similar equations for the higher-order sensitivity coefficients are the main working equations of the SGFM. Notice that using other two-point quadrature formulae can change the form of Eq. (24), but the nonnecessity of performing matrix inversion remains. The validity of Eq. (22) rests on the slowness in variation of the integrand, or the closeness of the two integration end-points. Clearly, these two conditions are satisfied if we rescale $\underline{G}$ at every step. That is, during the normal numerical integration of Eqs. (1) and (13), we let $\underline{G}$ equal $\underline{1}$ at the beginning of every step. This procedure, however, would lead to disaster for the following reason.

To solve a stiff system a practical way is to use a variable-step variable-order algorithm¹¹⁻¹³, such as that of Gear¹²⁻¹³. Now, in Gear's method the differential equation solver takes a large step only when the rapidly changing component has decayed to an insignificant level¹², a situation which could never happen if the condition of a stiff system is repeatedly altered at the start of every new step. In consequence, the step-size remains unbearably small throughout the whole integration process.

This sort of severe limitation upon step-size is the main factor which renders the step-by-step rescaling a prohibitively costly procedure. We now propose to raise this limitation by solving Eq. (13) simultaneously with Eq. (1) without any error

control on $\underline{G}$. In this way, the Green's function matrix equation never affects the choice of step-size and the integration process can proceed smoothly even in the presence of rescalings. Justification for this outrageous behavior comes from the very act of rescaling. As one can see clearly, setting the Green's function matrix $\underline{G}$ to a definite constant matrix $\underline{1}$ involves absolutely no error at all. Therefore, the possible amplification of errors as the ODE solver progresses from one step to another is eliminated. The accuracy of $\underline{G}$ in each step depends entirely upon the way it is computed there; the usual stability problem simply does not appear in SGFM.

To calculate $\underline{G}$, we first note that the solution to Eq. (13) always changes (exponentially) faster than the coefficient $\underline{J}$. (For example, the solution to $G' = t^n G$ is proportional to $\exp[t^{n+1}/(n+1)]$.) Therefore, the step-size h chosen by Eq. (1) is definitely unsuitable for $\underline{G}$ when h is large, if the same method as that used to calculate y is used to calculate $\underline{G}$. Let us proceed by writing Eq. (13) as

$$G'_{ij} = J_{ii} G_{ij} + \sum_{k \neq i} J_{ik} G_{kj}, \quad i, j = 1, 2, \dots, N. \quad (23)$$

If $\underline{J}$ is nearly a constant in $[t_0, t]$ (see below), then

$$G_{ij}(t) = \exp[J_{ii}(t-t_0)] \left(\delta_{ij} + \sum_{k \neq i} J_{ik} \int_{t_0}^t dx G_{kj}(x) \exp[-J_{ii}(x-t_0)] \right), \quad (24)$$

where the initial condition

$$\underline{G}(t_0) = \underline{1} \quad (25)$$

has been used. To obtain an explicit expression for $G_{ij}(t)$, we let, as guided by the form of Eq. (24),

$$G_{kj}(x) = g_{kj}(x) \exp[J_{kk}(x-t_0)]. \quad (26)$$

The functions $g_{kj}(x)$ are evaluated as follows.

$$\begin{aligned} g_{kj}(x) &= \exp[-J_{kk}(x-t_0)] \sum_{n=0}^{\infty} \frac{(x-t_0)^n}{n!} G_{kj}^{(n)}(t_0) \\ &= G_{kj}(t_0) \sum_{m=0}^L \frac{(-J_{kk})^m}{m!} (x-t_0)^m + G_{kj}^{(1)}(t_0) \sum_{m=0}^{L-1} \frac{(-J_{kk})^m}{m!} (x-t_0)^{m+1} \\ &\quad + \dots + \frac{G_{kj}^{(L)}(t_0)}{L!} (x-t_0)^L + O((x-t_0)^{L+1}), \end{aligned} \quad (27)$$

where

$$G_{kj}^{(n)}(t_0) = \left(\frac{\partial^n G_{kj}}{\partial t^n} \right) (t_0). \quad (28)$$

It can be shown readily that Eq. (27) gives correct time derivatives of G_{kj} at t_0 up to the L th-order. Substituting Eq. (27) into Eq. (24) then yields

$$\begin{aligned} G_{ij}(t) &= \exp[J_{ii}(t-t_0)] \{ \delta_{ij} + \\ &\quad (1-\delta_{ij}) J_{ij} \sum_{m=0}^L \frac{(-J_{jj})^m}{m!} A_m(J_{jj}-J_{ii}; t-t_0) + \\ &\quad \sum_{k \neq i} J_{ik} [G_{kj}^{(1)}(t_0) \sum_{m=0}^{L-1} \frac{(-J_{kk})^m}{m!} A_{m+1}(J_{kk}-J_{ii}; t-t_0) + \\ &\quad \dots + \frac{G_{kj}^{(L)}(t_0)}{L!} A_L(J_{kk}-J_{ii}; t-t_0)] + O((t-t_0)^{L+2}), \end{aligned} \quad (29)$$

where

$$A_m(u;v) = \int_0^v dx x^m \exp(ux). \quad (30)$$

The A_m can be evaluated in a recursive manner:

$$A_m(u;v) = u^{-1}[v^m \exp(uv) - mA_{m-1}(u;v)], \quad m=1,2,\dots, \quad (31)$$

$$A_0(u;v) = u^{-1}[\exp(uv) - 1]. \quad (32)$$

We would like to emphasize here that Eq. (29) does not necessarily imply the behavior of G_{ij} is governed by $\exp[J_{ii}(t-t_0)]$. This can be appreciated more easily through examining the factors $\exp[J_{ii}(t-t_0)]A_m(J_{jj}-J_{ii};t-t_0)$:

$$\begin{aligned} B_m(J_{jj}, J_{ii}; t-t_0) &\equiv \exp[J_{ii}(t-t_0)]A_m(J_{jj}-J_{ii};t-t_0) \\ &= (J_{jj}-J_{ii})^{-1}\{(t-t_0)^m \exp[J_{jj}(t-t_0)] - \\ &\quad mB_{m-1}(J_{jj}, J_{ii}; t-t_0)\}, \end{aligned} \quad (33)$$

$$\begin{aligned} B_0(J_{jj}, J_{ii}; t-t_0) &= (J_{jj}-J_{ii})^{-1}\{\exp[J_{jj}(t-t_0)] - \\ &\quad \exp[J_{ii}(t-t_0)]\}. \end{aligned} \quad (34)$$

Clearly, $G_{ij}(t)$ as given by Eq. (29) consists of $\exp[J_{jj}(t-t_0)]$ for all j and also powers of $(t-t_0)^m$ for $m=0,1,2,\dots,L$.

The free parameter L in the theory is related to the order of accuracy of Eq. (27) and can be chosen in a variety of ways. For example, L can be equated simply to the current order q of the backward differentiation formula¹²⁻¹³ used in solving

Eq. (1). The $G_{kj}^{(n)}(t_0)$, $n=1,2,\dots,L$, are then calculated through direct differentiation^{1g}. Alternatively, one may fix L at 1 while dividing each step into q intervals. Within each interval, Eq. (29) is applied to solve for the "component" Green's function matrix. The final answer is then obtained by multiplying together the resulting q "component" Green's function matrices. For example, let us say the step $[t_0, t_1]$ is divided into two parts: $[t_0, t']$, $[t', t_1]$. Equation (29) can be applied to each interval to solve for $\underline{G}(t')$ and $\underline{G}_1(t_1)$, where $\underline{G}$ and $\underline{G}_1$ satisfy $\underline{G}(t_0) = \underline{G}_1(t') = \underline{1}$. The desired $\underline{G}(t_1)$ is then given by [cf., Eq. (17)]

$$\underline{G}(t_1) = \underline{G}_1(t_1) \underline{G}(t'). \quad (35)$$

The test problems next section were solved by the latter procedure. There are two reasons for this choice. Firstly, fixing L at 1 avoids the time-consuming computation of higher-order derivatives of $\underline{G}$. Secondly, dividing each step into q intervals reduces the time variation of $\underline{J}$ (recall that Eq. (29) was derived under the assumption that $\underline{J}$ is nearly a constant).

The theory developed in this section is not exact, especially so when no error control is performed on $\underline{G}$. Therefore, some indications as to the reliability of computed sensitivity coefficients are required. One useful measure for this purpose is $\det \underline{G}$, the determinant of $\underline{G}$: As can be shown^{1a}, the exact $\det \underline{G}(t)$ satisfies

$$\det \underline{G}(t) = \exp\left[\int_0^t dx \sum_{i=1}^N J_{ii}(x)\right], \quad (36)$$

which allows $\det \underline{G}$ to be calculated almost effortlessly as Eqs. (1) and (13) are being solved. Other possible candidates are the various conservation relations among elements of the $\underline{G}$ matrix (see Appendix). When serious deviations from these measures occur, inferences concerning the system sensitivity behavior are at best doubtful. In this case, a hybrid method which switches to DM⁷ may have to be used. Notice, however, that there are many interesting kinetic problems of which sensitivity information is required only for a relatively short time span. Take the whole class of combustion problems for example. Here, an analysis of the reaction mechanism near ignition is highly desirable, both for academic and for practical reasons. As ignition time is typically 10^{-5} sec, SGFM of sensitivity analysis may usually be applied with confidence.

3. Discussions

The SGFM of sensitivity analysis was applied to (1) a simple first-order reaction system ($C_1 \xrightleftharpoons[k_2]{k_1} C_2$; $C_1(0)=10^3$, $C_2(0)=1$, $k_1=10^3$, $k_2=1$), (2) Chapman mechanism for atmospheric ozone kinetics (2 species, 4 steps)⁷, (3) nitrous oxide decomposition [7 species, 9 steps; $T=2512^\circ K$, $P(\text{atm})=0.37$ (73.7% Ar, 2.0% N_2O , 21.3% N_2 , 3% O_2)]¹⁴, and (4) Bowman mechanism for methane oxidation (13 species, 23 steps)⁴. The time spans of sensitivity computation are 1 sec (system 1), 20 sec (system 2), 10^{-4} sec

(system 3), and 2.5×10^{-5} sec (system 4). All calculations were done in single precision on a CDC-CYBER-172 computer. Tolerance varies from 10^{-6} (systems 1 and 2) to 10^{-4} (systems 3 and 4). For all systems studied, the first- and second-order sensitivity coefficients computed by SGFM and DM generally agreed with each other. However, SGFM enjoyed a considerably higher computation speed. For example, SGFM is ~10 times faster than DM for system 2, and over 400 times faster for system 3. More details will be presented at the conference.

There are several possible areas where improvements can likely be made. The first area concerns the method of numerical integration: Trapezoidal rule is evidently not the best, and a two-point exponential rule^{1c} (or other more sophisticated ones) could do better. Another area is the manner in which sensitivity coefficients are computed. Presently, the sensitivity coefficients are computed once for each step, while Green's function matrix is computed q times for each step. It is clear that higher accuracy may be achieved for the first-order sensitivity coefficients if they are also computed q times. We do not expect this procedure to require too much extra computational time. However, the resultant improvements to second-order sensitivity coefficients should be significant. Finally, the way $g_{kj}(x)$ are computed [see Eq. (27)] may be improved. For example, one may try to eliminate the necessity of $G_{kj}^{(n)}(x)$ by techniques similar to the multi-step method in numerical solution of

ordinary differential equations.

Appendix

We examine in this appendix some general properties of the Green's function matrix $\underline{G}$ defined by Eqs. (13)-(14). The discussions are pertinent to chemical kinetics only.

First we prove that the determinant of $\underline{G}$ is in general a monotonic decreasing function of time. In chemical kinetics, the rate of change of species i can usually be written as

$$C_i' = -D_i C_i + P_i \quad (A1)$$

In Eq. (A1), the prime indicates differentiation with respect to time; C_i is the concentration of species i ; $-D_i C_i$ is the contribution to C_i' resulting from destruction of species i ; P_i is the production part which is independent of C_i . Clearly,

$$\begin{aligned} J_{ii} &\equiv \partial C_i' / \partial C_i \\ &= -D_i - (\partial D_i / \partial C_i) C_i \\ &\leq 0. \end{aligned} \quad (A2)$$

Therefore,

$$\det \underline{G}(t) = \exp \left[\int_0^t dx \sum_i J_{ii}(x) \right]$$

(See Ref. 1a) is a monotonic decreasing function of t .

AD-A122 171

PROCEEDINGS OF THE INTERNATIONAL CONFERENCE ON STIFF
COMPUTATION APRIL 12..(U) UTAH UNIV SALT LAKE CITY DEPT
OF CHEMICAL ENGINEERING R C AIKEN 1982
AFOSR-TR-82-1036-VOL-3 AFOSR-82-0038

4/4

UNCLASSIFIED

F/G 12/1

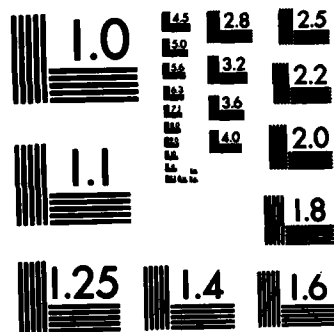
NL



END

FILMED

DATA



MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

We then show that columns of the $\underline{G}$ matrix satisfy certain conservation relations: For example, corresponding to conservation of mass,

$$\left(\sum_{i=1}^N m_i C_i \right)' = 0, \quad (\text{A3})$$

where m_i is the molecular weight of species i , we have

$$\begin{aligned} & \left(\sum_i m_i G_{ij} \right)' \\ &= \sum_k \left(\sum_i m_i J_{ik} \right) G_{kj} \\ &= \sum_k \frac{\partial}{\partial C_k} \left(\sum_i m_i C_i \right) G_{kj} \\ &= 0. \end{aligned} \quad (\text{A4})$$

That is, $\sum_i m_i G_{ij}$ is a conserved quantity. Relation (A4) holds for all columns of the $\underline{G}$ matrix. Other conservation relations of $\underline{G}$ (say, those corresponding to the conservation of atomic species) may be derived in a similar fashion.

References

1. a) J.-T. Hwang, E. P. Dougherty, S. Rabitz and H. Rabitz, J. Chem. Phys., **69**, 5180 (1978); b) J.-T. Hwang and H. Rabitz, J. Chem. Phys., **70**, 4609 (1979); c) E. P. Dougherty, J.-T. Hwang and H. Rabitz, J. Chem. Phys., **71**, 1794 (1979); d) J.-T. Hwang and K. H. Wang, On nonlinear sensitivity analysis and logarithmic solutions of chemical rate equations, 1980 Chinese Chemical Society Conference; (e) J.-T. Hwang, Proc. Natl. Sci. Counc. B, Rep. of China, **6**, 20 (1982); (f) —, ibid., **6**, 37 (1982); (g) —, A computational algorithm for the scaled Green's function method in chemical kinetics, ibid., in press; (h) —, On the proper usage of sensitivities of chemical kinetics models to the uncertainties in rate coefficients, ibid., in press.
2. R. W. Atherton, R. B. Schainker and E. R. Ducot, AIChE J., **21**, 441 (1975).
3. D. E. Bartine, E. M. Oblow, and F. R. Mynatt, Nuc. Sci. Eng., **55**, 147 (1974).
4. A. A. Boni, R. C. Penner, Combust. Sci. Tech., **15**, 99 (1977).
5. R. I. Cukier, H. B. Levine and K. E. Shuler, J. Comp. Phys., **26**, 1 (1978).
6. M. Demiralp, and H. Rabitz, J. Chem. Phys., **74**, 3362 (1981).
7. R. P. Dickinson, and R. J. Gelinas, J. Comp. Phys., **21**, 123 (1976).
8. K. H. Ebert, H. J. Ederet, and G. Isbarn, Angew. Chem., **19**, 333 (1980).

9. D. Edelson, and D. L. Allera, Int. J. Chem. Kinetics,
12, 605 (1980).
10. M. Koda, A. H. Dogru, and J. H. Seinfeld, J. Comp. Phys.,
30, 259 (1979).
11. D. Jacobs, The State of the Art in Numerical Analysis.
Academic Press, N.Y., USA (1977).
12. C. W. Gear, Numerical Initial Value Problems in Ordinary
Differential Equations. Prentics-Hall, N.J., USA (1971).
13. A. C. Hindmarsh, GEAR: ordinary differential equation system
solver. UCID-3001 (Rev. 3). Lawrence Livermore Laboratory
(1974).
14. J. P. Monat, R. K. Hanson and C. H. Kruger, Combust. Sci.
Tech., 16, 21 (1977).

Some Experiences in the Selection of Integrators
for Large-scale ODE Problems in Chemical Engineering

Katherine L. Chen

and

W. E. Schiesser

Lehigh University

Bethlehem, Pa. USA

The computer-based analysis of chemical engineering plants is progressing rapidly beyond the conventional steady state energy and material balance calculations. This increased interest in computer-based analysis is due in particular to: (1) consideration of new chemical processing units and plants, e.g., coal conversion plants, and (2) the need to better understand these proposed units and plants because of the large costs involved in building them. Any analytical tool that can be applied to better focus the design effort is potentially valuable in avoiding costly design errors. Thus, there is increasing interest in applying first-principle models, i.e., models based on fundamental conservation principles, and thermodynamics, kinetics and transport processes, which offer the possibility of a fundamental understanding of the proposed units. Generally, these models are expressed as systems of differential equations, particularly if unsteady state (dynamic) operation is to be considered. Dynamic simulation is indispensable for the understanding of the stability and controllability of proposed units and plants.

When a first-principle model is formulated, it typically involves a large (greater than 100) system of nonlinear ordinary differential equations (ODEs). Such large-scale ODE problems are demanding in terms of the selection and use of a numerical integrator. Fortunately, major advances in the development and implementation of numerical integrators for initial-value ODEs have been made in the past ten years. Yet, the solution of large-scale ODE problems is anything but routine. The success in achieving a solution with reasonable effort is still quite dependent on the particular problem system and the experience of the analyst.

In this paper, we relate some of our experiences in integrating a system of 128 nonlinear ODEs which were formulated for the dynamic simulation of the H_2 plant in a high-Btu coal gasification plant. The individual units in the H_2 plant model and the number of equations for each unit are listed below:

Unit	No. of PDEs	No. of ODEs
Reformer	1	21-point NMOL
Shift reactors		
Unit no. 1	1	21-point NMOL
Unit no. 2	1	21-point NMOL
CO ₂ Absorber		
Absorber		34
Stripper		10
Cleanup Methanator	1	21-point NMOL
Total	4	128

To the best of our knowledge, this is the first reported attempt to dynamically simulate an entire plant with interconnected units modelled both as discrete (perfectly mixed) systems and distributed systems. The latter lead to partial differential equations (PDEs) which were approximated by the numerical method of lines (NMOL) thereby giving a final set of ODEs for the plant model.

The 128-ODE system has a complex structure when viewed in terms of its Jacobian matrix. This matrix is not even approximately banded (it is nearly lower-triangular because of the serial nature of the H_2 plant interconnection of units) and it is not sparse. Therefore, if we conclude that the ODEs are probably stiff and therefore an implicit integrator should be used, the selection of an integrator designed for banded or sparse systems may not accommodate the ODE system very well. In fact, we found this to be the case for a banded integrator, as demonstrated by unsuccessful attempts to completely integrate the ODE system, even with large computer run times. Contrary to expectations, we found an explicit integrator to be more efficient for the calculation of a complete plant transient response for one specific set of initial conditions. Of course, an explicit integrator avoids:

- (1) the problem of matching the Jacobian matrix structure of the ODE system to the integrator and
- (2) the matrix calculations of the implicit integrator which may be quite time consuming.

However, the use of an explicit integrator was not the final answer to successful integration of the ODE system. When the initial conditions were changed, the explicit integrator failed. Thus we conclude that another complicating feature of large-scale chemical

process dynamic simulation is the effect of nonlinearities. This was clearly evident as we mapped the Jacobian matrix of the ODE system during the solution, i.e., the structure changed significantly, and the magnitude of the individual elements in the matrix also changed, sometimes by orders of magnitude. The nonlinearity clearly had a detrimental and largely unpredictable effect on the performance of both the explicit and implicit integrators applied to the ODE problem system.

As a result of this experience, we would like to pose a series of questions for the numerical analysts and developers of ODE integrators:

(1) As users of ODE integrators, rather than developers of integrators, can we reduce the often painful learning experience apparently required for each new, large-scale problem, e.g., the performance, or lack of performance of an integrator, is typically determined by the parameters selected by the user when the integrator is called such as the type of error and its magnitude, minimum step size, etc.?

(2) How can integrators be selected more rationally than just by trial and error? Will sparse matrix integration help?

(3) How does nonlinearity affect the stability of the calculation, i.e., the apparent stiffness?

We have some limited answers to these questions:

(1) Look at the structure of the ODE system in terms of a map of the Jacobian matrix. This picture of the structure will assist in selecting an implicit integrator that processes a Jacobian

matrix with approximately the same structure as the ODE problem system.

(2) Look at the derivatives of the system in detail. Specifically, print all of the temporal derivatives, and the spatial derivatives in the case of PDE models, corresponding to the initial conditions. This will detect a mismatch between the problem time scale and the time scale defined for the integrator. If the magnitudes of the temporal derivatives look unreasonable, print the magnitudes of the individual terms which comprise the derivatives. In general, the derivatives dictate what the solution will do. Therefore, a detailed understanding of the derivatives is indispensable in finding errors and determining why problems exist with the numerical integration.

(3) Compute and print the temporal eigenvalues of the linearized ODE system. This gives a direct indication of the stability of the simulation around the point of linearization (the real parts of the eigenvalues must be either zero or negative). If (1) and (3) are repeated during the course of the calculation, changes in the Jacobian matrix and stiffness due to nonlinearities can be observed as the solution evolves.

(4) If an implicit integrator is selected, use the map of part (1) to reorder the ODE to compress the apparent bandwidth of the Jacobian matrix. This reordering can generally be devised from consideration of the physical problem as well as from a priori consideration of the Jacobian matrix map.

We have found that these limited answers often make the difference between an unsuccessful solution of the problem ODE

system and a successful solution achieved with a reasonable computer run time. Some of these concerns will be illustrated in terms of our experiences with the H₂ plant simulation.

SOME EXPERIENCES IN THE SELECTION OF INTEGRATORS
FOR LARGE-SCALE ODE PROBLEMS IN CHEMICAL ENGINEERING

KATHERINE L. CHEN*

AND

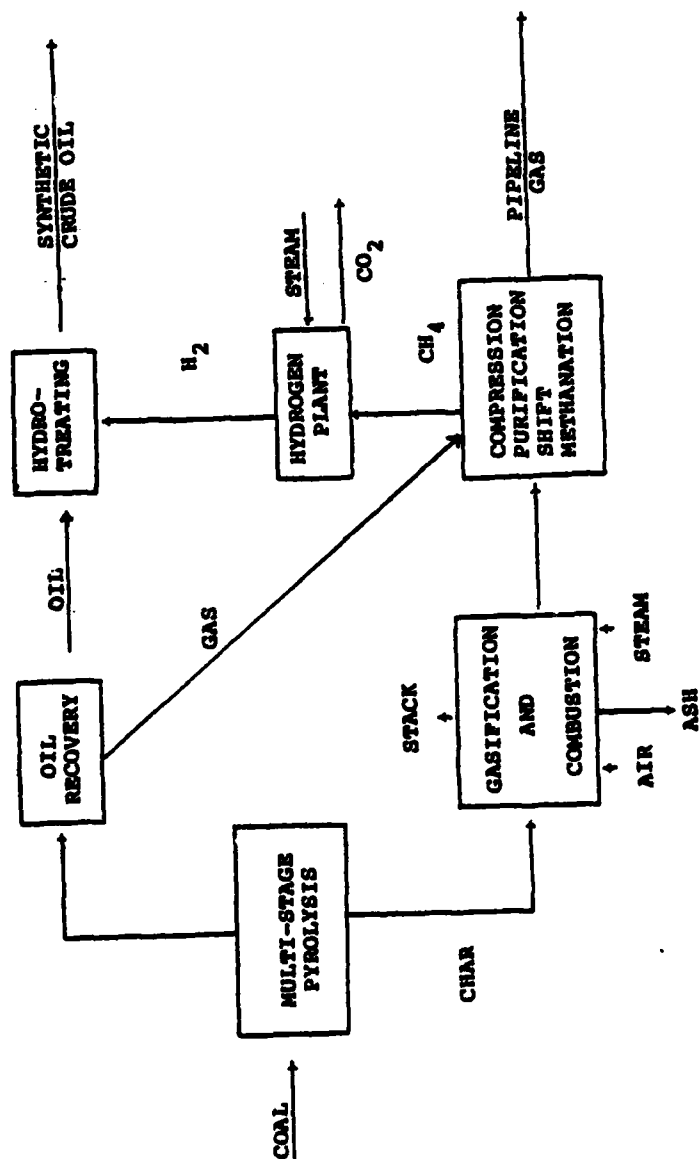
WILLIAM E. SCHIESSER

LEHIGH UNIVERSITY

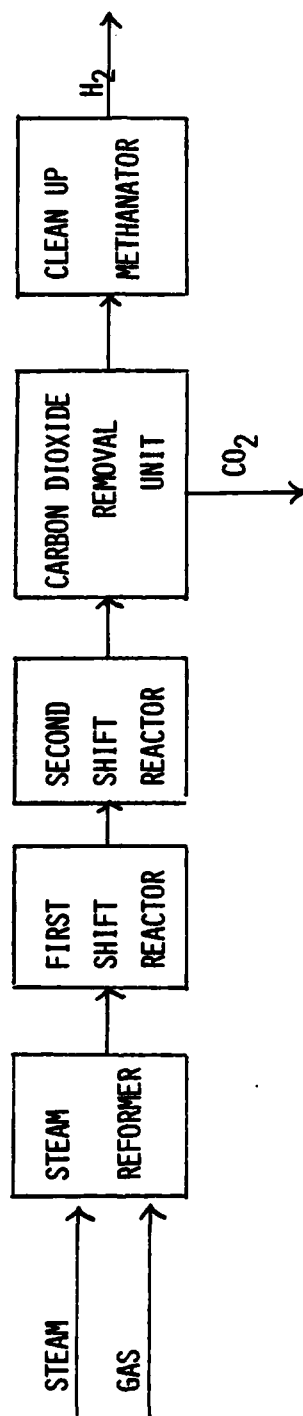
BETHLEHEM, PA. USA

* PRESENT ADDRESS: RAW MATERIALS & CHEMICAL PROCESSES RESEARCH
HOMER RESEARCH LABORATORIES
BETHLEHEM STEEL CORPORATION
BETHLEHEM, PA.

BLOCK DIAGRAM OF COAL GASIFICATION PROCESS



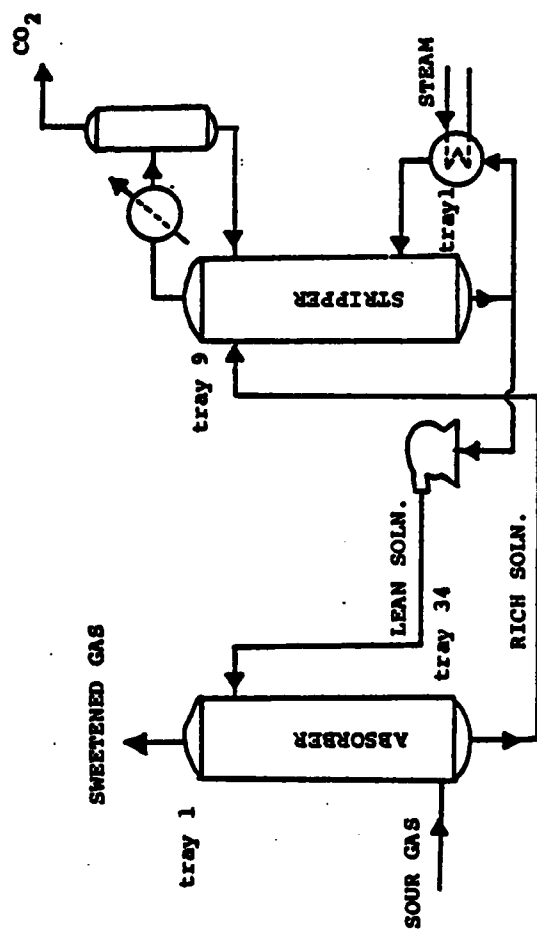
BLOCK DIAGRAM OF ENTIRE HYDROGEN PLANT



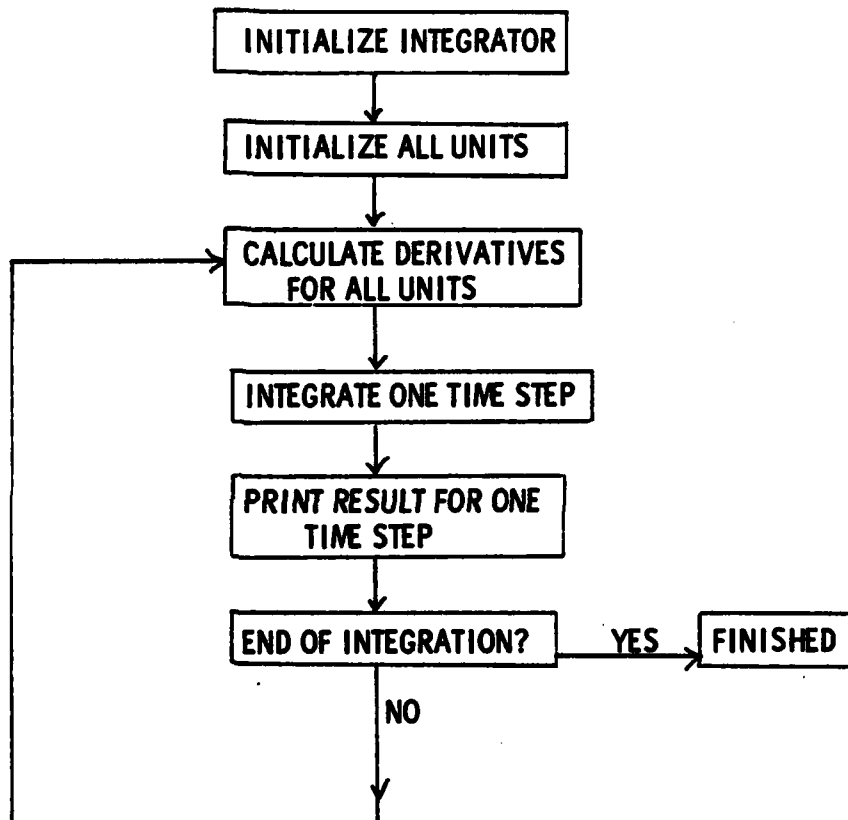
PDE	1	1	1	1	1
ODE	21	21	21	21	21

TOTAL NO. OF ODEs 128

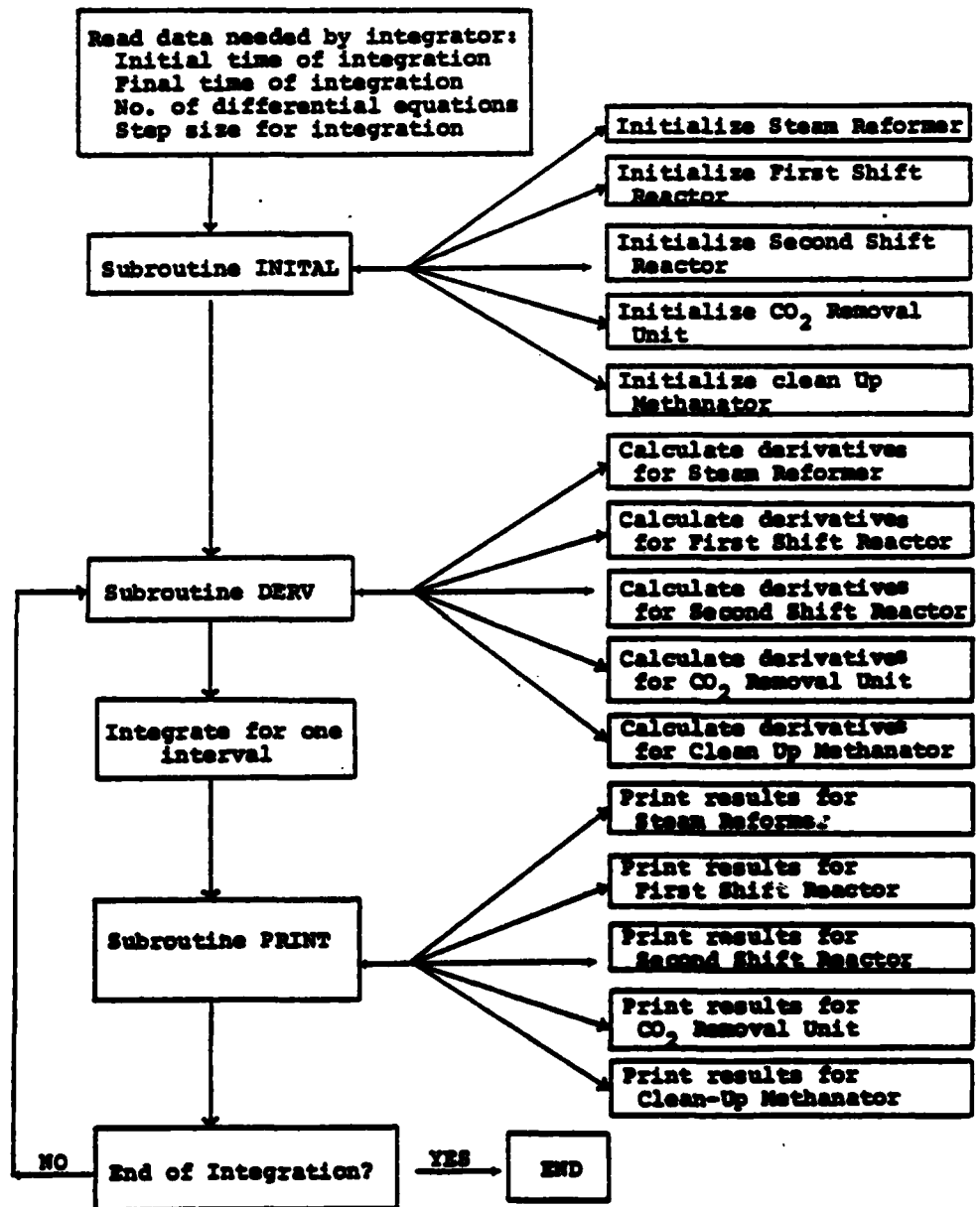
CONVENTIONAL HOT-CARBONATE PROCESS



BLOCK DIAGRAM OF DYNAMIC SIMULATION PROGRAM



BLOCK DIAGRAM OF DYNAMIC SIMULATION PROGRAM



**CANNOT INTEGRATE
FROM 0.035-0.040 HR
WITHIN NEXT 1000
COMPUTER SECONDS**

TEMPORAL DERIVATIVES AT 0.0 HR

REFORMER	SHIFT 1		SHIFT 2		ABSORBER		STRIPPER		CONNECTING STREAM		METHANATOR
	DDT	DDT	DDT	DDT	DDT	DDT	DDT	DDT	DDT	DDT	
0.0	0.0	0.0	0.0	0.0	2.34E+00	-1.01E-04	-2.23E-05	0.0			
8.49E+03	2.85E+01	2.41E+01	2.41E+01	2.41E+01	9.03E-02	-1.21E-04		1.06E+02			
8.49E+03	2.60E+01	1.28E+01	1.28E+01	1.28E+01	.	.		1.00E+02			
.	.	.	.	.	.	.		.			
.	.	.	.	.	.	-1.90E-04		.			
8.49E+03	1.79E+01	1.01E+00	1.01E+00	1.01E+00	1.56E-01	-2.20E-04		8.05E+01			
.	.	.	.	.	.	-4.70E-01		.			
8.49E+03	1.24E+01	8.04E-02	8.04E-02	8.04E-02	.			6.46E+01			
.	.	.	.	.	.			.			
8.49E+03	8.56E+00	6.37E-03	6.37E-03	6.37E-03	2.70E-01			5.19E+01			
.	.	.	.	.	.			.			
.	.	.	.	.	.			.			
8.49E+03	5.91E+00	5.05E-04	5.05E-04	5.05E-04	.			4.17E+01			
8.49E+03	5.39E+00	2.68E-04	2.68E-04	2.68E-04	4.69E-01			3.94E+01			
8.49E+03	4.92E+00	1.42E-04	1.42E-04	1.42E-04	.			3.73E+01			
					.						
					.						
					.						
					.						
					1.12E+00						

0-100-1-76-8

TEMPORAL DERIVATIVES AT 0.5 HR (STEADY STATE)							
REFORMER	SHIFT 1	SHIFT 2	ABSORBER	STRIPPER	CONNECTING STREAM	METHANATOR	
DTDT	DTDT	DTDT	DXDT	DXDT	DXDT	DTDT	
0.0	0.0	0.0	1.98E-04	9.89E-05	3.28E-04	0.0	
0.0	2.05E-03	5.15E-03	3.27E-04	8.85E-05		1.90E-01	
0.0	2.25E-03	2.39E-02	.	.		4.80E-01	
.	.	.	.	.		.	
.	.	.	.	4.25E-05		.	
3.69E-01	-7.37E-04	1.56E-02	.	.		4.96E+00	
.	.	.	4.89E-05	5.60E-04		.	
.	.	.	.	-4.76E-03		.	
2.91E-01	4.95E-03	-3.17E-03	.	.		1.21E+01	
.	.	.	.	.		.	
5.06E-02	1.56E-02	1.68E-02	-2.76E-04	.		1.57E+01	
.	.	.	.	.		.	
4.84E-01	5.37E-02	-1.28E-02	.	.		1.78E+01	
4.52E-02	-1.35E-01	1.75E-02	-2.61E-03	.		1.87E+01	
2.53E-01	1.75E-01	3.40E-03	.	.		1.99E+01	
			.	.			
			.	.			
			.	.			
			1.64E-01	.			

TEMPORAL DERIVATIVES AT 0.0 HR

REFORMER	SHIFT 1	SHIFT 2	ABSORBER	STRIPPER	STREAM 2	METHANATOR
DT01	DTROT2	DTROT3	OXA	OXS	OX2	DTROT4
0.	0.	0.	2.3406E+00	-1.0124E-04	-2.2320E-05	0.
0.4097E+03	2.0505E+01	2.4115E+01	9.0202E-02	-1.2125E-04		1.0597E+02
0.4097E+03	2.5903E+01	1.2795E+01	9.7621E-02	-1.4000E-04		1.0031E+02
0.4097E+03	2.3605E+01	6.7800E+00	1.0556E-01	-1.5762E-04		9.4949E+01
0.4097E+03	2.1590E+01	3.6022E+00	1.1414E-01	-1.7427E-04		0.9075E+01
0.4097E+03	1.9601E+01	1.9114E+00	1.2343E-01	-1.9006E-04		0.5073E+01
0.4097E+03	1.7941E+01	1.0142E+00	1.3347E-01	-2.0512E-04		0.0527E+01
0.4097E+03	1.6356E+01	5.3016E-01	1.4434E-01	-2.1956E-04		7.6224E+01
0.4097E+03	1.4910E+01	2.0556E-01	1.5609E-01	-4.7025E-01		7.2151E+01
0.4097E+03	1.3593E+01	1.5152E-01	1.6000E-01			6.0296E+01
0.4097E+03	1.2392E+01	0.0402E-02	1.0255E-01			6.4647E+01
0.4097E+03	1.1297E+01	4.2663E-02	1.9743E-01			6.1192E+01
0.4097E+03	1.0299E+01	2.2630E-02	2.1353E-01			5.7923E+01
0.4097E+03	9.3694E+00	1.2012E-02	2.3095E-01			5.4028E+01
0.4097E+03	8.5601E+00	6.3740E-03	2.4900E-01			5.1090E+01
0.4097E+03	7.8041E+00	3.3622E-03	2.7020E-01			4.9125E+01
0.4097E+03	7.1150E+00	1.7947E-03	2.9220E-01			4.6500E+01
0.4097E+03	6.4067E+00	9.5229E-04	3.1617E-01			4.4015E+01
0.4097E+03	5.9140E+00	5.0531E-04	3.4204E-01			4.1664E+01
0.4097E+03	5.3910E+00	2.6013E-04	3.7005E-01			3.9437E+01
0.4097E+03	4.9150E+00	1.4227E-04	4.0037E-01			3.7330E+01
			4.3320E-01			
			4.6876E-01			
			5.0720E-01			
			5.4901E-01			
			5.9422E-01			
			6.4322E-01			
			6.9633E-01			
			7.5391E-01			
			8.1636E-01			
			8.8409E-01			
			9.5750E-01			
			1.0373E+00			
			1.1239E+00			

TEMPORAL DERIVATIVES AT 0.3 HR

REFORMER	SHIFT 1	SHIFT 2	ABSORBER	STRIPPER	STREAM 2	METHANATOR
OTDI	OTRDI2	OTRDI3	DXA	DXS	DX2	OTRDI4
0.	0.	0.	5.5014E-03	1.4959E-03	5.4425E-03	0.
1.0997E-01	4.2331E-03	2.0770E-03	5.0997E-03	1.3751E-03		5.2654E+00
1.0997E-01	2.0929E-02	3.6500E-03	5.0090E-03	1.2544E-03		1.3007E+01
1.0997E-01	-1.9492E-02	-2.0720E-02	6.0040E-03	1.1347E-03		2.4352E+01
2.1206E-01	1.0002E-02	0.3714E-03	5.9293E-03	1.0254E-03		3.0760E+01
2.0652E-01	0.6097E-03	-1.1175E-02	5.0169E-03	9.1077E-04		5.0361E+01
2.0652E-01	-1.4949E-02	1.6070E-02	5.3740E-03	7.6070E-04		0.2306E+01
1.3129E-01	2.1671E-02	-1.2552E-02	4.9220E-03	1.5027E-03		1.1119E+02
5.5691E-02	-9.0743E-03	6.7520E-03	4.1962E-03	-6.7154E-01		1.4177E+02
5.5691E-02	2.4193E-02	-1.6026E-02	3.4931E-03			1.7235E+02
5.5691E-02	-7.0705E-03	5.4411E-03	2.6156E-03			1.9904E+02
5.9270E-01	2.5072E-02	-2.5059E-02	1.9711E-03			2.2094E+02
5.9270E-01	-2.0095E-03	1.1247E-02	1.2207E-03			2.3677E+02
2.4406E-01	1.7533E-02	-1.6702E-02	0.1224E-04			2.4060E+02
2.4406E-01	4.0923E-03	-3.3502E-02	2.2410E-04			2.5770E+02
7.1597E-01	3.0627E-02	-2.1470E-02	1.1925E-04			2.6667E+02
7.1597E-01	1.3454E-02	-5.1435E-02	-4.0920E-04			2.7502E+02
3.0130E-01	1.5519E-02	-0.9354E-02	-1.0043E-04			2.8703E+02
3.0130E-01	-7.1127E-03	-2.0664E-01	-1.1500E-03			2.9952E+02
2.6607E-01	1.3521E-01	-3.6177E-01	-1.1549E-04			3.1303E+02
2.6607E-01	-1.6575E-01	-6.2910E-01	-2.0936E-03			3.3064E+02
			5.3006E-04			
			-3.0557E-03			
			2.3744E-03			
			-7.5060E-03			
			6.9905E-03			
			-1.5525E-02			
			1.0100E-02			
			-3.3643E-02			
			4.4390E-02			
			-7.5040E-02			
			1.0599E-01			
			-1.7030E-01			
			2.5021E-01			

TEMPORAL DERIVATIVES AT 0.5 HR

(STEADY STATE)

REFORMER	SHIFT 1	SHIFT 2	ABSORBER	STRIPPER	STREAM 2	METHANATOR
OTOT	OTROT2	OTROT3	OXA	OXS	OX2	OTROT4
0.	0.	0.	1.9700E-04	9.0939E-05	3.2774E-04	0.
0.	2.0497E-03	5.1469E-03	3.2661E-04	0.0466E-05		1.9030E-01
0.	2.2519E-03	2.3913E-02	2.3707E-04	0.4750E-05		4.7953E-01
1.3102E-01	-4.9044E-03	-7.5065E-03	3.0376E-04	7.2703E-05		1.6924E+00
1.3102E-01	-9.4779E-05	-2.5642E-03	2.0632E-04	6.2025E-05		2.0266E+00
3.6927E-01	-1.0252E-03	-1.2976E-02	2.5600E-04	4.2451E-05		3.5010E+00
3.6927E-01	-7.3717E-04	1.5614E-02	1.6911E-04	-3.2901E-05		4.9607E+00
9.4137E-02	2.0651E-03	-1.3420E-02	1.4740E-04	5.6044E-04		6.4672E+00
9.4137E-02	4.4992E-03	6.1560E-03	4.0050E-05	-4.7554E-03		0.7017E+00
9.4137E-02	3.7097E-03	7.9026E-03	5.9626E-05			1.0260E+01
2.9052E-01	4.9549E-03	-3.1706E-03	-7.9031E-05			1.2114E+01
2.9052E-01	4.5970E-03	-7.9122E-04	-1.0764E-04			1.3212E+01
4.1104E-01	-1.5961E-02	2.0009E-03	-2.2379E-04			1.4276E+01
4.1104E-01	4.1405E-03	-1.6741E-02	-2.0003E-04			1.4030E+01
5.0626E-02	1.5554E-02	1.6044E-02	-3.0500E-04			1.5692E+01
5.0626E-02	-1.7437E-02	-1.2926E-02	-2.7573E-04			1.5604E+01
4.0376E-01	7.5191E-03	1.5966E-02	-5.7403E-04			1.6650E+01
4.0376E-01	6.9052E-04	-1.3377E-02	-2.0401E-04			1.7065E+01
4.0376E-01	5.3724E-02	-1.2709E-02	-9.0405E-04			1.7004E+01
4.5205E-02	-1.3470E-01	1.7543E-02	-1.1774E-04			1.0602E+01
2.5279E-01	1.7543E-01	3.3957E-03	-1.4534E-03			1.9947E+01
			3.6769E-04			
			-2.6051E-03			
			1.6295E-03			
			-5.0504E-03			
			4.7942E-03			
			-1.0450E-02			
			1.2207E-02			
			-2.2557E-02			
			2.9762E-02			
			-4.9971E-02			
			7.0340E-02			
			-1.1253E-01			
			1.6426E-01			

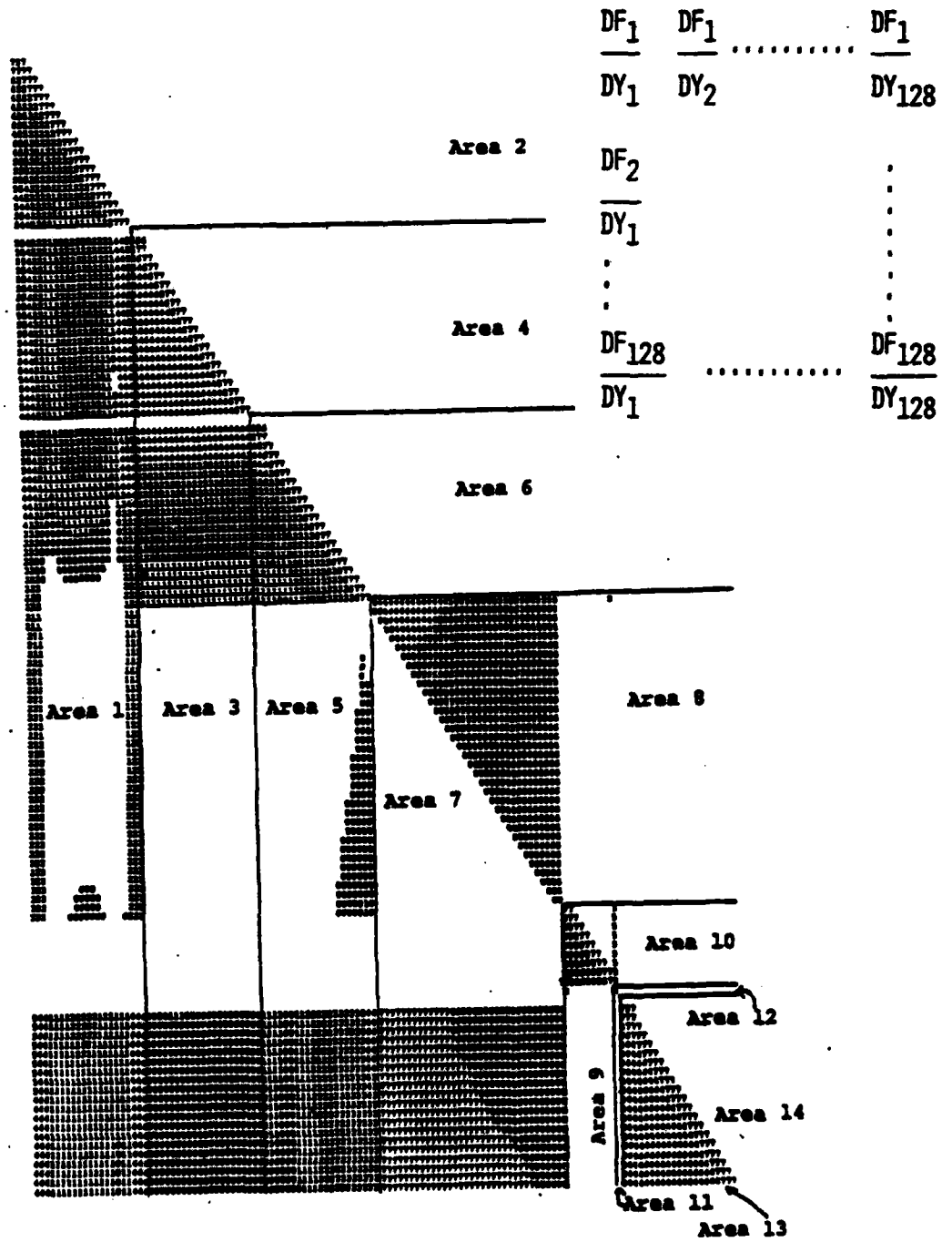
DEFINITION OF Y AND F

Y TEMPERATURE OF STEAM REFORMER, SHIFT REACTORS, METHANATION UNIT
MOLE FRACTION OF CO₂ IN LIQUID STREAM

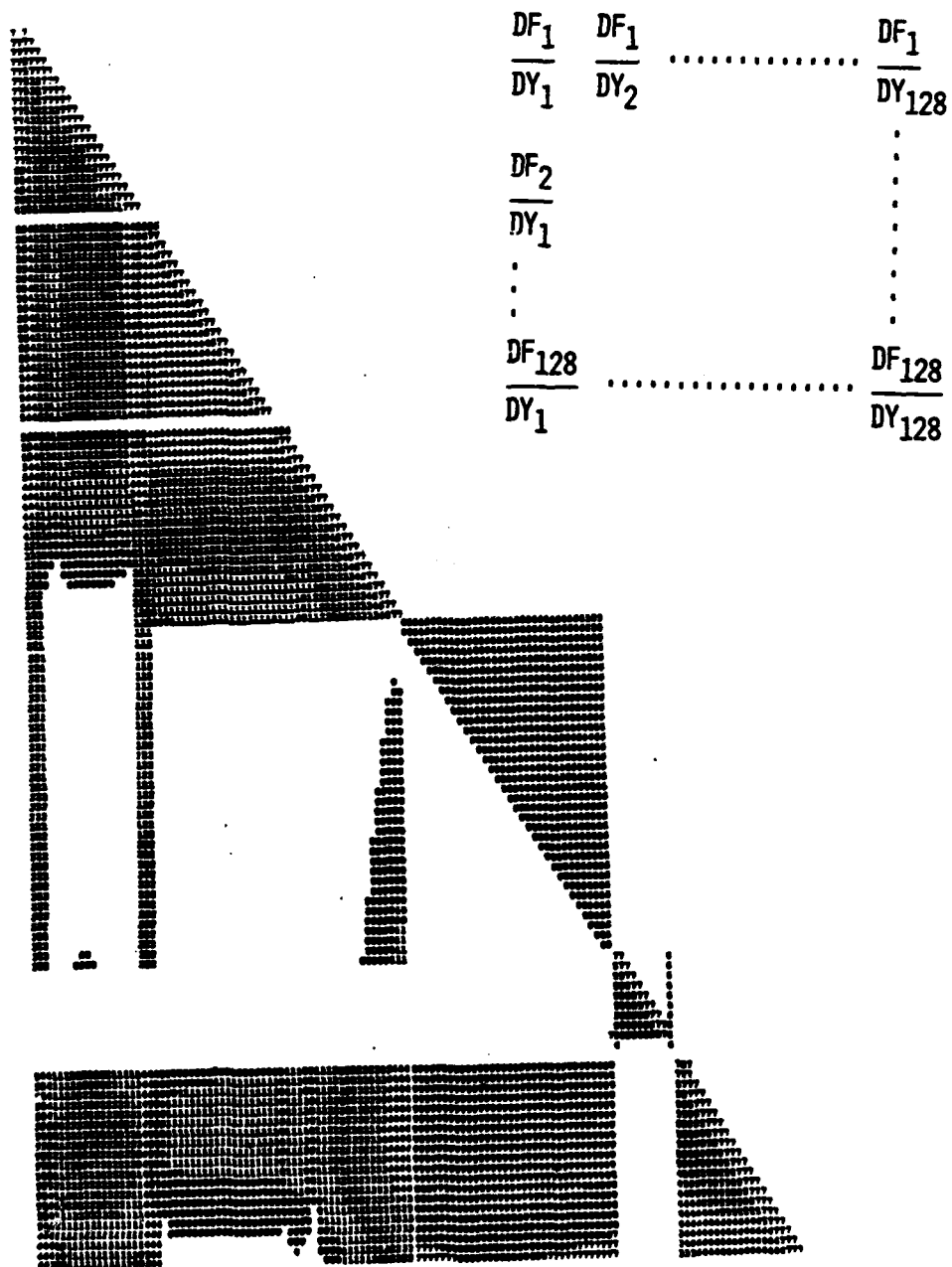
F TEMPORAL DERIVATIVE OF Y

$$\frac{DF_i}{DY_j} = \frac{D \left(\frac{DT_i}{DT} \right)}{DT_j}$$

MAP OF JACOBIAN MATRIX AT 0.05 HR.



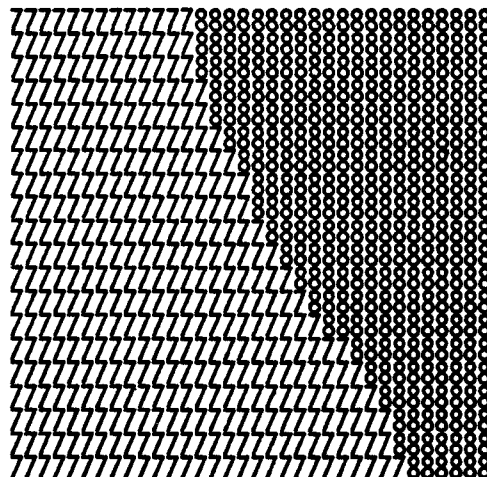
MAP OF JACOBIAN MATRIX AT STEADY STATE



$$\begin{array}{ccccccc}
 \frac{DF_1}{DY_1} & \frac{DF_1}{DY_2} & \dots & \dots & \dots & \dots & \frac{DF_1}{DY_{128}} \\
 & & & & & & \vdots \\
 \frac{DF_2}{DY_1} & & & & & & \vdots \\
 \vdots & & & & & & \vdots \\
 \vdots & & & & & & \vdots \\
 \frac{DF_{128}}{DY_1} & \dots & \dots & \dots & \dots & \dots & \frac{DF_{128}}{DY_{128}}
 \end{array}$$

COMPARISON OF A PORTION OF THE JACOBIAN MAP AT 0.05 HR AND 0.5 HR

0.05 HR



0.50 HR

