THE UNIVERSITY OF MICHIGAN

COLLEGE OF ENGINEERING

Departments of

Aerospace Engineering
Meteorology and Oceanography

High Altitude Engineering Laboratory

Technical Report

AMPLCT, A NUMERICAL INTEGRATION ROUTINE FOR SYSTEMS OF STIFF DIFFERENTIAL EQUATIONS

By (Stuart Wayne)
Stuart W. Bowen

ORA Project 033390

under contract with:

NATIONAL AERONAUTICS AND SPACE ADMINISTRATION

Grant NGR 23-005-360

WASHINGTON, D. C.

administered through:

OFFICE OF RESEARCH ADMINISTRATION, ANN ARBOR

June 1971

engn

UMR0578

CONTENTS

I. INTRODUCTION

The numerical integration of the ordinary (nonlinear) differential equations describing the steady flow of a reacting gas by the usual Runge-Kutta, Adams-Moulton or Hamming's methods typically are found to require a vanishingly small step size because of an extreme stability property of the differential equations known as "stiffness". Physically this situation arises from the very large spread in the time scales associated with the various hydrodynamic and chemical rates in the flow. The step size required by these usual explicit methods is always determined by the fastest rates even though the overall solution may not be determined by these fastest solution components. One would like to use a step size more in keeping with the changes in the overall solution.

Rather than discussing this phenomena further here, we refer the interested reader to Ref. 1-7 and the references contained therein. Of particular interest are Chapters 3 and 6 in Ref. 7.

In the following sections the equations required to integrate large sets of stiff differential equations which may have physically meaningful parasitic eigenvalues are developed following the procedure given in Ref. 2, 4-6. Since an algorithm which implements this very powerful method has not been previously given, a FORTRAN IV listing of a completely self contained subroutine is presented, together with the calling

requirements and helpful explanatory comments concerning the computer program. Some remarks are made regarding the program modifications needed if the matrix eigenvalues are to be computed and displayed, and an example solution is provided. Finally, comparisons between this method and other methods are given for several systems of equations.

II. DERIVATION OF EQUATIONS

A set of coupled differential equations can be written in vector notation as

$$\overline{w}' = \frac{d\overline{w}}{ds} = \overline{F}(\overline{w}) \tag{1}$$

The equation is assumed to have been written in the automorphic form so that the vector $\overline{F}$ does not explicitly contain the independent variable s. We then locally linearize the right hand side by performing an expansion of the ith component of the vector $\overline{F}$ (denoted by F_i) in a multidimensional Taylor series about the point $s_n = nh$. This gives, after substitution in (1) for the ith component of $\overline{w}'$ (denoted by w_i')

$$\begin{aligned} w_i' = F_{in} &+ (w_1 - w_{1n})\,(\partial F_i/\partial w_1)_n + \cdots \\ &+ (w_m - w_{mn})\,(\partial F_i/\partial w_m)_n \\ &+ O(|\overline{w} - \overline{w}_n|^2) \qquad i = 1 \text{ to } m \end{aligned} \tag{2}$$

The number of components of $\overline{w}$ is taken to be m; hence there are m coupled equations such as (2) in the set.

Note that when

$$\overline{w} = \overline{w}(s + nh) = \overline{w}_{n+1}$$

$$|\overline{w} - \overline{w}_n|^2 = |\overline{w}_{n+1} - \overline{w}_n|^2$$

$$= h^2 |w_n'|^2 + O(h^3)$$

Define the Jacobian matrix

$$[A] = (a_{ij}) = (\partial F_i / \partial w_j) \tag{3}$$

In many problems , explicit expressions for the matrix elements, a_{ij} cannot be conveniently given. In such cases the elements a_{ij} can be numerically computed by the approximate formula

$$a_{ij} = \partial F_i / \partial w_j \cong [F_i(1.005\ w_j) - F_i(.995\ w_j)]/.01\ w_j \tag{3a}$$

which is accurate to third order.

In terms of $[A]$ evaluated at s_n and denoted by $[A_n]$, Eq. (1) can be written

$$\overline{w}' = \overline{F}_n + [A_n]\ (\overline{w} - \overline{w}_n) + O(h^2)$$

$$= [A_n]\ \overline{w} + \overline{F}_n - [A_n]\ \overline{w}_n + O(h^2)$$

or

$$\overline{w}' = [A_n]\ \overline{w} + \overline{f}_n + O(h^2) \tag{4}$$

where

$$\overline{f}_n = \overline{F}_n - [A_n]\ \overline{w}_n \tag{5}$$

Note although Eq. (4) represents the local linearization of Eq. (1) (referenced to $s_n = nh$), it is the derivative $\overline{w}'$ that is represented as linear in the step-size h.

We use the modified Euler implicit difference scheme which is consistent with this error and is unconditionally stable to advance the solution

$$\overline{w}_{n+1} = \overline{w}_n + (h/2)(\overline{w}'_{n+1} + \overline{w}_n') \tag{6}$$

The local linearization has converted the set of differential equations into a set of m algebraic equations that must be solved by the methods of linear algebra. Inserting Eq. (4) into (6) gives

$$\overline{w}_{n+1} = \overline{w}_n + (h/2)([A_n]\ \overline{w}_{n+1} + \overline{f}_n + [A_n]\ \overline{w}_n + \overline{f}_n)$$

which can be rearranged to form

$$\begin{aligned}([I] - (h/2)[A_n])(\overline{w}_{n+1} - \overline{w}_n) &= h[A_n]\ \overline{w}_n + h\,\overline{f}_n \\ &= h([A_n]\ \overline{w}_n + \overline{F}_n - [A_n]\ \overline{w}_n) \\ &= h\,\overline{F}_n\end{aligned}$$

where $[I]$ is the unit matrix.

The increment in the dependent argument vector is given by

$$\Delta\overline{w} = \overline{w}_{n+1} - \overline{w}_n = ([I] - (h/2)[A_n])^{-1}\ h\,\overline{F}_n \tag{7}$$

where $([I] - (h/2)[A_n])^{-1}$

is the inverse of the matrix $([I] - (h/2)[A_n])$.

A measure of the "stiffness" of Eq. (1) is given by the ratio of the greatest to least eigenvalues of the matrix $[A]$. Standard explicit methods such as the fourth order Runge Kutta typically require a step size less than $\sim 2.78/|\lambda_{max}|$ for stability, where λ_{max} is the largest eigenvalue of $[A]$[7]. This small step size is governed not by the behavior of the solution as a whole but by the rapidly varying transients, i. e., largest eigenvalue, which may have ceased to be numerically important to the overall solution at large times.

The method by which the step size is controlled is quite important to the success and usefulness of a method. This will be illustrated in Section IX in which the same formula as Eq. (7) is used to advance the solution under two other step size control techniques.

The technique used here is described as follows:

1. Compute the increment in the independent argument $\Delta w_i = w_i^{(n+1)} - w_i^n$ for each i = 1 to m.

2. If any $|w_i| < \epsilon$, ignore it in the following tests. The value of ϵ used is 1×10^{-30}.

3. Find the largest $|\Delta w_i/w_i| = |\Delta w/w|_{max}$.

4. If $|\Delta w/w|_{max} > \delta$, divide the current step size in half, discard the calculated increments Δw_i and recalculate the step.

5. If $|\Delta w/w|_{max} < \delta/b$, double the current step size, discard the calculated increments Δw_i and recalculate the step.

6. If $\delta/b < |\Delta w/w|_{max} < \delta$, the calculations are considered satisfactory, and the calculated step is accepted.

Values for b and δ that have proved satisfactory are

$$b = 5$$

$$\delta = .05$$

This step size control works well for implicit techniques. It decreases the step size in regions where the largest eigenvalues are important in numerically determining the solution.

Since implicit techniques are stable for all negative eigenvalues, the value of the maximum eigenvalue itself should not be used to control step size.

All implicit techniques require the inversion of a matrix, usually at each step, to advance the solution. This matrix inversion typically requires of order $N^3/3$ operations for a matrix of order N and so becomes the operation requiring the most time for very large N, (say of order 50-100). For small N the evaluation of the derivative vector $\overline{F}$ is usually the most time consuming operation.

III. ACCURACY AND STABILITY

We now discuss in an elementary manner those factors that are important in determining the accuracy and stability of the numerical solutions to a set of differential equations and in particular discuss the

properties of the trapezoidal rule used in Section II. For a fuller discussion, Ref. (5, 7, 8 and 10) have proved helpful.

The accuracy of the numerical solution to an ordinary differential equation depends on two things; the truncation error at each step due to the finite step size and the retention of only a finite number of terms in the equivalent power series solution, and the stability of the finite difference method used. Stability of the finite difference method refers to the manner in which unwanted solutions that are inevitably introduced, are propagated from one step to another. We shall not be concerned here with what Fox[8] calls, inherent instability, wherein a small amount of a strongly increasing complementary solution to the differential equations, whose coefficients should be zero for the given initial conditions numerically, is later introduced and swamps the wanted solution, or strong instability in which spurious finite difference complementary solutions are introduced through the use of higher order difference equations than the differential equations to which they are applied. Fox refers to the stability of interest to us as partial instability since in principle it can always be overcome by selecting a sufficiently small step size.

To test the stability of a numerical method we consider a linear set of simultaneous differential equations written in the form

$$\overline{w}' = [A]\,\overline{w}(x) \tag{1}$$

where $[A]$ is a constant matrix which may be identified with the local Jacobian matrix of Section II for a nonlinear set of equations. The initial conditions are presumed to be specified by $\overline{w}(0)$.

It can be shown[7] that if the matrix $[A]$ is diagonalizable, that is, there exists a matrix $[S]$ such that $[S][A][S]^{-1} = [I]\,\overline{\lambda}$, then we can write

$$\overline{y} = [S]\,\overline{w}$$

and transform Eq. (1) to the form

$$\overline{y}' = [I]\,\overline{\lambda}\quad \overline{y} \tag{2}$$

If $[S][A][S]^{-1} = [I]\,\overline{\lambda}$, then multiplying from the left on $[S]^{-1}$ and transforming the right hand side yields $[A] = [I]\,\overline{\lambda}$ so that the elements of the vector $\overline{\lambda}$ are the eigenvalues of the matrix $[A]$. Expanding Eq. (2) we see that it is therefore sufficient to discuss a set of linear differential equations of the form

$$y_i' = \lambda_i\, y_i \tag{3}$$

for each i. Note it is only the eigenvalues λ_i of the matrix $[A]$ that are important in determining the behavior of the solution.

The analytic solution to Eq. (3) is

$$y = e^{\lambda x}\, y(0) \tag{4}$$

for each component.

We apply the modified Euler finite difference method

$$y_{n+1} = y_n + \frac{h}{2}(y'_{n+1} + y'_n) \tag{5}$$

to Eq. (3) and find

$$y_{n+1} = y_n + \frac{h\lambda}{2}\ y_{n+1} + \frac{h\lambda}{2} y_n$$

or

$$y_{n+1} = \left(\frac{1 + \frac{h\lambda}{2}}{1 - \frac{h\lambda}{2}} \right) y_n \tag{6}$$

The quantity in the brackets is known as the characteristic root μ and for the trapezoidal rule

$$\mu = \frac{1 + \frac{h\lambda}{2}}{1 - \frac{h\lambda}{2}} \tag{7}$$

The characteristic root when the simplest first order Euler method $y_{n+1} = y_n + h y'_n$ is used, can easily be shown to be

$$y_{n+1} = (1 + h\lambda)\, y_n \tag{8}$$

and

$$\mu = 1 + h\lambda$$

To understand the importance of the μ let Y = the true value of the computed solution y which also satisfies Eq. (3), i. e.

$$Y' = \lambda Y$$

If $e = Y - y$ is the error, then the error at the n^{th} step is $e_n = Y_n - y_n$. Now

$$e' = Y' - y'$$

$$\simeq \frac{\partial(\lambda Y)}{\partial Y} (Y - y)$$

or

$$e' = \lambda e$$

for small values of e. Thus the error e satisfies the same differential equation as does y. Applying the modified Euler method to the error equation yields

$$e_{n+1} = \frac{1 + \frac{h\lambda}{2}}{1 - \frac{h\lambda}{2}} \; e_n$$

and clearly if the absolute error is not to grow with n we must have

$$|\mu| = \left| \frac{1 + \frac{h\lambda}{2}}{1 - \frac{h\lambda}{2}} \right| \leq 1$$

Inherent instability is avoided by requiring all $\lambda_i < 0$. If a higher order difference is applied to Eq. (3) one generally finds more than one

characteristic root μ. If some of these μ are greater than unity then the corresponding difference solution will grow faster than the true solution. This is the case of strong instability.

For the simple Euler method we have $|\mu| \leq 1$ only for $-2 \leq h\lambda_i \leq 0$, while the modified Euler or trapezoidal rule allows for all i

$$-\infty < h\lambda_i < 0$$

in order that

$$|\mu| < 1$$

Thus the trapezoidal rule is at least stable for any negative eigenvalue no matter how large.

The characteristic roots of Eq. (8) and (6) are special cases of what are known as rational Padé approximations to the true solution $e^{h\lambda}$ given by Eq. (4). In general any implicit method generates a characteristic root $\mu(x)$ (where $x = h\lambda$) that can be identified with a particular Padé approximation to exp(x) which is written in the form

$$P_{n,m} = \sum_{i=0}^{n} a_i x^i \Big/ \sum_{k=0}^{m} b_k x^k$$

Equation (8) is $P_{1,0}$ while Eq. (6) is $P_{1,1}$

Even though the trapezoidal rule is stable for an arbitrarily large range of negative $h\lambda_i$, the accuracy of the $P_{1,1}$ approximation to $\exp(h\lambda_i)$ will be poor for the very largest negative values of λ_i if h is chosen to give

an accurate solution for some λ's whose absolute value is closer to zero and which may be more important in numerically determining the local solution. Note as $h\lambda \to -\infty$, $\mu \to -1$ for the implicit trapezoidal rule while $\exp(h\lambda) \to 0$. Even though the "stiff" components are not then accurately calculated, their numerical contribution to the overall solution is small. The virtue of the implicit method is that these stiff components no longer important in determining the solution, do not influence the stability and accuracy of the overall solution, and the step size can be adjusted to the remaining components to give accurate results. In other words the step size h can be freely adjusted to give accurate results with a practical value tailored to fit just those components most important in determining the local solution. This cannot be done using $P_{1,0}$, i. e. the simple Euler method. It is clear that once any component of a solution has gone unstable, subsequent values of the whole solution are in error. This may happen even if the truncation error per step may be unimportant.

If $\lambda_i > 0$, Eq. (7) indicates $|\mu| > 1$ and hence the absolute error per step will increase. In this case however the true solution is also increasing in value and it is the relative error that is important in determining the step size. The solution will be valid as long as no spurious component grows faster than the true solution.

For positive eigenvalues, $\lambda_i > 0$ it is clear that $P_{1,1}$ is a good approximation to exp(hλ) only for values of hλ considerably less than 1. For hλ = .4, the relative error of $P_{1,1}$ is 5.48×10^{-3}. The allowable step size is therefore limited by the largest positive eigenvalue for the case of the parasitic eigenvalues. Local eigenvalues of real physical problems can be locally parasitic[9], i. e., have opposite signs. This is the strong instability case of Fox[8].

In summary we see that for the modified Euler method h must be chosen to have $|h\lambda| < 1$ for the most important eigenvalues that determine the local solution but that the presence of very large negative λ_i will not affect the stability of the solution.

IV. CALLING REQUIREMENTS OF AMPLCT

Name:	AMPLCT
Purpose:	To obtain a solution of a system of first order ordinary differential equations $dY_i/dx = F_i(x, y_1, y_2, \ldots, y_{NEQ})$ i = 1 to NEQ which show extreme numerical "stiffness", and have parasitic eigenvalues.
Calling Sequence, FORTRAN:	CALL AMPLCT (V, NEQ, DERIV, QMAT, L, M)

Parameters:

V — is a real vector of dimension $\geq$ 8*NEQ + 3 containing the following information:

V(1) is an accuracy parameter controlling step size changes. V(1) is the maximum relative change in any of the independent variables. A value of V(1) $\approx$ .05 has proved satisfactory in past use.

V(2) = x, initial and current independent variable.

V(3) = h, the initial and current step size.

V(4), V(5), . . . V(4 + NEQ - 1) contain the NEQ values of the Y vector, both the initial and current values.

V(4 + NEQ), . . . V(4 + 2NEQ - 1) contain the NEQ values of the dY(I)/dx vector which are supplied by the derivative subroutine (i. e., DERIV).

NEQ — is the number of differential equations, written in the form dY(I)/dx = F(I). These must be written in the automorphic form if x appears explicitly in the dY/dx vector. This is done by having the last differential equation written in

the form dx/dx = 1 and using the dependent variable x in evaluating dY/dx.

As written NEQ $\leq$ 50.

DERIV — is the name of a subroutine supplied by the user that evaluates the derivatives dY/dx and stores them in V(4 + NEQ) through V(4 + 2*NEQ - 1). The name of this subroutine must be declared EXTERNAL. This subroutine has no arguments. The variable V must be declared in a block COMMON between the main program and DERIV. In the dY/dx evaluation in DERIV, the dependent value of x (if it appears) must be used, not the independent value of x.

QMAT — is a working matrix, which must be dimensioned exactly NEQ by NEQ in the main program.

L, M — are integer work vectors which must be dimensioned exactly NEQ in the main program.

QMAT, L, N — are used by the matrix inversion routine.

Usage:

Prior to calling, the user must set V(1) (=. 05 recommended), V(2) = initial value of x, V(3) = initial value of step size and V(4) through V(4 + NEQ - 1) to the NEQ values of the initial Y vector.

Each call to AMPLCT advances the solution one step. Upon return the new value of the independent variable x is stored in V(2) and the corresponding values of the solution vector Y are stored in V(4) through V(3 + NEQ). Only 8*NEQ + 3 elements of V are needed.

The auxiliary vectors Q and C used internally in AMPLCT need dimension $\geq$ NEQ**2. As written, the vectors Q and C limit NEQ $\leq$ 50.

Successful solutions have been obtained to systems of equations having the ratio of the absolute value of the greatest to least eigenvalues on the order of 10^8. Situations in which physically meaningful positive and negative eigenvalues occur (parasitic case) pose no special problem although the step size must be reduced inversely with the magnitude of the positive eigenvalue. The value of the largest positive eigenvalue has been found to be sensitive to the degree of precision in extreme cases[9].

The subroutine can be converted to double precision by removing the C from the IMPLICIT REAL*8 statement in AMPLCT, line 2, and in MINV, line 43, changing the function references ABS, AMAX1 in lines 50 and 51 of AMPLCT and line 69 of MINV to DABS and DAMAX1 respectively. In addition in AMPLCT, lines 12, 18, 25 and 50 the single precision power E should be changed to the double precision D power.

The subroutine MINV called from AMPLCT is the standard IBM routine, supplied from SSP.

V. FORTRAN IV LISTING OF AMPLCT AND MINV

The leading carat > indicates the beginning of a line, while the leading star * in MINV signifies the continuation of a wrapped around line. The leading number at the left is the line number and is not part of the FORTRAN statement.

```
>       1            SUBRØUTINE AMPLCT(V,NEQ,DERIV,QMAT,L,M)
>       2      C     IMPLICIT REAL*8(A-H,Ø-Z)
>       3            DIMENSIØN Q(2500),C(2500),V(1),
>       4           1 QMAT(NEQ,NEQ),L(1),M(1)
>       5            CALL DERIV
>       6            DØ 1 JEQ=1,NEQ
>       7            V(2*NEQ+3+JEQ)=V(3+JEQ)
>       8          1 V(5*NEQ+3+JEQ)=V(NEQ+3+JEQ)
>       9            DØ 2 JTERM=1,NEQ
>      10            DØ 9 JT=1,NEQ
>      11          9 V(3+JT)=V(2*NEQ+3+JT)
>      12            V(3+JTERM)=1.005*V(2*NEQ+3+JTERM)+1.E-30
>      13            CALL DERIV
>      14            DØ 3 JEQ=1,NEQ
>      15          3 V(6*NEQ+3+JEQ)=V(NEQ+3+JEQ)
>      16            DØ 109 JT=1,NEQ
>      17        109 V(3+JT)= V(2*NEQ+3+JT)
>      18            V(3+JTERM)=.995*V(2*NEQ+3+JTERM)-1.E-30
>      19            CALL DERIV
>      20            DØ 103 JEQ=1,NEQ
>      21        103 V(7*NEQ+3+JEQ)= V(NEQ+3+JEQ)
>      22         15 DØ 5 JEQ=1,NEQ
>      23            JEQTER=(JTERM-1)*NEQ+JEQ
>      24          5 C(JEQTER)=(V(6*NEQ+3+JEQ)-V(7*NEQ+3+JEQ))/
>      25           1  (0.010*V(2*NEQ+3+JTERM)+2.E-30)
>      26          2 CØNTINUE
>      27            JMAT=1
>      28         20 DØ 17 ICØL=1,NEQ
>      29            DØ 17 IRØW=1,NEQ
>      30            JEQTER=(ICØL-1)*NEQ+IRØW
>      31            QMAT(IRØW,ICØL)=-.5*V(3)*C(JEQTER)
>      32            IF(ICØL.EQ.IRØW) QMAT(IRØW,ICØL)=QMAT(IRØW,ICØL)+1.
>      33         17 CØNTINUE
>      34            CALL MINV(QMAT,NEQ,DETERM,L,M)
>      35            DØ 16 ICØL=1,NEQ
>      36            DØ 16 IRØW=1,NEQ
>      37            JEQTER=(ICØL-1)*NEQ+IRØW
>      38         16 Q(JEQTER)=QMAT(IRØW,ICØL)
>      39            DØ 7 JEQ=1,NEQ
>      40            V(3*NEQ+3+JEQ)=0.0
```

AMPLCT LISTING

```
41            DØ 8 JTERM=1,NEQ
42            JEQTER=(JTERM-1)*NEQ+JEQ
43          8 V(3*NEQ+3+JEQ)=V(3*NEQ+3+JEQ)+V(3)*Q(JEQTER)*
44           1 V(5*NEQ+3+JTERM)
45          7 CØNTINUE
46            JMAT=JMAT+1
47            IF(JMAT-10)21,19,19
48         21 AM=0.0
49            DØ 10 JEQ=1,NEQ
50            IF(ABS(V(2*NEQ+3+JEQ)).LE.1.E-30) GØ TØ 10
51            AMA=ABS(V(3*NEQ+3+JEQ)/V(2*NEQ+3+JEQ))
52            AM=AMAX1(AM,AMA)
53         10 CØNTINUE
54            IF(AM-V(1)) 11,11,12
55         11 IF(AM-0.2*V(1)) 18,18,19
56         12 V(3)=0.5*V(3)
57            GØ TØ 20
58         18 V(3)=2.0*V(3)
59            GØ TØ 20
60         19 DØ 14 JEQ=1,NEQ
61         14 V(3+JEQ)=V(2*NEQ+3+JEQ)+V(3*NEQ+3+JEQ)
62            V(2)=V(2)+V(3)
63            RETURN
64            END
```

AMPLCT LISTING, CONT.

```
>       1      C DECK  MINV
>       2      C
*
>       3      C     ..................................................
*............
>       4      C
*
>       5      C         SUBRØUTINE MINV
*
>       6      C
*
>       7      C         PURPØSE
*
>       8      C            INVERT A MATRIX
*
>       9      C
*
>      10      C         USAGE
*
>      11      C            CALL MINV(A,N,D,L,M)
*
>      12      C
*
>      13      C         DESCRIPTIØN ØF PARAMETERS
*
>      14      C            A - INPUT MATRIX, DESTRØYED IN CØMPUTATIØN AND
*REPLACED BY
>      15      C                RESULTANT INVERSE.
*
>      16      C            N - ØRDER ØF MATRIX A
*
>      17      C            D - RESULTANT DETERMINANT
*
>      18      C            L - WØRK VECTØR ØF LENGTH N
*
>      19      C            M - WØRK VECTØR ØF LENGTH N
*
>      20      C
*
>      21      C         REMARKS
*
>      22      C            MATRIX A MUST BE A GENERAL MATRIX
*
>      23      C
*
>      24      C         SUBRØUTINES AND FUNCTIØN SUBPRØGRAMS REQUIRED
*
```

MINV LISTING

```
>      25      C           NØNE
*
>      26      C
*
>      27      C        METHØD
*
>      28      C           THE STANDARD GAUSS-JØRDAN METHØD IS USED. THE D
*ETERMINANT
>      29      C           IS ALSØ CALCULATED. A DETERMINANT ØF ZERØ INDIC
*ATES THAT
>      30      C           THE MATRIX IS SINGULAR.
*
>      31      C
*
>      32      C     ....................................................
*............
>      33      C
*
>      34            SUBRØUTINE MINV(A,N,D,L,M)
*
>      35            DIMENSIØN A(1),L(1),M(1)
*
>      36      C
*
>      37      C        ...............................................
*............
>      38      C
*
>      39      C        IF A DØUBLE PRECISIØN VERSIØN ØF THIS RØUTINE IS D
*ESIRED, THE
>      40      C        C IN CØLUMN 1 SHØULD BE REMØVED FRØM THE DØUBLE PR
*ECISIØN
>      41      C        STATEMENT WHICH FØLLØWS.
*
>      42      C
*
>      43      C     DØUBLE PRECISIØN A,D,BIGA,HØLD
*
>      44      C
*
>      45      C        THE C MUST ALSØ BE REMØVED FRØM DØUBLE PRECISIØN S
*TATEMENTS
>      46      C        APPEARING IN ØTHER RØUTINES USED IN CØNJUNCTIØN WI
*TH THIS
>      47      C        RØUTINE.
*
>      48      C
*
>      49      C        THE DØUBLE PRECISIØN VERSIØN ØF THIS SUBRØUTINE MU
*ST ALSØ
>      50      C        CØNTAIN DØUBLE PRECISIØN FØRTRAN FUNCTIØNS.  ABS I
*N STATEMENT
```

```
>      51      C         10 MUST BE CHANGED TØ DABS.
*
>      52      C
*
>      53      C         ..................................................
*.............
>      54      C
*
>      55      C         SEARCH FØR LARGEST ELEMENT
*
>      56      C
*
>      57                D=1.0
*
>      58                NK=-N
*
>      59                DØ 80 K=1,N
*
>      60                NK=NK+N
*
>      61                L(K)=K
*
>      62                M(K)=K
*
>      63                KK=NK+K
*
>      64                BIGA=A(KK)
*
>      65                DØ 20 J=K,N
*
>      66                IZ=N*(J-1)
*
>      67                DØ 20 I=K,N
*
>      68                IJ=IZ+I
*
>      69           10 IF( ABS(BIGA)- ABS(A(IJ))) 15,20,20
*
>      70           15 BIGA=A(IJ)
*
>      71                L(K)=I
*
>      72                M(K)=J
*
>      73           20 CØNTINUE
*
>      74      C
*
```

MINV LISTING, CONT.

```
>      75      C         INTERCHANGE RØWS
*
>      76      C
*
>      77              J=L(K)
*
>      78              IF(J-K) 35,35,25
*
>      79           25 KI=K-N
*
>      80              DØ 30 I=1,N
*
>      81              KI=KI+N
*
>      82              HØLD=-A(KI)
*
>      83              JI=KI-K+J
*
>      84              A(KI)=A(JI)
*
>      85           30 A(JI) =HØLD
*
>      86      C
*
>      87      C         INTERCHANGE CØLUMNS
*
>      88      C
*
>      89           35 I=M(K)
*
>      90              IF(I-K) 45,45,38
*
>      91           38 JP=N*(I-1)
*
>      92              DØ 40 J=1,N
*
>      93              JK=NK+J
*
>      94              JI=JP+J
*
>      95              HØLD=-A(JK)
*
>      96              A(JK)=A(JI)
*
>      97           40 A(JI) =HØLD
*
>      98      C
*
>      99      C         DIVIDE CØLUMN BY MINUS PIVØT (VALUE ØF PIVØT ELEME
*NT IS
>     100      C         CØNTAINED IN BIGA)
*
```

MINV LISTING, CONT.

```
>     101      C
*
>     102         45 IF(BIGA) 48,46,48
*
>     103         46 D=0.0
*
>     104            RETURN
*
>     105         48 DØ 55 I=1,N
*
>     106            IF(I-K) 50,55,50
*
>     107         50 IK=NK+I
*
>     108            A(IK)=A(IK)/(-BIGA)
*
>     109         55 CØNTINUE
*
>     110      C
*
>     111      C         REDUCE MATRIX
*
>     112      C
*
>     113            DØ 65 I=1,N
*
>     114            IK=NK+I
*
>     115            HØLD=A(IK)
*
>     116            IJ=I-N
*
>     117            DØ 65 J=1,N
*
>     118            IJ=IJ+N
*
>     119            IF(I-K) 60,65,60
*
>     120         60 IF(J-K) 62,65,62
*
>     121         62 KJ=IJ-I+K
*
>     122            A(IJ)=HØLD*A(KJ)+A(IJ)
*
>     123         65 CØNTINUE
*
>     124      C
*
```

MINV LISTING, CONT.

```
>     125      C         DIVIDE RØW BY PIVØT
*
>     126      C
*
>     127               KJ=K-N
*
>     128               DØ 75 J=1,N
*
>     129               KJ=KJ+N
*
>     130               IF(J-K) 70,75,70
*
>     131          70 A(KJ)=A(KJ)/BIGA
*
>     132          75 CØNTINUE
*
>     133      C
*
>     134      C         PRØDUCT ØF PIVØTS
*
>     135      C
*
>     136               D=D*BIGA
*
>     137      C
*
>     138      C         REPLACE PIVØT BY RECIPRØCAL
*
>     139      C
*
>     140               A(KK)=1.0/BIGA
*
>     141          80 CØNTINUE
*
>     142      C
*
>     143      C         FINAL RØW AND CØLUMN INTERCHANGE
*
>     144      C
*
>     145               K=N
*
>     146         100 K=(K-1)
*
>     147               IF(K) 150,150,105
*
>     148         105 I=L(K)
*
>     149               IF(I-K) 120,120,108
*
>     150         108 JQ=N*(K-1)
*
```

MINV LISTING, CONT.

```
>     151            JR=N*(I-1)
*
>     152            DØ 110 J=1,N
*
>     153            JK=JQ+J
*
>     154            HØLD=A(JK)
*
>     155            JI=JR+J
*
>     156            A(JK)=-A(JI)
*
>     157        110 A(JI) =HØLD
*
>     158        120 J=M(K)
*
>     159            IF(J-K) 100,100,125
*
>     160        125 KI=K-N
*
>     161            DØ 130 I=1,N
*
>     162            KI=KI+N
*
>     163            HØLD=A(KI)
*
>     164            JI=KI-K+J
*
>     165            A(KI)=-A(JI)
*
>     166        130 A(JI) =HØLD
*
>     167            GØ TØ 100
*
>     168        150 RETURN
*
>     169            END
*
```

MINV LISTING, CONT.

VI. COMMENTS FOR SUBROUTINE AMPLCT LISTING (F, w refer to Eq. (II-7))

Line 5	DERIV is called to evaluate F's from current w's.
Line 7	The independent values, i. e., the w's JEQ = 1 to NEQ read into V(3 + JEQ) are stored in V(2*NEQ + 3 + JEQ).
Line 8	The values of the derivative functions F's, JEQ = 1 to NEQ, evaluated in DERIV and passed through in V(NEQ + 3 + JEQ), are stored in V(5*NEQ + 3 + JEQ).
Line 11	The w's are reset in V(3 + JT), JT = 1 to NEQ.
Line 12	The current value w_i only is multiplied by 1. 005 in preparation to call DERIV.
Line 15	The new F's for $w_i = 1.005\ w_i$ are stored in V(6*NEQ + 3 + JEQ) JEQ = 1 to NEQ
Line 17	The w's are reset
Line 18	The current value of w_i only is multiplied by . 995 in preparation for call to DERIV.
Line 21	The new F's for $w_i = .995\ w_i$ are stored in V(7*NEQ + 3 + JEQ) JEQ = 1 to NEQ.
Line 24	The value of the matrix [A], whose elements are $a_{ij} = \partial F_i / \partial w_j$ is calculated and stored column by column in the linear array C.

Line 27 — JMAT is a counter incremented by one each time a trial solution cycle has been completed. No more than 10 successive doublings or halvings of the step size are permitted.

Line 31, 32 — The matrix [[I] - (h/2)[A]] is made up and stored in the two dimensional matrix QMAT.

Line 34 — The subroutine MINV inverts QMAT and returns the inverted elements in QMAT itself.

Line 38 — The two dimensional array QMAT is loaded column by column into the linear array Q.

Line 43 — The increment for each independent argument

$$\Delta w_i = w_i^{(n+1)} - w_i^{(n)} = h F_i \sum_j (\delta_{ij} - (h/2)\, a_{ij})^{-1}$$

is calculated and stored in V(3*NEQ + 3 + JEQ), JEQ = 1 to NEQ.

Line 47 — No more than 10 step size changes and subsequent recalculations of the whole step are allowed. If JMAT $\geq$ 10, the test for the maximum relative change is deleted and the results of the step are accepted.

Line 48-59 — The largest value of $|\Delta w_i / w_i|$ is found and stored in AM; to avoid division by zero, this test is deleted for any $|w_i| < 1 \times 10^{-30}$. The step size h is changed as follows: If $|\Delta w_i / w_i|_{max} > V(1)$, divide current step size h (stored in V(3)) by 2.0, discard the results of

the calculated increments Δw_i and recalculate. If $|\Delta w_i/w_i|_{max} < (1/5)\ V(1)$, double the current step size, discard the calculated results and recalculate. If $(1/5)\ V(1) < |\Delta w_i/w_i|_{max} < V(1)$ the calculations are considered satisfactory.

Line 61 — The dependent arguments, w's are advanced.

Line 62 — The independent argument is advanced.

VII. MODIFICATIONS TO AMPLCT FOR EIGENVALUE DISPLAY

A principle advantage of the scheme given here is that the eigenvalues of the matrix [A] do not need to be explicitly determined. There may be situations however wherein the actual values of the eignevalues are wanted.

To do this create a storage matrix A dimensioned NEQ X NEQ. This can be easily done by including A as an additional argument in the argument list for AMPLCT and dummy dimensioning A(NEQ, NEQ) in AMPLCT. The actual dimensioning of A is then done in the main program where the value of NEQ is known. Between lines 30 and 31 in AMPLCT, load A from the array C as follows:

A(IROW, ICOL) = C(JEQTER)

Then at preselected intervals set by a counter (which may be added as an additional argument in AMPLCT) after a step has been successful, the eigenvalues of A can be obtained for display by performing a decomposition

of the matrix A into a lower triangular matrix having 1's on principle diagonal and an upper triangular matrix[10]. The values on the principle diagonal of the upper triangular matrix are the eigenvalues of A. Such decompositions, done by Gaussian elimination and partial pivoting, are usually available as library subroutines. In Ref. (11), a FORTRAN IV program that does this decomposition by the Rutishauser method is given.

VIII. EXAMPLE SOLUTION

To illustrate the use of AMPLCT, a complete FORTRAN IV program and the resulting computer output are given for the following equations[12]:

$$\frac{dy_1}{dx} = y'_1 = -2000\, y_1 + 1000\, y_2 + 1$$

$$\frac{dy_2}{dx} = y'_2 = y_1 - y_2$$

with $y_1(0) = 0$, $y_2(0) = 0$, from $x = 0.$ to 4.

The eigenvalues are $\lambda_1 = -2000.5$ and $\lambda_2 = -.5$ and are constant because the equations are linear. These equations are already in automorphic form since the independent variable x does not appear on right sides so that it really was unnecessary to include the third equation,

$$\frac{dx}{dx} = y'_3 = 1. \quad , \quad y_3(0) =).$$

but doing so effectively includes the independent argument in the step size test.

Due to the step size test on $|\Delta y/y|$, the solution was not started at $x = 0$, but at $x = 1 \times 10^{-3}$ to avoid an unnecessarily small initial step size, since all initial conditions were zero. The 360/70 CPU time required for this example was .991 sec for 65 steps using a rather coarse value of the accuracy parameter G(1) = .50.

The CPU times and number of steps for G(1) = .10 and .05 were 4.072 sec and 8.050 sec for 625 steps respectively. In each case about 30% of the steps were required to reach x = .01.

```
>       EXTERNAL DERA
>       C0MM0N/D/G(50)
>C    INPUT NEEDED: G(1)=ACCURACY,G(2)=XSTART,G(50)=XMAX
>C    AND G(3)=INITIAL STEP SIZE
>C    WITH G(4),G(5),G(6) AS INITIAL C0NDITI0NS
>       NAMELIST/INPUT/G
>       DIMENSI0N QMAT(3,3) ,L(3),M(3)
>     1 READ(5,INPUT)
>       WRITE(6,110) G(1),G(2),G(3),G(4),G(5),G(6)
>   110 F0RMAT('1 ACCURACY PARAM.,  G(1)=',1PE15.6/
>      1'  INDEPEN. VARIABLE, G(2)=',E15.6/
>      2' STEP SIZE, G(3)=',E15.6/
>      3' DEPEN. VARIABLE, G(4)=',E15.6/
>      4' DEPEN. VARIABLE, G(5)=',E15.6/
>      5 ' DEPENDENT VARIABLE G(6) = ', E15.6)
>       WRITE(6,101)
>       N=3
>       WRITE(6,100)G(2),G(4),G(5),G(6)
>   100 F0RMAT(' ',1P4E16.6)
>   101 F0RMAT('1',8X,'X',15X,'Y(1)',12X,'Y(2)',12X,'Y(3)',/)
>     6 CALL AMPLCT(G,N,DERA,QMAT,L,M)
>       WRITE(6,100)G(2),G(4),G(5),G(6)
>       IF(G(2).LE.G(50)) G0 T0 6
>       END
>       SUBR0UTINE DERA
>       C0MM0N/D/G(50)
>       G(7)=-2000.*G(4)+1000.*G(5)+1.
>       G(8)=G(4)-G(5)
>       G(9)=1.
>       RETURN
>       END
```

EXAMPLE PROGRAM LISTING

```
> &INPUT    G=.50,1.E-3,.01,5.0012E-4,2.4993E-7,1.E-3,G(50)=4.,    &END
```

INPUT DATA FOR EXAMPLE PROGRAM

```
>  ACCURACY PARAM., G(1)=   5.000000E-01
>  INDEPEN. VARIABLE, G(2)=   9.999999E-04
> STEP SIZE, G(3)=   9.999998E-03
> DEPEN. VARIABLE, G(4)=   5.001200E-04
> DEPEN. VARIABLE, G(5)=   2.499300E-07
> DEPENDENT VARIABLE G(6) =    9.999999E-04
```

X	Y(1)	Y(2)	Y(3)
9.999999E-04	5.001200E-04	2.499300E-07	9.999999E-04
1.156250E-03	5.001265E-04	3.280294E-07	1.156250E-03
1.312500E-03	5.001419E-04	4.061180E-07	1.312500E-03
1.468749E-03	5.001635E-04	4.841975E-07	1.468749E-03
1.624999E-03	5.001901E-04	5.622686E-07	1.624999E-03
1.781249E-03	5.002199E-04	6.403317E-07	1.781249E-03
1.937499E-03	5.002522E-04	7.183875E-07	1.937499E-03
2.093749E-03	5.002865E-04	7.964362E-07	2.093749E-03
2.406249E-03	5.003582E-04	9.525139E-07	2.406249E-03
2.718749E-03	5.004329E-04	1.108565E-06	2.718749E-03
3.031249E-03	5.005093E-04	1.264591E-06	3.031249E-03
3.343749E-03	5.005863E-04	1.420593E-06	3.343749E-03
3.656249E-03	5.006639E-04	1.576569E-06	3.656249E-03
4.281245E-03	5.008194E-04	1.888448E-06	4.281245E-03
4.906245E-03	5.009754E-04	2.200231E-06	4.906245E-03
5.531244E-03	5.011314E-04	2.511916E-06	5.531244E-03
6.156243E-03	5.012872E-04	2.823504E-06	6.156243E-03
6.781243E-03	5.014429E-04	3.134995E-06	6.781243E-03
8.031242E-03	5.017542E-04	3.757686E-06	8.031242E-03
9.281240E-03	5.020658E-04	4.379988E-06	9.281240E-03
1.053124E-02	5.023766E-04	5.001902E-06	1.053124E-02
1.178124E-02	5.026872E-04	5.623426E-06	1.178124E-02
1.303124E-02	5.029982E-04	6.244561E-06	1.303124E-02
1.553123E-02	5.036192E-04	7.485672E-06	1.553123E-02
1.803123E-02	5.042385E-04	8.725228E-06	1.803123E-02

COMPUTER OUTPUT

>	2.053123E-02	5.048583E-04	9.963241E-06	2.053123E-02
>	2.303123E-02	5.054765E-04	1.119971E-05	2.303123E-02
>	2.553122E-02	5.060937E-04	1.243463E-05	2.553122E-02
>	3.053122E-02	5.073270E-04	1.489984E-05	3.053122E-02
>	3.553122E-02	5.085573E-04	1.735889E-05	3.553122E-02
>	4.053122E-02	5.097836E-04	1.981182E-05	4.053122E-02
>	4.553122E-02	5.110074E-04	2.225861E-05	4.553122E-02
>	5.053122E-02	5.122274E-04	2.469930E-05	5.053122E-02
>	6.053122E-02	5.146612E-04	2.956245E-05	6.053122E-02
>	7.053119E-02	5.170791E-04	3.440131E-05	7.053119E-02
>	8.053118E-02	5.194889E-04	3.921604E-05	8.053118E-02
>	9.053117E-02	5.218831E-04	4.400678E-05	9.053117E-02
>	1.005312E-01	5.242690E-04	4.877364E-05	1.005312E-01
>	1.205311E-01	5.289924E-04	5.823614E-05	1.205311E-01
>	1.405311E-01	5.336907E-04	6.760456E-05	1.405311E-01
>	1.605311E-01	5.383205E-04	7.687985E-05	1.605311E-01
>	1.805311E-01	5.429175E-04	8.606281E-05	1.805311E-01
>	2.005311E-01	5.474603E-04	9.515442E-05	2.005311E-01
>	2.405310E-01	5.564245E-04	1.130676E-04	2.405310E-01
>	2.805310E-01	5.651971E-04	1.306262E-04	2.805310E-01
>	3.205310E-01	5.738169E-04	1.478372E-04	3.205310E-01
>	3.605309E-01	5.822424E-04	1.647075E-04	3.605309E-01
>	4.005309E-01	5.905265E-04	1.812439E-04	4.005309E-01
>	4.805309E-01	6.065653E-04	2.133440E-04	4.805309E-01
>	5.605308E-01	6.220061E-04	2.441856E-04	5.605308E-01
>	6.405308E-01	6.368090E-04	2.738182E-04	6.405308E-01
>	7.205308E-01	6.510646E-04	3.022887E-04	7.205308E-01
>	8.005308E-01	6.647294E-04	3.296430E-04	8.005308E-01
>	9.605308E-01	6.905228E-04	3.811962E-04	9.605308E-01
>	1.120530E 00	7.143167E-04	4.287849E-04	1.120530E 00
>	1.280530E 00	7.362934E-04	4.727137E-04	1.280530E 00
>	1.440530E 00	7.565685E-04	5.132644E-04	1.440530E 00
>	1.600530E 00	7.752941E-04	5.506966E-04	1.600530E 00
>	1.920529E 00	8.085738E-04	6.172450E-04	1.920529E 00
>	2.240529E 00	8.369293E-04	6.739367E-04	2.240529E 00
>	2.560529E 00	8.610785E-04	7.222313E-04	2.560529E 00
>	2.880528E 00	8.816605E-04	7.633730E-04	2.880528E 00
>	3.200528E 00	8.991791E-04	7.984205E-04	3.200528E 00
>	3.840528E 00	9.269945E-04	8.540163E-04	3.840528E 00
>	4.480527E 00	9.471201E-04	8.942788E-04	4.480527E 00

COMPUTER OUTPUT, CONT.

IX. COMPARISON TO OTHER METHODS

It is of interest to compare AMPLCT with other methods devised to integrate stiff equations. Lapidus and Seinfeld[7], p. 286, have tabulated the running time and accuracy of numerical experiments on several examples of stiff differential equations. Their results were obtained with an IBM 7094 while the results reported here for AMPLCT were obtained using an IBM 360/67. IBM 7094 and 360/67 machine times are not directly comparable, so to make the run time comparisons meaningful the results for each example on each machine have been normalized by their respective CPU times required for the classical fixed step 4^{th} order Runge Kutta technique using the same step size over the same range of independent variable. If only a portion of the Runge Kutta solution was computed due to excessive run time, the full run time was estimated on the basis that the time per step was approximately constant.

Two of Lapidus and Seinfeld's four examples were chosen, the example numbers being theirs.

Example 1

$$y' = -200(y - F(x)) + F'(x)$$

where $F(x) = 10 - (10 + x)\, e^{-x}$ with $y(0) = 10$. The solution is desired between $x = 0$ and $x = 15$. The exact solution is

$$y = F(x) + 10\, e^{-200x}$$

Example 1 has two components, the eigenvalues being -200 and -1 and is only moderately stiff and not parasitic. The solution component e^{-200x} decays very rapidly compared to e^{-x}.

Example 4

$$y_1' = -.04y_1 + 1 \times 10^4 y_2 y_3$$

$$y_2' = .04y_1 - 1 \times 10^4 y_2 y_3 - 3 \times 10^7 y_2^2$$

$$y_3' = 3 \times 10^7 y_2^2$$

with $y_1(0) = 1$, $y_2(0) = 0$, $y_3(0) = 0$. The solution is wanted between x = 0 and x = 40. The eigenvalues are $\lambda_1 = 0$ and,

$$\lambda^2 + (.04 + 1 \times 10^4 y_3 + 6 \times 10^7 y_2)\lambda + (.24 \times 10^7 y_2 + 6 \times 10^{11} y_2^2) = 0$$

At x = 0 the eigenvalues are (0, 0, and -.04) while for large x they become (0, 0, and -1×10^4) with the absolute value of the largest eigenvalue changing from .04 to 2405 in the range x = 0 to .02. The system is fairly stiff.

The various implicit methods use the following basic equation to advance the step

$$y_{n+1} = y_n + w_1 k_1 + w_2 k_2$$

where

$$k_1 = h[f(y_n) + a_1 A(y_n) k_1]$$

$$k_2 = h[f(y_n + b_1 k_1) + a_2 A(y_n + c_1 k_1) h_2]$$

for the equation $y' = f(y)$ with $A(y) = \partial f/\partial y$ being either a scalar or the Jacobian matrix. The values of a_1, a_2, b_1, c_1, w_1 and w_2 depend on the particular method. The trapezoidal rule uses

$$a_1 = 1/2,\ w_1 = 1,\ w_2 = 0$$

Some of the better methods cited by Lapidus and Seinfeld were:

RK4 — Classical, fixed step fourth order Runge Kutta, used for the time normalization.

TR — Trapezoidal procedure in which from y_n at x_n, the trapezoidal rule is used twice to give two successive values y_{n+1}, y_{n+2} and then the first new value y_{n+1} is replaced by $(y_n + 2y_{n+1} + y_{n+2})/4$

TR-EX — Trapezoidal rule using the input filtering described above and global extrapolation. In global extrapolation after integration over the whole interval, the step size is decreased by a factor of 2 and the whole problem is redone. The values at a particular point were then obtained by Romberg extrapolation to the limit after three such cycles of halving.

CAL

Calahan's method using $a_1 = a_2 = .788675134$, $b_1 = -1.15470054$, $c_1 = 0$, $w_1 = 3/4$, $w_2 = 1/4$. The characteristic root μ is given by

$$\mu = \frac{1 - .578\,h\lambda - .456\,(h\lambda)^2}{1 - 1.578\,h\lambda + .622\,(h\lambda)^2}$$

LW1

Liniger and Willoughby's method using exponential fitting. The step is advanced using

$$y_{n+1} = y_n + h[(1 - \beta)y'_{n+1} + \beta y'_n]$$

which has a characteristic root

$$\mu = \frac{1 + \beta h\lambda}{1 - (1 - \beta)\, h\lambda}$$

The constant β is determined by requiring μ to approximate the exponential $\mu(h\lambda) = e^{h\lambda}$ which yields

$$\beta = -\frac{1}{h\lambda} - \frac{1}{e^{-h\lambda} - 1}$$

The largest negative eigenvalue is chosen to make the exponential fit. The backwards Euler method uses $\beta = 0$ which corresponds to a fit at $h\lambda \to -\infty$.

ARK

A classical fixed step fourth order Runge-Kutta method, set up so as to use the same calling conventions as AMPLCT and used for the time normalization.

The numerical results using these methods are summarized in Tables 1-4. The relative accuracy R_n is given by $|y_n - Y(x_n)|/Y(x_n)$ where y_n is the computed value and Y is the true value. The errors are shown at two values of x. The CPU times, normalized by the classical fourth order Runge-Kutta method using the step size noted, are given in the last column.

For example 4, the solution obtained with RK4 or ARK using a step size h = .001 was assumed exact. The parameter V(1) noted with AMPLCT is the relative accuracy parameter which controls the step size.

In addition to the previous methods, a recent stiff method developed by Gear[7, 13, 14] was run with the 360/67 using example 4. Gear's method is a predictor and iterated corrector method in which both the order and step size are automatically selected to give the largest step and hence reduce the overall calculation time.

A FORTRAN IV listing of the alogrithm is given in Ref. (14). The result of Gear's method is given in Table 5.

Table 2 shows the almost linear increase of computing time with ARK that occurs with decreasing step size. The largest relative error in example 1 occurs near x = .016 for all methods. Even though a step size of h = .01 was small enough so that a stable solution was possible using ARK , the accuracy at x = .016 is unacceptable for this step size. For example 1, AMPLCT appears to offer no marked advantage over the stiff methods cited by Lapidus and Seinfeld.

Method	Step Size h	Relative Accuracy(R_n)		Relative Time t/t_{RK4}
		x = .4	x = 10.	
RK4	.01	1.0^{-5}*	2.0^{-9}	1.0
TR	.2	1.85^{-2}	4.3^{-5}	.182
TR-EX	.2	1.4^{-4}	1.0^{-8}	3.26
CAL	.01/.2**	1.7^{-2}	4.0^{-8}	.091
LW1	.2	1.1^{-3}	5.0^{-8}	.273

TABLE 1. Example 1 Accuracy and Relative Run Times from (7). t_{RK4} = 11 sec to x = 15 using h = .01.

* $1.0^{-5} \equiv 1.0 \times 10^{-5}$

** h changed from .01 to .2 at x = .1

Method	Step Size h	Relative Accuracy (R_n) x = .016	.4	10	Relative Time t/t_{ARK}
ARK	.001	1.91^{-5}	$<1^{-7}$	$<1^{-9}$	8.85
	.002	7.04^{-4}	$<1^{-7}$	$<1^{-9}$	4.65
	.005	4.66^{-2}	6.55^{-7}	$<1^{-9}$	1.95
	.010	2.57	2.35^{-5}	1.03^{-9}	1.0
AMPLCT V(1) = .05		1.13^{-4}	1.14^{-5}	2.98^{-7}	.713
AMPLCT V(1) = .10		2.61^{-4}	4.58^{-5}	5.72^{-7}	.332

TABLE 2. Example 1 Accuracy and Relative Run Times Using ARK and AMPLCT. t_{ARK} = 12.2 sec to x = 15 using h = .01

Method	Step Size h	R_{1n} x = .4	R_{1n} 10	R_{2n} .4	R_{2n} 10	R_{3n} .4	R_{3n} 10	Relative Time t/t_{RK4}
RK4	.001	0	0	0	0	0	0	1.0
TR	.2	1.35^{-3}	1.05^{-3}	2.12^{-1}	2.4^{-1}	9.0^{-2}	1.5^{-2}	.0674
TR-EX	.2	1.72^{-5}	3.6^{-4}	3.5^{-2}	4.3^{-4}	6.8^{-4}	1.2^{-3}	.246
CAL	.005/.02*	2.4^{-3}	1.01^{-1}	2.5^{+0}	6.0^{-1}	1.62^{-1}	5.4^{-1}	.0725
LW1	.02	1.6^{-4}	4.9^{-4}	2.4^{-4}	1.3^{-4}	3.2^{-3}	4.4^{-4}	.145

TABLE 3. Example 4 Accuracy and Relative Run Times from (7). t_{RK4} = 138 sec to x = 40 using h = .001 (estimated).

* h changed from .005 to .02 at x = 1.

Method	Step Size h	Relative Accuracy R_{1n}		R_{2n}		R_{3n}		Rel. Time t/t_{ARK}
		x = .40	7.40	.40	7.40	.40	7.40	
ARK	.001	0	0	0	0	0	0	1.0
AMPLCT V(1) = .05		x = .40	7.40	.40	7.40	.40	7.40	.00933
		6.68×10^{-4}	6.85×10^{-5}	3.50×10^{-3}	1.12×10^{-3}	4.58×10^{-2}	2.15×10^{-3}	
AMPLCT V(1) = .10		x = .645	7.34	.645	7.34	.645	7.34	.00516
		2.59^{-2}	6.42×10^{-4}	1.84^{-2}	2.68×10^{-3}	1.11^{-1}	6.68×10^{-3}	

TABLE 4. Example 4 Accuracy and Relative Run Times using ARK and AMPLCT. t_{ARK} = 382 sec to x = 40 using h = .001 (estimated).

Method	Relative Accuracy						Rel. Time
	R_{1n}		R_{2n}		R_{3n}		t/t_{ARK}
	x = .40	7.4	.40	7.4	.40	7.4	
GEAR EPS = 1^{-4} * h = 1 x 10^{-4} **	9.1^{-5}	4.5^{-4}	2.3^{-4}	4.95^{-4}	1.4^{-2}	2.94^{-4}	1.69^{-3}

TABLE 5. Example 4 Accuracy and Relative Run Time of GEAR's Method.
t_{ARK} = 382 sec to x = 40 using h = .001 (estimated)

* Error control, Euclidian norm of estimated relative errors for all components per step.
** initial step size, automatic step size control used.

In example 4 AMPLCT shows up quite well compared to the methods discussed by Lapidus and Seinfeld even considering that ARK took about 3 times longer than RK4 to integrate this example. It was found that the run time of AMPLCT depends almost entirely on the relative accuracy parameter V(1) and is independent of the initial step size. In example 4 the errors at 7. 40 and 10 were similar; the point of error evaluation at x = 7. 40 being a matter of convenience. The results of RK4 or ARK using h = . 001 were taken as exact for example 4. ARK and RK4 both become unstable using h = . 001 for $x > 16$ where $|\lambda_{max}| \simeq 2780$. The generally higher accuracy of TR-EX has been obtained at the expense of an increased run time.

Gear's method is faster than AMPLCT by about a factor of 5 for example 4 using EPS = . 0001. The order varied between 1^{st} and 2^{nd} up to x = . 70 at which point it selected 3^{rd} order for the remainder of the problem. Using EPS = . 001, Gear's method was unstable.

REFERENCES

1. M. P. Sherman, "Radiation Coupled Chemical Nonequilibrium Normal Shock Waves," J. Quant. Spect. Rad. Transf. 8, 1968, p. 569.

2. H. Lomax, H. E. Bailey, "A Critical Analysis of Various Numerical Integration Methods for Computing the Flow of a Gas in Chemical Nonequilibrium," NASA TN D-4109, 1967.

3. H. E. Bailey, "Numerical Integration of the Equations Governing the One Dimensional Flow of a Chemically Reactive Gas," Phys. Fluids 12, 1969, pp. 2292-2300.

4. H. Lomax, "On the Construction of Highly Stable Explicit, Numerical Methods for Integrating Coupled Ordinary Differential Equations with Parasitic Eigenvalues," NASA TN D 4547, April 1968.

5. H. Lomax, "Stable Implicit and Explicit Numerical Methods for Integrating Quasi-Linear Differential Equations with Parasitic-Stiff and Parasitic-Saddle Eigenvalues," NASA TN D 4703, July 1968.

6. H. Lomax, H. E. Bailey, F. B. Fuller, "On Some Numerical Difficulties in Integrating the Equations for One Dimensional Nonequilibrium Nozzle Flow," NASA TN D 5176, April 1969.

7. L. Lapidus, J. Seinfeld, Numerical Solution of Ordinary Differential Equations, Academic Press, 1971.

8. L. Fox, Numerical Solution of Ordinary and Partial Differential Equations, Pergammon Press, 1962, Ch. 4.

9. S. W. Bowen, C. Park, "Computer Study of Nonequilibrium Excitation in Recombining Nitrogen Plasma Nozzle Flows," AIAA Paper 70-44, New York, 1970. Also, AIAA J., 9, No. 3, March 1971, p. 493.

10. A. Ralston, A First Course in Numerical Analysis, McGraw Hill, 1965, p. 510.

11. B. Carnahan, H. A. Luther, J. O. Wilkes, Applied Numerical Methods, Wiley, 1969, pp. 236, 249.

12. M. E. Fowler, R. M. Warten, "A Numerical Integration Technique for Ordinary Differential Equations with Widely Separated Eigenvalues," IBM J. of Res. and Dev., 11, 1967, p. 537.

13. C. W. Gear, "The Automatic Integration of Ordinary Differential Equations," Comm. ACM, 14, 1971, pp. 176-179.

14. C. W. Gear, "Algorithm 407, DIFSUB for Solution of Ordinary Differential Equations," Comm. ACM, 14, 1971, pp. 185-190.