Modern Systems Analysis, Inc.
1225 WINES DRIVE
ANN ARBOR, MICHIGAN 48103

CR - 134021

Telephone (313)
761-7295

9-12319-73F

FINAL REPORT

for

CONTRACT NAS 9-12319 (Mod. No. 1S)

NUMERICAL INTEGRATION ROUTINES FOR NEAR-EARTH OPERATIONS

by

William F. Powers

September 1973

Modern Systems Analysis, Inc.
1235 WINES DRIVE
ANN ARBOR, MICHIGAN 48103

Telephone (313)
761-7295

9-12319-73F

FINAL REPORT

for

CONTRACT NAS 9-12319 (Mod. No. 1S)

NUMERICAL INTEGRATION ROUTINES FOR NEAR-EARTH OPERATIONS

by

William F. Powers

September 1973

TABLE OF CONTENTS

1. INTRODUCTION

The major goals of the extension of NASA Contract NAS 9-12319 were concerned with building two general purpose numerical integration schemes into the NASA-JSC computer system. There were other minor goals associated with the contract and these will be discussed in Chapters 5 and 6.

Since most of the work was concerned with the numerical integration schemes, the major portion of this report will be devoted to describing the state-of-the-art of numerical integration, the particular integrators built into the JSC computer system, and the use of the new integration packages. Those who are only interested in determining how to use the integrators may proceed immediately to the Appendices, which are self-contained.

The remaining portion of the report is as follows: Chapter 2 contains comments about numerical integration in general; Chapter 3 discusses the extrapolation numerical integration technique; Chapter 4 discusses the variable-order, variable-stepsize Adams numerical integration technique; Chapter 5 presents results concerned with numerical integration and optimization in the JSC PEACE parameter optimization program; and Chapter 6 presents conclusions and recommendations.

2. GENERAL COMMENTS ON NUMERICAL INTEGRATORS

In this chapter we shall briefly discuss the state-of-the-art of numerical integration, especially with respect to the development of general purpose numerical integration subroutines. For additional information, References 1 and 2 should be consulted.

2.1 General Purpose Integrators

As with any engineering system, the selection of a numerical integration subroutine usually involves a tradeoff between stability and performance. However, in the past few years a number of stable (reliable), relatively high-performance general purpose numerical integration subroutines have been developed. It is the purpose of this report to introduce these routines to the Mission Planning and Analysis Division in as simple a format as possible so that the routines will be used. As noted in Chapter 1, the user may proceed directly to the Appendices if he is only interested in learning how to use the subroutines.

The subroutines of this report are referred to as general purpose subroutines because the schemes are relatively flexible and apply to a large class of problems. The schemes derive their flexibility mainly from the fact that they are both variable order and variable stepsize. This fact allows the routines to adapt efficiently to many different physical situations. In addition, the numerical integration routine of Chapter 4 (and Appendix B) has excellent diagnostic capabilities which indicate numerical problems, e.g., excessive roundoff errors.

One of the first questions which must be answered is: "When should I use a general purpose integrator?" The integrators described in Chapters 3 and 4 are very reliable, especially DVDQ (Chap.4), and they have been used with ease on problems from many different disciplines. If a new problem requiring numerical integration is to be solved, a general purpose integrator will not only produce reliable answers in a short time (with a minimum of expensive and tedious human effort) but usually also teach the user something about the physical problem, e.g., the stepsize usually contracts in areas of high acceleration or "action." Thus, it is advisable to use a general purpose integrator when solving a physical problem for the first time.

As to whether or not one should continue to use a general purpose integrator, after fully understanding the process of the physical situation, is problem dependent. That is, since a general purpose integrator is in some sense conservative (stable), one could probably invent rules-of-thumb to be used with a problem-dependent integrator to give a high-performance integrator for that particular problem. (This, of course, is the case with problems in celestial mechanics where many efficient, problem dependent integration schemes are employed.) In this process one must always keep in mind the tradeoff of human effort and computer effort, e.g., it would not be worthwhile to spend many man-hours inventing rules-of-thumb for a particular problem if the problem is to be solved only a few times, whereas the opposite would be true if thousands of simulations are anticipated.

2.2 Main Existing Procedures

The numerical integration schemes which are most often used in engineering analyses are the various fixed stepsize fourth-order Runge-Kutta and Adams predictor-corrector methods,[1,2] especially the Runge-Kutta methods. One reason for this is that the Runge-Kutta method is easy to understand, easy to program, and relatively reliable (once a workable stepsize has been selected). Comparable properties of these two approaches (for fourth-order only) are listed below:

> Runge-Kutta: Based on Taylor series expansion; self-starting; requires four function evaluations per step; functions evaluated on a symmetrical, but unequal grid; difficult to approximate the local truncation error, which makes automatic variable stepsizing difficult.
>
> Adams Predictor-Corrector: Based on interpolation formulas; not self-starting, usually requires fourth-order Runge-Kutta scheme to develop the first few steps; after the starting phase, requires two* function evaluations per step; functions evaluated on an equal grid; relatively ease to approximate the local truncation error, which makes automatic variable stepsizing relatively easy; usually less stable than the Runge-Kutta method.

From the comparisions above, one can see that the Runge-Kutta (or single step) and predictor-corrector (or multistep) methods are basically different, and thus, the performance of an integrator is problem dependent. Also, at first glance, the predictor-corrector method may appear to be faster because it only requires half as many function evaluations (after startup) as the Runge-Kutta method. However, this is not

*This number may be higher in some schemes when more than one correction is allowed.

necessarily the case because on some problems a smaller step-size is required for the predictor-corrector (because it is less stable) and/or the "overhead" (time associated with carrying out the requirements of the integrator) of a predictor-corrector scheme is higher than for a corresponding Runge-Kutta scheme.

With regard to a variable stepsize, which is a necessity in many problems, one usually uses physical reasoning or a halving-and-doubling procedure to adjust the stepsize in a Runge-Kutta scheme. However, in the past few years a more attractive technique has been developed, and this will be described in Chapter 2.3. By comparing predictor and corrector values in a predictor-corrector scheme, it is relatively easy to adjust the stepsize.

2.3 Recent Developments

In the opinion of this author, the three main developments in numerical integration in the past eight years have been the following (in chronological order):

(i) The development of an efficient rational function extrapolation numerical integration scheme;[3]

(ii) The development of higher-order Runge-Kutta formulas with relatively efficient means for stepsize control;[4,5]

(iii) The development of high-order, variable-order, variable-stepsize Adams methods in user oriented subroutine form.[1,6]

This report is based on the advances noted in (i) and (iii).

The higher-order Runge-Kutta schemes, sometimes called Fehlberg integration, are not included because MPAD already has a capability in this area.

All of the general purpose integrators use a variable stepsize (although they can be used with a fixed stepsize). The basis for the adjustment of the stepsize in (i) and (iii) will be discussed in Chapters 3 and 4. The means for efficient adjustment of stepsize in Runge-Kutta schemes[4,5] is as follows. Runge-Kutta formulas can be developed for any order of accuracy, and for a given order the formula is not unique. For example, there exist numerous well-known fourth-order Runge-Kutta formulas. It can be shown that the <u>least</u> number of function evaluations required for a Runge-Kutta formula of order four is four, of order five is six, of order six is seven, of order seven is nine, and so on. Of course, formulas exist for the orders mentioned above which require more than the least number mentioned, and this is the basis for the efficient estimation of the stepsize. For example, suppose we wish to develop a fifth-order Runge-Kutta scheme with a reliable stepsize adjustment procedure, and we select a fifth-order formula which requires the least number of function evaluations possible, i.e., six. A sixth-order formula requires at <u>least</u> seven function evaluations. However, suppose we construct a sixth-order formula with parameter values which match those of the fifth-order formula on the same six function evaluations required by the fifth-order formula, i.e., we embed the fifth-order formula in the sixth-order formula. If this is done, then the remaining terms in the sixth-order formula give an excellent estimate of the truncation error for the fifth-order formula, which can then be used to adjust the stepsize. Fehlberg[4] has shown that the resultant sixth-order formula requires eight function evaluations (whereas the minimum number for an arbitrary sixth-order formula is seven). In summary then, for a fifth-order Runge-Kutta formula with the Fehlberg stepsize modification procedure, eight function evaluations are required (as opposed to six for a fixed stepsize fifth-order formula).

To conclude this section, we mention some recent comparative studies of numerical integration schemes.[7-12] The first statement to be made is that it is very difficult to draw definite conclusions from these studies. For example, the extrapolation scheme will perform "below par" with a very poor choice of initial stepsize estimate, and thus, this scheme performs very well in Refs. 7-10, 12, but relatively poorly in Ref. 11. Also, some reports use the specified error tolerance as the accuracy indicator whereas others compare the number of significant digits (compared to either an analytical solution or a finely-tuned numerical solution which is assumed to be exact to a large number of digits). These two methods of comparison can cause differing conclusions.

Another important point in comparing integrators is the accuracy required. For low accuracies, e.g., 3 to 4 significant digits, the sophisticated integration packages will probably perform poorly. However, in aerospace applications, high accuracy is usually desired and the situation reverses.

Finally, a summary of major considerations in selecting or comparing integration schemes is listed below, assuming comparable accuracies for the integrators.

(1) CPU Time: This is probably the most important parameter to MPAD users since it indicates how fast the integrator is, i.e., how much computer time is involved.

(2) Number of Function Evaluations: This parameter indicates how many times the right-hand sides of the differential equations are evaluated in a given run. In many comparisons this parameter is given the most emphasis (as opposed to CPU time) with the assumption that the smaller the number, the better the integrator. However, the scheme with the least number of function evaluations may not be the fastest (i.e., least CPU time) because of more overhead. The importance of the number of function evaluations parameter increases as the complexity of the right-hand sides of the differential equations increases. For example, an integrator with little overhead, large number of function evaluations may be best when a spherical earth is assumed in a gravitational model, whereas an integrator with large overhead, small number of function evaluations is probably best if an extensive oblate model is employed.

(3) Overhead: It is hard to put a number on this property of an integrator because it has to do with the time required to carry-out the logical structure of the integrator, e.g., checks, rules-of-thumb, etc. built into

the integrator. Usually, Runge-Kutta methods involve less overhead than predictor-corrector methods, and the DVDQ (Chap.4) integration package involves more overhead than any other method reported in the comparisons in Refs. 7-12. However, on many problems it is the fastest integrator because of the small number of function evaluations required, which is due to the extensive logical structure built into the program.

(4) Storage: In some cases the amount of computer storage required for an integrator is of importance. As integrators are developed which are applicable to almost all problems, this may be of some concern for small computer systems (e.g., DVDQ requires much more storage than other popular integrators; see Appendix B).

(5) Scaling: The scaling of a physical problem can make or break some integrators. We have found that the routines of Chapters 3 and 4 retain many of their desirable properties even on poorly scaled problems, which is not the case for most other integrators.

(6) Stiffness: A subject of considerable interest in numerical integration in recent years is "stiff" systems of differential equations. A stiff system is one which possesses at least two variables whose natural time scales are significantly different (on the order of 10^3), e.g., a linear system with characteristic roots of -1, -1000. See Refs. 1 and 2 for further discussions of stiff systems.

Reference 1 contains a subroutine for such systems, and F. T. Krogh of the Jet Propulsion Laboratory is currently modifying DVDQ to include stiff systems.[13]

3. THE EXTRAPOLATION METHOD

The goal of this chapter and Chapter 4 is not to develop the underlying equations in the integration packages since these are readily available in textbook form, e.g., Refs. 1 and 2. Instead we shall mainly discuss the basis for the methods and present simple examples to illustrate the main features. It is assumed that the reader is already familiar with the basic concepts of numerical integration, e.g., knowledge of the workings of the basic Runge-Kutta and predictor-corrector schemes.

3.1 History and Basic Motivation of the Method

The simplest formula for the numerical integration of the scalar differential equation

$$\dot{x} = f(t,x) \quad , \; x(t_o)=x_o \tag{3.1}$$

is Euler's formula

$$x(t_{k+1}) = x(t_k) + hf(t_k,x_k), \; (k=0,1,\ .\ .\ .) \tag{3.2}$$

where h is the stepsize and $x(t_k)$ denotes the value of x obtained at the $k^{\underline{th}}$ step in the process. This is a first-order formula, and it can be shown[1] that the global error (i.e., the difference between the true and approximate solutions) can be represented by

$$\text{error at } t_k=x^T(t_k)-x_h(t_k)=hE(t_k)+O(h^2), \tag{3.3}$$

where $x^T(t_k)$ denotes the true solution, $x_h(t_k)$ is the numerical solution for stepsize h, and $E(t_k)$ is a portion of the error which is a function of the differential equation only if round-off error is neglected. In 1927, Richardson[14] noted the following property: suppose (3.2) is used twice, first with a stepsize of h and second with a stepsize of h/2. Then, the error for each calculation must be of the following form:

$$x^T(t_k) - x_h(t_k)=hE(t_k)+0(h^2) \tag{3.4}$$

$$x^T(t_k) - x_{h/2}(t_k)=(h/2)E(t_k)+0(h^2). \tag{3.5}$$

Elimination of $E(t_k)$ from these two equations results in

$$x^T(t_k)=2x_{h/2}(t_k)-x_h(t_k)+0(h^2), \tag{3.6}$$

i.e., a second-order approximation has been obtained by combining two first-order approximations. In many cases such a procedure gives higher accuracy with fewer function evaluations. For example, consider the integration of $\dot{x}=-x$ from $t_o=0$ to $t_f=1$ with $x(0)=1$. The results[1] using Euler's formula (i.e., Eq. (3.2)) and the "Richardson deferred-approach-to-the-limit" or the "extrapolation" approach (e.g., Eq. (3.6)) are shown in Table 3.1.

h	$x_h(1)$ Euler's Formula (# of fn. eval.)	$x_h(1)$ Richardson (# of fn. eval.)
1	0.0000000 (1)	---
1/2	0.2500000 (2)	0.5000000 (3)
1/4	0.3164063 (4)	0.3828125 (7)
1/8	0.3435089 (8)	0.3708116 (15)
1/16	0.3560741 (16)	0.3685393 (31)
1/32	0.3620552 (32)	0.3680364 (63)
1/64	0.3649865 (64)	0.3679177 (127)

TABLE 3.1 Comparison of Euler and Richardson's Deferred-Approach-to-the-Limit for $\dot{x}=-x$.

The true solution to the above problem is $x(1)=0.367879$ (correct to six places). A simple computation shows that the extrapolated value corresponding to fifteen function evaluations is comparable in accuracy to the Euler value requiring sixty-four function evaluations. Thus, in this case quite a gain is achieved with the Richardson idea of extrapolating values from results with a decreasing sequence of stepsizes.

Although the above idea appears very attractive, it was somewhat dormant until W. B. Gragg studied the method extensively in his doctoral dissertation[15] in 1963. The delay in the use of the idea was probably mainly due to the lack of a digital computer. Then, based upon the theoretical studies of Gragg, Bulirsch and Stoer[3] developed an efficient numerical integration scheme involving this idea. The routine presented in Appendix A is a recently improved version of the Ref. 3 subroutine. Since the scheme is relatively new, compared to Runge-Kutta and predictor-corrector methods, a good deal of current research in numerical analysis is concerned with the understanding and application of the extrapolation method. Thus, the scheme should be improved even more in the near future.

3.2 Motivation for Order Increasing Capability

As the example of the previous section demonstrated, the order of approximation was improved by one when the extrapolation was performed on two integrations of the first-order Euler formula. Actually each additional integration of Euler's method, with a monotonically decreasing stepsize, followed by extrapolation increases the order of approximation by one, as in the generation of Table 3.1. The reason for this is because the error term for Euler's method possesses an asymptotic expansion of the form

$$x(t_p;h) \doteq x^T(t_p)+E_1(t_p)h+E_2(t_p)h^2+\ldots+E_m(t_p)h^m+O(h^{m+1}), \qquad (3.7)$$

where $x(t_p;h)$ is the result of the Euler integration at t_p with stepsize h, $x^T(t_p)$ is the true solution at t_p, and $E_1(t_p), \ldots ,$ $E_m(t_p)$ are portions of the error which depend only upon the differential equation (and not h). In Eq. (3.7), the quantities $x^T(t_p), E_1(t_p), \ . \ . \ . \ , E_m(t_p)$ are, in general, unknown.

Consider the use of Euler's formula on m+1 integrations with stepsizes $h_o > h_1 > \ . \ . \ . > h_m$. Then, neglecting terms of order $O(h^{m+1})$, Eq. (3.7) could be evaluated with each of the stepsizes (where the left-hand side of the equation is the value of $x(t_p)$ determined by the integrations), and the result is m+1 linear equations in the unknowns $x^T(t_p), E_1(t_p), \ . \ . \ . \ , E_m(t_p)$. The resultant value for $x^T(t_p)$ is an $m^{\underline{th}}$ -order approximation of the true solution.

In practice one need not solve the system of linear equations mentioned above to obtain the higher-order approximation of $x^T(t_p)$. The sole reason for presenting the above example was to show how successive integrations increase the order by one if the Euler formula is the base formula.

Of course one may use the extrapolation idea with any integration formula as the basic formula. Thus, the question arises as to the existence of a best base formula. The answer to such a question would involve to some degree finding a formula which increases the order of approximation by more than one with each integration; that is, the formula would possess an error term with an asymptotic expansion of the form:

$$x(t_p;h)=x^T(t_p)+\sum_{k=r}^{\infty} E_k(t_p)h^{c\cdot k}. \qquad (c>1)(3.8)$$

Gragg[15] showed that if the modified mid-point rule is used as the basic formula, then $r=1$, $c=2$ in Eq. (3.8), i.e., the error for the modified mid-point rule has an asymptotic expansion of the form:

$$x(t_p;h) = x^T(t_p) + \sum_{k=1}^{\infty} E_k(t_p)h^{2k}. \qquad (3.9)$$

Gragg's[15] modified midpoint rule is as follows:

$$\begin{aligned} & h_i=H/n_i \quad , \quad \{n_i\} = \text{set of } \underline{\text{even}} \text{ increasing integers} \\ & x_o = x(t_o) \\ & x_1 = x_o+h_i f(t_o,x_o) \\ & x_{p+2} = x_p+2h_i f(t_{p+1},x_{p+1}) \ , \ p=0, \ . \ . \ . \ , n_i-1 \\ & x(t_o+H;h_i) = (1/4)x_{n_i+1}+(1/2)x_{n_i}+(1/4)x_{n_i-1}, \end{aligned} \qquad (3.10)$$

where H is the basic stepsize (e.g., the output interval). The integers in $\{n_i\}$ must be either all even or all odd for theoretical reasons; Gragg has shown that an even set has certain advantages over an odd set, and typical choices are {2,4,6,8, 12,16,24,...} and {2,4,8,16,32,64,...}.[3]

Finally, one other theoretical result due to Gragg is that the error between the true solution and the last extrapolated value, say $\overline{x}_m^T(t_p)$, is of the form

$$\left| \overline{x}_m^T(t_p)-x^T(t_p)\right| \leq Kh_o^2h_1^2\cdots h_m^2, \qquad (3.11)$$

where $h_o > h_1 > \ldots > h_m$, if the modified midpoint formula is employed.

3.3 Implementation Notes

The heart of the extrapolation numerical integration scheme is a table of values which we shall call the "T-table", after Bulirsch and Stoer.[3] To understand the operation of the method it is probably best to consider an example of its operation.

Example: As with our previous examples, assume that the base formula is Euler's method. In the previous section we showed that an mth order approximation could be obtained by solving a system of m+1 linear equations. Actually, the solution of the linear system can be avoided by employing a recursive extrapolation process due to Aitken.[17] Let us first write the resultant table and recursive formula, and then use a simple example to motivate the result.

Suppose four integrations with stepsizes $h_o > h_1 > h_2 > h_3$ are employed on the interval $[t_{p-1}, t_p]$ with the resultant values $x(t_p;h_o)$, $x(t_p;h_1)$, $x(t_p;h_2)$, $x(t_p;h_3)$. The following T-table is then constructed by the recursive formula

$$T^i_m = T^{i+1}_{m-1} + (T^{i+1}_{m-1} - T^i_{m-1})/[(h_i/h_{i+m})-1] \quad (3.12)$$

$$\begin{tikzcd}
\text{Order 1 column} \arrow[d] & \text{Order 2 column} \arrow[dd] & \text{Order 3 column} \arrow[ddd] & \text{Order 4 column} \arrow[dddd] \\
x(t_p;h_o)=T_o^o \arrow[dr] & & & \\
 & T_1^o \arrow[dr] & & \\
x(t_p;h_1)=T_o^1 \arrow[ur] \arrow[dr] & & T_2^o \arrow[dr] & \\
 & T_1^1 \arrow[ur] \arrow[dr] & & T_3^o \\
x(t_p;h_2)=T_o^2 \arrow[ur] \arrow[dr] & & T_2^1 \arrow[ur] & \\
 & T_1^2 \arrow[ur] & & \\
x(t_p;h_3)=T_o^3 \arrow[ur] & & &
\end{tikzcd}$$

FIGURE 3.1. T-Table for Polynomial Extrapolation with Euler's Method.

In the table above only the first column is computed by numerical integration. The remaining columns are generated by the algebraic equation (3.12). Thus, with four integrations of the first-order Euler's method, a fourth-order result, T_3^o, can be inferred. In fact, if one sets $m=3$ in Eq. (3.7) and evaluates the equation at $h=h_o,h_1,h_2,h_3$, then the solution of the four equations for $x^T(t_p)$ will be precisely T_3^o, which is obtained by a well-defined recursive process. Intuitively one would expect that the solution becomes more accurate as one either moves down a column or moves to the right along a diagonal. Gragg[15] has proved that under reasonable conditions, the T-table for polynomial extrapolation with the modified midpoint rule has the following properties:

(i) The order n column converges to the solution faster than the order m column, with $n > m$.

(ii) The principal diagonal, i.e., $T_0^0, T_1^0, \ldots, T_j^0, \ldots$, converges to the solution faster than any column.

Thus, one should take the bottom element from the last column as the best estimate of the solution, e.g., T_3^0 in the example of Figure 3.1.

To fix the interpretation of the T-table, let us perform some simple computations. We shall show why T_2^0 is a third-order approximation, and verify that T_2^0 obtained by the recursive relation is the same as the value obtained by solving the corresponding system of linear equations.

The minimum number of integrations for a third-order approximation with Euler's method is three. Thus, assuming $h_0=h, h_1=h/2, h_2=h/4$ we have (at t_p):

$$\begin{aligned} x(h) &= x^T+E_1h+E_2h^2+O(h^3) \\ x(h/2) &= x^T+E_1h/2+E_2h^2/4+O(h^3) \\ x(h/4) &= x^T+E_1h/4+E_2h^2/16+O(h^3) \end{aligned} \tag{3.13}$$

One can easily verify that Eq. (3.12) implies

$$\begin{aligned} T_1^0 &= 2x(h/2)-x(h) \\ T_1^1 &= 2x(h/4)-x(h/2), \end{aligned} \tag{3.14}$$

which correspond to Eq. (3.6) developed earlier. An alternate interpretation of Eqs. (3.14) is that they represent Eq. (3.13) with E_1 eliminated and $O(h^2)$ terms neglected. In fact the elimination of E_1 from Eqs. (3.13) implies

$$\begin{aligned} x^T &= 2x(h/2)-x(h)+h^2E_2/2+O(h^3) \\ x^T &= 2x(h/4)-x(h/2)+h^2E_2/8+O(h^3). \end{aligned} \tag{3.15}$$

Then, upon elimination of E_2 from Eqs. (3.15), the following third-order approximation of x^T is obtained

$$x^T=[x(h)-6x(h/2)+8x(h/4)]/3+O(h^3). \qquad (3.16)$$

This value, of course, corresponds to the value T_2^o obtained by repeated use of Eq. (3.12).

Before we consider variable stepsize and order aspects of the method, note that the name "polynomial extrapolation" is due to the fact that the value of $x(t_p;0)$ is obtained by using a polynomial fit of the data $x(t_p;h_o)$, $x(t_p;h_1)$, ... to extrapolate the value at $h=0$. It is not necessary to use a polynomial fit of the data, and in fact, the Bulirsch-Stoer extrapolation scheme uses a rational function fit of the data. In most simulations to date, the rational function procedure has given better results. In the case of rational function extrapolation with the midpoint rule, the form of the T-table remains the same, however the recursive formula (3.12) is replaced by[3]

$$T^i_{-1}=0$$

$$T^i_o=x(t_p;h_i) \qquad (3.17)$$

$$T^i_k=T^{i+1}_{k-1}+\frac{T^{i+1}_{k-1}-T^i_{k-1}}{\left(\frac{h_i}{h_{i+k}}\right)^2\left[1-\frac{T^{i+1}_{k-1}-T^i_{k-1}}{T^{i+1}_{k-1}-T^{i+1}_{k-2}}\right]-1}. \quad (k\geq 1)$$

These are the formulas in the subroutine of Appendix A.

3.4 Variable-Stepsize, Variable-Order Considerations

The main reason for using variable stepsize and order methods is to develop the most efficient scheme possible for a specified accuracy. From the previous sections it is apparent that more than one stepsize is utilized in the basic operation of the extrapolation algorithm, and the only questions with respect to stepsize have to do with the initial choice and the final choice, e.g., h_o and h_m if $[h_o/2, h_o/4, \ldots, h_m = h_o/2m]$ is the sequence.

The comparative studies in Refs. 7-11 indicate that a poor choice for h_o can cause inefficiency in the scheme, e.g., if h_o is too large the extrapolated values may be contaminated by round-off and if h_o is too small it is not taking full advantage of the extrapolation process. Both the scheme of Appendix A and the original version (pp. 96-99, Ref. 1) have means for adjusting h_o by checking how many values are required for the first column of the T-table (i.e., if too few are required, the stepsize is too small; if too many are required, the stepsize is too large).

The determination of order and the final stepsize, h_m, are somewhat coupled. The algorithm of Appendix A can generate six columns, which implies a possible twelth-order scheme since the orders of the columns go up by twos with the midpoint formula (see Eqs. (3.9), (3.10)). The termination of integrations (or the determination of h_m) varies from basic step to basic step. Integration is terminated when two successive approximations T_k^{m-k} and T_k^{m-k+1} differ by less than the specified error tolerance, and the value T_k^{m-k+1} is taken as the best estimate. This feature is one of the strongest aspects of the scheme since the cutoff is determined by comparing successive better approximations to the solution. One may consult the table on page 90 of Ref. 1 to see a striking example of this feature.

4. VARIABLE ORDER, VARIABLE STEPSIZE INTEGRATORS

In this chapter we shall discuss the basic ideas behind variable order, variable stepsize Adams predictor-corrector methods. Although there exist relatively efficient variable stepsize Runge-Kutta integrators, e.g., Ref. 4, there does not appear to be an efficient way of changing the order in a Runge-Kutta scheme. The two main variable order, variable stepsize Adams methods are those due to Krogh[6] and Gear[1]. Krogh's program is discussed and listed in Appendix B, while Gear's program is listed on pp. 158-166 of Ref. 1.

4.1 Storage Methods

Before we discuss means of storing information available in a predictor-corrector method, let us write the first few Adams formulas for future reference. With

$$\dot{x} = f(t,x),\ x_p = x(t_p), f_p = (t_p, x_p),\ h = t_{p+1} - t_p \qquad (4.1)$$

we have the following.

Adams-Bashforth (Predictors)	Error
$x_{p+1} - x_p = h\ f_p$	$(h^2/2)\ddot{x}$
$x_{p+1} - x_p = (h/2)(3f_p - f_{p-1})$	$(5h^3/12)\dddot{x}$
$x_{p+1} - x_p = (h/12)(23f_p - 16f_{p-1} + 5f_{p-2})$	$(9h^4/24)x^{(4)}$
$x_{p+1} - x_p = (h/24)(55f_p - 59f_{p-1} + 37f_{p-2} - 9f_{p-3})$	$(251h^5/720)x^{(5)}$

$$\vdots \qquad\qquad \vdots \qquad\qquad \vdots$$

Adams-Moulton (Correctors)	Error
$x_{p+1}-x_p=h\ f_{p+1}$	$-(h^2/_2)\ddot{x}$
$x_{p+1}-x_p=(h/2)(f_{p+1}+f_p)$	$-(h^3/_{12})\dddot{x}$
$x_{p+1}-x_p=(h/12)(5f_{p+1}+8f_p-f_{p-1})$	$-(h^4/_{24})x^{(4)}$
$x_{p+1}-x_p=(h/24)(9f_{p+1}+19f_p-5f_{p-1}+f_{p-2})$	$-(19h^5/_{720})x^{(5)}$
$\vdots \qquad \vdots$	$\vdots$

The above formulas are used with a $k^{\underline{th}}$ (or k-1) order predictor and $k^{\underline{th}}$ order corrector, where a formula is $k^{\underline{th}}$ order if its error term is of the form $C_{k+1}\ h^{k+1}\ x^{(k+1)}$. Classically, the Adams predictor and corrector formulas were written as

$$x_{p+1}=x_p + h \sum_{j=o}^{k} a_j f_{p-j} \qquad \text{(Predictor)(4.2)}$$

$$x_{p+1}=x_p + h \sum_{j=o}^{k} a_j^* f_{p+1-j}, \qquad \text{(Corrector)(4.3)}$$

and stored in the computer in the form $x_p, f_p, f_{p-1}, \ . \ . \ . \ , f_{p-k}$. Actually one loses a great deal of free information by choosing this method for storing back information. Let us first state two other means for storing back information, and then discuss the advantages of using these methods.

Backward Difference Table

$$\text{Store:} \quad x_p, \quad \nabla^{i-1} f_p \qquad (4.4)$$

(Used in DVDQ[6]; the notation will be explained below.)

Scaled Derivatives

$$\text{Store:} \quad z_p=[x_p, h\dot{x}_p, \ . \ . \ . \ , (h^q/_{q!})x_p^{(q)}]^T \qquad (4.5)$$

(Used by Gear[1] and popularized by Nordsieck.[16])

It can be shown that well-defined linear transformations connect all three means of representation.[1] We shall see below, by means of an example, how one would construct such a linear transformation.

The Adams predictor-corrector formulas written in backward difference form are

$$x_{p+1}^{(P)}=x_p+ h\sum_{j=0}^{k-1} q_j \nabla^j f_p \qquad (k^{\underline{th}} \text{ order}) \qquad (4.6)$$

$$x_{p+1}^{(C)}=x_p+ h\sum_{j=0}^{k-1} q_j^* \nabla_P{}^j f_{p+1}, \qquad (k+1\text{-order}) \qquad (4.7)$$

where the different order predictor and corrector formulas are employed since these are precisely the formulas used in DVDQ. The notation is as follows:

$$\nabla^o f_p=f_p \ , \ \nabla^{j+1} f_p= \nabla^j f_p- \nabla^j f_{p-1} \quad (j\geq 0) \qquad (4.8)$$

$$\nabla_P^o f_p=f[t_{p+1},x_{p+1}^{(P)}] \ , \ \nabla_P^{j+1} f_{p+1}= \nabla_P{}^j f_{p+1} - \nabla^j f_p (j\geq 0). \qquad (4.9)$$

Let us consider an example which clarifies the notation and demonstrates how one would construct a linear transformation between the various means of representation, e.g., between Eqs. (4.2), (4.3), (4.6), and (4.7).

Example: Consider the second-order Adams-Bashforth formula, i.e.,

$$x_{p+1}^{(P)} - x_p = (h/2)(3f_p-f_{p-1}). \qquad (4.10)$$

With respect to Eq. (4.6), k=2, so in backward difference form Eq. (4.10) can be written as

$$x_{p+1}^{(P)} = x_p + h \sum_{j=0}^{1} q_j \nabla^j f_p \qquad (4.11)$$

or

$$x_{p+1}^{(P)} = x_p + h\,[q_o\ \nabla^o f_p + q_1\ \nabla^1 f_p]. \qquad (4.12)$$

We shall now employ the notation definitions of Eq. (4.8) to write Eq. (4.12) in terms of function evaluations only, and then determine q_o and q_1 so that Eq. (4.11) matches Eq. (4.10). By definition:

$$\nabla^o f_p = f_p \ , \quad \nabla^1 f_p = \nabla^o f_p - \nabla^o f_{p-1} = f_p - f_{p-1}.$$

Upon substitution into Eq. (4.12), we have

$$x_{p+1}^{(P)} = x_p + h[q_o f_p + q_1(f_p - f_{p-1})]$$

or

$$x_{p+1}^{(P)} = x_p + h[(q_o + q_1)f_p - q_1 f_{p-1}]. \qquad (4.13)$$

Comparison of Eqs. (4.10) and (4.13) implies

$$q_o + q_1 = 3/2 \quad , \quad q_1 = 1/2 \ , \qquad (4.14)$$

which is the desired linear transformation between the coefficients of Eqs. (4.2) and (4.6) for the second-order case.

Now that we have seen that the backward difference representation of the predictor-corrector formulas is no more difficult than the standard means of representation, the question arises as to what we gain out of such a representation. The answer is given by noting the following equations:

$$\begin{aligned}
&\nabla^o f_p = f_p = \dot{x}_p \\
&\nabla^1 f_p = f_p - f_{p-1} = h[(f_{p-1} - f_p)/(-h)] = h[(\dot{x}_{p-1} - \dot{x}_p)/(-h)] \approx h\ddot{x}_p \\
&\nabla^2 f_p \approx (h^2/_2)\dddot{x}_p \\
&\quad \vdots \qquad\qquad \vdots \\
&\nabla^n f_p \approx (h^n/_{n!})x_p^{(n+1)} \qquad (4.15)
\end{aligned}$$

Equation (4.15) has great significance since the major portion of the local truncation error can be represented by (see Gear, p. 111)

$$C_{k+1}h^{k+1}x_p^{(k+1)} + O(h^{k+2}), \qquad (4.16)$$

i.e., the higher-order differences indicate the local truncation errors for various order integrators since they are proportional to the higher-order derivatives of x at t_p. Thus, by employing this means of representation, or the scaled derivative representation, one has the means for not only adjusting the stepsize but also for choosing the most efficient order integration formulas.

4.2 Variable Order, Variable Stepsize Procedures

In the previous section we saw that by storing back information in either backward difference or scaled derivative form, we have the means for estimating the local truncation error at various orders with no extra computation. In this section we shall present a method due to Gear[1] for adjusting the stepsize and order automatically.

Suppose that the integration has proceeded long enough to generate a sufficiently long "tail" of scaled derivatives or backward differences. Later we shall discuss problems associated with starting the algorithms. Suppose we are currently at t_p with $k^{\underline{th}}$ order integration formulas, stepsize h, and scalar truncation error parameter E. Use the $k^{\underline{th}}$ order formulas and stepsize h to compute $x(t_{p+1})$, where $t_{p+1}=t_p+h$. From our "tail" of backward differences or scaled derivatives we can easily compute the truncation error associated with the $k^{\underline{th}}$ order formula, and make the following check:

$$C_{k+1} h^{k+1} x_{p+1}^{(k+1)} \overset{?}{\leq} E. \tag{4.17}$$

If Inequality (4.17) is satisfied, we accept the computed value for $x(t_{p+1})$; if the inequality is not satisfied, we must determine a smaller stepsize h to compute a value $x(t_{p+1})$ which satisfies (4.17). In either case we proceed to the tests below.

We now wish to either possibly increase the stepsize if (4.17) was satisfied or decrease the stepsize if (4.17) was not satisfied. In addition we wish to check to see if the order should be increased or decreased. Define:

$$\text{New stepsize} = zh, \ z \text{ scalar unknown.} \tag{4.18}$$

If all quantities were known exactly and the k+1-derivative of x_{p+1} remained constant, then the optimal stepsize would be defined by (4.17) with the equality, i.e.,

$$C_{k+1} (zh)^{k+1} x_{p+1}^{(k+1)} = E. \tag{4.19}$$

Then, solving for z we obtain

$$z = \left\{ E/[C_{k+1} h^{k+1} x_{p+1}^{(k+1)}] \right\}^{\frac{1}{k+1}} . \tag{4.20}$$

However, the quantities are not known exactly and $x^{(k+1)}$ does not, in general, remain constant so a safety factor is included in Eq. (4.20), e.g.,

$$z_k = \frac{1}{1.2} \left\{ E/[C_{k+1} h^{k+1} x_{p+1}^{(k+1)}] \right\}^{\frac{1}{k+1}} , \tag{4.21}$$

where the k subscript is attached to z to imply that this is the stepsize multiplier associated with the $k^{\underline{th}}$ order integration formula. Since we also have backward differences which imply $x_{p+1}^{(k)}$ and $x_{p+1}^{(k+2)}$, we can determine the optimal stepsizes at the neighboring orders, i.e., at the k-1 and k+1 orders. Operating in exactly the same way as we did in forming Eq. (4.21), we obtain (with appropriate safety factors)

$$z_{k-1} = \frac{1}{1.3} \left\{ E/[C_k h^k x_{p+1}^{(k)}] \right\}^{\frac{1}{k}} \qquad \text{(Order k-1)} \quad (4.22)$$

$$z_{k+1} = \frac{1}{1.4} \left\{ E/[C_{k+2} h^{k+2} x_{p+1}^{(k+2)}] \right\}^{\frac{1}{k+2}} . \qquad \text{(Order k+1)} (4.23)$$

The safety factors 1.2, 1.3, and 1.4 are chosen so that we have the following order of priority: no order change, lower the order, and increase the order. The order which produces the largest value of z in Eqs. (4.21), (4.22), (4.23) is selected as the integration formula order for the next integration.

With the procedure above, an arbitrary value for h may result, and thus to employ the current "tail" of information, interpolation must be used to shift the information to the new stepsize. As shown by Gear[1], this may be done by means of a matrix multiplication. Also, because of the overhead associated with implementing the tests and interpolation, the above procedure is not performed on each step in Gear's program; see page 157 of Ref. 1 for a discussion of how the procedure is implemented.

The same ideas form the basis for changing order and stepsize in DVDQ. However, DVDQ only allows halving and doubling of the stepsize, which Krogh claims is more efficient when the order is to be varied. Also, DVDQ surveys the local truncation error at four orders to build more stability into the order changing procedure.

4.3 Self-Starting Adams Methods

To conclude this chapter, we shall present an example to demonstrate how predictor-corrector methods may be used without employing alternate integration schemes to get them started, e.g., many existing fourth-order predictor-corrector subroutines employ a fourth-order Runge-Kutta scheme to generate the required starting values. Before we consider the example, let us consider an alternate method for estimating the major portion of the local truncation error in a predictor-corrector scheme. Suppose the predictor and corrector are of the same order. Then, denoting the true solution of the differential equation at t_p by x_p^T, it can be shown that (Ref. 1, p. 111):

$$x_p^{(P)} = x_p^T + C_k h^{k+1} x_p^{(k+1)} + O(h^{k+2}) \qquad (4.24)$$

$$x_p^{(C)} = x_p^T + \bar{C}_k h^{k+1} x_p^{(k+1)} + O(h^{k+2}) \qquad (4.25)$$

Upon elimination of x_p^T between these two equations, we obtain

$$x_p^{(P)} - x_p^{(C)} = h^{k+1} x_p^{(k+1)} (C_k - \bar{C}_k) + O(h^{k+2}) \;, \qquad (4.26)$$

where everything is known to order k+1 except $x_p^{(k+1)}$. Thus, solving for $x_p^{(k+1)}$:

$$x_p^{(k+1)} \approx [x_p^{(P)} - x_p^{(C)}]/[h^{k+1}(C_k - \bar{C}_k)]. \qquad (4.27)$$

Then, given an estimate of $x_p^{(k+1)}$, we can estimate the difference between the true and corrected solutions from Eq. (4.25), i.e.,

$$|x^{(C)} - x^T| \approx |\bar{C}_k h^{k+1} x_p^{(k+1)}| \;, \qquad (4.28)$$

where $x_p^{(k+1)}$ is estimated from Eq. (4.27). Upper and lower limits, E_u and E_L, respectively, are usually specified to indicate when the stepsize should be halved or doubled, e.g.,

$$|x^{(C)}-x^T| \begin{cases} > E_u \Rightarrow \text{Halve stepsize} & (4.29) \\ , & \\ < E_L \Rightarrow \text{Double stepsize} & (4.30) \end{cases}$$

Let us now use this method in an example which demonstrates a self-starting Adams procedure.

Example: Given $\dot{x}=f(t,x)$, $t_o, x(t_o)=x_o$, and error constants E_L, E_u, and E. Also, an initial stepsize h_o must be specified and this value should be smaller than one would expect to result with a fourth-order integration scheme. For sake of illustration, assume that only one correction is employed after each predictor step. Consider the first-order Adams predictor and corrector formulas. Then,

$$x_1^{(P)} = x_o+h_of_o \tag{4.31}$$

$$x_1^{(C)} = x_o+h_of[t_1,x_1^{(P)}]. \tag{4.32}$$

We now check to see if the stepsize should be changed by using Eq. (4.27) to estimate $\ddot{x}_1$, and then checking

$$E_L \leq |x_1^{(C)} - x_1^T| \leq E_u$$

or,

$$E_L \leq |\tfrac{1}{2}(x_1^{(P)} - x_1^{(C)})| \leq E_u. \tag{4.33}$$

After making this test, we either halve, double, or keep the same stepsize. Let h be the resultant stepsize. We then form the first components of the backward difference table

$$[x_1^{(C)},\ \nabla^o f_1 = f_1,\ \nabla^1 f_1 = f_1 - f_o]. \tag{4.34}$$

We now have enough information to use the second-order Adams formulas. Again using a single correction we form:

$$x_2^{(P)} = x_1 + (h/2)(3f_1 - f_o) \tag{4.35}$$

$$x_2^{(C)} = x_1 + (h/2)[f_2(t_2, x_2^{(P)}) + f_1]. \tag{4.36}$$

Of course if we are storing backward differences, we would use the backward difference version of the predictor-corrector equations in the actual calculation (i.e., Eqs. (4.6), (4.7)).

Given $x_2^{(C)}$, we again employ Eq. (4.27) to estimate $\dddot{x}_2$, and check to see if the stepsize should be changed by

$$E_L \leq | x_2^{(C)} - x_2^T | \leq E_u. \tag{4.37}$$

After this test, we either halve, double, or keep the same stepsize, and then augment the difference table to form

$$[x_2^{(C)},\ \nabla^o f_2,\ \nabla^1 f_2,\ \nabla^2 f_2]. \tag{4.38}$$

One then proceeds in the same manner to the third-order formula, and eventually the difference table will possess sufficient information to allow a switch over to the automatic order and stepsize changing procedure described in the previous section. Of course, if one desires a fixed order predictor-corrector method, e.g., fourth-order, the above starting procedures would be used up to fourth-order and then sufficient information will exist for the use of the fourth-order formulas.

In Appendix B a listing of DVDQ is presented along with a sample program.

5. RESULTS WITH NASA-JSC COMPUTER PROGRAMS

In this section further results involving the NASA-JSC PEACE parameter optimization program will be discussed. Since most of the work in this area was done in the first portion of the contract (as opposed to the extension), and is thoroughly documented in Ref. 18, we shall only discuss recent results.

5.1 Parameter Optimization

The main modification to the PEACE program involved the addition of the Fletcher method with a one-dimensional search. The method was simulated on the stage-and-half configuration used for the simulations in Ref. 18. It was found that the Fletcher method with a one-dimensional search did not perform as well as either the DFP or Broyden algorithms (see Table 5.1). It appears that the terminal convergence properties are poor because the H-matrix probably becomes "contaminated" by the switching of the formulas for the H-matrix. Note that, from Table 5.1, the rate of convergence of the Fletcher method slowed considerably after the first time that the formulas were switched. That is, the test within Fletcher's method implied that the DFP formula should be used until the sixth iterate when the Broyden formula was used. Thereafter the convergence was very slow. Recommendations for use of the schemes will be given in Chapter 6.

Iteration	COST (All with 1-D Search)		
Number	DFP	Broyden	Fletcher
0	1611.	1611.	1611.
1	271.1	271.1	271.1
2	13.75	13.75	13.75
3	2.792	2.792	2.792
4	1.311	1.311	1.311
5	.8897	.8897	.8897
6	.2636	.2636	.2636*
7	.1621	.1621	.2462
8	.1563	.1563	.1605
9	.1554	.1554	.1580*
10	.1461	.1461	.1580
12	.1183	.1183	.1572
14	.1046	.1045	.1548
16	.0972	.0967	.1472
18	.0923	.0921	.1470
20	.0888	.0886	.1413

*(In Fletcher column, * indicates Broyden formula used; otherwise DFP formula used. The Broyden formula was also used on the 15th and 19th iterates.)

TABLE 5.1 Comparison of DFP, Broyden, and Fletcher (all with 1-D search) on Stage-And-Half Configuration.

5.2 Numerical Integration

The extrapolation numerical integration scheme of Appendix A has been built into the stage-and-half configuration version of the PEACE parameter optimization program. Since the program had its former integration scheme (fourth-order Runge-Kutta) built into the program (as opposed to being subroutined), the extrapolation scheme will probably not perform as efficiently as possible. However, the resultant integrations should give some indication of its efficiency in a problem with a considerable number of discontinuities, and it should give some indication as to the accuracy of the Runge-Kutta integrations previously obtained. The computer program will be tested by NASA-JSC personnel.

6. SUMMARY AND RECOMMENDATIONS

6.1 Summary

Two general purpose numerical integration computer programs have been delivered (and checked out) to NASA-JSC personnel. These are the Bulirsch-Stoer extrapolation scheme (1972 version) and Krogh's variable-order, variable-stepsize Adams method. User's guides and listings of the programs are presented in the Appendices.

The PEACE parameter optimization program was modified to include Fletcher's method with a one-dimensional search and the extrapolation integrator. In addition, W. F. Powers presented seven lectures on optimal control and numerical integration to Mission Planning and Analysis personnel in August 1973.

6.2 Recommendations

1.) With respect to parameter optimization, the Broyden and DFP algorithms are recommended if good terminal convergence is desired. In this respect, Broyden's method has always performed better than DFP, but not appreciably better. Fletcher's method (without a 1-D search) appears to work well in the early stages, and especially on problems where the H-matrix in the DFP method is having trouble. However, because of the H-matrix switching it appears to have trouble obtaining rapid terminal convergence.

2.) The Broyden method appears to do naturally and continuously what the Fletcher method does roughly, i.e., take tendency to singularity of the H-matrix into account. From a theoretical point of view, the Broyden method appears to be the most attractive.

3.) With respect to numerical integration, the DVDQ scheme has the best set of diagnostics and it is strongly recommended for use on new problems. The resultant output should indicate to the user a natural partitioning of the problem since the changing stepsize and order usually indicate a change in physical characteristics of the solution, also. One may then use this information to define fixed-stepsize, fixed-order formulas in the various phases of the physical problem if a large number of production runs are anticipated. That is, there is no use paying the overhead of DVDQ if all of the runs will be similar with respect to physical characteristics.

4.) The extrapolation numerical integration subroutine has less diagnostic capability than DVDQ, but appears to be faster on runs where the right-hand side of the differential equation is not unduly complicated. This scheme is relatively new and should be continuously improved by research in the next decade. It may be more optimal for MPAD problems than DVDQ because it is a one-step method which should not be as adversely affected by multi-stage problems as DVDQ.

5.) It is not claimed that the numerical integration routines listed in the Appendices will speed up existing programs, which usually employ a fixed stepsize that changes from one well-defined physical phase of the problem to another. However, they should speed up the process of choosing optimal stepsizes and determining accurate reference cases which will aid in the development of production programs. In this respect, the possibility of including both routines in SVDS should be considered.

6.) Rough rules-of-thumb for choosing a numerical integration scheme are the following:

(i) if low accuracy is all that is required (e.g., three significant digits) and/or there exists a large number of physical phases (which require starting and stopping the integrator), then a low-order Runge-Kutta scheme (e.g., order one to four) with sufficiently small fixed stepsize is probably optimal;

(ii) if high accuracy is required (e.g., five or more significant digits), then either DVDQ or extrapolation should be employed. The relative efficiency of these schemes increases with the order of accuracy required. To date, comparative studies[7-12] have shown the two schemes to have the following properties: DVDQ has higher overhead, but smaller number of function evaluations than extrapolation, and for many problems, roughly the same computer time. In Ref. 9 a rule for choosing between the two is proposed: if the right-hand side of the differential equation is relatively simple, extrapolation will probably give the least CPU time, whereas DVDQ will give the least if the right-hand side is lengthy (e.g., if gravitational anomalies are taken into account).

7. REFERENCES

1. Gear, C.W., Numerical Initial Value Problems in Ordinary Differential Equations, Prentice-Hall, Englewood Cliffs, N.J., 1971.

2. Lapidus, L., and Seinfeld, J.H., Numerical Solution of Ordinary Differential Equations, Academic Press, New York, 1971.

3. Bulirsch, R., and Stoer, J., "Numerical Treatment of Ordinary Differential Equations by Extrapolation Methods," Numer. Math., Vol. 8, No. 1, pp. 1-13, 1966.

4. Fehlberg, E., "Classical Fifth-, Sixth-, Seventh-, and Eighth-Order Runge-Kutta formulas with Stepsize Control," NASA TR R-287, October 1968.

5. Sarafyan, D., "Estimation of Errors for the Approximate Solution of Differential Equations and Their Systems," Louisiana State University Technical Report No. 15, August 1966.

6. Krogh, F.T., "VODQ/SVDQ/DVDQ-Variable Order Integrators for the Numerical Solution of Ordinary Differential Equations," TU Doc. No. CP-2308, NPO-11643, Jet Propulsion Laboratory, May 1969.

7. Clark, N.W., "A Study of Some Numerical Methods for the Integration of Systems of First-Order Ordinary Differential Equations," Report No. ANL-7428, Argonne National Laboratory, March 1968.

8. Fox, P., "A Comparative Study of Computer Programs for Integrating Differential Equations," Numerical Mathematics, Vol. 15, No. 11, pp. 941-948, 1972.

9. Hull, T.E., et. al., "Comparing Numerical Methods for Ordinary Differential Equations," SIAM J. on Numerical Analysis, Vol. 9, No. 4, pp. 603-637, 1972.

10. Frazho, D.B., Powers, W.F., and Canale, R.P., "Numerical Integration Aspects of a Nutrient Utilization Ecological Problem," to appear in Proceedings of SIAM Conference on Numerical Solution of Differential Equations, 1973. (Also, available as University of Michigan, Dept. of Aerospace Engineering Report AC-101, February 1973).

11. Dickmann, E.D., "Comparison of the Performance of Two Integration Routines," NASA-MSFC internal memorandum (no number), 1973.

12. Pesapane, J.P., and Powers, W.F., "Evaluation of the Performance of Numerical Integration Schemes on a Reentry Problem," Presented at 1972 SIAM Conference on Numerical Solution of Differential Equations, Austin, Texas.

13. Krogh, F.T., Personal Communication, Jet Propulsion Laboratory, October 1972.

14. Richardson, L.F., "The Deferred Approach to the Limit, I-Single Lattice," Trans. Roy. Soc., London, Vol. 226, pp. 299-349, 1927.

15. Gragg, W.B., "Repeated Extrapolation to the Limit in the Numerical Solution of Ordinary Differential Equations," Doctoral Dissertation, UCLA, 1963.

16. Nordsieck, A., "On Numerical Integration of Ordinary Differential Equations," Math. Comp., Vol. 16, No. 1, pp. 22-49, 1962.

17. Aitken, A.C., "On Interpolation by Iteration of Proportional Parts," Proc. Edinburgh Math. Soc., Vol. 2, pp. 56-76, 1932. (Also, discussed in Hildebrand, F.B., Introduction to Numerical Analysis, McGraw-Hill, New York, p. 49, 1956.)

18. Powers, W.F., "Near-Earth Orbital Guidance and Remote Sensing," Modern Systems Analysis Report 9-12319-F, December 1972. (Final report for NASA-MSC Contract 9-12319.)

APPENDIX A
DIFSYS
USER'S GUIDE AND LISTING

SUBROUTINE DIFSYS is a double precision rational function extrapolation numerical integration scheme. It is an improved version of the ALGOL subroutine reported in Ref. 3. The original subroutine is also documented in FORTRAN in Ref. 1 (pp 96-99). The version presented in this appendix was supplied by its developer, R. Bulirsch. Some user's may prefer to use the original version (presented in Ref. 1) because it contains more comment cards and error checks. However, to date, we have not encountered any difficulties with the faster, more streamlined version presented in this appendix.

DESCRIPTION

DIFSYS is called by:

CALL DIFSYS (N, YF, EPS, H, X, Y)

where:

N=order of system (number of differential equations)

YF=a user supplied subroutine which calculates the derivatives, and has the form

SUBROUTINE YF(X,Y,DY)

X = independent variable

Y = dependent variable vector (must be dimensioned in YF)

DY = derivatives (right-hand side of the differential equation vector; must be dimensioned in YF)

(Note: YF must be declared in an EXTERNAL statement in the program which calls DIFSYS.)

EPS = stepsize error control (DIFSYS will reset EPS to 10^{-11} if the user supplies a smaller number)

H = maximum integration interval

X = independent variable

Y = dependent variable vector (must be dimensioned N in calling program)

The quantities N, EPS, H, X, and Y(N) must be supplied before DIFSYS is called.

OPERATION

This subroutine does only one integration step per call. Hence, if the interval of integration is $[x_o, x_f]$, where x_f may be defined implicitly, the user must test for x_f (or the implicit condition) and adjust H at the end so that x_f is satisfied exactly.

EXAMPLE PROGRAM

Consider the integration of:

$$\ddot{x} = -\dot{x}^2 + tx$$

$t_o, t_f, x(t_o), \dot{x}(t_o)$ specified.

Define $y_1 = x$, $y_2 = \dot{x}$. Then, the following program will execute the integration with DIFSYS.

```
      IMPLICIT REAL*8(A-H,O-Z)
      EXTERNAL YF
      DIMENSION Y(2)
      READ (5,100)EPS,H,TO,TF,(Y(I),I=1,2)
      N=2
      T=TO
1     CALL DIFSYS (N,YF,EPS,H,T,Y)
      WRITE(6,101)T,(Y(I),I=1,2)
      IF(T.GE.TF)STOP
      IF((T+H).GT.TF)H=TF-T
      GO TO 1
100   FORMAT(6D13.6)
101   FORMAT('TIME=',D20.8,'Y(I)=',2D20.8)
      END
      SUBROUTINE YF(T,Y,DY)
      IMPLICIT REAL*8(A-H,O-Z)
      DIMENSION Y(2),DY(2)
      DY(1)=Y(2)
      DY(2)=-Y(2)**2+Y(1)*T
      RETURN
      END
      SUBROUTINE DIFSYS(N,YF,EPS,H,X,Y)
      REAL*8 Y(N),YA(10),YL(10),YM(10),DY(9),DZ(10),DT(10,7),
     1 D(7),S(10),X,XN,H,G,B,B1,U,V,C,TA,W
      REAL*4 EP(4)/0.4E-1,0.16E-2,0.64E-4,0.256E-5/
      LOGICAL*1 KONV,BO,KL,GR
      JTI=0
      FY=1.
      ETA=ABS(EPS)
      IF(ETA.LT.1.E-11)  ETA=1.E-11
      DO 100 I=1,N
  100 YA(I)=Y(I)
      CALL YF (X,Y,DZ)
   10 XN=X+H
      BO=.FALSE.
      DO 110 I=1,N
  110 S(I)=0.D0
      M=1
      JR=2
      JS=3
      DO 260 J=1,10
      IF(.NOT.BO) GOTO 200
      D(2)=1.7777777777777778D0
      D(4)=7.1111111111111111D0
      D(6)=2.8444444444444444D1
      GO TO 201
  200 D(2)=2.25D0
      D(4)=9.D0
      D(6)=3.6D1
  201 IF(J.LE.7) GOTO 202
      L=7
      D(7)=6.4D1
      GO TO 203
  202 L=J
      D(L)=M*M
  203 KONV=L.GT.3
      M=M+M
      G=H/M
      B=G+G
      DO 210 I=1,N
      YL(I)=YA(I)
  210 YM(I)=YA(I)+G*DZ(I)
      M=M-1
      DO 220 K=1,M
      CALL YF(X+K*G,YM,DY)
      DO 220 I=1,N
      U=YL(I)+B*DY(I)
      YL(I)=YM(I)
      YM(I)=U
      U=DABS(U)
      IF(U.GT.S(I)) S(I)=U
  220 CONTINUE
      CALL YF (XN,YM,DY)
      KL=L.LT.2
      GR=L.GT.5
      FS=0.
      DO 233 I=1,N
      V=DT(I,1)
      C=(YM(I)+YL(I)+G*DY(I))*0.5D0
      DT(I,1)=C
      TA=C
      IF(KL) GO TO 233
      DO 231 K=2,L
      B1=D(K)*V
      B=B1-C
      W=C-V
      U=V
      IF(B.EQ.0.D0) GO TO 230
      B=W/B
      U=C*B
      C=B1*B
  230 V=DT(I,K)
      DT(I,K)=U
  231 TA=U+TA
      IF(.NOT.KONV) GO TO 232
      IF(DABS(Y(I)-TA).GT.S(I)*ETA) KONV=.FALSE.
  232 IF(GR.OR.S(I).EQ.0.D0) GO TO 233
      FV=DABS(W)/S(I)
      IF(FS.LT.FV) FS=FV
  233 Y(I)=TA
      IF(FS.EQ.0.D0) GO TO 250
      FA=FY
      K=L-1
      FY=(EP(K)/FS)**(1./(L+K))
      IF(L.EQ.2) GO TO 240
      IF(FY.LT.0.7*FA) GO TO 250
  240 IF(FY.GT.0.7) GO TO 250
      H=H*FY
      JTI=JTI+1
      IF(JTI.GT.5) GO TO 30
      GO TO 10
  250 IF(KONV) GO TO 20
      D(3)=4.D0
      D(5)=1.6D1
      BO=.NOT.BO
      M=JR
      JR=JS
  260 JS=M+M
      H=H*0.5D0
      GO TO 10
   20 X=XN
      H=H*FY
      RETURN
   30 H=0.D0
      DO 300 I=1,N
  300 Y(I)=YA(I)
      RETURN
      END
```

APPENDIX B

DVDQ

USER'S GUIDE AND LISTING

SUBROUTINE DVDQ is a double precision variable order, variable stepsize Adams predictor-corrector numerical integration scheme developed by F. T. Krogh of the Jet Propulsion Laboratory. As one may see by inspecting the listing, the program is extremely well-documented and over half of the listing is devoted to comment cards. Before we discuss the implementation of the deck, some notable features of the program are listed below:

(i) Maximum integration formula order = 16.

(ii) Halves and doubles for variable stepsize.

(iii) Only MAIN and DVDQ are necessary, i.e., one need not employ a separate subroutine for the right-hand sides of the differential equations.

(iv) Order of the predictor = order of the corrector -1. (Recall the general property that

$$x_p^T = x_p^{(P)} + O(h^{k+1})$$

$$x_p^T = x_p^{(C),m} + O(h^{k+m+1}) + O(h^{r+1})$$

where k=order of predictor, r = order of the corrector, m = number of applications of the corrector. This implies that with k = r or r-1 the same order of accuracy is obtained for a single corrector application.)

(v) Has "GSTOP" feature, i.e., if the trajectory is to be terminated by a condition of the form $G(t_f,x_f)=0$, the program can handle it automatically.

Description

DVDQ is called by:

```
     CALL DVDQ(NEQ, T, Y, F, KD, EP, IFLAG, H,
     HMINA, HMAXA, DELT, TFINAL, MXSTEP, KSTEP,
     KEMAX, EMAX, KQ, YN, DT)
```

Not all of the arguments must be supplied by the user. The arguments which must be supplied by the user will be listed and briefly discussed below. Further explanation and definitions of the remaining terms may be found in the general comment section in the initial portion of the listing.

NEQ = number of differential equations

T = independent variable

Y = dependent variable vector

F = right-hand side of the vector differential equation

KD = order of the differential equation (not to be confused with the order of the integration scheme). For example, if $\ddot{x} = f(t,x,\dot{x})$ is to be integrated directly, KD = 2 ; usually KD = 1 since $\dot{x} = f(t,x)$ is the usual trajectory analysis system.

EP = local truncation error indicator. It is an absolute error indicator in the sense that the local error is kept less than EP in all components of the differential equation. If it is desired to control the error on each component of the vector differential equation separately, EP should be specified as a vector of negative numbers. The negative values alert the routine to a vector error specification. See the discussion of EP in the comment cards for further options.

H = initial stepsize estimate. (Probably better to guess smaller than one would expect with a fourth-order scheme since the routine builds up from a first-order formula. However, the choice is not critical since the stepsize is adjusted rapidly.)

HMINA = minimum allowable stepsize

HMAXA = maximum allowable stepsize

DELT = output interval. Since the scheme only halves and doubles, use DELT = 2^kH if possible; otherwise, it will interpolate for output values.

TFINAL = final value of the independent variable.

MXSTEP = maximum number of steps allowable between output points.

In addition DT(17,NEQ), YN( ), Y( ), F( ), KQ( ) must appear in a DIMENSION statement in MAIN. See the example below or the comment portion of the listing for the particular dimensions.

Operation

Only one call is made to DVDQ; thereafter the simple statement CALL DVDQ1 is used. The heart of operation of DVDQ is a computed GO TO statement which is driven by the parameter IFLAG, which has values 1, 2, . . ., 8. The full implications of each value of IFLAG are discussed in the initial comment section of the listing. Although the operation of DVDQ may appear complicated at first glance (because of the eight values for IFLAG), the operation is straightforward with excellent error detection capabilities. Probably the easiest way to get acquainted with DVDQ is to study the simple program given below. Informal comments are given to explain roughly what each value of IFLAG is indicating. The listing follows the example.

Example: Integrate $\dot{x}_1=x_2$

$$\dot{x}_2=-x_2^2+tx_1$$

with $t_0=0, t_f=2$, $x_1(0)=1.0$, $x_2(0)=0.0$. The following program will execute the integration with DVDQ.

```
          IMPLICIT REAL*8 (A-H,O-Z)
     C    SINCE SYSTEM IS FIRST_ORDER WITH TWO DIFFERENTIAL EQUATIONS,
     C    Y,F,KQ,YN ARE OF DIMENSION 2. DT IS ALWAYS DIMENSIONED DT(17,NEQ).
          DIMENSION Y(2),F(2),DT(17,2),KQ(2),YN(2)
          NEQ=2
          T=0.D0
          Y(1)=1.D0
          Y(2)=0.D0
          KD=1
          EP=1.D-5
          H=1.D-1
          HMINA=1.D-4
          HMAXA=1.D0
          DELT=2.D-1
          TFINAL=2.D0
          MXSTEP=1000
          CALL DVDQ(NEQ,T,Y,F,KD,EP,IFLAG,H,HMINA,HMAXA,DELT,TFINAL,
         1MXSTEP,KSTEP,KEMAX,EMAX,KQ,YN,DT)
          GO TO 10
     20   CALL DVDQ1
     10   GO TO (1,1,3,4,5,6,7,8),IFLAG
     1    F(1)=Y(2)
          F(2)=-Y(2)**2+Y(1)*T
          GO TO 20
     3     WRITE(6,100) T,(Y(I),I=1,2)
          GO TO 20
     4     WRITE(6,100) T,(Y(I),I=1,2)
          STOP 0
     5    STOP 5
     6    STOP 6
     7    HMINA=HMINA/2.D0
          GO TO 20
     8    STOP 8
     100  FORMAT('TIME= ',D23.15/'Y(I)= ',2D23.15)
          END
```

IFLAG = 1 or 2; GO TO DERIVATIVE EVALUATION & THEN CALL DVDQ1. (1=PREDICTOR, 2=CORRECTOR)

IFLAG=3 ⇒ OUTPUT POINT; AFTER WRITING, CALL DVDQ1.

IFLAG=4 ⇒ $t=t_f$, END OF INTEGRATION PHASE; CHANGE t_f OR STOP.

IFLAG=5 ⇒ MXSTEP ATTAINED; STOP OR INCREASE MXSTEP.

IFLAG=6 ⇒ EP CANNOT BE ATTAINED (PROBABLY DUE TO ROUNDOFF); STOP OR DECREASE EP.

IFLAG=7 ⇒ H < HMINA REQUIRED; STOP OR RESET HMINA OR EP.

IFLAG=8 ⇒ ILLEGAL PARAMETER IN CALLING SEQUENCE ⇒ STOP.

```
IISTICS*    SOURCE STATEMENTS =         32,PROGRAM SIZE =       1400
'ISTICS*  NO DIAGNOSTICS GENERATED
' IN MAIN
```

```
      SUBROUTINE  DVDQ(/NEQ/,/T/,/Y/,/F/,/KD/,/EP/,/IFLAG/,/H/,/HMINA/,
     * /HMAXA/,/DELT/,/TFINAL/,/MXSTEP/,/KSTEP/,/KEMAX/,/EMAX/,
     * /KQ/,/YN/,/DT/)

      DOUBLE PRECISION VARIABLE ORDER INTEGRATION SUBROUTINE
      FOR THE SOLUTION OF ORDINARY DIFFERENTIAL EQUATIONS.

      ANALYSIS AND CODING BY FRED T. KROGH,  AT THE JET PROPULSION
      LABORATORY, PASADENA, CALIF.  APRIL 1, 1969.
      THIS DECK IN EBCDIC FORMAT.

      CONVERTED FOR USE ON 360/75 BY MELBA W. NEAD, JPL  APRIL, 1970.

      AT THE END OF THIS LISTING INSTRUCTIONS ARE GIVEN FOR REMOVING
      SOME FEATURES AND FOR ADDING OTHERS. THE GSTOP FEATURE IS
      EXPLAINED NEAR THE END OF THE LISTING.

      VARIABLES IN THE CALLING SEQUENCE HAVE THE FOLLOWING TYPES.
      INTEGER NEQ,KD(1),IFLAG,MXSTEP,KSTEP,KEMAX,KQ(1)
      REAL    EP(1),HMINA,HMAXA,EMAX
      DOUBLE PRECISION T,Y(1),F(1),H,DELT,TFINAL,YN(1),DT(17,1)

      PARAMETERS WHICH MUST BE ASSIGNED VALUES BEFORE CALLING
      DVDQ ARE    NEQ, T, Y, KD, H, HMINA, HMAXA, DELT,
                  TFINAL, AND MXSTEP.
      DVDQ IS USED ONLY ON THE INITIAL ENTRY. ALL OTHER
      ENTRIES ARE MADE BY CALLING DVDQ1. IN ADDITION TO
      THE PARAMETERS MENTIONED ABOVE THE USER MUST ASSIGN
      VALUES TO F (ONCE PER STEP INITIALLY, AND TWICE PER STEP
      AFTER GETTING STARTED) AND EP (EITHER INITIALLY, OR DURING
      THE INTEGRATION IF A RELATIVE ERROR TEST IS USED).
      THE FOLLOWING PARAMETERS GIVE ADDITIONAL INFORMATION ABOUT THE
      INTEGRATION AND ARE USED FOR STORAGE. THEY SHOULD NOT BE
      CHANGED BY THE USER.  IFLAG,KSTEP,KEMAX,EMAX,KQ,YN, AND DT.


C     AN EXAMPLE OF HOW ONE MIGHT SET UP THE CALLS TO DVDQ IS GIVEN
C     BELOW.

      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION F(2),DT(17,2),Y(4),YN(4),KQ(2)

C     SET PARAMETERS AND INITIAL CONDITIONS
      NEQ = 2
      KD = 2
      MXSTEP = 500
      EP = 1.D-6
      HMINA = 0.
      HMAXA = 100.
      T = 0.
      Y(1) = 1.
      Y(2) = 0.
      Y(3) = 0.
      Y(4) = 1.
      H = 1.
      DELT = 1.
      TFINAL = 12.
      NEVALS = 0
```

```
CC     NOW MAKE FIRST ENTRY
C      CALL DVDQ(NEQ,T,Y,F,KD,EP,IFLAG,H,HMINA,HMAXA,DELT,
C     1          TFINAL,MXSTEP,KSTEP,KEMAX,EMAX,KQ,YN,DT)
C      GO TO  2
C
CC     ALL SUBSEQUENT ENTRIES MADE HERE
C    1 CALL DVDQ1
C    2 GO TO (10,10,30,30,50,60,50,50), IFLAG
C
CC     EVALUATE DIFFERENTIAL EQUATION, IFLAG = 1 OR 2
C   10 R = Y(1)**2 + Y(3)**2
C      R = R*DSQRT(R)
C      F(1) = -Y(1)/R
C      F(2) = -Y(3)/R
C      NEVALS = NEVALS+1
C      GO TO  1
C
CC     OUTPUT, IFLAG = 3, OR FINISHED, IFLAG = 4
C   30 PRINT 13,T,Y(1),Y(3),KSTEP,NEVALS
C   13 FORMAT(' ',F6.2,1P2D20.12,2I5)
C      IF (IFLAG.EQ.3) GO TO  1
C      STOP
C
CC     ERROR CONDTION, IFLAG = 5, 7 OR 8
C   50 PRINT  12,IFLAG,H
C   12 FORMAT(' IFLAG,H=',I3,D20.10)
C      GO TO 30
C
CC     EP TOO SMALL, IFLAG = 6
C   60 EP = 32.*EMAX*EP
C      PRINT  14,T,EP
C   14 FORMAT(' T,NEW EP=',F6.2,D15.6)
C      GO TO  1
C      END
C
C
C.....................................................................
C
C
C      THE USAGE OF THE VARIABLES IS GIVEN BELOW.
C
C      NEQ=NUMBER OF EQUATIONS (INPUT)
C
C      T=INDEPENDENT VARIABLE (INITIAL VALUE SUPPLIED BY THE USER)
C
C      Y=CURRENT VALUE OF DEPENDENT VARIABLE. THE INITIAL
C        VALUE OF Y MUST BE SPECIFIED BY THE USER BEFORE
C        THE FIRST ENTRY. THE DIMENSION OF Y MUST BE
C        AT LEAST AS GREAT AS THE SUM OF THE ORDERS OF
C        THE DIFFERENTIAL EQUATIONS WHICH ARE BEING
C        INTEGRATED. IF WE LET KD(I) DENOTE THE ORDER
C        OF THE I-TH DIFFERENTIAL EQUATION, THEN Y(J)
C        IS THE K-TH DERIVATIVE OF THE L-TH CCMPONENT,
C        WHERE L IS THE SMALLEST INTEGER FOR WHICH
C        KD(1)+KD(2)+...+KD(L).GE.J  AND K=KD(L)+J-1-(KD(1)
C        +KD(2)+...+KD(L)), J=1,2,...,(KD(1)+KD(2)+...+KD(NEQ)).
C        (FOR EXAMPLE, FOR THE SYSTEM  F(1)=UPP, F(2)=VPP, WHERE P
C        DENOTES A PRIME, Y(1)=U, Y(2)=UP, Y(3)=V, Y(4)=VP.)
C
C      F(I)=KD(I)-TH DERIVATIVE OF THE I-TH COMPONENT WITH RESPECT
```

```
C          TO T, I=1,2,...,NEQ. THE USER MUST PROVIDE
C          THE CODE WHICH COMPUTES F GIVEN Y AND T.
C
C      KD GIVES THE ORDER OF THE DIFFERENTIAL EQUATIONS IN THE
C         SYSTEM. KD MUST BE LESS THAN OR EQUAL TO 4.
C         (FOR DIFFERENTIAL EQUATIONS WITH DIFFERENT ORDERS SET
C         KD.LT.0. IF THIS IS DONE IT IS ASSUMED THAT KD IS A VECTOR
C         AND THAT ABS(KD(I)) GIVES THE ORDER OF THE I-TH EQUATION.)
C
C      EP IS A PARAMETER USED TO CONTROL THE LOCAL ERROR.
C      IF EP IS POSITIVE THE LOCAL ERROR IS KEPT LESS
C      THAN EP IN ALL COMPONENTS OF THE DIFF. EQ.
C      (THE ESTIMATED LOCAL ERROR IS KEPT LESS THAN EP IN
C      THE (KD(I)-1)-ST DERIVATIVE OF THE I-TH COMPONENT. THUS
C      FOR EQUATIONS WITH ORDER GREATER THAN ONE, THE ERROR
C      IN A DERIVATIVE IS ESTIMATED. IN THIS CASE THE VALUE OF
C      EP REQUIRED TO OBTAIN A GIVEN ACCURACY IN THE DEPENDENT
C      VARIABLE DEPENDS ON THE SCALING.)
C      IF EP.LT.0, THEN IT IS ASSUMED THAT EP
C      IS A VECTOR. LET K BE THE SMALLEST VALUE
C      OF I FOR WHICH EP(I).GE.0.  FOR I.LT.K
C      THE LOCAL ERROR CONTROL IS BASED ON
C      ABS(EP(I)), AND FOR I.GE.K IT IS BASED ON
C      EP(K). IF ONE WANTS A RELATIVE ERROR TEST--
C      FOR EXAMPLE, THE LOCAL ERROR IS TO BE KEPT
C      LESS THAN C*P WHERE C IS A CONSTANT
C      AND P IS A POSITIVE FUNCTION OF T AND Y,
C      THEN ONE SHOULD SET EP=C*P WHEN IFLAG=1.
C      IF EP=0 AND HMAXA.NE.0, IFLAG IS SET EQUAL 8. IF EP=0 AND HMAXA=0,
C      NO ERROR TESTS ARE PERFORMED AND THE ORDER(S) AND STEPSIZE ARE
C      NOT CHANGED. THIS OPTION SHOULD NOT BE USED IF KQ(I)=1 FOR ANY I.
C
C      IFLAG IS USED FOR COMMUNICATION BETWEEN THE INTEGRATOR
C            AND THE PROGRAM WHICH CALLS IT. THE VALUE
C            OF IFLAG SHOULD NOT BE CHANGED BY THE USER.
C        THE FOLLOWING VALUES OF IFLAG HAVE THE FOLLOWING MEANINGS.
C      =1  THE VALUE OF Y FOR THE CURRENT STEP HAS BEEN
C          PREDICTED. THE USER SHOULD COMPUTE F AND CALL DVDQ1.
C          IF A RELATIVE ERROR TEST IS USED THE NEW VALUE
C          OF EP SHOULD ALSO BE COMPUTED HERE.
C      =2  THE VALUE OF Y FOR THE CURRENT STEP HAS BEEN
C          CORRECTED. THE USER SHOULD COMPUTE F AND CALL DVDQ1.
C      =3  AN OUTPUT POINT HAS BEEN REACHED (SEE DESCRIPTION
C          OF DELT), PRINT RESULTS AND CALL DVDQ1.
C      =4  T=TFINAL    IF DVDQ1 IS CALLED WITH T=TFINAL AND
C          IFLAG=4, IFLAG IS SET EQUAL TO 8. IF THE VALUE OF
C          TFINAL IS CHANGED THE INTEGRATION WILL CONTINUE.
C      =5  KSTEP=KSOUT   (SEE THE DESCRIPTION OF MXSTEP).
C      =6  EMAX.GT..1 AND IT APPEARS TO THE SUBROUTINE THAT
C          REDUCING H WILL NOT HELP BECAUSE OF ROUND-OFF ERROR.
C          IF THIS OCCURS A LARGER VALUE OF EP (OR OF ABS(EP(KEMAX)) IF
C          EP IS A VECTOR) SHOULD PROBABLY BE USED. IF EP IS NOT
C          INCREASED, TOO SMALL A STEPSIZE IS LIABLE TO BE USED. (WE HAVE
C          FOUND THAT REPLACING EP WITH 32.*EMAX*EP WORKS QUITE WELL.)
C          INCREASING EP IN THIS WAY WILL NOT DEGRADE THE ACCURACY,
C          HOWEVER IF THE NATURE OF THE PROBLEM CHANGES IT MAY PAY TO
C          USE A SMALLER VALUE OF EP LATER IN THE INTEGRATION.
C      =7  ABS(H).LT.HMINA.  TO CONTINUE WITH THE CURRENT
C          VALUE OF H, SET HMINA.LE.ABS(H) AND CALL DVDQ1.
C          IF THE INTEGRATOR HAS JUST HALVED H ONE MAY CONTINUE
```

```
C         WITH TWICE THE STEPSIZE BY SIMPLY CALLING DVDQ1. (SUCH
C         AN ACTION IS RISKY WITHOUT A CAREFUL ANALYSIS OF THE
C         SITUATION.) IF THE STEPSIZE HAS NOT JUST BEEN HALVED
C         (ABS(H).LT.HMINA MAY BE DUE TO THE USER INCREASING THE
C         VALUE OF HMINA OR TO HAVING TOO SMALL AN H AT THE END
C         OF THE STARTING PHASE.) THE INTEGRATION WILL CONTINUE
C         WITH THE CURRENT VALUE OF H AND A RETURN TO THE USER WITH
C         IFLAG=7 WILL BE MADE ON EVERY STEP UNTIL ABS(H).GE.HMINA.
C     =8  ILLEGAL PARAMETER IN THE CALLING SEQUENCE. IF DVDQ1
C         IS CALLED WITH IFLAG=8 THE PROGRAM IS STOPPED.
C
C     H=CURRENT VALUE OF THE STEPSIZE. IN SELECTING THE INITIAL
C       VALUE FOR H, THE USER SHOULD REMEMBER THE FOLLOWING--
C      1. THE INTEGRATOR IS CAPABLE OF CHANGING H QUITE QUICKLY AND
C         THUS THE INITIAL CHOICE IS NOT CRITICAL.
C      2. IF IT DOES NOT LEAD TO PROBLEMS IN COMPUTING THE DERIVATIVES
C         (E.G. BECAUSE OF OVERFLOW OR TRYING TO EXTRACT THE SQUARE
C         ROOT OF A NEGATIVE NUMBER), IT IS BETTER TO CHOOSE H MUCH
C         TOO LARGE THAN MUCH TOO SMALL.
C      3. IF H*DELT.LE.0 INITIALLY, AN IMMEDIATE RETURN IS MADE
C         WITH IFLAG=8. THE SIGN OF H IS WHAT DETERMINES THE
C         DIRECTION OF INTEGRATION.
C      4. IF DELT=H*(2**K) K A NONNEGATIVE INTEGER THEN OUTPUT
C         VALUES WILL BE OBTAINED WITHOUT DOING AN INTERPOLATION.
C
C     HMINA   AFTER GETTING STARTED, AND WHENEVER H
C             IS HALVED, ABS(H) IS COMPARED WITH HMINA.
C             IF ABS(H).LT.HMINA CONTROL IS RETURNED TO
C             THE USER WITH IFLAG=7.
C
C     HMAXA   THE STEPSIZE IS NOT DOUBLED IF
C             DOING SO WOULD MAKE ABS(H).GT.HMAXA
C
C     DELT  ENABLES THE USER TO SPECIFY THE POINTS WHERE
C       OUTPUT IS DESIRED. LET TOUT=DELT + THE VALUE OF T THE LAST
C       TIME CONTROL WAS RETURNED TO THE USER WITH IFLAG=3. (INITIALLY
C       TOUT=THE INITIAL VALUE OF T.) CONTROL IS RETURNED TO THE
C       USER WITH IFLAG=3 WHENEVER T=TOUT. IF TOUT DOES NOT FALL
C       ON AN INTEGRATION STEP, OUTPUT VALUES ARE OBTAINED BY
C       INTERPOLATION ON THE FIRST STEP THAT (T-TOUT)*H.GT.0.
C       INTERPOLATED VALUES FOR BOTH Y AND F ARE COMPUTED.
C       (NOTE THAT A RETURN WITH IFLAG=3 IS ALWAYS MADE
C       BEFORE TAKING THE FIRST STEP.)
C
C     TFINAL   CONTROL IS RETURNED TO THE USER WITH IFLAG=4 WHEN
C       T REACHES TFINAL. IF TFINAL DOES NOT FALL ON AN INTEGRATION
C       STEP VALUES AT TFINAL ARE OBTAINED BY EXTRAPOLATION.
C
C     MXSTEP   ON THE INITIAL ENTRY, AND ON ENTRIES
C       WITH  2.LT.IFLAG.LT.6  KSOUT IS SET EQUAL TO
C       KSTEP+MXSTEP. AT THE END OF EACH STEP KSTEP IS INCREMENTED
C       AND COMPARED WITH KSOUT. IF KSTEP.GE.KSOUT CONTROL IS
C       RETURNED TO THE USER WITH IFLAG=5. (THUS IF DELT IS
C       SUFFICIENTLY LARGE, CONTROL WILL BE RETURNED TO THE USER
C       WITH IFLAG=5 EVERY MXSTEP STEPS.)
C
C     KSTEP=NUMBER OF INTEGRATION STEPS TAKEN (COMPUTED
C       BY THE INTEGRATOR.)
C
C     KEMAX=INDEX OF COMPONENT RESPONSIBLE FOR THE
```

```
C           VALUE OF EMAX (SEE BELOW).
C
C     EMAX=LARGEST VALUE IN ANY COMPONENT OF (ESTIMATED ERROR)/EP
C       ORDINARILY THE STEPSIZE IS HALVED IF EMAX.GT..1. WITH A
C       RECENT HISTORY OF LOCAL ROUND-OFF PROBLEMS VALUES OF EMAX AS
C       LARGE AS 1 ARE PERMITTED. THE STEPSIZE IS NOT HALVED ON ANY
C       STEP THAT ROUND OFF ERROR APPEARS TO BE LIMITING THE PRECISION.
C
C     KQ(I)=HIGHEST ORDER DIFFERENCE USED IN INTEGRATING
C       THE I-TH EQUATION. (COMPUTED BY THE INTEGRATOR)
C
C     YN=A VECTOR WITH THE DIMENSION OF Y USED TO STORE
C       THE VALUE OF Y AT THE END OF EACH INTEGRATION STEP.
C
C     DT=AN ARRAY WITH DIMENSION DT(17,NEQ) USED TO
C       STORE THE DIFFERENCE TABLE.
C
C
      DOUBLE PRECISION TOUT,TL,TPD,TPD1,TPD2,HH,FAC
      DOUBLE PRECISION DD,D,GAM,GAS,PT,TP
      DIMENSION DD(19),D(18),GAM(17,4),GAS(17),PT(18),FAC(3)
      EQUIVALENCE(DD(2),D(1))
      DIMENSION ETA(15,15)
C
      DATA KMAXO/4/
C     KMAXO IS THE MAXIMUM ORDER DIFFERENTIAL EQUATION
C     THIS SUBROUTINE WILL INTEGRATE.
C
      DATA FAC/1.D0,.5D0,.16666666666666667D0/
C     FAC(J)=1/(FACTORIAL J), J=1,2,...,MAX(2,KMAXO-1)
C
      DATA KQMAX/16/
C     KQMAX GIVES THE MAXIMUM ORDER.
C     THERE IS LITTLE POINT IN HAVING KQMAX MUCH BIGGER THAN THE NUMBER
C     OF DECIMAL DIGITS IN THE MANTISSA.
C     IF KQMAX IS SET LESS THAN 6, DT, D, AND PT SHOULD BE DIMENSIONED
C     AS IF KQMAX=6.
C
      DATA RND,KBIT2/8.88E-16,108/
C     RND IS APPROXIMATELY 2**(3-B) WHERE B IS
C     THE NUMBER OF BITS IN THE MANTISSA.
C     KBIT2=2*B+2 WHERE B IS THE NUMBER OF BITS IN THE MANTISSA.
C     IF THE DERIVATIVES ARE NOT COMPUTED TO THE ACCURACY EXPECTED
C     FROM THE WORD LENGTH OF THE COMPUTER (FOR EXAMPLE BECAUSE OF
C     CANCELLATION PROBLEMS OR TABULAR DATA), THEN THESE CONSTANTS
C     CAN BE CHANGED TO REFLECT THE NUMBER OF BITS WHICH ARE
C     SIGNIFICANT IN THE COMPUTED DERIVATIVES. (THIS IS NOT NECESSARY,
C     BUT IS WISE IF THE ACCURACY REQUESTED IS DIFFICULT TO OBTAIN
C     BECAUSE THE DERIVATIVES HAVE SO FEW SIGNIFICANT DIGITS.)
C
      DATA P1,P01,P25,P3E1/.1,.01,.25,3./
C     THE ABOVE DATA STATEMENT CONTAINS VARIOUS CONSTANTS
C     USED IN THE SUBROUTINE.
C
      DATA PT/1.D0,2.D0,4.D0,8.D0,16.D0,32.D0,64.D0,128.D0,256.D0,
     1  512.D0,1024.D0,2048.D0,4096.D0,8192.D0,16384.D0,32768.D0,
     2  65536.D0,131072.D0/
C     PT(J)=2**(J-1), J=1,2,...,KQMAX+2
C     DATA PTS1,PTS2,PTS3,PTS4,PTS5,P5/1.,2.,4.,8.,16.,.5/
      DATA PTS1,PTS2,     PTS4,PTS5,P5/1.,2.,   8.,16.,.5/
```

```
C
      DATA GAS/1.D0,-.5D0,-8.33333333333333333D-02,
     1  -4.16666666666666667D-02,-2.63888888888888889D-02,
     2  -1.875D-02,              -1.42691798941798942D-02,
     3  -1.13673941798941799D-02,-9.35653659611992945D-03,
     4  -7.89255401234567901D-03,-6.78584998463470686D-03,
     5  -5.92405641233766234D-03,-5.23669325795028507D-03,
     6  -4.67749840704226452D-03,-4.21495223900547286D-03,
     7  -3.82689955321188442D-03,-3.49734984534991765D-03/
C     GAS(I) GIVES THE I-TH ADAMS-MOULTON CORRECTOR
C     COEFFICIENT, I=1,2,...,KQMAX+1.
C
      DATA       GAM(01,01),GAM(02,01),GAM(03,01),GAM(04,01),GAM(05,01),
     *           GAM(06,01),GAM(07,01),GAM(08,01),GAM(09,01),GAM(10,01),
     *           GAM(11,01),GAM(12,01),GAM(13,01),GAM(14,01),GAM(15,01),
     *           GAM(16,01),GAM(17,01)/
     1  1.D0,.5D0,.416666666666666667D0,.375D0,
     2  .348611111111111111D0,.329861111111111111D0,
     3  .315591931216931217D0,.304224537037037037D0,
     4  .294868000440917108D0,.286975446428571429D0,
     5  .280189596443936722D0,.274265540031599059D0,
     6  .269028846773648774D0,.264351348366606510D0,
     7  .260136396127601037D0,.256309496574389153D0,
     8  .252812146729039235D0/
      DATA       GAM(01,02),GAM(02,02),GAM(03,02),GAM(04,02),GAM(05,02),
     *           GAM(06,02),GAM(07,02),GAM(08,02),GAM(09,02),GAM(10,02),
     *           GAM(11,02),GAM(12,02),GAM(13,02),GAM(14,02),GAM(15,02),
     *           GAM(16,02),GAM(17,02)/
     1  .5D0,.166666666666666667D0,.125D0,.105555555555555556D0,
     2  9.375D-2,               8.56150793650793651D-2,
     3  7.95717592592592593D-2,7.48522927689594356D-2,
     4  7.10329861111111111D-2,6.78584998463470686D-2,
     5  6.51646205357142857D-2,6.28403190954034208D-2,
     6  6.08074792915494387D-2,5.90093313460766200D-2,
     7  5.74034932981782663D-2,5.59575975255986825D-2,
     8  5.46464393325006467D-2/
      DATA       GAM(01,03),GAM(02,03),GAM(03,03),GAM(04,03),GAM(05,03),
     *           GAM(06,03),GAM(07,03),GAM(08,03),GAM(09,03),GAM(10,03),
     *           GAM(11,03),GAM(12,03),GAM(13,03),GAM(14,03),GAM(15,03),
     *           GAM(16,03),GAM(17,03)/
     1  .166666666666666667D0,4.16666666666666667D-2,
     2  2.91666666666666667D-2,2.36111111111111111D-2,
     3  2.03373015873015873D-2,1.81299603174603175D-2,
     4  1.65181327160493827D-2,1.52772266313932981D-2,
     5  1.42851881914381914D-2,1.34693965531639143D-2,
     6  1.27836579217097570D-2,1.21970388231238926D-2,
     7  1.16879616455733216D-2,1.12408663352884755D-2,
     8  1.08442182943468791D-2,1.04892655447842863D-2,
     9  1.01692338611494262D-2/
      DATA       GAM(01,04),GAM(02,04),GAM(03,04),GAM(04,04),GAM(05,04),
     *           GAM(06,04),GAM(07,04),GAM(08,04),GAM(09,04),GAM(10,04),
     *           GAM(11,04),GAM(12,04),GAM(13,04),GAM(14,04),GAM(15,04),
     *           GAM(16,04),GAM(17,04)/
     1  4.16666666666666667D-2,8.33333333333333333D-3,
     2  5.55555555555555556D-3,4.36507936507936508D-3,
     3  3.67890211640211640D-3,3.22365520282186949D-3,
     4  2.89544753086419753D-3,2.64543583988028432D-3,
     5  2.44737491482283149D-3,2.28579543818052416D-3,
     6  2.15093669481483635D-3,2.03630871020228384D-3,
     7  1.93741301123433302D-3,1.85102419106078320D-3,
     8  1.77476374781296400D-3,1.70683564605258723D-3,
     9  1.64585591005465158D-3/
C       GAM(I,J) GIVES THE I-TH ADAMS-FALKNER PREDICTOR
C       COEFFICIENT FOR INTEGRATING J-TH ORDER DIFFERENTIAL
C       EQUATIONS, I=1,2,...,KQMAX+1,  J=1,2,...,KMAXO.
C
        DATA      ETA(01,01),ETA(02,01),ETA(03,01),ETA(04,01),ETA(05,01),
     *            ETA(06,01),ETA(07,01),ETA(08,01),ETA(09,01),ETA(10,01),
     *            ETA(11,01),ETA(12,01),ETA(13,01),ETA(14,01),ETA(15,01)/
     *                                 3.33333330E-01, 2.50000000E-01,
     1  1.13636360E-01, 6.73076930E-02, 4.60526330E-02, 3.43749980E-02,
     2  2.71381590E-02, 2.22547310E-02, 1.87484580E-02, 1.61123220E-02,
     3  1.40603000E-02, 1.24197060E-02, 1.10802170E-02, 9.96793590E-03,
     4  9.03137260E-03/
        DATA      ETA(01,02),ETA(02,02),ETA(03,02),ETA(04,02),ETA(05,02),
     *            ETA(06,02),ETA(07,02),ETA(08,02),ETA(09,02),ETA(10,02),
     *            ETA(11,02),ETA(12,02),ETA(13,02),ETA(14,02),ETA(15,02)/
     *                                 2.00000000E-01, 4.00000000E-01,
     1  3.40909090E-01, 2.01923080E-01, 1.38157900E-01, 1.03124990E-01,
     2  8.14144780E-02, 6.67641930E-02, 5.62453730E-02, 4.83369670E-02,
     3  4.21809010E-02, 3.72591170E-02, 3.32406510E-02, 2.99038080E-02,
     4  2.70941180E-02/
        DATA      ETA(01,03),ETA(02,03),ETA(03,03),ETA(04,03),ETA(05,03),
     *            ETA(06,03),ETA(07,03),ETA(08,03),ETA(09,03),ETA(10,03),
     *            ETA(11,03),ETA(12,03),ETA(13,03),ETA(14,03),ETA(15,03)/
     *                                 1.42857140E-01, 2.85714280E-01,
     1  3.42857140E-01, 3.46153840E-01, 2.45614040E-01, 1.87500000E-01,
     2  1.50303650E-01, 1.24626490E-01, 1.05873640E-01, 9.15858320E-02,
     3  8.03445710E-02, 7.12783130E-02, 6.38220510E-02, 5.75925170E-02,
     4  5.23196800E-02/
        DATA      ETA(01,04),ETA(02,04),ETA(03,04),ETA(04,04),ETA(05,04),
     *            ETA(06,04),ETA(07,04),ETA(08,04),ETA(09,04),ETA(10,04),
     *            ETA(11,04),ETA(12,04),ETA(13,04),ETA(14,04),ETA(15,04)/
     *                                 1.11111110E-01, 2.22222220E-01,
     1  2.85714280E-01, 2.53968250E-01, 3.07017540E-01, 2.50000000E-01,
     2  2.08755060E-01, 1.78037850E-01, 1.54399060E-01, 1.35682710E-01,
     3  1.20516850E-01, 1.07997450E-01, 9.75059080E-02, 8.86038720E-02,
     4  8.09709320E-02/
        DATA      ETA(01,05),ETA(02,05),ETA(03,05),ETA(04,05),ETA(05,05),
     *            ETA(06,05),ETA(07,05),ETA(08,05),ETA(09,05),ETA(10,05),
     *            ETA(11,05),ETA(12,05),ETA(13,05),ETA(14,05),ETA(15,05)/
     *                                 9.09090910E-02, 1.81818180E-01,
     1  2.42424240E-01, 2.42424240E-01, 1.73160170E-01, 2.50000000E-01,
     2  2.27732800E-01, 2.05428290E-01, 1.85278880E-01, 1.67608050E-01,
     3  1.52231820E-01, 1.38853860E-01, 1.27181620E-01, 1.16957100E-01,
     4  1.07961240E-01/
        DATA      ETA(01,06),ETA(02,06),ETA(03,06),ETA(04,06),ETA(05,06),
     *            ETA(06,06),ETA(07,06),ETA(08,06),ETA(09,06),ETA(10,06),
     *            ETA(11,06),ETA(12,06),ETA(13,06),ETA(14,06),ETA(15,06)/
     *                                 7.69230760E-02, 1.53846150E-01,
     1  2.09790210E-01, 2.23776220E-01, 1.86480190E-01, 1.11888110E-01,
     2  1.91295550E-01, 1.91733070E-01, 1.85278880E-01, 1.75988460E-01,
     3  1.65763530E-01, 1.55516330E-01, 1.45680770E-01, 1.36449950E-01,
     4  1.27892540E-01/
        DATA      ETA(01,07),ETA(02,07),ETA(03,07),ETA(04,07),ETA(05,07),
     *            ETA(06,07),ETA(07,07),ETA(08,07),ETA(09,07),ETA(10,07),
     *            ETA(11,07),ETA(12,07),ETA(13,07),ETA(14,07),ETA(15,07)/
     *                                 6.66666660E-02, 1.33333330E-01,
     1  1.84615380E-01, 2.05128200E-01, 1.86480190E-01, 1.34265730E-01,
     2  6.96192690E-02, 1.39442230E-01, 1.52023690E-01, 1.56434190E-01,
    3  1.56012730E-01, 1.52787970E-01, 1.47993160E-01, 1.42382550E-01,
    4  1.36418720E-01/
     DATA      ETA(01,08),ETA(02,08),ETA(03,08),ETA(04,08),ETA(05,08),
    *          ETA(06,08),ETA(07,08),ETA(08,08),ETA(09,08),ETA(10,08),
    *          ETA(11,08),ETA(12,08),ETA(13,08),ETA(14,08),ETA(15,08)/
    *                              5.88235290E-02, 1.17647060E-01,
    1  1.64705880E-01, 1.88235290E-01, 1.80995470E-01, 1.44796380E-01,
    2  9.21431500E-02, 4.21225830E-02, 9.77295190E-02, 1.14931240E-01,
    3  1.25367380E-01, 1.30961120E-01, 1.33193850E-01, 1.33136920E-01,
    4  1.31546610E-01/
     DATA      ETA(01,09),ETA(02,09),ETA(03,09),ETA(04,09),ETA(05,09),
    *          ETA(06,09),ETA(07,09),ETA(08,09),ETA(09,09),ETA(10,09),
    *          ETA(11,09),ETA(12,09),ETA(13,09),ETA(14,09),ETA(15,09)/
    *                              5.26315790E-02, 1.05263160E-01,
    1  1.48606810E-01, 1.73374610E-01, 1.73374610E-01, 1.48606810E-01,
    2  1.06692070E-01, 6.09668970E-02, 2.49410030E-02, 6.63064840E-02,
    3  8.35782530E-02, 9.62949390E-02, 1.05153040E-01, 1.10947430E-01,
    4  1.14388350E-01/
     DATA      ETA(01,10),ETA(02,10),ETA(03,10),ETA(04,10),ETA(05,10),
    *          ETA(06,10),ETA(07,10),ETA(08,10),ETA(09,10),ETA(10,10),
    *          ETA(11,10),ETA(12,10),ETA(13,10),ETA(14,10),ETA(15,10)/
    *                              4.76190480E-02, 9.52380950E-02,
    1  1.35338350E-01, 1.60401000E-01, 1.65118680E-01, 1.48606810E-01,
    2  1.15583080E-01, 7.54828240E-02, 3.91930050E-02, 1.45159280E-02,
    3  4.37790860E-02, 5.88469080E-02, 7.14002090E-02, 8.13614340E-02,
    4  8.89687070E-02/
     DATA      ETA(01,11),ETA(02,11),ETA(03,11),ETA(04,11),ETA(05,11),
    *          ETA(06,11),ETA(07,11),ETA(08,11),ETA(09,11),ETA(10,11),
    *          ETA(11,11),ETA(12,11),ETA(13,11),ETA(14,11),ETA(15,11)/
    *                              4.34782610E-02, 8.69565210E-02,
    1  1.24223600E-01, 1.49068320E-01, 1.56914020E-01, 1.46453090E-01,
    2  1.20608430E-01, 8.61488760E-02, 5.16893250E-02, 2.46139640E-02,
    3  8.33088030E-03, 2.82465160E-02, 4.03201180E-02, 5.13861540E-02,
    4  6.10071000E-02/
     DATA      ETA(01,12),ETA(02,12),ETA(03,12),ETA(04,12),ETA(05,12),
    *          ETA(06,12),ETA(07,12),ETA(08,12),ETA(09,12),ETA(10,12),
    *          ETA(11,12),ETA(12,12),ETA(13,12),ETA(14,12),ETA(15,12)/
    *                              4.00000000E-02, 7.99999990E-02,
    1  1.14782610E-01, 1.39130430E-01, 1.49068320E-01, 1.43105590E-01,
    2  1.23020590E-01, 9.37299770E-02, 6.20271900E-02, 3.44595500E-02,
    3  1.51622020E-02, 4.72588120E-03, 1.78691420E-02, 2.69906870E-02,
    4  3.60496320E-02/
     DATA      ETA(01,13),ETA(02,13),ETA(03,13),ETA(04,13),ETA(05,13),
    *          ETA(06,13),ETA(07,13),ETA(08,13),ETA(09,13),ETA(10,13),
    *          ETA(11,13),ETA(12,13),ETA(13,13),ETA(14,13),ETA(15,13)/
    *                              3.70370370E-02, 7.40740740E-02,
    1  1.06666670E-01, 1.30370370E-01, 1.41706920E-01, 1.39130430E-01,
    2  1.23671500E-01, 9.89371980E-02, 7.02974820E-02, 4.33935080E-02,
    3  2.24625220E-02, 9.18921340E-03, 2.65466170E-03, 1.11137880E-02,
    4  1.77085690E-02/
     DATA      ETA(01,14),ETA(02,14),ETA(03,14),ETA(04,14),ETA(05,14),
    *          ETA(06,14),ETA(07,14),ETA(08,14),ETA(09,14),ETA(10,14),
    *          ETA(11,14),ETA(12,14),ETA(13,14);ETA(14,14),ETA(15,14)/
    *                              3.44827580E-02, 6.89655170E-02,
    1  9.96168580E-02, 1.22605360E-01, 1.34865900E-01, 1.34865900E-01,
    2  1.23138430E-01, 1.02348820E-01, 7.67616190E-02, 5.11744120E-02,
    3  2.96272910E-02, 1.43647470E-02, 5.49240340E-03, 1.47872400E-03,
    4  6.81096220E-03/
     DATA      ETA(01,15),ETA(02,15),ETA(03,15),ETA(04,15),ETA(05,15),
    *          ETA(06,15),ETA(07,15),ETA(08,15),ETA(09,15),ETA(10,15),
     *          ETA(11,15),ETA(12,15),ETA(13,15),ETA(14,15),ETA(15,15)/
     *                                  3.22580640E-02, 6.45161290E-02,
     1 9.34371530E-02, 1.15684090E-01, 1.28537880E-01, 1.30515390E-01,
     2 1.21814360E-01, 1.04412310E-01, 8.17139820E-02, 5.77775630E-02,
     3 3.63173250E-02, 1.98094500E-02, 9.03588950E-03, 3.24365270E-03,
     4 8.17727570E-04/
C     ETA(I,J) I=1,2,...,J IS USED IN THE FIRST MODIFICATION OF THE
C     I-TH DIFFERENCE OF A J-TH ORDER METHOD AFTER THE STEPSIZE IS
C     HALVED.
C     ETA(I,J) J=1,2,...,I-1 IS USED IN THE SECOND MODIFICATION OF
C     THE (J+1)-ST DIFFERENCE OF AN I-TH ORDER METHOD.
C     THE TWO MODIFICATIONS OF THE DIFFERENCE TABLE AFTER HALVING THE
C     STEPSIZE REMOVES MOST OF THE INSTABILITY INHERENT IN THE METHOD
C     USED HERE FOR HALVING THE STEPSIZE.
C
C
C     IF THE GSTOP FEATURE IS ELIMINATED, REMOVE THE FOLLOWING CARD.
      DATA LGSS,LGSD,LGSE/0,0,0/
C
C     INITIALIZE
      KSTEP=-1
      NE=NEQ
      IF (NE.LE.0) GO TO 1190
      HH=H
      NV=0
      KDMAX=0
      KDD=KD(1)
      KDS=KDD
      DO 10 I=1,NE
      KQ(I)=1
      DT(1,I)=0.D0
      IF (KDS.LE.0) KDD=IABS(KD(I))
      IF ((KDD.EQ.0).OR.(KDD.GT.KMAXO)) HH=0.D0
      IF (KDD.GT.KDMAX) KDMAX=KDD
   10 NV=NV+KDD
C
      IF ((DELT*HH).LE.0.D0) GO TO 1190
      ERRMX=P1
      EMAX=ERND
      RNDC=RND*P25
      LDOUB=0
      E2HFAC=P25
      LSC=8
      LSTC=4
C     LSC AND LSTC ARE USED IN COMBINATION AS FOLLOWS
C       LSTC=4,  LSC=4     FIRST TIME THROUGH THE FIRST STEP
C       LSTC=3,  LSC=4     SECOND TIME THROUGH THE FIRST STEP
C                          (NECESSARY TO CHECK STABILITY)
C       LSTC=2,  LSC=4     THIRD TIME THROUGH THE FIRST STEP
C                          (ONLY OCCURS IF INSTABILITY POSSIBLE)
C       LSTC=2,  LSC=2     SECOND STEP  (IF KQ(I)=2 , I=1,...,NEQ)
C       LSTC=1,  LSC=0     STARTING, ONE DERIVATIVE EVAL. PER STEP.
C       LSTC=1,  LSC.GT.0  SET WHEN STARTING TWO DERIV. EVAL. PER STEP
C       LSTC=-1  LSC.LT.0  SET WHEN HALVING THE STEPSIZE
C      IN THE LAST TWO CASES LSC IS SET EQUAL TO LSTC*(MAXIMUM KQ(I)
C      +1). AT THE END OF EACH STEP IF LSC.NE.0 IT IS REPLACED BY
C      LSC-LSTC UNTIL LSC=0, AT WHICH TIME LSTC IS SET EQUAL TO 0.
C       WHEN DOUBLING H, LSTC IS SET EQUAL TO -1 AND LSC TO -3.
C      UNDER CERTAIN CONDITIONS WHEN KQ(I)=1, LSTC IS SET =-1 AND LSC=-5
C
```

```
      KSOUT=MXSTEP
      TOUT=T
      IFL=13
   20 IFLAG=1
      GO TO 315
C     END OF INITIALIZATION
C
C
      ENTRY DVDQ1
C
C
C     TO OUTPUT VARIABLES IN THE CALLING SEQUENCE REMOVE THE C-S
C     IN COLUMN ONE OF THE FOLLOWING CARDS UNTIL REACHING THE COMMENT
C     END OF CODE FOR PRINTING VARIABLES IN CALLING SEQUENCE.
C     IF (NEQ.NE.0) GO TO 28
C     NEQ=1
C  22 WRITE(6,5000) T,DELT,HMINA,HMAXA,KEMAX,EMAX,IFLAG,TFINAL,MXSTEP
C5000 FORMAT(3HOT=1PD24.17,7H  DELT=D12.5,8H  HMINA=,E10.3,8H  HMAXA=,
C    1  E10.3,8H  KEMAX=,I2,7H  EMAX=E10.3,8H  IFLAG=,I2/
C    2  9H  I KQ KD,7X,4HF(I),9X,1HJ,12X,4HY(J),22X,5HYN(J),
C    3  10X,7HTFINAL=1PD15.8,9H  MXSTEP=I4)
C     J=1
C     DO 24 I=1,NE
C     IF (KDS.LT.0) KDD=IABS(KD(I))
C     K=KDD
C     WRITE(6,5001) I,KQ(I),KDD,F(I),J,Y(J),YN(J)
C5001 FORMAT(1H ,I2,2I3,1PD17.8,I4,2D26.17)
C  23 J=J+1
C     K=K-1
C     IF (K.EQ.0) GO TO 24
C     WRITE(6,5002) J,Y(J),YN(J)
C5002 FORMAT(26X,I4,1P2D26.17)
C     GO TO 23
C  24 CONTINUE
C     WRITE(6,5003)
C5003 FORMAT(3HO I,15X,16HDIFFERENCE TABLE)
C     DO 27 I=1,NE
C     KQQ=KQ(I)+1
C     K=MINO(KQQ,7)
C     WRITE(6,5004) I,(DT(IO,I),IO=1,K)
C5004 FORMAT(1H ,I2,1PD19.8,6D16.7)
C
C     IF (K.EQ.KQQ) GO TO 27
C     K=K+1
C     WRITE(6,5005) (DT(IO,I),IO=K,KQQ)
C5005 FORMAT(1H ,1PD21.5,7D14.5)
C  27 CONTINUE
C     IF (NEQ.EQ.0) RETURN
C     NEQ=0
C 28  CONTINUE
C     END OF CODE FOR PRINTING VARIABLES IN CALLING SEQUENCE.
C
      IF (2-IFL) 30,60,320
   30 IF (IFL.GT.5) GO TO 1180
C
C     SET STEP STOP
      KSOUT=KSTEP+MXSTEP
      IF (IFL-4) 40,1210,210
C
C
```

```
C     SET PRINT STOP
   40 TOUT=T+DELT
C
   50 TPS1=ABS(SNGL(DMOD((TOUT-T)/HH,2.D0))-PTS1)
      LFD=-1
      IF (TPS1.GE.P5) LFD=1
C
C     LFD IS USED TO INDICATE WHETHER DOUBLING H IS PERMITTED.
C     IF LFD.LT.0 AT THE END OF A STEP THEN DOUBLING H IS
C     NOT PERMITTED. THE SIGN OF LFD IS CHANGED JUST BEFORE THE
C     END OF EACH STEP. IF DELT#H*%POWER OF 2< THEN
C     OUTPUT VALUES WILL BE OBTAINED WITHOUT INTERPOLATION.
C
      GO TO 200
C
C
C     ENTRY WITH IFLAG=2
C
C     UPDATE DIFFERENCE TABLE
C     AND COMPUTE KQM=MAXIMUM VALUE OF KQ(I), I=1,2,...,NEQ.
C
   60 KQM=0
      DO 80 I=1,NE
      KQQ=KQ(I)
      IF (KQQ.GT.KQM) KQM=KQQ
      D(1)=F(I)
      DO 70 K=1,KQQ
      D(K+1)=D(K)-DT(K,I)
   70 DT(K,I)=D(K)
      DT(KQQ+1,I)=D(KQQ+1)
   80 CONTINUE
C     END OF UPDATING DIFFERENCE TABLE
C
C     STORE Y(J) IN YN(J)
      DO 90 J=1,NV
   90 YN(J)=Y(J)
C
      LFD=-LFD
      TL=T
      KSTEP=KSTEP+1
C
C     IF THE GSTOP FEATURE IS ELIMINATED, REMOVE THE 2 FOLLOWING CARDS.
      IF (LGSS) 1430,110,1510
  100 IFLAG=2
  110 IF (LSC.EQ.0) GO TO 140
      LSC=LSC-LSTC
      IF (LSC.EQ.0) GO TO 130
      IF (LSTC.NE.(-1)) GO TO 140
      IF (LDOUB.LT.0) RNDC=RND*P1
  120 E2HAVE=E2HMAX
      TPS1=PTS1
      GO TO 190
  130 IF (ABS(SNGL(HH)).LT.HMINA) GO TO 1000
      LSTC=0
  140 IF (LDOUB.NE.1) GO TO 150
      IF ((LFD.GT.0).AND.(ABS(SNGL(HH+HH)).LE.HMAXA)) GO TO 1030
      GO TO 200
  150 RQMAX=PTS1/FLOAT(KQM+3)
      IF (LSTC.NE.0) GO TO 120
      TPS1=E2HMAX/E2HAVE
      IF (TPS1-PTS1) 160,190,170
  160 E2HFAC=AMAX1(.075E0,E2HFAC-RQMAX,E2HFAC*TPS1)
      GO TO 180
  170 TPS1=TPS1*TPS1
      E2HFAC=AMIN1(PTS1,E2HFAC*TPS1)
  180 RNDC=(1.1-E2HFAC)*RND
      E2HAVE=P5*(E2HMAX+E2HAVE)
  190 ERRMX=AMAX1(P1,ERRMX-RQMAX*TPS1)
C     E2HFAC IS A FACTOR WHICH IS TAKEN TIMES AN INITIAL ESTIMATE OF
C            E2H TO GET A FINAL VALUE OF E2H. (E2H=ESTIMATE OF WHAT
C            (ESTIMATED ERROR)/(REQUESTED ERROR) WOULD BE IF H WERE
C            DOUBLED.).
C     E2HMAX IS THE MAXIMUM VALUE OF THE INITIAL ESTIMATE OF E2H OVER
C            ALL COMPONENTS WITH KQ(I).GT.1.
C     E2HAVE IS A WEIGHTED AVERAGE OF PAST VALUES OF E2HMAX.
C     THE VALUE OF E2HFAC TENDS TO BE SMALLER WHEN E2HMAX IS
C     CONSISTANTLY SMALLER THAN E2HAVE.
C
C
C     CHECK FOR PRINT STOP AND FOR T REACHING TFINAL
  200 TPD=(TOUT-TL)/HH
      TPD1=(TFINAL-TL)/HH
C
C     IF THE GSTOP FEATURE IS ELIMINATED, REMOVE THE FOLLOWING CARD.
      IF (LGSE.LT.0) GO TO 1780
      IF (TPD1.LT.FAC(1)) GO TO 1220
      IF (TPD.LE.0.D0) GO TO 1280
C
C     CHECK FOR STEP STOP
      IF (KSOUT.GT.KSTEP) GO TO 210
C
      IFL=5
      GO TO 310
C
C     CHECK TO SEE IF ROUND-OFF ERROR IS PROMINENT
  210 IF (EMAX.EQ.ERND) GO TO 220
C     IT IS
      IFL=6
      IF (EMAX.GE.P1) GO TO 310
      IF ((LSTC.GE.0).OR.(LDOUB.EQ.1)) ERRMX=PTS1
C
  220 IFL=1
  230 T=TL+HH
C
C     START A NEW STEP
C
C     PREDICT
  240 J=0
      DO 290 I=1,NE
      IF (KDS.LE.0) KDD=IABS(KD(I))
      KDC=KDD
  250 KQQ=KQ(I)
      TPD=0.D0
      K=KDC
  260 TPD=TPD+DT(KQQ,I)*GAM(KQQ,KDC)
      KQQ=KQQ-1
      IF (KQQ.GT.0) GO TO 260
  270 K=K-1
      IF (K.LE.0) GO TO 280
      L=J+K
      TPD=YN(L+1)*FAC(K)+HH*TPD
      GO TO 270
  280 J=J+1
      Y(J)=YN(J)+HH*TPD
      KDC=KDC-1
      IF (KDC.GT.0) GO TO 250
  290 CONTINUE
C     END OF PREDICT
C
C     IF THE GSTOP FEATURE IS ELIMINATED, REMOVE THE C  IN COLUMN ONE
C     OF THE 2 FOLLOWING CARDS
C     IF (IFL) 20,320,300
C 300 CONTINUE
C     AND THEN REMOVE THE 2 FOLLOWING CARDS.
      IF (IFL) 1240,320,300
  300 IF (LGSD.NE.0) GO TO 1520
C
  310 IFLAG=IFL
  315 CONTINUE
C
C     TO OUTPUT VARIABLES IN THE CALLING SEQUENCE REMOVE THE C IN
C     COLUMN ONE OF THE FOLLOWING CARD.
C     IF (NEQ.EQ.0) GO TO 22
C
      RETURN
C
C
C     ENTRY WITH IFLAG=1
  320 EPS=EP(1)
      ERND=0.
      EMAX=0.
      E2HMAX=0.
      J=0
      IF (LDOUB.GE.0) LDOUB=1
C
C     LDOUB IS SET IN THE LOOP BELOW AS FOLLOWS
C     LDOUB=0   HALVE
C     LDOUB=1   DOUBLE
C     LDOUB=2   DO NOT DOUBLE
C
C     LDOUB,LT.0 AT THE BEGINNING OF THE LOOP INDICATES THE FOLLOWING
C        =-3  STEPSIZE HAS JUST BEEN HALVED. IF A DISCONTINUITY IS
C             NOT INDICATED MODIFY THE DIFFERENCE TABLE AND REPEAT
C             THE STEP.
C        =-2  STEP AFTER LDOUB=-3. PROCEED AS USUAL (ORDER IS NOT
C             CHANGED)
C        =-1  STEP AFTER LDOUB=-2. MODIFY THE DIFFERENCE TABLE ONCE
C             AGAIN AND REPEAT THE STEP.
C     IF LDOUB IS SET EQUAL TO -4 THE ORDER IN AT LEAST ONE COMPONENT
C     HAS BEEN GREATLY REDUCED AND THE STEP IS REPEATED.
C
C
C     IF THE OUTPUT OPTION IS ELIMINATED, REMOVE THE 4 FOLLOWING CARDS.
      IF (NEQ.LE.0) WRITE (6,5020) LSC,LFD,LSTC,KSTEP,E2HFAC,ERRMX,HH
 5020 FORMAT (19H0 I   KQQ LRND LDOUB,5X,1HE,9X,3HE2H,
     1  8X,3HEPS,3X,4HLSC=,I3,6H  LFD=,I2,7H  LSTC=,I2,8H  KSTEP=,I4,
     2  9H  E2HFAC=,F4.2,8H  ERRMX=,F4.2,4H  H=,1PD9.2)
C
C
C     BEGINNING OF LOOP FOR CORRECTING, ESTIMATING THE ERROR,
C     AND ADJUSTING THE NUMBER OF DIFFERENCES USED
C
      DO 790 I=1,NE
      IF (KDS.LE.0) KDD=IABS(KD(I))
      KQQ=KQ(I)
C     KQQ GIVES THE ORDER OF THE PREDICTOR FORMULA AND KQQ+1 THE
C     ORDER OF THE CORRECTOR FORMULA.
C
      KQ1=KQQ+1
      D(1)=F(I)
C     FORM THE DIFFERENCE TABLE FROM PREDICTED DERIVATIVE VALUES.
      DO 330 K=1,KQ1
      D(K+1)=D(K)-DT(K,I)
  330 CONTINUE
C     D(K) GIVES THE (K-1)-ST DIFFERENCE FORMED FROM PREDICTED
C     DERIVATIVE VALUES
      TPS3=ABS(SNGL(D(KQQ+1)))
      IF (LDOUB.LT.0) GO TO 720
C
  340 IF (KQQ.NE.1) GO TO 520
C
C     KQ(I)=1  IS TREATED AS A SPECIAL CASE
      E2H=PTS2
      TPS5=DT(3,I)
      IF (LSTC.LT.2) GO TO 370.
C     FIRST STEP OF INTEGRATION
      IF (LSTC.NE.4) GO TO 350
      TPS4=0.
      IF (KDD.GT.1) TPS3=AMAX1(TPS3,ABS(SNGL(HH*D(1))))
      TPS3=TPS3*P1
      GO TO 510
  350 DT(2,I)=D(2)
      D(2)=D(1)-DT(5,I)
      TPS2=-D(2)
      TPS3=PTS5*ABS(TPS2)
C     FIRST STEP THAT KQ(I)=1
  360 DT(7,I)=PT(4)
      IF (LSTC-2) 420,380,380
  370 IF (TPS5.EQ.0.) GO TO 360
      IF (DT(6,I).EQ.0.D0) GO TO 400
      TPS2=DT(5,I)-DT(1,I)
  380 TPS4=DT(4,I)
      TPS1=ABS(TPS4)
      TPS4=TPS2*SIGN(PTS2,TPS4)-TPS5*TPS1
      IF (TPS4.GT.(-TPS1)) GO TO 410
  390 TPS6=-PTS1
      GO TO 450
C     FIRST STEP AFTER THE STEPSIZE HAS BEEN CHANGED
  400 DT(6,I)=PT(1)
      TPS6=0.
      GO TO 450
  410 IF (TPS4.LT.TPS1) GO TO 440
      IF (TPS1.EQ.0.) GO TO 390
  420 TPS6=PTS1
      GO TO 450
  430 KQ(I)=2
      IF (2-LSTC) 510,510,520
  440 TPS6=TPS4/TPS1
  450 TPS4=TPS5+TPS6
      IF (TPS4.LT.P25) GO TO 430
```

```
C     INCREASE E2H IF (-S).GT..25
      E2H=PTS4*TPS4
      IF (2-LSTC) 460,470,480
  460 LSC=0
      GO TO 510
  470 IF (TPS5-P25) 430,460,460
  480 IF (TPS4.GT.PTS2) GO TO 490
      IF (TPS4.GT.P5) D(2)=D(2)*GAM(2,1)
      GO TO 510
  490 IF (TPS4.LT.PTS4) GO TO 500
      TPS4=PTS4
      D(2)=D(2)/PT(3)
C     THE ESTIMATE OF E (AND HENCE OF E2H) IS INCREASED IF (-S).GE.8.
      TPS3=TPS3*SNGL(DT(7,I))
      GO TO 510
  500 D(2)=D(2)*DBLE(PTS2*(TPS4-PTS1)/(TPS4*TPS4))
      IF (TPS4.GE.P3E1) E2H=E2H*SNGL(DT(7,I))
C     STORE D(1)=PREDICTED DERIVATIVE  AND  D(2)=2*(CORRECTED Y -
C     PREDICTED Y)/H    D(1) AND D(2) ARE USED TO COMPUTE (-S) ON
C     THE NEXT STEP.
  510 DT(5,I)=D(1)
      DT(4,I)=D(2)
      D(4)=TPS4
C     STORE D(4)= CURRENT ESTIMATE OF (-S).  (-S).GT.3 IS AN INDICATION
C     THAT THE STEPSIZE SHOULD BE LIMITED BECAUSE OF STABILITY PROBLEMS.
C     S=H*(ESTIMATE OF EIGENVALUE OF F)=H*(DIFFERENCE BETWEEN PREDICTED
C     AND CORRECTED DERIVATIVE VALUES)/(DIFFERENCE BETWEEN PREDICTED
C     AND CORRECTED INTEGRALS OF THE DERIVATIVE VALUES)
C     THE TREATMENT OF THE CASE KQ(I)=1 COULD BE IMPROVED BY USING A
C     SPECIAL METHOD FOR STIFF EQUATIONS WHEN (-S).GT.3  (MAYBE).
C     (THE ENTIRE TREATMENT OF THE CASE KQ(I)=1 IS FAR FROM IDEAL.)
      DT(3,I)=D(4)
C
C     CORRECT
  520 KDC=0
      TPD=D(KQ1)
      J=J+KDD
      K=J
  530 TPD=HH*TPD
      KDC=KDC+1
      Y(K)=Y(K)+GAM(KQQ+1,KDC)*TPD
      K=K-1
      IF (KDC.LT.KDD) GO TO 530
C     END OF CORRECT
C
      IF (EPS) 540,550,560
  540 EPS=EP(I)
      IF (EPS.NE.0.) GO TO 560
  550 IF (HMAXA) 1190,780,1190
  560 TPS4=ABS(SNGL(D(KQQ+2)))
      TPS2=ABS(SNGL(D(KQQ)))
      TPS6=SNGL(HH)/EPS
C
      E=ABS(SNGL(GAS(KQQ+1))*TPS3*TPS6)
C     E GIVES  ABS((ESTIMATED ERROR)/EPS)
C
      LRND=1
C
C     LRND= 1  MEANS NO ROUND-OFF ERROR
C         = 0  MEANS SOME ROUND-OFF ERROR
```

```
C           =-1  MEANS EXTREME ROUND-OFF ERROR
C
      FRND=RNDC*ABS(SNGL(PT(KQQ+2)*D(1)))
C     CHECK TO SEE IF ROUND OFF ERROR IS DOMINANT
      IF ((TPS3+TPS4).GT.FRND) GO TO 570
      LRND=0
      IF ((PTS4*TPS2).LT.FRND) LRND=-1
C
  570 IF (E.LE.ERND) GO TO 590
      IF (E.LE.EMAX) GO TO 580
      EMAX=E
      KEMAX=I
  580 IF (LRND.LE.0) GO TO 590
      ERND=E
      IF (ERND.GT.ERRMX) LDOUB=0
  590 IF (LDOUB.LE.0) GO TO 780
      TPS1=ABS(SNGL(DD(KQQ)))
      TPS5=TPS1
      IF (KQQ-2) 600,610,620
  600 E2H=E*E2H
      IF (E2H.LT.P01) GO TO 780
      IF (SNGL(D(4)).LT.P3E1) GO TO 770
      LSTC=-1
      LSC=-5
      GO TO 770
  610 TPS1=TPS2
      IF (LSTC.NE.2) GO TO 620
      KQ(I)=3
      TPS2=0.
      TPS4=0.
      LRND=0
  620 E2H=TPS2+TPS3+TPS4
      E2H=ABS(SNGL(GAS(KQQ-1)*PT(KQQ+1))*E2H*TPS6)
C     E2H IS USED AS AN ESTIMATE OF WHAT THE VALUE OF E WOULD BE
C     IF H WERE DOUBLED. THE ESTIMATE IS CONSERVATIVELY LARGE.
      IF (E2H.GT.E2HMAX) E2HMAX=E2H
C
      IF (LRND) 630,640,660
C     EXTREME ROUND-OFF ERROR--REDUCE E2H
  630 K=(KBIT2/KQQ)-4
      IF (K.LE.3) GO TO 640
      IF (K.GT.KQMAX) K=KQMAX
      E2H=E2H/PT(K+1)
      GO TO 650
  640 E2H=AMIN1(E2H,E2H*P3E1*E2HFAC)
  650 E2H=E2H*P1
      TPS6=PTS4
      GO TO 670
C
  660 E2H=E2H*E2HFAC
      TPS6=FLOAT(KQQ+2)
C     TEST TO SEE IF DIFFERENCES DECREASE MORE RAPIDLY THAN NECESSARY
C
  670 IF (TPS5.LT.(TPS3*TPS6)) GO TO 680
      IF (TPS2.LE.(TPS4*TPS6)) GO TO 760
C     THEY DO   INCREASE KQ(I)
      IF (KQQ.NE.KQMAX) KQ(I)=KQ1
      GO TO 760
C
C     TEST TO SEE IF DIFFERENCES DECREASE TOO SLOWLY
```

```
  680 TPS6=TPS6*P25
      IF ((TPS1.GT.(TPS3*TPS6)).OR.(TPS2.GT.(TPS4*TPS6))) GO TO 760
C     THEY DO
      IF (LSTC.LE.0) GO TO 750
      IF (E2H.LT.P01) GO TO 750
      IF (LSC-LSTC) 690,750,770
  690 IF (KSTEP-4) 750,700,710
  700 KQ1=LSTC
  710 LSC=KQ1
C     END OF ONE DERIVATIVE EVALUATION PER STEP
      GO TO 770
C
C     AFTER HALVING H. REDUCE KQ(I) IF A DISCONTINUITY HAS OCCURRED.
  720 IF (LDOUB.EQ.(-2)) GO TO 340
      DT(KQQ+1,I)=D(KQQ+1)
      IF (LDOUB.EQ.(-1)) DT(KQQ+1,I)=D(KQQ+2)
      K=KQQ
  730 IF (K.EQ.1) GO TO 740
      IF ((DABS(D(K-1)).GT.(PT(2)*DABS(D(K+1)))).OR.
     1  (DABS(D(K)).GT.(PT(2)*DABS(D(K+2))))) GO TO 740
      K=K-1
      GO TO 730
  740 IF ((K+K).GE.KQQ) GO TO 780
      LDOUB=-4
      E2H=0.
      KQQ=K+1
C
C
C     DIFFERENCES DECREASE TOO SLOWLY REDUCE KQ(I).
  750 KQ(I)=KQQ-1
      IF (KQQ.EQ.2) DT(3,I)=0.D0
  760 IF (E2H.LT.P01) GO TO 780
  770 LDOUB=2
  780 CONTINUE
C
C     IF THE OUTPUT OPTION IS ELIMINATED, REMOVE THE 6 FOLLOWING CARDS.
      IF (NEQ.GT.0) GO TO 790
      IO2=MAXO(1,(KQQ-1))
      IO3=IO2+3
      WRITE (6,5021) I,KQQ,LRND,LDOUB,E,E2H,EPS,
     1  (IO1,D(IO1),IO1=IO2,IO3)
 5021 FORMAT (1H I2,I4,2I5,1PE13.3,2E11.3,4(3H  (,I2,1H),D10.3))
C
  790 CONTINUE
C
C     END OF LOOP FOR CORRECTING, ESTIMATING THE ERROR, ETC.
C
C
C     IF THE INTERPOLATION CAPABILITY IS ELIMINATED REMOVE THE
C     FOLLOWING CARD.
      IF (IFL.LT.0) GO TO 1250
C     TEST FOR HALVING H
      IF (LDOUB) 80C,950,870
  800 LDOUB=LDOUB+1
      IF (LDOUB+1) 810,870,820
  810 IF (LDOUB.EQ.(-2)) GO TO 820
C     ORDER IN AT LEAST ONE COMPONENT HAS BEEN GREATLY REDUCED
      LDOUB=0
      GO TO 220
  820 DO 860 I=1,NE
```

```
      KQQ=KQ(I)
      TP=DT(KQQ+1,I)
      IF (KQQ.LE.3) GO TO 860
      IF (LDOUB.NE.0) GO TO 840
      DO 830 K=3,KQQ
C     SECOND MODIFICATION OF DIFFERENCE TABLE AFTER HALVING H
  830 DT(K,I)=DT(K,I)+ETA(KQQ-1,K-2)*TP
      GO TO 860
  840 DO 850 K=2,KQQ
C     FIRST MODIFICATION OF DIFFERENCE TABLE AFTER HALVING H
  850 DT(K,I)=DT(K,I)+ETA(K-1,KQQ-1)*TP
  860 CONTINUE
      IFL=0
      GO TO 240
C
  870 IFL=2
      IF (LSTC.LE.0) GO TO 300
      IF (2-LSTC) 880,900,940
  880 LSTC=LSTC-1
      IF (LSTC.EQ.3) GO TO 890
      IF (LSC) 920,960,920
  890 IFL=1
      GO TO 300
  900 IF (LSC-2) 910,930,920
  910 LSTC=0
  920 LDOUB=2
      GO TO 60
  930 LSTC=1
      LSC=0
      GO TO 60
  940 IF (LSC) 300,60,300
C
C     HALVE H
  950 HH=FAC(2)*HH
      IF (LSTC.LT.2) GO TO 990
      ERND=P25*ERND
C     IN LOOP TO FIND A NEW INITIAL STEPSIZE
      IF (ERND.GE.P1) GO TO 950
      LSTC=4
  960 LSC=4
      DO 970 I=1,NE
  970 KQ(I)=1
      IF (LSTC-3) 890,890,1170
C
C     ENTRY AFTER IFLAG=7
  980 IF (LDOUB.EQ.0) GO TO 990
      LSC=1
      LSTC=1
      GO TO 140
C     TEST TO SEE IF H IS TOO SMALL FOR HALVING
  990 IF (ABS(SNGL(HH)).GE.HMINA) GO TO 1040
      IF ( IFL.EQ.7) GO TO 1010
 1000 IFL=7
      GO TO 1020
C
 1010 HH=HH+HH
      IFL=2
 1020 H=HH
      GO TO 310
C
```

```
C
C      ERROR CRITERIA PERMIT DOUBLING
 1030 HH=HH+HH
      IF (LSTC.EQ.1) GO TO 1050
      LSC=-3
 1040 LSTC=-1
C
C      CHANGE THE STEPSIZE
 1050 DO 1160 I=1,NE
      KQQ=KQ(I)
      IF (KQQ.NE.1) GO TO 1070
      DT(6,I)=0.D0
      D(3)=DT(3,I)*PT(2)
      IF (D(3).GT.PT(3)) LSC=-6
      IF (LDOUB.NE.0) GO TO 1060
      KQM=8
      IF (D(3).GE.PT(5)) DT(7,I)=DT(7,I)*PT(2)
      D(3)=D(3)/PT(3)
 1060 DT(3,I)=D(3)
      GO TO 1160
C
C      BEGINNING OF LOOP FOR CHANGING DIFFERENCE TABLE TO
C      CORRESPOND TO NEW VALUE OF H
 1070 DO 1080 K=1,KQQ
      D(K)=DT(K,I)/PT(K)
 1080 IF (LDOUB.EQ.0) D(K)=D(K)/PT(K)
      KQQ2=KQQ-2
      IF (KQQ2) 1160,1140,1090
 1090 DO 1130 J=1,KQQ2
      IF (LDOUB.NE.0) GO TO 1110
C
C      HALVE
      K=KQQ
 1100 D(K-1)=D(K-1)+D(K)
      K=K-1
      IF (K+J-KQQ) 1130,1130,1100
C
C      DOUBLE
 1110 DO 1120 K=J,KQQ2
 1120 D(K+1)=D(K+1)-D(K+2)
 1130 CONTINUE
C
 1140 DO 1150 K=2,KQQ
      IF (LDOUB.NE.0) D(K)=D(K)*PT(K)
      DT(K,I)=D(K)*PT(K)
 1150 CONTINUE
C      DIFFERENCE TABLE NOW CORRESPONDS TO NEW VALUE OF H
C
 1160 CONTINUE
 1170 H=HH
      IF (LDOUB.NE.0) GO TO 50
      LFD=1
      IF (LSTC.GE.0) GO TO 220
      LDOUB=-3
      LSC=LSTC-KQM
      GO TO 220
C      END OF CHANGING STEPSIZE
C
C
C      IF THE GSTOP FEATURE IS ELIMINATED, REMOVE THE C  IN COLUMN ONE
```

```
C     OF THE 2 FOLLOWING CARDS
C1180 IF (7-IFL) 1181,980,220
C1181 IF (IFL-8) 60,1200,60
C     AND THEN REMOVE THE 2 FOLLOWING CARDS.
 1180 K=IFL-5
      GO TO (220,980,1200,1570,1570,1720,1720,60,1480,1450,1630,1570), K
C
C     ILLEGAL VALUE OF PARAMETER   INTEGRATION CAN NOT PROCEED
 1190 IFL=8
      GO TO 310
 1200 WRITE (6,4000)
 4000 FORMAT (26H0IFLAG=8 IN CALL TO DVDQ1.)
      STOP
C
C
 1210 IF (T-TFINAL) 200,1190,200
C
C     IF ONE DOES NOT WANT THE INTERPOLATICN FEATURE, REMOVE ALL CARDS
C     BELOW THIS POINT (EXCEPT FOR THE END STATEMENT), AND ADD THE
C     FIVE FOLLOWING STATEMENTS.
C1220 IFL=4
C     IF (TPD1.GT.TPD) GO TO 1280
C     GO TO 310
C1280 IFL=3
C     GO TO 310
C
 1220 IFL=4
      IF (KSTEP.NE.0) GO TO 1270
      TPD2=TPD
C     ESTIMATE ERROR WHEN EXTRAPOLATION FROM INITIAL POINT IS REQUESTED
 1230 HH=HH*TPD1*.75D0
C
C     IF THE GSTOP FEATURE IS ELIMINATED, REMOVE THE FOLLOWING CARD.
      IFLS=IFL
      IFL=-1
      GO TO 230
C
C     IF THE GSTOP FEATURE IS ELIMINATED, REMOVE THE 4 FOLLOWING CARDS.
 1240 IF ((LGSD.EQ.0).OR.(IFLS.NE.4)) GO TO 20
      LGSE=-1
      TPD=FAC(1)
      GO TO 1820
 1250 HH=H
      IF (EMAX.LT.P01) GO TO 1260
C     ERROR IS TOO LARGE, REDUCE H AND REPEAT THE FIRST STEP
      IF (TPD1.LT.0.D0) GO TO 1190
      LDOUB=1
      ERND=FAC(1)/TPD1
      ERND=ERND*ERND*P25
      GO TO 950
C
C     IF THE GSTOP FEATURE IS ELIMINATED, REMOVE THE C IN COLUMN ONE
C     OF THE FOLLOWING CARD
C1260 IFL=4
C     AND THEN REMOVE THE 2 FOLLOWING CARDS.
 1260 IFL=IFLS
      IF (IFL.NE.4) GO TO 1790
      TPD=TPD2
      IFLAG=3
 1270 IF (TPD1.GT.TPD) GO TO 1280
```

```
      T=TFINAL
      TPD=TPD1
      GO TO 1290
 1280 T=TOUT
      IFL=3
 1290 IF ((TPD.EQ.0.D0).AND.(IFLAG.LE.2)) GO TO 310
C
C     INTERPOLATE FOR OUTPUT
 1300 TP=TPD
      D(2)=TP
      KQQ2=0
      KDC=0
      D(1)=PT(1)
      DD(1)=PT(1)
      DO 1310 K=2,KQM
      DD(1)=DD(1)+PT(1)
      TP=TP+PT(1)
 1310 D(K+1)=(D(K)*TP)/DD(1)
      GO TO 1350
C
C     COMPUTE THE INTERPOLATING INTEGRATION COEFFICIENTS
 1320 KQQ2=1
      L=KQM-KDC
      KDC=KDC+1
 1330 IF (L.LE.0) GO TO 1350
      TP=0.D0
      K=L
      J=L+KDC
 1340 JS=J-K
      TP=TP+GAS(K)*D(JS+1)
      K=K-1
      IF (K.GT.0) GO TO 1340
      D(J)=TP
C
C     D(J) IS THE INTEGRATION COEFFICIENT FOR THE INTERPOLATION WHICH
C     CORRESPONDS TO GAM(J-KDC,KDC).
C
      L=L-1
      GO TO 1330
C     END OF COMPUTING INTEGRATION COEFFICIENTS
C
C     PERFORM THE PARTIAL STEP INTEGRATION
 1350 J=0
      DO 1420 I=1,NE
      IF (KDS.LE.0) KDD=IABS(KD(I))
      IF (KDC.GT.KDD) GO TO 1410
      TP=0.D0
      KQQ=KQ(I)+KQQ2
 1360 L=KQQ-KDC
      IF (L.LE.0) GO TO 1370
      TP=TP+D(KQQ)*DT(L,I)
      KQQ=KQQ-1
      IF (KQQ) 1390,1390,1360
 1370 K=J+KDD
      L=KDC
 1380 L=L-1
      IF (L.EQ.0) GO TO 1400
      TP=TP*HH+YN(K)*FAC(L)*TPD
      K=K-1
      GO TO 1380
 1390 F(I)=TP
      GO TO 1420
 1400 Y(K)=YN(K)+HH*TP
 1410 J=J+KDD
 1420 CONTINUE
      IF (KDC.NE.KDMAX) GO TO 1320
C     END OF PARTIAL STEP INTEGRATION
C
C     IF THE GSTOP FEATURE IS ELIMINATED, REMOVE THE C IN COLUMN ONE
C     OF THE FOLLOWING CARD
C     GO TO 310
C     ALL STATEMENTS BELOW THIS POINT SHOULD THEN BE REMOVED (EXCEPT
C     FOR THE END STATEMENT)
      IF (LGSE) 1800,310,1810
C
C
C     SECTION FOR COMPUTING GSTOPS
C
      ENTRY DVDQG(NG,NSTOP,G,GT)
C
C     VARIABLES IN THE CALLING SEQUENCE HAVE THE FOLLOWING TYPES.
      INTEGER NG,NSTOP
      DOUBLE PRECISION G(1),GT(1)
C
C     A GSTOP IS DEFINED AS A RETURN WHICH IS MADE TO THE USER WHEN A
C     USER SPECIFIED FUNCTION G PASSES THROUGH ZERO. THE USER MAY
C     SPECIFY ANY NUMBER OF FUNCTIONS G OF TWO TYPES. ZEROS OF THE FIRST
C     TYPE ARE LOCATED WITHOUT REQUIRING A DERIVATIVE EVALUATION
C     BEYOND THE ZERO. THIS TYPE OF GSTOP REQUIRES THAT G BE EVALUATED
C     BEFORE EACH DERIVATIVE EVALUATION. ZEROS OF THE SECOND TYPE ARE
C     LOCATED USING INTERPOLATION, WHICH IS MORE ACCURATE THAN THE
C     EXTRAPOLATION USED IN THE PRECEDING CASE AND ONLY REQUIRES ONE
C     EVALUATION OF G PER STEP. THUS ONE SHOULD USE THE SECOND TYPE OF
C     GSTOP IF POSSIBLE. USERS NOT USING THE GSTOP FEATURE NEED READ
C     NO FURTHER.
C
C     DVDQG IS USED AS A SET UP CALL TO INDICATE A CHANGE IN THE NUMBER
C     OR TYPES OF GSTOPS. DVDQG SHOULD BE CALLED JUST BEFORE OR JUST
C     AFTER CALLING DVDQ IF
C     1. ONE WANTS TO TEST FOR GSTOPS BEGINNING WITH THE FIRST STEP.
C     2. A JOB IS BEING RUN AFTER ANOTHER JOB THAT USES THE GSTOP
C        FEATURE. DVDQG MUST BE CALLED EVEN IF ALL THE VARIABLES IN
C        THE NEW JOB ARE THE SAME.
C     IN ADDITION DVDQG MAY BE CALLED AT ANY TIME IN THE INTEGRATION
C     TO CHANGE THE NUMBER OR TYPE OF GSTOPS.
C
C     THE USAGE OF THE VARIABLES IS GIVEN BELOW.
C
C     NG=   THE NUMBER OF COMPONENTS IN G TO BE EXAMINED FOR A ZERO.
C           IF DVDQG IS CALLED AFTER THE FIRST STEP OF THE INTEGRATION,
C           THEN G IS EVALUATED FOR THE FIRST TIME AT THE END OF THE
C           NEXT STEP AND THUS A GSTOP IS NOT DETECTED IF G CHANGES
C           SIGN ON THE CURRENT STEP. IF IT IS IMPORTANT THAT G BE
C           EVALUATED IMMEDIATELY SET NG EQUAL TO THE NEGATIVE OF THE
C           NUMBER CF COMPONENTS TO BE TESTED FOR A ZERO. SETTING NG
C           LESS THAN ZERO WHEN CALLING DVDQG BEFORE THE FIRST STEP IS
C           NOT NECESSARY AND IS LIABLE TO BE DISASTEROUS. IF DVDQG IS
C           CALLED DURING THE INTEGRATION THE FOLLOWING STATEMENT SHOULD
C           BE A GO TO (THE COMPUTED GO TO FCLLOWING THE CALL TO DVDQ1).
C
```

```
C      NSTOP=THE NUMBER OF COMPONENTS OF G THAT MUST BE EXAMINED FOR
C            A ZERO BEFORE COMPUTING THE DERIVATIVES (FIRST TYPE OF
C            GSTOP). IF NSTOP.LT.0 OR NSTOP.GT.ABS(NG) IFLAG IS SET
C            EQUAL 8 AND AN IMMEDIATE RETURN IS MADE. IF NSTOP.GT.0,
C            G(1),G(2),...,G(NSTOP) ARE EXAMINED FOR A ZERO BEFORE EACH
C            DERIVATIVE EVALUATION, THE REMAINING COMPONENTS (IF ANY)
C            ARE EXAMINED AT THE END OF EACH STEP. WHEN A GSTOP IS FOUND
C            THE SUBROUTINE SETS NSTOP EQUAL TO THE INDEX OF THE
C            COMPONENT OF G RESPONSIBLE FOR THE STOP.
C
C      G=    A VECTOR CONTAINING THE CURRENT VALUES OF THE FUNCTIONS
C            WHOSE ZEROS ARE TO BE DETERMINED.
C
C      GT=   A VECTOR WITH THE SAME DIMENSION AS G USED BY THE
C            SUBROUTINE FOR TEMPORARY STORAGE.
C
C      RETURNS FROM CALLING DVDQ1 WITH IFLAG.GT.8 SHOULD BE INTERPETED
C      AS FOLLOWS. (WE USE NSTOPI TO DENOTE THE INITIAL VALUE OF NSTOP.)
C      IFLAG
C      = 9 COMPUTE G(NSTOPI+1),...,G(ABS(NG)) (THE COMPONENTS OF G WITH
C          ZEROS TO BE LOCATED USING INTERPOLATION). THEN CALL DVDQ1.
C          NO RETURN IS MADE WITH IFLAG=9 IF NSTOPI=ABS(NG).
C      =10 COMPUTE G(1),G(2),...,G(NSTOPI) (THE COMPONENTS OF G WITH
C          ZEROS TO BE LOCATED USING EXTRAPOLATION). THEN CALL DVDQ1.
C          NO RETURN IS MADE WITH IFLAG=10 IF NSTOPI=0.
C      =11 G(NSTOP) IS APPROXIMATELY ZERO. IF THERE ARE NO
C          DISCONTINUITIES SIMPLY CALL DVDQ1 TO CONTINUE THE INTEGRATION.
C      =12 G(NSTOP) CHANGES SIGN, BUT THERE IS DIFFICULTY IN CONVERGING
C          TO A ZERO. THE USER MAY WISH TO MAKE A SPECIAL CHECK TO BE
C          CERTAIN THAT EVERYTHING IS ALL RIGHT. TO CONTINUE THE
C          INTEGRATION CALL DVDQ1.
C
       DOUBLE PRECISION RG
       DOUBLE PRECISION GI
       DIMENSION GI(2),RG(3)
C
C      INITIALIZE FOR GSTOPS
       NGA=IABS(NG)
       LGSS=-NGA
       LGSD=0
       LGSE=0
       IFLG=-20
       IF (NG) 1425,315,315
 1425  IFLG=-IFL
       IFLG=-IFL
 1430  LGSD=NSTOP
       IF (LGSD) 1190,1450,1440
 1440  IFL=15
       GO TO 1470
C      ENTRY WITH IFL=15
 1450  LGSS=0
       IF (LGSD-NGA) 1460,1480,1190
 1460  LGSS=LGSD+1
       IFL=14
 1470  IFLAG=IFL-5
       GO TO 315
C      ENTRY WITH IFL=14
 1480  DO 1490 I=1,NGA
 1490  GT(I)=G(I)
       GO TO 1730
```

```
C     END OF INITIALIZATION FOR GSTOPS
C
C     ENTRY TO EVALUATE G AT THE END OF THE STEP
 1500 LGSE=1
 1510 IGK=LGSS
      IFLG=0
      IFL=9
      GO TO 310
C     ENTRY TO EVALUATE G BEFORE EVALUATING THE DERIVATIVES
 1520 IFLG=IFL
      IFL=10
 1530 IFLAG=10
      IGKM=LGSD
 1540 IGK=1
 1550 GO TO 315
 1560 IGK=IGK+1
      IF (IGK.GT.IGKM) GO TO 1650
C     ENTRY WITH IFL=9,10, AND 17
C     TEST FOR G CHANGING SIGN
 1570 IF (G(IGK)*GT(IGK)) 1600,1580,1590
 1580 IF (GT(IGK).NE.0.D0) GO TO 1600
      IF (TL.EQ.TG) GO TO 1560
 1590 IF (LGSE.GT.0) GT(IGK)=G(IGK)
      GO TO 1560
C     G CHANGES SIGN -- PREPARE FOR ITERATION TO FIND ZERO
 1600 NSTOP=IGK
      NSTOPI=IGK
      IFLGS=IFL
C     COMPUTE INITIAL VALUE FOR RG (=RATIO OF PARTIAL STEPSIZE WHERE
C     G IS KNOWN/THE INTEGRATION STEPSIZE)
      IF (IFLG.EQ.0) GO TO 1610
      RG(3)=FAC(1)
      RG(2)=0.D0
      IF ((IFLG.EQ.2).AND.(IGK.LT.LGSS)) RG(2)=FAC(1)
      GO TO 1620
 1610 RG(3)=0.D0
      RG(2)=-FAC(1)
 1620 IF (LGSE.LT.0) RG(3)=TPD
      LGSE=-3
      GI(2)=GT(IGK)
      EPSGS=RND
      IFL=16
      K=1
      GO TO 1640
C     END OF PREPARATION TO BEGIN THE ITERATION
C
C     ENTRY WITH IFL=16
C     ITERATE TO FIND GSTOP
 1630 K=1
      IF ((GI(2)*G(IGK)).GT.0.D0) K=2
      IF (DABS(GI(K)).GT.DABS(G(IGK))) GO TO 1640
C     CONVERGENCE PROBLEMS
      LGSE=LGSE-1
      IF (LGSE.EQ.(-5)) EPSGS=PTS1
      EPSGS=EPSGS*PTS4
 1640 GI(K)=G(IGK)
      RG(K)=RG(3)
C     SECANT ITERATION (GIVES NEW PARTIAL STEPSIZE/H)
      TPD=RG(1)-(GI(1)*(RG(2)-RG(1)))/(GI(2)-GI(1))
      T=TL+TPD*HH
```

```
C     TEST FOR CONVERGENCE OF ITERATION
      IF (DABS(TPD-RG(3)).LE.EPSGS) GO TO 1560
      RG(3)=TPD
      GO TO 1300
 1650 IF (10-IFL) 1660,1700,100
 1660 IF (IGKM.NE.NGA) GO TO 1710
      IF (LGSE.GT.(-3)) GO TO 1690
      IF (LSTC.NE.4) GO TO 1670
C     ESTIMATE ERROR -- GSTOP IS THE RESULT OF EXTRAPOLATING FROM
C     THE INITIAL POINT
      TPD1=TPD
      RG(3)=TPD
      GO TO 1230
 1670 IFL=11
      IF (LGSE.LT.(-4)) IFL=12
 1680 IFLAG=IFL
C     TEST TO SEE IF GSTOP IS PRECEDED BY ANOTHER STOP
      IF (((T-TOUT)*HH.LE.0.D0).AND.((T-TFINAL)*HH.LE.0.D0)) GO TO 1300
C     IT IS
      RG(3)=TPD
      IFLS=IFL
      GO TO 200
 1690 LGSE=1
      IFL=IFLG
      IF (IFL.LT.0) GO TO 20
 1700 IGKM=NGA
      IFL=IFLG
      GO TO 310
 1710 IFL=17
      IFLAG=9
      IGKM=NGA
      GO TO 315
C     ENTRY WITH IFL=11 AND 12
C     SET PARAMETERS TO INDICATE A GSTOP HAS BEEN FOUND
 1720 GT(NSTOPI)=0.D0
 1730 LGSE=1
      IGKM=NGA
      TG=TL
      IF (IFLG) 1740,1760,1770
 1740 IF (IFL.LT.13) GO TO 1750
      IF (IFLG.EQ.(-20)) GO TO 100
      IFL=-IFLG
      GO TO 310
 1750 HH=H
      GO TO 200
 1760 TPD=0.D0
      T=TL
      LGSE=-2
      GO TO 1300
 1770 IF (IFLG-3) 220,200,200
 1780 IF (LGSE.EQ.(-1)) GO TO 1790
      LGSE=-1
      GO TO 1220
 1790 TPD=RG(3)
      T=TL+TPD*HH
      IF (LGSE.NE.(-1)) GO TO 1670
      IFL=IFLS
      LGSE=-3
      GO TO 1680
 1800 IF (LGSE+2) 1550,1500,310
 1810 IF (TPD.LE.0.D0) GO TO 310
      LGSE=-2
 1820 IFLG=IFL
      IFL=17
      IFLAG=9
      IF (LGSD .GT. 0) GO TO 1530
      GO TO 1540
C     END OF SECTION FOR COMPUTING GSTOPS
C
C
C     IN SOME APPLICATIONS, FOR EXAMPLE MULTIPLE QUADRATURE, MORE THAN
C     ONE INTEGRATION SUBROUTINE IS REQUIRED. THIS IS NOT NECESSARY IF
C     ALL OF THE VARIABLES ASSOCIATED WITH ONE INTEGRATION ARE SAVED
C     OUTSIDE OF THE INTEGRATOR WHILE DOING OTHER INTEGRATIONS, AND
C     THEN RESTORING THEM WHEN NEEDED. THESE VARIABLES CAN BE RESTORED
C     BY CALLING AN ENTRY WHICH CONTAINS ALL OF THE VARIABLES IN THE
C     CALLING SEQUENCE AND ALL OF THE VARIABLES THAT MUST BE SAVED
C     WHENEVER CHANGING TO A DIFFERENT INTEGRATION. (THIS ENTRY MUST
C     BE ADDED BY THE USER AND SHOULD BE FOLLOWED BY A RETURN STATEMENT.
C     AFTER CALLING THIS ENTRY EITHER DVDQ OR DVDQ1 SHOULD BE CALLED
C     DEPENDING ON WHETHER THE INTEGRATION IS BEING STARTED OR NOT.)
C     THE VARIABLES WHICH MUST BE SAVED ARE'
C     NE,NV,KDS,KDMAX,KSOUT,LDOUB,LFD,LSC,LSTC,IFL,IFLS,KQM   (INTEGERS)
C     ERND,QDEC,ERRMX,E2HAVE,E2HFAC,E2HMAX,RNDC   (REAL)
C     HH,TOUT,TL   (DOUBLE PRECISION)
C     IF THE GSTOP FEATURE IS USED, THE EVALUATION OF G REQUIRES AN
C     INTEGRATION, AND THIS INTEGRATION MAY RESULT IN ANOTHER GSTOP,
C     THEN SOME ADDITIONAL VARIABLES MUST BE SAVED.
C     IN MANY APPLICATIONS NE(=NEQ),NV(=SUM OF ORDERS OF THE
C     DIFFERENTIAL EQUATIONS),KDS(=KD),AND KDMAX(=MAXIMUM ORDER OF ANY
C     DIFFERENTIAL EQUATION) WILL BE THE SAME FOR EVERY INTEGRATION,
C     AND HENCE NEED NOT BE SAVED.
C
C     INSTRUCTIONS FOR MAKING CERTAIN CHANGES IN THIS SUBROUTINE ARE
C     GIVEN THROUGHOUT THE LISTING. TO FIND THESE INSTRUCTIONS, SEE
C     BELOW.
C
C     TO ELIMINATE THE GSTOP CAPABILITY, SEE JUST BELOW CARDS SEQUENCED
C     541,703,745,791,1201,1234,1239,1253, AND 1330.
C     THIS MAKES THE SUBROUTINE SHORTER AND REDUCES OVERHEAD A LITTLE.
C
C     TO REMOVE THE INTERPOLATION CAPABILITY, SEE JUST BELOW CARDS
C     SEQUENCED 1070 AND 1219.
C     THE GSTOP FEATURE MUST ALSO BE ELIMINATED SINCE IT REQUIRES THE
C     INTERPOLATION CAPABILITY. IF OUTPUT POINTS ARE NOT HIT EXACTLY
C     (THEY ARE HIT EXACTLY IF HMAXA.LE.ABS(DELT), AND INITIAL H=
C     DELT*(2**(-K)), K=0,1,2...), THEN IFLAG=3 ON THE FIRST STEP THAT
C     (T-TOUT)*H.GT.0 (SEE THE USAGE OF DELT). IFLAG IS SET EQUAL TO 4
C     ON THE LAST STEP THAT (T-TFINAL)*H.LE.0.
C
C     THE OUTPUT OPTION GIVES OUTPUT OF VARIABLES USED IN THE
C     INTEGRATION ON EVERY STEP THAT NEQ.LE.0. (WHICH OF COURSE MUST
C     BE SET AFTER THE INITIAL CALL TO THE INTEGRATOR)  TO ELIMINATE
C     THIS OPTION, SEE JUST BELOW CARDS SEQUENCED 834 AND 1057.
C
C     THE CHECK OPTION WHEN ADDED TO THE OUTPUT OPTION OUTPUTS EVERY
C     VARIABLE IN THE CALLING SEQUENCE JUST AFTER ENTERING AND JUST
C     BEFORE LEAVING THE INTEGRATOR WHEN NEQ=0. THIS OUTPUT IS
C     SOMETIMES USEFUL IN DEBUGGING A PROGRAM. TO INCLUDE THIS OPTION
C SEE JUST BELOW CARDS SEQUENCED 613 AND 802.
C
      END
```