

TECHNICAL
LIBRARY

AD A080998

AD

TECHNICAL REPORT ARBRL-TR-02206

A NUMERICAL METHOD TO INTEGRATE STIFF
SYSTEMS OF ORDINARY DIFFERENTIAL
EQUATIONS

T. P. Coffee
J. M. Heimerl
M. D. Kregel

January 1980



US ARMY ARMAMENT RESEARCH AND DEVELOPMENT COMMAND
BALLISTIC RESEARCH LABORATORY
ABERDEEN PROVING GROUND, MARYLAND

Destroy this report when it is no longer needed.
Do not return it to the originator.

Secondary distribution of this report by originating
or sponsoring activity is prohibited.

Additional copies of this report may be obtained
from the National Technical Information Service,
U.S. Department of Commerce, Springfield, Virginia
22151.

The findings in this report are not to be construed as
an official Department of the Army position, unless
so designated by other authorized documents.

*The use of trade names or manufacturers' names in this report
does not constitute indorsement of any commercial product.*

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER TECHNICAL REPORT ARBRL-TR-02206	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) A NUMERICAL METHOD TO INTEGRATE STIFF SYSTEMS OF ORDINARY DIFFERENTIAL EQUATIONS		5. TYPE OF REPORT & PERIOD COVERED BRL Technical Report
		6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) T. P. Coffee J. M. Heimerl M. D. Kregel		8. CONTRACT OR GRANT NUMBER(s)
9. PERFORMING ORGANIZATION NAME AND ADDRESS USA Armament Research and Development Command USA Ballistic Research Laboratory ATTN: DRDAR-BLP Aberdeen Proving Ground, MD 21005		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS RDT&E Project #1L161102AH43
11. CONTROLLING OFFICE NAME AND ADDRESS USA Armament Research and Development Command USA Ballistic Research Laboratory ATTN: DRDAR-BL Aberdeen Proving Ground, MD 21005		12. REPORT DATE JANUARY 1980
		13. NUMBER OF PAGES 54
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office)		15. SECURITY CLASS. (of this report) Unclassified
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE
16. DISTRIBUTION STATEMENT (of this Report) Approved for public release; distribution unlimited.		
17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)		
18. SUPPLEMENTARY NOTES		
19. KEY WORDS (Continue on reverse side if necessary and identify by block number) Numerical Integration Stiff Ordinary Differential Equations Jacobian Matrix Predictor-Corrector Methods		
20. ABSTRACT (Continue on reverse side if necessary and identify by block number) srf A method is described for the efficient integration of stiff systems of ordinary differential equations. The method, based on a predictor-corrector formulation, uses the Jacobian in a non-standard fashion. The resulting program is compared with EPISODE, a standard stiff integrator, for a number of systems of ordinary differential equations. The results show that the procedure is usually competitive with EPISODE and is more efficient for some problems.		

TABLE OF CONTENTS

	Page
I. INTRODUCTION	5
II. THE NUMERICAL METHOD	10
III. COMPARISONS	16
IV. NUMERICAL RESULTS	17
REFERENCES	26
APPENDIX A	27
APPENDIX B	31
DISTRIBUTION LIST	51

I. INTRODUCTION

The K-integrator for stiff ordinary differential equations was developed in the late 1960's by one of us (MDK), and has been repeatedly improved since that time¹⁻³. It is intended to solve systems of the form

$$\dot{Y}(t) = G(Y(t), t), \quad t \in [a, b], \quad (1)$$

$$Y(a) = Y_0,$$

where $\dot{Y}(t) = dY(t)/dt$,

$$Y(t) = [Y^1(t), Y^2(t), \dots, Y^N(t)]^T,$$

$$\text{and} \quad G(u, t) = [G^1(u, t), G^2(u, t), \dots, G^N(u, t)]^T.$$

The concept of stiffness is difficult to define formally. However, it can be described in terms of the Jacobian matrix $J(u, t)$, where the element in row i and column j is

$$J^{ij}(u, t) = \partial G^i(u, t) / \partial y^j. \quad (2)$$

A stiff system will have one or more eigenvalues in the Jacobian whose real parts are negative and large in modulus. As a result the corresponding components in the solution will decay very rapidly in comparison to the other terms present.

¹M. D. Kregel and E. L. Lortie, "Description and Comparison of the K-Method for Performing Numerical Integration of Stiff Ordinary Differential Equations", BRL Report No. 1733, July 1974 (ADA #A003855).

²M. D. Kregel and J. M. Heimerl, "Comments on the Solutions of Coupled Stiff Differential Equations", BRL Memorandum Report No. 2769, July 1977; or Proceedings of the 1977 Army Numerical Analysis and Computers Conference, ARO Report 77-3, November 1977, pp. 553-563 (ADA #A043122).

³T. P. Coffee, J. M. Heimerl, and M. D. Kregel, "A Numerical Method for Large Stiff Systems of Ordinary Equations", Transactions of the 24th Conference of Army Mathematicians, ARO Report 79-1, January 1979, pp. 249-257.

It will be useful to rewrite the system (1) using the diagonal Jacobian elements. We define the diagonal matrix $R(Y(t),t)$ and the vector $F(Y(t),t)$ with components,

$$R^i(Y(t),t) = -J^{ii}(Y(t),t), \text{ and} \quad (3)$$

$$F^i(Y(t),t) = \dot{Y}^i(t) + R^i(Y(t),t)Y^i(t).$$

Then equation (1) becomes

$$\dot{Y}(t) = F(Y(t),t) - R(Y(t),t)Y(t). \quad (4)$$

The term R^i measures the sensitivity of $\dot{Y}^i$ with respect to changes in Y^i .

The system is solved at a set of discrete points in time. Thus, step sizes, h_i , and times, $t_n = a + \sum_{i=1}^n h_i$, are introduced. Approximations y_n to $Y(t_n)$ are then produced for $n=1,2, \dots$ by the integrator.

These approximations will be found using multistep formulas of the form
$$y_n = \sum_{i=1}^q \alpha_i y_{n-i} + h_n \sum_{i=0}^q \beta_i \dot{y}_{n-i}. \quad (5)$$

The truncation error will depend on the order of the formula. For stiff equations, there is also the problem of stability. That is, a small error in one step may propagate and grow in subsequent steps. Standard non-stiff methods will be restricted to very small step sizes to preserve stability. Dahlquist⁴ has shown that to maintain reasonable stability for stiff systems, the formulas must be implicit, that is, $\beta_0 \neq 0$.

Most stiff multi-step methods use a modified Newton iteration to solve (5). First an explicit predictor y_n^P is found. Then its time derivative and corresponding corrector are defined:

$$\dot{y}_n^P = F(y_n^P, t_n) - R(y_n^P, t_n)y_n^P \quad \text{and} \quad (6)$$

⁴G. G. Dahlquist, AMS Symp. Appl. Math. 15, (1963), 147.

$$y_n^C = \sum_{i=1}^q \alpha_i y_{n-i} + h_n \sum_{i=1}^q \beta_i \dot{y}_{n-i} + h_n \beta_o \dot{y}_n^P. \quad (7)$$

Finally the vector Δ_n is defined such that

$$y_n = y_n^P + \Delta_n. \quad (8)$$

An approximation d_n to Δ_n can be found from the difference $(y_n^P - y_n^C)$, by expanding y_n in a series about the y_n^P and truncating. This procedure leaves the following system of linear equations to be solved,

$$(y_n^P - y_n^C) = [h\beta_o J_n - I]d_n, \quad (9)$$

where $J_n = J(y_n^P, t_n)$ and I is the identity matrix. The accuracy of the approximation to y_n can be found by monitoring the size of the quantity $(y_n^{P2} - y_n^{C2})$, where

$$y_n^{P2} = y_n^P + d_n \quad \text{and} \quad (10)$$

$$y_n^{C2} = \sum_{i=1}^q \alpha_i y_{n-i} + h_n \sum_{i=1}^q \beta_i \dot{y}_{n-i} + h_n \beta_o \dot{y}_n^{P2}. \quad (11)$$

If necessary, the Newton iteration can be repeated.

Solving the system of equations (9) requires approximately $N^3/3$ multiplications and divisions, where N is the number of equations. Thus, in general the larger N , the greater the computation time. Reduction of this time has been an objective of several algorithms including the one presented herein.

One approach to this problem is the well-known algorithm DIFSUB, by C. W. Gear⁵. This is a variable-order method, based on formulas of the form

⁵C. W. Gear, "Numerical Initial Value Problems in Ordinary Differential Equations", Prentice-Hall, 1971.

$$y_n = \sum_{i=1}^q \alpha_i y_{n-i} + h \beta_0 \dot{y}_n, \quad 1 \leq q \leq 6. \quad (12)$$

The program automatically uses the higher order formulas if more accuracy is required. All the formulas are based on a fixed-step size h . To change the step size, the appropriate values for y_{n-i} are found by interpolation.

For a given step, the matrix based on the Jacobian is found and inverted. The program reevaluates this matrix only when it fails to obtain convergence in three Newton iterations. One therefore does not have to invert a matrix every step.

More recently, variable-step formulas have been developed. Any sequence of step sizes h_i can be used. The coefficients α_i and β_i in (5) are variable and depend on the previous step sizes. In practice, such codes have been more stable.

One such code is EPISODE, developed by Byrne and Hindmarsh⁶. It uses a variable-order, variable-step formula. Like DIFSUB, it uses an aged Jacobian. An LU decomposition is computed instead of a matrix inversion.

The K-integrator uses a fixed-order, variable-step formula. Except for starting, it always uses a third-order formula. The lack of higher order formulas will not be important, provided only moderate accuracy is required.

The major innovation in the K-integrator is the method of solving the set of equations (9). Both DIFSUB and EPISODE approximate $(h\beta_0 J_{n-1})$ by the values of the matrix at an earlier time step t_{n-1} . The K-integrator evaluates the matrix at each time step; but, if possible, it is one of reduced size. Finding a suitable approximation to any of the d_n^k permits the corresponding row and column to be eliminated from the matrix.

To illustrate, we write (9) in component form; i.e.,

$$y_n^{Pk} - y_n^{Ck} = h\beta_0 \sum_{\substack{j=1 \\ j \neq k}}^N J_n^{kj} d_n^j + [h\beta_0 J_n^{kk} - 1] d_n^k. \quad (13)$$

⁶G. D. Byrne and A. C. Hindmarsh, ACM Trans. Math. Software (1975), 71.

We attempt to approximate d_n^k by

$$d_n^k = (y_n^{Pk} - y_n^{Ck}) / (h\beta_o J_n^{kk} - 1). \quad (14)$$

There are several cases where (14) is a reasonable approximation. First, if h is very small, such as at the beginning of the integration, so that $|h\beta_o \sum_{j \neq k} J_n^{kj} d_n^j| \ll |d_n^k|$, then the off-diagonal terms may be neglected. Second, even if h is large, the equation for y_n^k may be weakly coupled to the other equations, i.e., $|J_n^{kk} d_n^k| \gg |\sum_{j \neq k} J_n^{kj} d_n^j|$. Finally, at any value of h , y_n^{Pk} may be a good approximation to y_n^{Ck} , and so $|y_n^{Pk} - y_n^{Ck}| \approx 0 \approx d_n^k$.

In general, the set of equations (13) is solved in two stages. First a set of indices, L , is determined for which (14) is valid, i.e.,

$$d_n^l = (y_n^{Pl} - y_n^{Cl}) / (h\beta_o J_n^{ll} - 1), \quad l \in L. \quad (15)$$

Then, using these values, the system (13) becomes

$$y_n^{Pk} - y_n^{Ck} - h\beta_o \sum_{j \in L} J_n^{kj} d_n^j = h\beta_o \sum_{\substack{j \notin L \\ j \neq k}} J_n^{kj} d_n^j + [h\beta_o J_n^{kk} - 1] d_n^k, \quad k \notin L. \quad (16)$$

The reduced system (16) is then solved using an LU decomposition and back substitution.* Where the reduced system involves a small number of the original set of equations, for example when h is small, there will be a significant reduction in computation time.

The difficulty in using this algorithm lies in determining which set of equations can be solved with sufficient accuracy using (14). The version reported here monitors the error terms $|y_n^{P2k} - y_n^{C2k}|$ and uses them to determine the set of indices L for the next time step. The details are given in the next section.

*Earlier versions of the K-method (Refs. 1 and 2) included a Gauss-Seidel iteration procedure before the matrix factorization. The version reported here omits this iteration procedure.

II. THE NUMERICAL METHOD

The K-integrator is based on a fixed third order formula of the form

$$y_n = \alpha_1 y_{n-1} + \alpha_2 y_{n-2} + \alpha_3 y_{n-3} + h (\beta_0 \dot{y}_n + \beta_1 \dot{y}_{n-1} + \beta_2 \dot{y}_{n-2}) \quad (17)$$

It is a variable step formula. If h , h' , and h'' are the last three step sizes, then

$$\beta_2 = 0.05$$

$$\beta_1 = 0.35$$

$$p = h'/h$$

$$q = (h' + h'')/h$$

$$\beta_0 = [(1+p)(1+q) - \beta_2 p(p-q) - \beta_1 pq]/(3+2p+2q+pq)$$

$$\alpha_3 = [1+p-\beta_2 p^2 - \beta_0(3+2p)]/[q^2(p-q)]$$

$$\alpha_2 = (\beta_2 + \beta_1 - \alpha_3 q + \beta_0 - 1)/p$$

$$\alpha_1 = 1 - \alpha_2 - \alpha_3.$$

For constant step size, the stability can be described in terms of parameters α and D . A formula is said to be $A(\alpha, D)$ stable, $\alpha \in (0, \frac{\pi}{2})$, if all numerical solutions to $\dot{Y} = -\lambda Y$ converge to zero as $n \rightarrow \infty$ with h fixed for all $|\arg(-\lambda h)| < \alpha$, $|\lambda| \neq 0$ and for all $\text{Re}(h\lambda) \leq D$. This combines the features of the $A(\alpha)$ - stability of Widlund⁸ and the stiff stability of Gear⁵. For the above formula $\alpha = 81^\circ$ and $D = 0.4$. The truncation error, neglecting higher order terms, is $-0.0966 h^4 Y^{(4)}(t)$. If the step size is not held constant, the truncation error will vary slightly.

⁷G. K. Gupta, Math. of Comp. 30 (1976), 417.

⁸O. B. Widlund, BIT 7 (1967), 65.

In practice the K-integrator uses modified forms of equations (9) and (10) that are somewhat more convenient for chemical kinetics problems, where it has mainly been applied. The modified form of equation (9) is

$$y_n^P - y_n^C = A_n g_n, \quad (18)$$

where $A_n^{ij} = [h\beta_o J_n^{ij} - \delta_{ij}]y_n^{Pi}$ and

$$g_n^i = d_n^i / y_n^{Pi}.$$

In this notation equation (10) becomes

$$y_n^{P2i} = y_n^{Pi} (1 + g_n^i), \quad i = 1, 2, \dots, N. \quad (19)$$

The rationale for (18) and (19) is the following. For a given network or set of chemical reactions at a known temperature, the rate at which each reaction proceeds equals a constant multiplied by the concentration(s) of the chemical species involved. The modified Jacobian, with components $J_n^{ij}y_n^{Pi}$, can be generated by adding and subtracting rates, which is computationally easy to do. (To recover the Jacobian matrix written in terms of the rates alone, we divide by the y_n^{Pi} .) In the modified notation, the diagonal approximation (14) becomes

$$g_n^k = (y_n^{Pk} - y_n^{Ck}) / A_n^{kk}. \quad (20)$$

In order to minimize the computation time, we find as many of the g_n^k as possible using (20). As discussed in the Introduction, it is expedient to make $|y_n^{Pk} - y_n^{Ck}|$ as small as possible. To this end we have developed an unorthodox predictor. While not essential to the code, it does increase its efficiency.

The usual predictor is obtained from an explicit multistep formula. But difficulties may arise when y_n^P is used in equation (8) to obtain y_n^C .

Since the system is stiff, the quantities $R^i(y_n^P, t_n)$ from equation (4) will often differ in absolute value by many orders of magnitude. In order to finish the integration in a reasonable time, h must become large enough so that some of the terms $h\beta_0 R^i$ become very large. This means small errors in y_n^P can become greatly magnified in y_n^C .

This effect can be reduced by considering the form of equation (17) for the final value y_n . Using (4) this can be written as

$$y_n = \alpha_1 y_{n-1} + \alpha_2 y_{n-2} + \alpha_3 y_{n-3} + h \beta_1 \dot{y}_{n-1} \quad (21)$$

$$+ h \beta_2 \dot{y}_{n-2} + h \beta_0 F(y_n, t_n) - h \beta_0 R(y_n, t_n) y_n.$$

We shall derive a formula for y_n^P that follows this general form.

First, we obtain predicted values for F and R , using the formulas

$$F_n^P = \gamma_1 F_{n-1} + \gamma_2 F_{n-2} + \gamma_3 F_{n-3} \quad (22)$$

$$R_n^P = \gamma_1 R_{n-1} + \gamma_2 R_{n-2} + \gamma_3 R_{n-3}.$$

The quantities γ_1 , γ_2 , and γ_3 are determined uniquely for any step sizes by imposing the condition that the formulas be second order. For the case of constant step size, the truncation errors are $h^3 F^{(3)}(y, t)$ and $h^3 R^{(3)}(y, t)$, neglecting higher order terms.

Substituting these predicted values into equation (21), we obtain

$$y_n^P = \alpha_1 y_{n-1} + \alpha_2 y_{n-2} + \alpha_3 y_{n-3} \quad (23)$$

$$+ h \beta_1 \dot{y}_{n-1} + h \beta_2 \dot{y}_{n-2} + h \beta_0 F_n^P - h \beta_0 R_n^P y_n^P.$$

This equation can be solved for y_n^P .

Neglecting higher order terms,

$$y_n^i - y_n^{Pi} = \frac{h^4 \beta_0 (F^{(3)i}(y, t) - R^{(3)i}(y, t) y_n^{Pi})}{1 + h \beta_0 R^i(y_n, t_n)} \quad (24)$$

As $h \beta_0 R^i$ becomes large, the accuracy of y_n^{Pi} tends to improve. Of course, errors still occur in F_n^{Pi} and R_n^{Pi} ; but, we no longer obtain large errors just because $h \beta_0 R^i$ is large. In fact, in the special case where F and R are constant, $y_n - y_n^P = 0$.

After the first few steps, a further heuristic modification is made in y_n^P . We define an error term for the predictor by the equation

$$E_n^{Pi} = (y_n^{P2i} - y_n^{Pi}) / y_n^{Pi} \quad (25)$$

where y_n^{P2} is the final accepted predicted value. Then we define a weighted average of these errors by

$$w_n^i = 0.5 E_n^{Pi} + 0.5 w_{n-1}^i \quad (26)$$

Finally in the next step we define a modified predictor whose components are given by

$$y_{n+1}^{PMi} = y_{n+1}^{Pi} (1 + w_n^i). \quad (27)$$

This modified predictor is used in equation (7) to obtain the corrector y_n^C .

The purpose of this modification is to detect any systematic errors in the predictor. We assume that the accuracy of y_n^P is similar to the accuracy of y_{n-1}^P . If the errors are not systematic, the fact that (26) is a weighted average will tend to minimize any errors introduced by the w_n^i .

The existence of systematic errors in an explicit predictor can be shown for the simple case of a single equation $\dot{Y} = -\lambda Y$, $Y(a) = Y_0$.

Let y_n^P be given by an explicit multistep formula $y_n^P = \sum_{i=1}^q \alpha_i y_{n-i} + h \sum_{i=1}^q \beta_i \dot{y}_{n-i}$, where h is constant. Assume the values at the previous steps are known exactly, that is, $y_{n-i} = Y(t_{n-i})$ and $\dot{y}_{n-i} = \dot{Y}(t_{n-i})$, $i=1, \dots, q$. Then $(Y(t_n) - y_n^P)/y_n^P$ is a constant, independent of the values for both n and a . This can be seen simply by substitution into the above expression.

In general, we cannot prove that systematic errors occur. However, in practice this modification does lead to a noticeable improvement in the accuracy of the predictor.

The key step in the algorithm, the determination of the set of indices, L , for which the diagonal approximation (14) is valid, depends on a user supplied error tolerance, ϵ . (Error control is implemented on a per step basis.)

Since h at the start is made quite small, it is reasonable to solve for all the g_n^i by (14). Thus, in this version of the code, all indices $1, 2, \dots, N$ are put into L at the beginning of the integration. (So that the entire Jacobian does not have to be evaluated, the diagonal elements, $J_n^{kk} y_n^k$, are computed in a separate subroutine.) Next we define a convergence error, E_c^i , for each component. If an absolute error criterion is used

$$E_c^i = |y_n^{C2i} - y_n^{P2i}| ; \quad (28a)$$

if a relative error criterion is used

$$E_c^i = 2 |(y_n^{C2i} - y_n^{P2i}) / (y_n^{C2i} + y_n^{P2i})| , \quad (28b)$$

where y_n^{P2} and y_n^{C2} are found from (17) and (11), respectively. In addition an overall root mean square (rms) convergence error, E_c , is defined by

$$E_c = \left(\sum_{i=1}^N (E_c^i)^2 / N \right)^{1/2}. \quad (29)$$

If $E_c < \epsilon$ we accept y_n^{C2} as our final value for y_n . Otherwise the Newton method is repeated. Also, if $E_c^i < \epsilon/5$, the i th index remains in L . Otherwise the corresponding equation is solved using the analogue of (16) for subsequent steps. The value $\epsilon/5$ is heuristic; but, it expresses the basic idea that very good accuracy is required to continue solving an equation by the diagonal approximation.

Normally, as h increases, the matrix A_n becomes less diagonally dominant. Thus, as the integration proceeds, fewer of the equations are solved using (20). However, it is possible for an equation to become diagonally dominant during the integration. There is no easy way to detect this event when it is solved as part of system (16). But a crude check is made by monitoring E_c . The condition $E_c < \epsilon/1000$ shows extreme accuracy, so for the next step all the indices are again placed into L . This allows a new determination of the set of equations to be solved by the diagonal approximation.

The last important part of the integrator is the algorithm for controlling step size. This is based on an estimate of the truncation error and on the convergence error E_c , defined by (29).

For simplicity, we use the truncation error form for constant step size, that is,

$$E_T^i = 0.0966 h^4 y^{(4)i}(t).$$

The fourth derivative of Y^i is approximated by interpolating a fourth degree polynomial through the values $y_n, y_{n-1}, y_{n-2}, y_n$, and y_{n-1} . The fourth derivative is then given by $4!$ times the leading coefficient of this polynomial. An rms value for the fourth derivative is given by

$$Y_{rms}^{(4)} = \left(\sum_{i=1}^N (y^{(4)i})^2 / N \right)^{1/2}.$$

We then define h_T by the relation

$$0.0966 h_T^4 Y_{rms}^{(4)} = \epsilon. \quad (30)$$

The quantity h_T is an upper limit for the next step size.

The step size also depends on the convergence error E_c . We define

$$h_c = h_n [1.0 + 0.1 \log_{10} (\epsilon/E_c)], \quad (31)$$

and take the minimum of the values h_T and h_c as the next step size h_{n+1} .

Formula (31) provides for a slow, steady increase in step size. This approach helps avoid oscillations in the step size. The value of h can increase rapidly only if the predictor and the corrector are in very close agreement.

In summary, the major innovation of the K-integrator is the method of solving the system of linear equations associated with the Jacobian. By using a diagonal approximation, the effort required for this operation can be substantially reduced. To increase the efficiency of this procedure, more effort than usual is invested in obtaining an accurate predicted value. A nonorthodox rational predictor is used, with a further heuristic modification based on the results obtained in the previous steps. The step-size changing algorithm is based on both truncation and convergence errors. It attempts to increase the step size by a small amount each step rather than by a large amount every several steps.

III. COMPARISONS

The K-integrator has been compared with EPISODE for selected problems. EPISODE can be run with several variants; we use the backward differentiation formulas (BDF), suitable for stiff problems, with a user supplied Jacobian.

Some general observations can be made concerning the two integrators. First, neither one will be efficient if a problem has eigenvalues near the imaginary axis. This area is outside the region of stability for the K-corrector (see page 10) and the BDF formulas. In a problem with eigenvalues $-10 \pm 100i$ ⁹ neither method performed adequately. Such examples are not considered further.

Also, since both programs are variable - step size, they should be able to handle discontinuities, by reducing the step size sufficiently. This was one of the reasons for developing EPISODE. This aspect is tested by the last three problems.

⁹W. H. Enright, T. C. Hull, and B. Lindberg, *BIT* 15 (1975), 10.

There are also important differences. The K-method was developed primarily for equations involving chemical reactions. These depend on reaction rates, most of which are known only approximately. Also, measurements of concentrations of species cannot be done to a very high accuracy. So we normally are satisfied with one or two place accuracy, since any further accuracy in solving the equations will not be physically meaningful.

Since the K-integrator is a fixed order method, we expect EPISODE to be more efficient at stricter error tolerances, where truncation error becomes more important. The treatment of the Jacobian is also important. At a stricter error tolerance, more steps will be taken, and the Jacobian will not change as much per step. So EPISODE will in general update the Jacobian less frequently and will become more efficient.

For a given error tolerance, the relative efficiency of the two methods is quite problem dependent. EPISODE is more efficient if the Jacobian changes slowly. Otherwise, the form of the Jacobian is not important. The K-integrator will do better if the Jacobian is diagonally dominant. Since it updates the relevant part of the matrix each step, it is not bothered by a rapidly changing Jacobian. In particular, the K-integrator will tend to be more efficient at the start of an integration, when h is small and at least some of the Y values change rapidly. Since many stiff problems are integrated until a steady state is attained, EPISODE will be more efficient at the end of the integration, when the Y values change slowly.

If the strategies of both the K - and Gear integrators could be successfully combined, the result might lead to a more efficient, general, stiff O.D.E. algorithm.

IV. NUMERICAL RESULTS

Both the K-integrator and EPISODE have been run for a set of twelve problems, given in Appendix A. The results of the comparisons are in Tables I through IV. All the runs were made on a CDC 7600 in single precision.

We are interested in the efficiency and accuracy of the methods. Efficiency can be measured by the run time. The times given in the tables are the actual times used by the core integrator. The tables also give the number of derivative calls, the number of matrix factorizations, the average dimension of the matrices involved, and the total number of integration steps.

For determining the error, the "correct" answer Y is found using an error criterion of 10^{-8} or 10^{-10} , depending on the problem. The error

Table I. Error Tolerance is 10^{-2}

		RUN TIME	FINAL ERROR	MAX ERROR	DER CALLS	MATRIX FAC	AVERAGE DIM	NO. OF STEPS
A	E	.039	.10	.18	75	32	10.0	57
	K	.017	.00	.03	88	0	0.0	43
B	E	.022	.19	1.21	83	20	6.0	54
	K	.022	.00	.98	130	39	2.0	56
C	E	.041	.07	12.19	220	28	4.0	117
	K	.048	.09	17.93	394	152	1.0	178
D	E	.003	.46	.46	18	8	3.0	11
	K	.008	.00	.00	86	10	1.0	42
E	E	.002	.01	.01	7	6	4.0	6
	K	*	*	*	*	*	*	*
F	E	.088	.16	.16	251	71	7.0	161
	K	.049	.58	.58	263	62	2.3	130
G	E	.086	.00	.21	128	54	9.0	84
	K	.079	.00	.01	203	62	5.6	99
H	E	14.604	.12	5.41	4323	1318	26.0	1938
	K	2.062	.00	3.13	684	254	16.9	318
I1	E	14.870	.52	3.20	883	144	64.0	522
	K	10.662	.14	3.18	832	278	30.7	412

*See text.

TABLE II. ERROR TOLERANCE IS 10^{-4}

		RUN TIME	FINAL ERROR	MAX ERROR	DER CALLS	MATRIX FAC	AVERAGE DIM	NO. OF STEPS
A	E	.082	.41	.41	179	39	10.0	119
	K	.038	.07	.40	196	0	0.0	97
B	E	.049	.35	1.90	184	24	6.0	115
	K	.039	.08	2.35	236	85	1.6	107
C	E	.091	.56	13.05	452	70	4.0	248
	K	.123	.17	65.10	1012	413	1.0	451
D	E	.008	.18	.64	48	13	3.0	28
	K	.011	.05	.05	108	19	1.6	53
E	E	.002	.67	.67	8	6	4.0	6
	K	.016	.00	.00	116	34	2.7	57
F	E	.134	.85	.85	369	94	7.0	225
	K	.094	3.34	3.34	499	145	1.4	247
G	E	.152	.00	.27	267	65	9.0	172
	K	.119	.00	.14	288	97	5.8	142
H	E	75.586	.00	8.76	15096	7995	26.0	8934
	K	3.913	.05	10.00	1510	504	14.8	726
I1	E	17.015	2.98	11.22	1337	137	64.0	857
	K	21.911	.80	13.72	2022	638	27.8	1004

TABLE III. ERROR TOLERANCE IS 10^{-6}

		RUN TIME	FINAL ERROR	MAX ERROR	DER CALLS	MATRIX FAC	AVERAGE DIM	NO. OF STEPS
A	E	.156	.23	.74	373	42	10.0	231
	K	.106	2.12	2.12	550	0	0.0	274
B	E	.111	.07	20.15	416	35	6.0	268
	K	.095	1.20	9.44	606	152	1.6	287
C	E	.168	.15	5.49	798	87	4.0	494
	K	.337	.68	166.79	2810	1041	1.0	1318
D	E	.014	.59	1.93	88	15	3.0	52
	K	.016	.47	.47	166	39	1.9	80
E	E	.005	1.82	1.82	23	9	4.0	14
	K	.019	.19	.23	116	34	2.7	69
F	E	.224	1.06	1.06	613	120	7.0	410
	K	.225	12.42	12.42	1314	183	1.5	654
G	E	.260	.04	1.11	502	79	9.0	310
	K	.288	.00	.48	728	224	5.9	359
H	E	46.046	.00	19.49	10795	4461	26.0	6533
	K	7.333	.51	29.23	3861	952	12.8	1886
I1	E	25.651	1.19	7.00	2306	177	64.0	1688
	K	394.610	.71	44.62	15719	6578	43.5	7813

TABLE IV. ERROR TOLERANCE IS 10^{-2}

		RUN TIME	FINAL ERROR	MAX ERROR	DER CALLS	MATRIX FAC	AVERAGE DIM	NO. OF STEPS
I2	E	19.650	.23	3.49	889	208	64.0	525
	K	20.159	.01	.75	976	367	37.7	457
I3	E	61.553	.43	.43	2214	713	64.0	1376
	K	28.872	.02	.23	2322	907	27.1	901
I4	E	38.228	.16	8.68	1313	455	64.0	808
	K	5.695	.01	2.87	779	250	22.3	361

may be absolute $(\sum_{i=1}^N (Y^i - y^i)^2 / N)^{1/2}$ or relative $(\sum_{i=1}^N ((Y^i - y^i) / Y^i) / N)^{1/2}$.

Error is measured at four times, equally spaced on a logarithmic scale between the initial step size h_0 and the final time t_f .

Both methods actually control error on a per step basis. The error that we have computed is the global error, which is of more interest to the user. The relation between the user supplied local error tolerance ϵ and the global error is highly problem dependent. However, a reliable code should keep global errors below a bound proportional to the local error tolerance, for a given problem. The tables give the error at the end time and the maximum error at the four output times. For convenience, the errors are given in units of the error tolerance.

Appendix B gives a listing of the code for the K-integrator for a sample problem. The output is given at different times and for different error criterions, as described above.

The first five problems are from an article by Enright, Hull, and Lindberg⁹. These are part of a larger set of problems, proposed as a test set for comparing stiff integrators. All the systems are small, but they show that both methods can handle a variety of types of problems. All the problems are run with an absolute error criterion.

The K-method is very efficient for problems A and B, because it can take advantage of the diagonal dominance of the Jacobians. In fact, in solving problem A no matrix factorizations are performed.

For problem C, the K-integrator becomes less efficient and less accurate at $\epsilon = 10^{-6}$. The difficulty is due to the fact that one component becomes very large, on the order of 10^4 . Since an absolute error criterion is being used, extreme accuracy is required. For this type of problem, the lack of higher order formulas in the K-method can cause difficulties.

For problems D and E, EPISODE is more efficient. The difference here is due to EPISODE's step changing strategy. Both problems are relatively easy, so EPISODE increases the step size rapidly. The K-method has a more conservative algorithm for changing step size, and takes many more steps. However, both programs are extremely fast here.

The K-integrator does fail on problem E for $\epsilon = 10^{-2}$. Here the problem is caused by the fact that the largest component is of the order 10^{-3} . So a step can be accepted by the integrator even with no significant digits in the final y values. As a result, the program uses the diagonal approximation too long. The resulting inaccuracies lead to instabilities that do not damp out.

In general, both methods are efficient and reasonably accurate on these problems. However, the K-integrator does experience some difficulties, due to very large or very small components combined with an absolute error criterion, whereas EPISODE does not.

The remaining systems are chemical reaction problems, and here a relative error criterion is used. To prevent control of the step size by species whose concentrations have become negligible, an artificial formation term of 10^{-30} is added to $\dot{y}$ every time the derivative subroutine is called. This is also an easy way of preventing underflow.

Problem F is a demonstration reaction set proposed by Edelson¹⁰. It is a simplified version of an atmospheric chemistry problem. The K-method is faster than EPISODE for $\epsilon = 10^{-2}$ and $\epsilon = 10^{-4}$. The problem is sufficiently diagonally dominant that the K-integrator can just use the diagonal approximation for a good part of the integration, and works with fairly small matrixes even near the end of the integration time. However, at $\epsilon = 10^{-6}$ its run time is equal to EPISODE, and it is less accurate. Again, the K-method will sometimes experience difficulties at $\epsilon = 10^{-6}$, due to the lack of higher order formulas.

Problem G and H are simulations of the chemistry in a gun barrel, i.e., under conditions of high temperature and high pressure. The problem is quite stiff.

Problem G involves nine species. Both programs are roughly comparable.

Problem H involves twenty-six species, some of whose concentrations are relatively small. EPISODE experiences major difficulties, due to its step size changing strategy. EPISODE attempts to make large changes in the step size, on the order of 33%. This leads to instability in the minor species. The program cannot meet its error criterion and must reduce the step size. This prevents the step size from increasing normally, and the integration takes a very large number of steps. The K-integrator, with changes of 5% to 10%, can increase its step size consistently.

Problem I1 is an atmospheric model of charge flow under the influence of a large electron flux. The electron density starts at a high level and decays to zero. The systems consist of 64 species. The reactions involved are given by Heimerl and Niles¹¹.

At $\epsilon = 10^{-2}$, the K-integrator is somewhat faster. However, the integrators proceeded in very different ways. For example, to reach the third output time, 3.16 seconds, takes EPISODE 12.4 seconds and the K-integrator only 4.5 seconds.

¹⁰D. Edelson, *J. Chem. Ed.* 52 (1975), 642.

¹¹J. M. Heimerl and F. E. Niles, "BENCHMARK -76: Model Computations for Disturbed Atmospheric Conditions. 1. Input Parameters", BRL Report No. 2022, October 1977 (ADA #A050355).

The K-method is extremely efficient at the start of the integration, where it works with very small matrixes. But by the end of the integration, the K-integrator is working with fifty-four by fifty-four matrixes. Meanwhile, EPISODE updates the Jacobian about every 3.6 steps, and is reasonably efficient throughout the integration. So the efficiency of the programs is highly problem dependent.

At $\epsilon = 10^{-6}$, the K-integrator is very slow. Because of truncation error, it cannot increase the step size adequately.

The remaining three problems involve discontinuities in the driving function, that is, the electron density, for the above problem. The discontinuities occur at the powers of 10, starting at 10^{-6} seconds. For comparison purposes, the equations are integrated out to a time between discontinuities, since the values are somewhat ambiguous at the discontinuities. Results are given only for $\epsilon = 10^{-2}$.

I2 is a sawtooth driving function. The K-integrator is slightly less efficient here.

I3 is a square wave driving function. The comparison is made at 5×10^2 because EPISODE cannot integrate past the discontinuity at 10^3 . Even using the smallest increment possible on a CDC 7600 in single precision, the program cannot meet its error criterion.

The program enters an infinite loop. Using a step size of 4×10^{-12} , it attempts to cross the discontinuity. Failing its convergence tests it reduces the step size to 2×10^{-12} . But $10^3 + 2 \times 10^{-12} = 10^3$, and no progress is made. It increases the step size to 3×10^{-12} , which is still too small an increment to register. Finally, it increases to the value 4×10^{-12} again. This process repeats indefinitely.

The reason for this is the way EPISODE determines the step size. EPISODE compares the original predictor with the final accepted y value. At a discontinuity, the original explicit predictor behaves very badly, since it has no information concerning the changed conditions. The step size must be reduced severely to obtain agreement.

The K-integrator, however, compares the modified predictor y^{p2} with the corresponding corrector y^{c2} . The existence of the discontinuity is fed in through the Jacobian, and the step size does not have to be reduced so far.

The K-method is also much faster than EPISODE. At the discontinuities, where h is reduced, it can use the simpler diagonal approximation to solve the equations.

Problem I4 is a combination of I2 and I3. The electron density forms a series of ramps, alternately increasing and decreasing, but starting at the same value at each decade. So there is a discontinuous derivative as well as function.

Here EPISODE cannot get past the discontinuity at 1.0 second; so the comparison was made at 0.5 second. Not only is the K-method able to integrate the problem more rapidly than EPISODE, but it also is able to integrate over the entire range of interest, i.e., 10^4 seconds.

In conclusion the K-integrator does appear to be comparable to the variable-order EPISODE program, at least for the looser error tolerance. It can be more efficient, depending on the problem, and permits integration over some severe functional discontinuities.

REFERENCES

1. M. D. Kregel and E. L. Lortie, "Description and Comparison of the K-Method for Performing Numerical Integration of Stiff Ordinary Differential Equations", BRL Report No. 1733, July 1974 (ADA #A003855).
2. M. D. Kregel and J. M. Heimerl, "Comments on the Solutions of Coupled Stiff Differential Equations", BRL Memorandum Report No. 2769, July 1977; or Proceedings of the 1977 Army Numerical Analysis and Computers Conference, ARO Report 77-3, November 1977, pp. 553-563 (ADA #A043122).
3. T. P. Coffee, J. M. Heimerl, and M. D. Kregel, "A Numerical Method for Large Stiff Systems of Ordinary Equations", Transactions of the 24th Conference of Army Mathematicians, ARO Report 79-1, January 1977, pp. 249-257.
4. G. G. Dahlquist, AMS Symp. Appl. Math. 15, (1963), 147.
5. C. W. Gear, "Numerical Initial Value Problems in Ordinary Differential Equations", Prentice-Hall, 1971.
6. G. D. Byrne and A. C. Hindmarsh, ACM Trans. Math. Software (1975), 71.
7. G. K. Gupta, Math. of Comp. 30 (1976), 417.
8. O. B. Widlund, BIT 7 (1967), 65.
9. W. H. Enright, T. C. Hull, and B. Lindberg, BIT 15 (1975), 10.
10. D. Edelson, J. Chem. Ed. 52 (1975), 642.
11. J. M. Heimerl and F. E. Niles, "BENCHMARK -76: Model Computations for Disturbed Atmospheric Conditions. 1. Input Parameters", BRL Report No. 2022, October 1977 (ADA # A050355).

APPENDIX A. SPECIFICATION OF PROBLEMS

Listed below are the systems of differential equations used in the tests. In all cases the initial time $t_0 = 0$. The final time t_f and the initial step size h_0 are given. The last four systems are not specified in detail because of their complexity.

A. Linear with real eigenvalues

$$\dot{Y}^i = i^5 Y^i \quad Y^i(0) = 1 \quad i = 1, 2, \dots, 10$$

$$t_f = 1 \quad h_0 = 10^{-5}$$

B. Linear with non-real eigenvalues.

$$\dot{Y}^1 = -10Y^1 + 25Y^2 \quad Y^1(0) = 1$$

$$\dot{Y}^2 = -25Y^1 - 10Y^2 \quad Y^2(0) = 1$$

$$\dot{Y}^3 = -4Y^3 \quad Y^3(0) = 1$$

$$\dot{Y}^4 = -Y^4 \quad Y^4(0) = 1$$

$$\dot{Y}^5 = -0.5Y^5 \quad Y^5(0) = 1$$

$$\dot{Y}^6 = -0.1Y^6 \quad Y^6(0) = 1$$

$$t_f = 20 \quad h_0 = 10^{-2}$$

C. Non-linear coupling.

$$\dot{Y}^1 = -Y^1 + 2 \quad Y^1(0) = 1$$

$$\dot{Y}^2 = -10Y^2 + 20(Y^1)^2 \quad Y^2(0) = 1$$

$$\dot{Y}^3 = -40Y^3 + 80[(Y^1)^2 + (Y^2)^2] \quad Y^3(0) = 1$$

$$\dot{Y}^4 = -100Y^4 + 200[(Y^1)^2 + (Y^2)^2 + (Y^3)^2] \quad Y^4(0) = 1$$

$$t_f = 20 \quad h_0 = 10^{-2}$$

D. Non-linear with real eigenvalues.

$$\dot{Y}^1 = -Y^1 + 10^8 Y^3 (1 - Y^1) \quad Y^1(0) = 1$$

$$\dot{Y}^2 = -10 Y^2 + 3 \times 10^7 Y^3 (1 - Y^2) \quad Y^2(0) = 0$$

$$\dot{Y}^3 = -\dot{Y}^1 - \dot{Y}^2 \quad Y^3(0) = 0$$

$$t_f = 1 \quad h_0 = 3.3 \times 10^{-8}$$

E. Non-linear with non-real eigenvalues.

$$\dot{Y}^1 = -7.89 \times 10^{-10} Y^1 - 1.1 \times 10^7 Y^1 Y^3 \quad Y^1(0) = 1.76 \times 10^{-3}$$

$$\dot{Y}^2 = 7.89 \times 10^{-10} Y^1 - 1.13 \times 10^9 Y^2 Y^3 \quad Y^2(0) = 6$$

$$\dot{Y}^3 = 7.89 \times 10^{-10} Y^1 - 1.1 \times 10^7 Y^1 Y^3 \quad Y^3(0) = 0$$

$$+ 1.13 \times 10^3 Y^4 - 1.13 \times 10^9 Y^2 Y^3$$

$$\dot{Y}^4 = 1.1 \times 10^7 Y^1 Y^3 - 1.13 \times 10^3 Y^4 \quad Y^4(0) = 0$$

$$t_f = 1000 \quad h_0 = 5 \times 10^{-5}$$

F. Atmospheric reaction set. A set of 7 chemical species and 7 reactions.

$$t_f = 1000 \quad h_0 = 10^{-16}$$

G. Hot Gas. A set of 9 chemical species and 57 reactions under high temperature and pressure.

$$t_f = 10^{-5} \quad h_0 = 10^{-15}$$

H. Expanded Hot Gas. A set of 26 chemical species and 227 reactions under high temperature and pressure.

$$t_f = 10^{-5} \quad h_0 = 10^{-15}$$

I. BENCHMARK - 76. A set of 64 chemical species and 498 reactions simulating chemistry in the upper atmosphere. The reactions are driven by the electron density.

1. Relaxation of driving force.

$$t_f = 10^4 \quad h_0 = 10^{-10}$$

2. Saw tooth driving force (discontinuous derivative).

$$t_f = 5 \times 10^3 \quad h_0 = 10^{-10}$$

3. Square Wave driving force (discontinuous function).

$$t_f = 5 \times 10^2 \quad h_0 = 10^{-10}$$

4. Ramp driving force (discontinuous function and derivative).

$$t_f = 0.5 \quad h_0 = 10^{-10}$$

APPENDIX B. PROGRAM LISTING

A listing of the computer code follows. The code is set-up to solve problem C, page 27.

```

1      PROGRAM KINT(OUTPUT,TAPE6=OUTPUT)
2      DIMENSION YC(30)
3      THE MAIN PROGRAM AND THE SUBROUTINE DEK ARE SUPPLIED
4      BY THE USER.
5      AS AN EXAMPLE
6      SOLVE SYSTEM C5 ENRLIGHT, HULL, AND LINDBERG.
7      DY1/DT=-Y1+2
8      DY2/DT=-10*Y2+20*Y1**2
9      DY3/DT=-40*Y3+80*(Y1**2+Y2**2)
10     DY4/DT=-100*Y4+200*(Y1**2+Y2**2+Y3**2) Y4(0)=1.0
11     Y1(0)=1.0 Y2(0)=1.0 Y3(0)=1.0
12     DIMENSION Y(30)
13     COMMON/TAB1/TIN,HIN,TEND,T,EC
14     COMMON/TAB3/NDIM,1DIM
15     COMMON/TAB4/NITG,NDER,NJAC,NMI,NDMI
16     SET INPUT DATA.
17     NDIM=DIMENSION OF VECTORS AND MATRICES.
18     NDI=NUMBER OF UNKNOWN.
19     TIN=START TIME
20     HIN=INITIAL STEP SIZE
21     TEND=END TIME
22     T=CURRENT TIME
23     EC=ERROR CRITERION
24     DO 1000 KOUT=1,4
25     HIN=1.0E-2
26     TEND=20.
27     RED=FLOAT(KTOUT)*ALOG10(TEND)/4.0+
28     * (4.0-FLOAT(KTOUT))*ALOG10(HIN)/4.0
29     TEND=10.**RED
30     DO 1000 KEC=4,10,2
31     KR=12-KEC
32     1DIM=30
33     NDIM=4
34     TIN=0.0
35     EC=.1**KR
36     WRITE(6,1010)EC
37     FORMAT(///2X,10HERR CRIT =,1PE12.4)
38     CHECK=TEND-TEND*1.0E-10
39     1010 CHECK=TEND-TEND*1.0E-10
40     IERROR=1 MEANS ABSOLUTE ERROR CRITERION.
41     IERROR=2 MEANS RELATIVE ERROR CRITERION.
42     IERROR=1
43     SET THE INITIAL VALUES.
44     Y(1)=1.0
45     Y(2)=1.0
46     Y(3)=1.0
47     Y(4)=1.0
48     SET THE COUNTERS TO 0
49     NITG=NO. OF INTEGRATION STEPS.
50     NDER=NO. OF DERIVATIVE EVALUATIONS
51     NJAC=NO. OF JACOBIAN EVALUATIONS
52     NMI=NO. OF MATRIX INVERSIONS
53     NDMI=SUM OF THE DIMENSIONS OF THE MATRICES INVERTED
54     NITG=0
55     NDER=0
56     NJAC=0
57     NMI=0
58     NDMI=0

```

03/19/78 14.19.29

FTN 4.6+452

PROGRAM KINT 76/76 ORT=1 ROUNO=-*/ TRACE

```

60      FT=SECONO(CP)
        00 4 M=1.100000
        C  CALL THE INTEGRATION SUBROUTINE.
          CALL MKR(Y,SING,IERROR)
          IF(I.GT.CHECK)GOTO 30
          IF(SING.GT.0.5)GOTO 30
        4 CONTINUE
65      C  PRINT DESIRED OUTPUT DATA.
          30 GT=SECONO(CP)
          HT=GT-FT
          WRITE(6,500)HT
          500 FORMAT(1H0,10H RUN TIME =,1PE12.4)
          IF(NMI.EQ.0)GOTO 35
          AO=FLOAT(NMI)/FLOAT(NMI)
          35 WRITE(6,510)NITG,NOER,NJAC
          510 FORMAT(1H0,11H MK CALLS =,16.6X,18H DERIVATIVE CALLS =,
            *16.6X,16H JACOBEAN CALLS =,16)
75      WRITE(6,520)NMI,AO
          520 FORMAT(1H0,26H NO. OF MATRIX INVERSIONS =,16.6X,
            *19H AVERAGE DIMENSION =,1PE12.4)
          WRITE(6,505)(Y(K),K=1,NOIM)
          505 FORMAT(1H0,1R5E20.12)
          WRITE(6,550)T
          550 FORMAT(1H0,12H FINAL TIME =,1PE12.4)
          IF(KEC.GT.4)GO TO 800
          00 810 K=1,NDIM
          810 YC(K)=Y(K)
          800 ERR=0.0
          DO 850 K=1,NOIM
            IF(IERROR.EG.1)OENOM=1.0
            IF(IERROR.EG.2)OENOM=YC(K)
            IF(OENOM.EG.0.0)GO TO 1000
            850 ERR=ERR+(YC(K)-Y(K))/OENOM)**2
            ERR=SQRT(ERR/FLOAT(NOIM))
            WRITE(6,840)ERR
            840 FORMAT(1/5X,7H ERROR =,1PE12.4)
          1000 CONTINUE
          99 STOR
          ENO
95

```



```

115      AUX3(J)=AU*AUX11(J)+BU*AUX4(J)+A2*AUX10(J)+CU*AUX12(J)+B1*AUX2(J)
      41 AUX5(J)=AUX11(J)+H*AUX12(J)
39      T=T+H
      CALL DER(AUX5,DERY)
      C GENERATION OF THE CORRECTOR VALUE, IN Y.
      DO 34 J=1,NUM
34      Y(J)=AUX3(J)+OU*DERY(J)
      C * * * * *
      C * * * * *
      C NEWTON-RAPHSON ITERATION.
      C IF KEY(J)=0, THE CORRESPONDING Y VALUE IS ACCEPTED
      C IN THE INITIAL SCREEN.
      DO 100 I11=1,10
      CO1=1.0/OU
      CALL JACOB
      ITEST=0
      K=1
      DO 13 J=1,NDIM
      AU=AUX5(J)
      AE(K)=AE(K)-AU/OU
      BOLD=AU-Y(J)
      IF(KEY(J).EQ.1) ITEST=1TEST+1
      Z(J)=BOLD*CO1
      G(J)=Z(J)/AE(K)
      13 K=K+NPI
      IF(ITEST.EQ.0.OR.NITG.EQ.0)GOTO 33
      CALL JAC
      C THE FOLLOWING BLOCK UPDATES THE Z ARRAY FOR THOSE Y VALUES WHICH HAVE
      C CONVERGED.
      DO 116 J=1,NDIM
      IF(KEY(J).EQ.1)GOTO 116
      J1=(J-1)*IOIM
      SURJ=G(J)
      DO 15 I1=1,NDIM
      J1=J1+1
      Z(I1)=Z(I1)-AE(J1)*SURJ
      15 CONTINUE
      116 CONTINUE
      C ANY Y VALUES THAT HAVE NOT CONVERGED ARE FOUND BY EXPLICITLY
      C SOLVING THE SYSTEM OF EQUATIONS. THE FINAL PREDICTED AND
      C CORRECTED VALUES ARE GENERATED.
      4 IF(ITEST.GT.0) CALL FIX (Z,G,SING)
      NMI=NMI+1
      NDMI=NDMI+1TEST
      IF (SING.GT. HALF)RETURN
      33 DO 12 J=1,NDIM
      12 AUX5(J)=AUX5(J)*(1.0+G(J))
      CALL DER(AUX5,DERY)
      C * * * * *
      C * * * * *
      C FIND A LIMIT ERR ON THE TRUNCATION ERROR.
      C FIND THE CONVERGENCE ERROR DELT.
      37 IF(NITG.GE.C)CALL CONT(H,MP,0.00,Z1,G1,Z2)
      ERR=1.0E-34
      DELT=0.0
      DO 442 J=1,NUMIM
      BOLD=AUX3(J)+DU*DERY(J)

```



```

230      HPP=HP
      HP=H
      H=H*(1.0+0.12*ZLOG10(EC/(12.5*SOELT)))
      IF (H.LT.HP) H=Z2MIN(H,0.85*HP)
      H=Z2MIN(H,H LIM)
235      C IF WE ARE NEAR THE REQUESTED OUTPUT TIME,
      C CHOOSE THE STEP SIZES SO THE OUTPUT TIME IS REACHED EXACTLY.
      IF (1+5.0*H.LT.TENO) GO TO 65
      IF (H.LT.HP) KENO=0
      H=HP
      IF (KENO.EQ.1) GO TO 65
      KENO=1
      H=(TENO-T)/5.0
240      65 CONTINUE
      CALL JACOB
      RHP=(H/HP)**3
      SFAC=0.5*Z2MIN(1.0,RHP)
      TFAC=SFAC
      K=1
      DO 40 J=1,NDIM
        AU=AUX5(J)
        AUX6(J)=AUX7(J)
        AUX7(J)=AUX8(J)
        F2(J)=F1(J)
        F1(J)=F0(J)
        F0(J)=OERY(J)-AE(K)
        F2(J)=R1(J)
        R1(J)=R0(J)
        R0(J)=-AE(K)/AU
        AUX2(J)=AUX12(J)
        AUX12(J)=OERY(J)
        AUX10(J)=AUX4(J)
        AUX4(J)=AUX11(J)
        AUX11(J)=Y(J)
        AUX12(J)=Y(J)
        IF (LOGIC3) ALF(J)=(AU/W(J)-1.0)*SFAC+ALF(J)*TFAC
40      K=K+NP1
      NITG=NITG+1
      IF (NITG.GE.2) LOGIC1=.TRUE.
      IF (NITG.GE.4) LOGIC2=.FALSE.
      IF (NITG.GE.5) LOGIC3=.TRUE.
265      C * * * * *
      C * * * * *
      NUPD=0
      IF (NITG.LE.5) RETURN
      IF (SOELT.GT.ACC2) RETURN
      IF (KEND.EQ.1) RETURN
      NUPD=1
      DO 45 J=1,NDIM
        KEY(J)=0
45      RETURN
      ENO

```

```

1      SUBROUTINE FIX(Z,G,SING)
      C CALLED BY MRK.
      C FIX GENERATES THE REDUCED MATRIX.
      DIMENSION AE(1),Z(1),G(1)
      DIMENSION N(30),Y(30)
      COMMON/TAB2/A(30,30),KEY(30)
      COMMON/TAB3/NDIM,IDIM
      EQUIVALENCE(AE(1),A(1,1))
      K=0
      DD 3 J=1,NDIM
      IF(KEY(J).LT.1)GOTD 3
      K=K+1
      N(K)=J
      Y(K)=Z(J)
      3 CONTINUE
      JI=0
      DO 2 J=1,K
      L=N(J)
      LM=(L-1)*IDIM
      LML=LM+L
      DO 1 I=1,K
      JI=JI+1
      LCM=LM+N(I)
      AE(JI)=AE(LCM)
      1 CONTINUE
      2 CONTINUE
      J=N(1)
      IF(K.EQ.1)G(J)=Y(1)/AE(1)
      IF(K.GT.1) CALL MATSDL(K,SING,Y,G,N)
      RETURN
      END
30

```

```

FIX 2
FIX 3
FIX 4
FIX 5
FIX 6
FIX 7
FIX 8
FIX 9
FIX 10
FIX 11
FIX 12
FIX 13
FIX 14
FIX 15
FIX 16
FIX 17
FIX 18
FIX 19
FIX 20
FIX 21
FIX 22
FIX 23
FIX 24
FIX 25
FIX 26
FIX 27
FIX 28
FIX 29
FIX 30
FIX 31
FIX 32

```

03/19/78 14.19.29

FTN 4.6+452

SUBROUTINE MATSOL 76/76 OPT=I ROUNO=-*/ TRACE

```

1  SUBROUTINE MATSOL(N,ERROR,Y,V,NA)
C   CALLED BY FIX.
C   MATSOL PERFORMS A LOWER-UPPER DECOMPOSITION AND SOLVES
C   THE SYSTEM OF EQUATIONS BY BACK SUBSTITUTION.
C   PIVOTING IS DONE BY COLUMNS BECAUSE THE MODIFIED FORM
C   OF THE JACOBIAN USED RESULTS IN WIDELY DIFFERENT ORDERS
C   OF MAGNITUDE IN THE COLUMNS.
C   DIMENSION Y(1),V(1),NA(1),IX(30)
COMMON/TAB2/A(900),KEY(30)
INTEGER P,Q,PIVOT
ZABS(X)=ABS(X)
C   ZABS(X)=OABS(X)
C   PERFORM THE LOWER-UPPER DECOMPOSITION.
ERROR=0.0
00 1 I=1,N
1 IX(I)=I
00 2 J=PIVOT,N
00 3 PIVOT=I,N
BIG=0.0
K=PIVOT*PIVOT*N-N
00 4 J=PIVOT,N
T=ZABS(A(K))
K=K*N
IF (BIG.GT.T) GOTO 2
BIG=T
Q=J
2 CONTINUE
3 IF (BIG) 3,10,3
3 IF (Q.EQ.PIVOT) GOTO 5
II=N*PIVOT-N
JJ=N*Q-N
00 4 I=I,N
II=II+I
JJ=JJ+I
T=A(II)
A(II)=A(JJ)
4 A(JJ)=T
IT=IX(PIVOT)
IX(PIVOT)=IX(Q)
IX(Q)=IT
5 IF (PIVOT.EQ.N) GOTO 9
P=PIVOT+I
II=N*PIVOT-N
IIP=III+PIVOT
00 6 I=P,N
IPII=I+II
T=-A(IPII)/A(IIP)
IF (T) 6,8,6
6 JJ=N*P-N
00 7 J=P,N
II=I+JJ
JJPP=JJ+PIVOT
JJ=JJ+N
7 A(II)=A(II)+T*A(JJPP)
8 CONTINUE
9 ERROR=0.0
GOTO 13
10 WRITE(6,II)N

```

```

        ERROR=1.0
        11 FORMAT (40H0***** SUBROUTINE MATSOL ERROR *****/.51H THE
        1 SYSTEM OF EQUATIONS IS SINGULAR. (NOTE - N=.14/)
        RETURN
C SOLVE THE SYSTEM BY BACK SUBSTITUTION.
        13 L=N-1
        DO 15 I=1,L
            1P=I+1
            11=N*I-N
            1PII=I+11
            YA=Y(I)/A(1PII)
            DO 15 J=IP,N
                1JII=J+11
                15 Y(J)=Y(J)-A(1JII)*YA
            1=N
            1N=I+N*1-N
            1N1=Y(1)/A(1N)
            16 IQ=I
            1=I-1
            IF(I) 19,19,17
            17 T=Y(1)
            1NJ=1+N*IQ-N
            DO 18 J=IQ,N
                T=T-A(1NJ)*Y(J)
            1NJ=1NJ+N
            18 CONTINUE
            1N1=1+N*I-N
            1N1=Y(1)/A(1N1)
            GOTO 16
        DO 20 I=1,N
            J=IX(1)
            J=NA(J)
            20 V(J)=Y(1)
        RETURN
        END

```


03/19/78 14.19.29

FTN 4.6+452

76/76 OPT=1 ROUNO=+--*/ TRACE

SUBROUTINE CONT

```

1      SUBROUTINE CONT(HP,HPP,ZO,ZOP,Z1,Z1P,Z2)
      C CALLED BY MRK.
      C CONT COMPUTES THE COEFFICIENTS TO ESTIMATE THE TRUNCATION ERROR.
      C A FOURTH DEGREE POLYNOMIAL IS FITTED TO THE DATA, AND ITS
      UP=HP
      UPP=HP+HPP
      UO=UPP-UP
      UPS=UP**2
      UPPS=UPP**2
      OET=UP*UO**2
      ZO=((UP-UO-UO)/(UPS-UP/UPPS))/OET
      ZOP=((UP**2/(UP*UPP))/OET
      Z1=((UO+UO-UP)/UPS)/OET
      Z1P=UD/(UP*OET)
      Z2=UP/(UPPS*OET)
      RETURN
      ENO
2      CONT
3      CONT
4      CONT
5      CONT
6      CONT
7      CONT
8      CONT
9      CONT
10     CONT
11     CONT
12     CONT
13     CONT
14     CONT
15     CONT
16     CONT
17     CONT
18     CONT

```

03/19/78 14.19.29

FTN 4.6+452

76/76 DPT=1 RDUND=+==/ TRACE

SUBROUTINE PDTR

```

1      SUBROUTINE PDTR(CD0,C01,C02,DU,H*HP,HPP)
      C   CALLED BY MRK.
      C   PDTR FINDS THE COEFFICIENTS FOR THE TWO STEP FORMULA
      C   PREDICTING F AND R.
      CD2=(H*(H*HP))/(HPP*(HP*HPP))
      C01=(-H*(H*HP+HPP))/(HP*HPP)
      CD0=1.0-C01-CD2
      C00=1.0/C00
      C01=CD1*C00
      CD2=CD2*C00
      RETURN
      END
10
2      PDTR
3      PDTR
4      PDTR
5      PDTR
6      PDTR
7      PDTR
8      PDTR
9      PDTR
10     PDTR
11     PDTR
12     PDTR
13     PDTR

```

```

1      SUBROUTINE DER(Y,DERY)
2      C CALLED BY MRK.
3      C DER IS USER SUPPLIED.
4      C IT COMPUTES THE REQUIRED DERIVATIVES AND JACOBIAN ELEMENTS.
5      DIMENSION Y(30),DERY(30)
6      COMMON/TAB2/W(30,30),KEY(30)
7      COMMON/TAB3/NDIM,NDIM
8      COMMON/TAB4/NITG,NDER,NJAC,NM1,NDM1
9      Y1=Y(1)
10     Y2=Y(2)
11     Y3=Y(3)
12     Y4=Y(4)
13
14     C FIND THE TIME DERIVATIVES.
15     DERY(1)=-Y1+2.0
16     DERY(2)=-10.*Y2+20.*Y1*Y1
17     DERY(3)=-40.*Y3+80.*(Y1*Y1+Y2*Y2)
18     DERY(4)=-100.*Y4+200.*(Y1*Y1+Y2*Y2+Y3*Y3)
19     NDER=NDER+1
20     RETURN
21     ENTRY JACOB
22     C FIND THE DIAGONAL MODIFIED JACOBIAN ELEMENTS.
23     W(1,1)=-Y1
24     W(2,2)=-10.0*Y2
25     W(3,3)=-40.0*Y3
26     W(4,4)=-100.0*Y4
27     RETURN
28     ENTRY JAC
29     C FIND THE OFF DIAGONAL MODIFIED JACOBIAN ELEMENTS THAT ARE
30     C BEING USED IN THE CURRENT TIME STEP.
31     DO 500 I=1,NDIM
32     DO 500 J=1,NDIM
33     IF(1.EQ.J)GO TO 500
34     W(I,J)=0.0
35     CONTINUE
36     IF(KEY(2).LT.1)GO TO 3
37     W(2,1)=40.*Y1*Y1
38     IF(KEY(3).LT.1)GO TO 4
39     W(3,1)=160.*Y1*Y1
40     W(3,2)=160.*Y2*Y2
41     IF(KEY(4).LT.1)GO TO 5
42     W(4,1)=400.*Y1*Y1
43     W(4,2)=400.*Y2*Y2
44     W(4,3)=400.*Y3*Y3
45     CONTINUE
46     NJAC=NJAC+1
47     RETURN
48     END

```

BLOCK	ADDRESS	LENGTH	FILE
/TAB1/	110	5	LGO
/TAB3/	115	2	LGO
/TAB4/	117	5	LGO
KINT	124	454	LGO
/TAB2/	600	1642	LGO
MRK	2442	2340	LGO
FIX	5002	173	LGO
MATSOL	5175	370	LGO
CONT	5565	35	LGO
POTR	5622	21	LGO
OER	5643	141	LGO
/STP.ENO/	6004	1	SL-FINLIB
/FCL.C./	6005	23	SL-FINLIB
/QB.10./	6030	136	SL-FINLIB
QBNTRY=	6166	1	SL-FINLIB
COMIO=	6167	44	SL-FINLIB
FECMSK=	6233	41	SL-FINLIB
FLTOUT=	6274	315	SL-FINLIB
FMTP=	6611	373	SL-FINLIB
FORSYS=	7204	533	SL-FINLIB
FORUTL=	7737	44	SL-FINLIB
GETFIT=	10003	43	SL-FINLIB
KODER=	10046	467	SL-FINLIB
OUTC=	10535	171	SL-FINLIB
OUTCOM=	10726	204	SL-FINLIB
CLOCK=	11132	43	SL-FINLIB
ALOG	11175	77	SL-FINLIB
EXP	11274	100	SL-FINLIB
SGRT	11374	46	SL-FINLIB
SYSALO=	11442	1	SL-FINLIB
SYS=1ST	11443	62	SL-FINLIB
XT01*	11525	33	SL-FINLIB
XT01=	11560	10	SL-FINLIB
XT0Y*	11570	55	SL-FINLIB

ERR CRIT = 1.0000E-08
 THE UPPER BOUND FOR THE INITIAL STEP SIZE IS .1000E-03

RUN TIME = 6.2000E-02

MRK CALLS = 302 DERIVATIVE CALLS = 608 JACOBEAN CALLS = 6

NO. OF MATRIX INVERSIONS = 6 AVERAGE DIMENSION = 2.0000E+00

1.064686985169E+00 1.560027657738E+00 5.820783762433E+00 6.283142314819E+01

FINAL TIME = 6.6874E-02

ERROR = 0.

ERR CRIT = 1.0000E-06

THE UPPER BOUND FOR THE INITIAL STEP SIZE IS .1000E-03
 RUN TIME = 2.0000E-02
 MRK CALLS = 98 DERIVATIVE CALLS = 198 JACOBEAN CALLS = 7
 NO. OF MATRIX INVERSIONS = 7 AVERAGE DIMENSION = 1.1429E+00
 1.064866985165E+00 1.560027643336E+00 5.820782077036E+00 6.283138999464E+01
 FINAL TIME = 6.6874E-02
 ERROR = 1.6598E-05

ERR CRIT = 1.0000E-04
 THE UPPER BOUND FOR THE INITIAL STEP SIZE IS .1000E-03
 RUN TIME = 8.0000E-03
 MRK CALLS = 36 DERIVATIVE CALLS = 75 JACOBEAN CALLS = 5
 NO. OF MATRIX INVERSIONS = 5 AVERAGE DIMENSION = 1.0000E+00
 1.064866985040E+00 1.560027220819E+00 5.820729635103E+00 6.283018040171E+01
 FINAL TIME = 6.6874E-02
 ERROR = 6.2196E-04

ERR CRIT = 1.0000E-02
 THE UPPER BOUND FOR THE INITIAL STEP SIZE IS .1000E-03
 RUN TIME = 4.0000E-03
 MRK CALLS = 19 DERIVATIVE CALLS = 40 JACOBEAN CALLS = 1
 NO. OF MATRIX INVERSIONS = 1 AVERAGE DIMENSION = 1.0000E+00
 1.064866983562E+00 1.560022085058E+00 5.820054963861E+00 6.281755530273E+01
 FINAL TIME = 6.6874E-02
 ERROR = 6.9435E-03

ERR CRIT = 1.0000E-08
 THE UPPER BOUND FOR THE INITIAL STEP SIZE IS .1000E-03
 RUN TIME = 1.5200E-01
 MRK CALLS = 710 DERIVATIVE CALLS = 1433 JACOBEAN CALLS = 117
 NO. OF MATRIX INVERSIONS = 117 AVERAGE DIMENSION = 1.0599E+00
 1.3605926680748E+00 3.331503023687E+00 2.459212912655E+01 1.184760458930E+03
 FINAL TIME = 4.4721E-01

ERROR = 0.

ERR CRIT = 1.0000E-06
THE UPPER BOUND FOR THE INITIAL STEP SIZE IS .1000E-03
RUN TIME = 5.4000E-02
MRK CALLS = 227 DERIVATIVE CALLS = 481 JACOBEAN CALLS = 107
NO. OF MATRIX INVERSIONS = 107 AVERAGE DIMENSION = 1.0187E+00
1.360592679192E+00 3.331502847640E+00 2.459212622392E+01 1.184760155095E+03
FINAL TIME = 4.4721E-01
ERROR = 1.5192E-04

ERR CRIT = 1.0000E-04
THE UPPER BOUND FOR THE INITIAL STEP SIZE IS .1000E-03
RUN TIME = 1.9000E-02
MRK CALLS = 76 DERIVATIVE CALLS = 161 JACOBEAN CALLS = 44
NO. OF MATRIX INVERSIONS = 44 AVERAGE DIMENSION = 1.1818E+00
1.360592638623E+00 3.331498164931E+00 2.459204996244E+01 1.184752221933E+03
FINAL TIME = 4.4721E-01
ERROR = 4.1187E-03

ERR CRIT = 1.0000E-02
THE UPPER BOUND FOR THE INITIAL STEP SIZE IS .1000E-03
RUN TIME = 8.0000E-03
MRK CALLS = 35 DERIVATIVE CALLS = 74 JACOBEAN CALLS = 16
NO. OF MATRIX INVERSIONS = 16 AVERAGE DIMENSION = 1.0000E+00
1.360592212851E+00 3.331444363491E+00 2.459120140067E+01 1.184662824373E+03
FINAL TIME = 4.4721E-01
ERROR = 4.8819E-02

ERR CRIT = 1.0000E-08
THE UPPER BOUND FOR THE INITIAL STEP SIZE IS .1000E-03
RUN TIME = 5.8300E-01
MRK CALLS = 2470 DERIVATIVE CALLS = 5090 JACOBEAN CALLS = 1271
NO. OF MATRIX INVERSIONS = 1271 AVERAGE DIMENSION = 1.0110E+00

1.949747629598E+00 7.559625514936E+00 1.215530923603E+02 2.960426794111E+04

FINAL TIME = 2.9907E+00

ERROR = 0.

ERR CRIT = 1.0000E-06
THE UPPER BOUND FOR THE INITIAL STEP SIZE IS .1000E-03

RUN TIME = 2.1100E-01

MRK CALLS = 792 DERIVATIVE CALLS = 1738 JACOBEAN CALLS = 645

NO. OF MATRIX INVERSIONS = 645 AVERAGE DIMENSION = 1.0186E+00

1.949747627407E+00 7.559625496519E+00 1.215530817893E+02 2.960426770378E+04

FINAL TIME = 2.9907E+00

ERROR = 1.3866E-04

ERR CRIT = 1.0000E-04
THE UPPER BOUND FOR THE INITIAL STEP SIZE IS .1000E-03

RUN TIME = 6.6000E-02

MRK CALLS = 243 DERIVATIVE CALLS = 532 JACOBEAN CALLS = 211

NO. OF MATRIX INVERSIONS = 211 AVERAGE DIMENSION = 1.1137E+00

1.949747551756E+00 7.559624851115E+00 1.215530613720E+02 2.960425772858E+04

FINAL TIME = 2.9907E+00

ERROR = 5.1263E-03

ERR CRIT = 1.0000E-02
THE UPPER BOUND FOR THE INITIAL STEP SIZE IS .1000E-03

RUN TIME = 2.4000E-02

MRK CALLS = 87 DERIVATIVE CALLS = 188 JACOBEAN CALLS = 64

NO. OF MATRIX INVERSIONS = 69 AVERAGE DIMENSION = 1.3478E+00

1.949745357133E+00 7.559606142033E+00 1.215524718396E+02 2.960397049424E+04

FINAL TIME = 2.9907E+00

ERROR = 1.4874E-01

ERR CRIT = 1.0000E-08
THE UPPER BOUND FOR THE INITIAL STEP SIZE IS .1000E-03

RUN TIME = 1.1310E+00
 MRK CALLS = 4517 DERIVATIVE CALLS = 9414 JACOBEAN CALLS = 3262
 NO. OF MATRIX INVERSIONS = 3282 AVERAGE DIMENSION = 1.0545E+00
 1.999999997939E+00 7.999999981676E+00 1.359999993817E+02 3.712799965962E+04
 FINAL TIME = 2.0000E+01
 ERROR = 0.

ERR CRIT = 1.0000E-06
 THE UPPER BOUND FOR THE INITIAL STEP SIZE IS .1000E-03
 RUN TIME = 3.7500E-01
 MRK CALLS = 1382 DERIVATIVE CALLS = 2979 JACOBEAN CALLS = 1248
 NO. OF MATRIX INVERSIONS = 1248 AVERAGE DIMENSION = 1.1482E+00
 1.999999997927E+00 7.999999981572E+00 1.359999993782E+02 3.712799965770E+04
 FINAL TIME = 2.0000E+01
 ERROR = 9.6008E-07

49

ERR CRIT = 1.0000E-04
 THE UPPER BOUND FOR THE INITIAL STEP SIZE IS .1000E-03
 RUN TIME = 1.4300E-01
 MRK CALLS = 480 DERIVATIVE CALLS = 1115 JACOBEAN CALLS = 467
 NO. OF MATRIX INVERSIONS = 467 AVERAGE DIMENSION = 1.2934E+00
 1.999999997843E+00 7.999999980828E+00 1.3599999967363E+02 3.712799964636E+04
 FINAL TIME = 2.0000E+01
 ERROR = 6.6390E-06

ERR CRIT = 1.0000E-02
 THE UPPER BOUND FOR THE INITIAL STEP SIZE IS .1000E-03
 RUN TIME = 5.7000E-02
 MRK CALLS = 183 DERIVATIVE CALLS = 428 JACOBEAN CALLS = 165
 NO. OF MATRIX INVERSIONS = 165 AVERAGE DIMENSION = 1.7273E+00
 1.999999996770E+00 7.999999971291E+00 1.359999990312E+02 3.712799946670E+04
 FINAL TIME = 2.0000E+01
 ERROR = 9.6462E-05

***** TERRYU2 0001006 LINES PRINTED

DISTRIBUTION LIST

<u>No. of</u> <u>Copies</u>	<u>Organization</u>	<u>No. of</u> <u>Copies</u>	<u>Organization</u>
12	Commander Defense Technical Info Center ATTN: DDC-DDA Cameron Station Alexandria, VA 22314	1	Commander US Army Communications Rsch and Development Command ATTN: DRDCO-PPA-SA Fort Monmouth, NJ 07703
1	Commander US Army Materiel Development and Readiness Command ATTN: DRCDMD-ST 5001 Eisenhower Avenue Alexandria, VA 22333	1	Commander US Army Electronics Research and Development Command Technical Support Activity ATTN: DELSD-L Fort Monmouth, NJ 07703
2	Commander US Army Armament Research and Development Command ATTN: DRDAR-TSS Dover, NJ 07801	1	Director US Army Electronics Command Atmospheric Sciences Laboratory ATTN: M.G. Heaps White Sands Missile Range, NM 88002
1	Commander US Army Armament Materiel Readiness Command ATTN: DRSAR-LEP-L, Tech Lib Rock Island, IL 61299	2	Commander US Army Missile Command ATTN: DRDMI-R DRDMI-YDL Redstone Arsenal, AL 35809
1	Director US Army ARRADCOM Benet Weapons Laboratory ATTN: DRDAR-LCB-TL Watervliet, NY 12189	1	Commander US Army Tank Automotive Research & Development Command ATTN: DRDTA-UL Warren, MI 48090
1	Commander US Army Aviation Research and Development Command ATTN: DRSAB-E P.O. Box 209 St. Louis, MO 63166	3	Commander US Army Research Office ATTN: J. Chandra D. Squire R. Lontz P.O. Box 12211 Research Triangle Park, NC 27709
1	Director US Army Air Mobility Research and Development Laboratory Ames Research Center Moffett Field, CA 94035		

DISTRIBUTION LIST

<u>No. of Copies</u>	<u>Organization</u>	<u>No. of Copies</u>	<u>Organization</u>
1	Director US Army TRADOC Systems Analysis Activity ATTN: ATAA-SL, Tech Lib White Sands Missile Range NM 88002	1	Bell Laboratories ATTN: D. Edelson Murray Hill, NJ 07974
5	Commander Naval Research Laboratory ATTN: W. Ali D. Strobel J. Brown, Code 7700 E. Oran, Code 6020 Tech Lib, Code 2020 Washington, DC 20375	1	Boeing Computer Services, Inc. Energy Technology Applications ATTN: K.W. Neves P.O. Box 24346 Seattle, WA 98124
3	AFGL (LKB, J. Paulson, K. Champion, W. Swider) Hanscom AFB, MA 01731	1	General Electric Company Valley Forge Space Technology Center ATTN: M. Bortner P.O. Box 8555 Philadelphia, PA 19101
2	Director National Bureau of Standards US Department of Commerce ATTN: R.F. Hampson R. Kraft, Div 205.01 Washington, DC 20234	1	General Electric Company Flight Propulsion Division ATTN: Tech Lib Cincinnati, OH 45215
3	Lockheed Palo Alto Rsch Lab ATTN: J. Reagan R. Sears Tech Info Ctr 3521 Hanover Street Palo Alto, CA 94304	1	Hercules Incorporated Alleghany Ballistic Lab ATTN: Tech Lib Cumberland, MD 21501
1	Director National Aeronautics and Space Administration Goddard Space Flight Center ATTN: R. Goldberg (Code 912) Greenbelt, MD 20771	1	Mission Research Corporation ATTN: M. Scheibe 735 State Street P.O. Drawer 719 Santa Barbara, CA 93102
		1	Pacific-Sierra Research Corp. ATTN: E. Fields 1456 Cloverfield Blvd Santa Monica, CA 90404
		1	Paul Gough Associates, Inc. ATTN: P.S. Gough P.O. Box 1614 Portsmouth, NH 03801
		1	Physics International Company 2700 Merced Street Leandro, CA 04577

DISTRIBUTION LIST

<u>No. of</u> <u>Copies</u>	<u>Organization</u>	<u>No. of</u> <u>Copies</u>	<u>Organization</u>
1	R&D Associates ATTN: F. Gilmore P.O. Box 9695 Marina Del Rey, CA 90291	2	Johns Hopkins University/APL Chemical Propulsion Info Agency ATTN: T. Christian Johns Hopkins Road Laurel, MD 20810
1	Science Applications, Inc. ATTN: R.B. Edelman 23146 Cumorah Crest Woodland Hills, CA 91364	1	Massachusetts Institute of Technology Dept of Mechanical Engineering ATTN: T. Toong Cambridge, MA 02139
3	SRI International ATTN: J. Peterson D. Golden Tech Lib 333 Ravenswood Avenue Menlo Park, CA 94025	1	Director Graduate Center of Applied Science New York University ATTN: M. Summerfield 26-36 Stuyvesant New York, NY 10003
2	Thiokol Corporation Huntsville Division ATTN: D. Flanigan Tech Lib Huntsville, AL 35807	1	Oregon State University Department of Mathematics ATTN: D.C. Jespersen Corvallis, OR 97331
1	Battelle Memorial Institute ATTN: Tech Lib 505 King Avenue Columbus, OH 43201	1	Pennsylvania State University Dept of Mechanical Engineering ATTN: K. Kuo University Park, PA 16801
1	Brigham Young University Dept of Chemical Engineering ATTN: R. Coates Provo, UT 84601	2	Princeton University Forrestal Campus Library ATTN: L. Caveny Tech Lib P.O. Box 710 Princeton, NJ 08540
1	Georgia Institute of Tech School Of Aerospace Engrg ATTN: B.T. Zinn Atlanta, GA 30332	1	Purdue University Indianapolis Regional Campus ATTN: J.M. Gersting 1201 East 38th Street Indianapolis, IN 46205
1	Institute of Gas Technology ATTN: D. Gidaspow 3424 S. State Street Chicago, IL 60616		

DISTRIBUTION LIST

<u>No. of Copies</u>	<u>Organization</u>	<u>No. of Copies</u>	<u>Organization</u>
2	Purdue University School of Mechanical Engrg ATTN: J. Osborn S.N.B. Murthy TSPC Chaffee Hall West Lafayette, IN 47906	2	University of Wisconsin - Madison Mathematics Research Center ATTN: J. Nohel, Director B. Noble 610 Walnut Street Madison, WI 53706
1	University of California Electrical Engineering and Computer Science Dept ATTN: W. Kahan 567 Evans Hall Berkeley, CA 94720	1	Utah State University Center for Research in Aeronomy ATTN: L. Megill Logan, UT 84321
1	University of California ATTN: A.C. Hindmarsh, L-310 Lawrence Livermore Laboratory P.O. Box 808 Livermore, CA 94550	1	Wayne State University Department of Engineering ATTN: R. Kummeler Detroit, MI 48202
1	University of California, San Diego AMES Department ATTN: F. Williams P.O. Box 109 La Jolla, CA 92037		<u>Aberdeen Proving Ground</u> Dir, USAMSAA ATTN: DRXSY-D DRXSY-MP, H. Cohen Cdr, USATECOM ATTN: DRSTE-TO-F Dir, Wpns Sys Concepts Team, Bldg E3516, EA ATTN: DRDAR-ACW
1	University of Illinois Dept of Aeronautical Engrg ATTN: H. Krier Transportation Bldg, Rm 105 Urbana, IL 61801		
1	University of Illinois Dept of Electrical Engineering ATTN: C. Sechrist, Jr. Urbana-Champaign Campus Urbana, IL 61801		
1	University of Wisconsin Dept of Chemical Engineering ATTN: J.R. Bowen 1415 Johnson Drive Madison, WI 53706		

USER EVALUATION OF REPORT

Please take a few minutes to answer the questions below; tear out this sheet and return it to Director, US Army Ballistic Research Laboratory, ARRADCOM, ATTN: DRDAR-TSB, Aberdeen Proving Ground, Maryland 21005. Your comments will provide us with information for improving future reports.

1. BRL Report Number _____

2. Does this report satisfy a need? (Comment on purpose, related project, or other area of interest for which report will be used.)

3. How, specifically, is the report being used? (Information source, design data or procedure, management procedure, source of ideas, etc.) _____

4. Has the information in this report led to any quantitative savings as far as man-hours/contract dollars saved, operating costs avoided, efficiencies achieved, etc.? If so, please elaborate.

5. General Comments (Indicate what you think should be changed to make this report and future reports of this type more responsive to your needs, more usable, improve readability, etc.) _____

6. If you would like to be contacted by the personnel who prepared this report to raise specific questions or discuss the topic, please fill in the following information.

Name: _____

Telephone Number: _____

Organization Address: _____

