STINT

STiff ORDINARY DIFFERENTIAL EQUATIONS INTegrator

PROGRAM AND USER MANUAL

by

Zdenek Picel
with
Theodore A. Bickart
and
Joel M. Tendler

issued by

Electrical and Computer Engineering Department
Syracuse University
Syracuse, New York 13210
USA

Technical Report TR-76-12

and

Electrical Engineering Department
Delft University of Technology
Delft
THE NETHERLANDS

December 1976

C O N T E N T S

Acknowledgement i
Availability i
Abstract 1

1. Introduction 2

2. Problem of "Stiffness" 4

3.1 The Computational Method 11
3.2 The Integration Algorithm 19
- Solution Segment 21
- Error Control Segment 26
- Step-size and Order Control Segment 30
- Initialization Segment 32

4. Program Code Realization 33

5. Usage of the Program 41
5.1 General Explanations 41
5.2 Examples of Use 46
Problem I 46
Problem II 55

6. Program Modifications 63

7. Testing 69

Bibliography 71

Figure 1 : Typical Integration Curves 5
2 : Block Diagram of Algorithm 20

Table 1 : Cyclic methods: Values of coefficients 16
2 : Comparison of \alpha-values 17
3 : Coefficients for the y_{ml+1}^{(0)} predictor . 24
4 : Reciprocal Values of the deltas and gammas 29
5 : Values of Bound \eta on Step-size Increase 31
6 : The values of the predictor-coefficients 35

Typing errors : i l

23 logical error
24 III
25
27 II
28
29 II
30 logical error
33 II
34 IIII
35
36 IIII
38
39 IIIII
42, 43, 55", 63, 64
65, 70
69

ACKNOWLEDGEMENT

STINT is the successor program to DIFJMT which was written by Joel M. Tendler and based upon cyclic methods developed by him, as reported herein and in

> J. M. Tendler, "A stiffly stable integration process using cyclic composite methods," Ph.D. Dissertation, Syracuse University, Syracuse, NY, May 1973.

The cyclic methods and the incorporating integration algorithm upon which STINT is based, together with test results, are also described in

> J. M. Tendler, T. A. Bickart, and Z. Picel, "A stiffly stable integration process using cyclic composite methods," submitted for publication.

AVAILABILITY

The subroutine STINT together with the called subroutines DEC and SOL are available by sending a request, together with a blank tape, to

Professor Theodore A. Bickart
Electrical and Computer Engineering Department
Link Hall
Syracuse University
Syracuse, New York 13210

ABSTRACT

STINT is a program for the numerical solution of stiff systems of ordinary differential equations, when the initial values are given. It is particularly well suited for solution of stiff systems, wherein the Jacobian matrix has complex eigenvalues near the imaginary axis. The program is based on a set of cyclic composite multistep methods of orders 1 thru 7.

1. INTRODUCTION

The program STINT (STiff INTegrator) described herein is intended for obtaining—efficiently and reliably—the numerical solution of a system of stiff ordinary differential equations (ODE's) of the form

$$\frac{dy_i}{dt} = f_i(y_i(t),\ldots,y_\nu(t),t),\ i = 1,\ldots,\nu \tag{1.1}$$

with $y_i(t_0)$ given. The system of equations (1.1) is usually expressed in the more compact, vector, notation

$$\dot{\underset{\sim}{y}}(t) = \underset{\sim}{f}(\underset{\sim}{y}(t),t), \tag{1.2}$$

where $\dot{\underset{\sim}{y}}$, $\underset{\sim}{y}$, and $\underset{\sim}{f}$ are vectors of length ν. Given the function $\underset{\sim}{f}$ and the initial condition $\underset{\sim}{y}(t_0)$, STINT then computes a numerical solution of (1.2) on the interval $[t_0,t_f]$ by generating approximations $\underset{\sim}{y}_n$ to $\underset{\sim}{y}(t_n)$ on a mesh $t_0 = t_0 < t_1 <\ldots < t_F = t_f$. This mesh is selected automatically by the program, subject to being overruled by the user. The selection process, for reasons of efficiency, has as its objective maximization of the step-size $h_{n+1} = t_{n+1} - t_n$, while not introducing a single-step error greater than the user's pre-specified value.

Note: The code STINT is unsuitable for obtaining numerical solutions of systems of retarded differential equations, stochastic differential equations, and higher (than first) order differential equations. However, as is well known, the last-mentioned system can be always reduced to a set of first order ODE's, such as (1.2), to which STINT is applicable.

As test results attest to, STINT is suitable for numerical solution of a wide class of systems of stiff ODE's. However, it is particularly well suited for solution of stiff systems, wherein the Jacobian matrix $\underset{\sim}{f}_{\underset{\sim}{y}}(\underset{\sim}{y}(t),t)$ has complex eigenvalues near the imaginary axis.

The motivation and importance of the problem of "stiffness" is discussed in Section 2. The class of methods which form the basis for the numerical integration algorithm on which, in turn, the code STINT is based is described in Section 3. Section 4 is devoted to a discussion of the implementation of the basic integration algorithm into the program code STINT. The usage of this code for numerical solution of a typical "stiff" problem is explained in Section

5. This section, which is to be viewed as the user manual portion of this document, is printed on light blue paper. Suggestions, for converting the program to solve other systems and for modifying it in order to further improve its efficiency for a certain class of problems, are outline in Section 6. Finally, in Section 7, some performance data for the program are given. A complete listing of the program is provided in the Appendix.

A casual user, who is not familiar with the problems of "stiffness", is advised to read Section 2 before solving his problem with STINT. If he decides to use STINT and just regard it as a "black box", it is sufficient to read Section 5 where explanations for this type of use are given.

The person primarily concerned with installation of new programs, will find the information in Section 6 useful. Lastly, the numerical analyst who needs to understand the code and wants to make an attempt to improve it, will find the reading of Sections 2 thru 7, we hope, worth while.

2. PROBLEM OF "STIFFNESS"

Numerical solution of ordinary differential equations has been the subject of a great deal of research, particularly in the past twenty years. While considerable attention was given to the analysis as well as design of computational methods which form the basis of any numerical integration program, it is fair to say, that the most significant results were obtained in areas of reliability, convenience, and accuracy of these programs. These achievements are due, in part, to the availability of digital computers. Thus, a user, who is neither a numerical analyst nor a computer programmer, but who has access to a computer, can solve a typical problem easily and cheaply. There exist, however, problems for which finding numerical solutions takes an inordinate amount of time. A class of such problems is known as "stiff". A typical stiff problem has a solution which, in some regions, is very smooth and rigid—that is, stiff.

Consider the following scalar problem

$$\frac{dy}{dt} = f(y(t),t) = q(y(t) - p(t)) + \dot{p}(t), \tag{2.1}$$

with $y(t_0)$ given. It is easy to verify that the solution of (2.1) is

$$y(t) = (y(t_0) - p(t_0))\, e^{q(t-t_0)} + p(t). \tag{2.2}$$

Note: $p(t)$ is the particular solution of (2.1).

If $q << 0$, the first part of the solution (2.2)—the transient part—will soon become insignificant when compared to the second part—the steady-state part. The typical integral curves i.e., solutions of (2.1), for different values of t_0 and $y(t_0)$, are shown in Figure 1. A common feature of all these curves is that after some very rapid variation, they all fan-in, i.e., converge, to a single solution curve with which they become almost identical. When the particular solution is slowly varying, an additional property, that of inherent stability of the differential equation is observed. To illustrate this property consider the propagation behavior of small errors introduced in the initial values. For inherently stable systems, these errors will have negligible effect as far as the exact (analytic) particular solution is concerned. This extremely good behavior is, however, of no consequence, when the numerical integration process is considered.

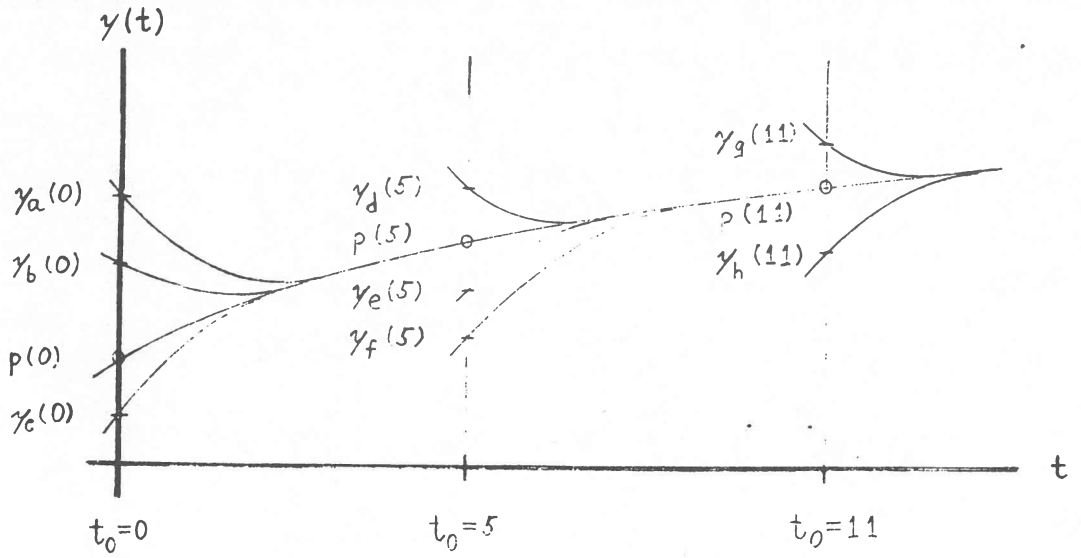


Figure 1. Typical Integral Curves of (2.1) for Different Values of t_0 and $y(t_0)$.

Since the exact (analytic) solution of many problems is often not known or cannot be calculated, the solution to a different but, hopefully, closely related problem which can be calculated is being sought. The replacement of the analytically unsolvable problem by a related, solvable one and its subsequent solution constitutes the numerical integration process. The replacement of the original problem by a new one is usually accomplished by invoking the principal of discretization. This principal has lead to the introduction of many numerical integration methods. Take, for example, the most simple one, the forward Euler method. When applied to a scalar equation, such as (2.1), it is given by

$$y_{n+1} = y_n + h\dot{y}_n, \tag{2.3}$$

where $\dot{y}_n = f(y_n,t_n)$ and $t_{n+1} = t_n + h$. The local truncation error*, i.e., the amount by which the exact solution of (2.1) does not satisfy (2.3), can be expressed as

$$\tau = \frac{1}{2}h^2\ddot{y}(t_n) + O(h^3). \tag{2.4}$$

Neglecting terms of order h^3, the error-per-step ε (specified by the user) will not be exceeded during this step, if the step-size h is chosen such that

---

* The names herein applied to the various error terms are those used in L.F. Shampine and M. K. Gordon, "Local error and variable order Adams codes," *Appl. Math. and Comp* **1** (1975), 47-66.

$$\left|\frac{1}{2}h^2 \ddot{y}(t_n)\right| \leq \varepsilon. \tag{2.5}$$

For (2.1), the value of $\ddot{y}(t_n)$ is

$$\ddot{y}(t_n) = q^2\left|y(t_0) - p(t_0)\right| e^{q(t_n - t_0)} + \ddot{p}(t_n). \tag{2.6}$$

Bearing in mind that $q << 0$ when the step occurs in the region where $(t_n - t_0)$ is small, $\ddot{y}(t_n) \approx q^2\left|y(t_0) - p(t_0)\right|$; thus, (2.5) will be (essentially) satisfied if

$$h \leq \left(\frac{2\varepsilon}{q^2\left|y(t_0) - p(t_0)\right|}\right)^{\frac{1}{2}}. \tag{2.7}$$

On the other hand, in the region, where $(t_n - t_0)$ is large, the transient term $\left|y(t_0) - p(t_0)\right| e^{q(t_n - t_0)}$ is insignificant. So, validity of (2.5) dictates that the step-size should satisfy

$$h \leq \left(\frac{2\varepsilon}{\left|\ddot{p}(t_n)\right|}\right)^{\frac{1}{2}}. \tag{2.8}$$

Since, by assumption, the particular solution $p(t)$ is slowly varying, the term $\left|\ddot{p}(t_n)\right|$ is small and the bound on h is large. It then follows, that as far as accuracy (which here means boundedness of the local truncation error by ε) is concerned, to resolve the rapidly decaying solution, a very small step-size is needed (as the bound on h, by (2.7), is proportional to $|1/q|$, a small number), but once the contribution of this solution component becomes insignificant, the step-size can become large and <u>independent of</u> $|q|$.

In general, the numerical solution of a problem such as (2.1) is obtained by repeated application of a method, such as (2.3). Even if one uses infinite precision arithmetic, only rarely will the calculated values y_n agree with the exact solution values $y(t_n)$. The question then arises, how the errors which are due to discretization and finite precision propagate through the step-by-step sequence of computations. Since the latter source of errors, although completely deterministic, is extremely complex to analyze (they are machine dependent, effected by the sequence of arithmetic operations, etc.), we turn our attention entirely to the former.

Let us suppose that an isolated error quantity δ_n, was introduced during the n-th step of our numerical integration process. Since this error will propagate into the (n+i)-st step, the difference equation (2.3) when applied to (2.1) will become

$$y_{n+1} + \delta_{n+1} = y_n + \delta_n + h[q(y_n + \delta_n + p(t_n)) + \dot{p}(t_n)] \quad (2.9)$$

which, in view of (2.3), reduces to

$$\delta_{n+1} = \delta_n + hq \times \delta_n = \delta_n(1 + hq). \quad (2.10)$$

Now, we do not want the error to grow during the step-by-step propagation; in fact, it should diminish, i.e., $|\delta_{n+j}| < |\delta_{n+j-1}|$, $j = 1,2,\ldots$ It is easily seen from (2.10) that this will be the case if and only if

$$|1 + hq| < 1. \quad (2.11)$$

Remember that the error δ_n was introduced in no specific (transient or steady-state) region. Now, it will be that some errors will be present right from the outset of the integration process. So, in order to prevent these errors from growing, the inequality (2.11) has to be satisfied throughout the whole course of the solution process. The problem of keeping the errors from growing is a stability problem for a particular numerical integration method. While for most ODE's accuracy is the requirement—i.e., constraints (2.7) and (2.8) are dominant over (2.11)—for stiff problems, stability—i.e., constraint (2.11)—is paramount. This is, for instance, the case when $q << 0$.

Although the concept of stability was discussed in context of preventing errors from growing, we can equivalently demand that the numerical solution of a problem whose exact solution decays be also decaying. The simplest ODE having this property is the scalar equation

$$\dot{y}(t) = qy(t), \quad (2.12)$$

where q is, in general, complex with negative real part. To derive the stability constraint, apply a particular numerical integration method with fixed step-size h to (2.12) and demand that $y_n \to 0$ as $n \to \infty$. All values $\lambda = qh$ in the complex λ-plane which satisfy the stability constraint thus obtained, define the stability region of this method. Various measures of the stability regions were recently introduced. Their main value is in helping to categorize different numerical integration methods which, in turn, is useful in the selection of suitable methods for various classes of problems.

The stability as discussed above is admittedly a crude simplification of the error propagation problem. However, for the most part it gives a useful insight of the local behavior of the numerical method.

It should be noted that any correctly implemented numerical integration program detects numerical instability automatically and adjusts the step-size so the stability constraints are satisfied. However, since such adjustments are always downwards, the integration process is slowed down and becomes less efficient than if no adjustments were required. For stiff problems solved by the forward Euler method, this downward adjustment process, most active in periods with a slowly varying solution, means a problem cannot be run to completion in a reasonable number of steps.

One type of remedy by which to overcome the problems associated with stiffness comes from using methods which have large stability regions. The simplest such method is the backward Euler method, given as

$$y_{n+1} = y_n + h\dot{y}_{n+1}, \tag{2.13}$$

where $\dot{y}_{n+1} = f(y_{n+1}, t_{n+1})$. Since the local truncation error of (2.13) is the same as for (2.3), accuracy dictates that the step-sizes in the transient and the steady-state regions satisfy the inequalities (2.7) and (2.8), respectively. Studying the effects of stability leads to the stability constraint

$$\left|\frac{1}{1 - hq}\right| < 1. \tag{2.14}$$

Now, for any h positive and q complex with negative real part, it is easy to show that (2.14) is always satisfied. This means that the step-size used with the backward Euler method need be constrained only by accuracy requirements. Thus, when the rapidly varying transients become quiescent, a step-size limited by (2.8) not (2.7) may be used.

This important feature is bought for a price, however. This price is the implicitness of (2.13) which is, in general, typical of all methods with good stability properties. Although solving an implicit equation is admittedly a complex task, the manyfold gain in overall efficiency of the integration process makes paying this price worth while.

While the preceding discussion of the problem of stiffness has been confined to the scalar case, more often than not we are faced with a task of solving a vector equation, such as (1.2). How can the concept of stiffness be extended to cover this case is shown next.

The solution of (1.2) can be started by linearizing it around a particular solution $\underset{\sim}{g}(t)$, i.e., by examining

$$\dot{\underset{\sim}{y}}(t) = \underset{\sim}{f}(\underset{\sim}{g}(t),t) + \underset{\sim}{f}_{\underset{\sim}{y}}(\underset{\sim}{g}(t),t) \times (\underset{\sim}{y}(t) - \underset{\sim}{g}(t))$$

$$= \underset{\sim}{f}_{\underset{\sim}{y}}(\underset{\sim}{g}(t),t) \times (\underset{\sim}{y}(t) - \underset{\sim}{g}(t)) + \dot{\underset{\sim}{g}}(t), \qquad (2.15)$$

where $\underset{\sim}{f}_{\underset{\sim}{y}}(\underset{\sim}{g}(t),t)$ is the Jacobian evaluated along the particular solution $\underset{\sim}{g}(t)$. If we assume that the Jacobian varies slowly, it can be approximated by a constant matrix $\underset{\sim}{J}$. If, moreover, its eigenvalues q_i are all distinct, there exists a similarity transformation such that the set of coupled equations

$$\dot{\underset{\sim}{y}}(t) = \underset{\sim}{J}(\underset{\sim}{y}(t) - \underset{\sim}{g}(t)) + \dot{\underset{\sim}{g}}(t) \qquad (2.16)$$

is decoupled into a set of scalar equations, each one of which is in the form of (2.1). While the accuracy constraints for both, the forward and the backward Euler methods, allow the step-size to increase every time a transient component becomes insignificant, the stability constraint (2.11) of the forward Euler method limits the step-size, throughout the whole range of integration, to a value dependent on the largest (in magnitude) q_i. On the other hand, since the stability constraint of the backward Euler method is always satisfied, once a transient looses significance, the step-size can be increased.

While only a few practical methods possess a stability property identical to that of the backward Euler method, i.e., every point of the entire left half complex λ-plane ($\lambda = qh$) satisfies the stability constraint, there exist a host of methods whose stability regions contain large portions of the left half λ-plane. Optimization of a particular stability region measure over a family of cyclic composite methods has lead to a set of methods which form the basis of the program STINT. However, before describing this particular set in detail, we devote the last part of this section to discussion of where stiff problems generally arise.

It has been observed that, typically, systems of stiff ODE's are encountered when a physical process in one of the following areas are modeled: electrical circuits and systems, chemical reactions, reactor kinetics, and various problems of dynamics. In addition, stiff problems arise in the mathematical processes of semi-discretization of partial differential equations and solutions of boundary

value problems via initial value problems. One way to detect stiffness in the first four areas is from physical considerations, e.g., greatly differing time constants, presence of large and small concentration change rates, etc. In general, one suspects a problem as being stiff if its solution is known to be very stable with the dependent variable changing significantly on widely different scales of the independent variable. Stiffness can also be detected by other than physical arguments: When a program not designed to handle stiff problems is applied to such, it will appear very inefficient. For example, more than 100 steps are taken over an interval in which the solution changes by less than 1%. Practical experience shows that even with programs designed to handle stiff problems the solution is still relatively expensive. Thus, in general, the accuracy requested should be closely related to the precision with which the fundamental quantities of the model are known and for which the model is valid.

3.1. THE COMPUTATIONAL METHOD

As was pointed out at the beginning of the preceeding section, the majority of the computational methods for numerical solution of differential equations are obtained by application of the principal of discretization. According to this principal an approximate solution to a problem is sought not over a continuous interval but, rather, on a discrete set of points. We are thus seeking a sequence of values $\underset{\sim}{y}_n$ which are approximations to the theoretical solution $\underset{\sim}{y}(t_n)$ at t_n. A computational method supplies the means to generate such a sequence. Since we are interested in a "good" approximation to the exact solution, an acceptable computational method must result in a discrete solution which is close to that of the problem whose solution is sought. This closeness is often determined from analytical (and, in a few cases, geometrical) interpretations of a particular computational method. The former often involves Taylor series expansions and interpolation, while the later is based on graphical observations. While the more simple computational methods (i.e., Euler methods, Trapezoidal rule, explicit Runge-Kutta methods, to name a few) were so interpreted, it was soon realized, that the resulting observations could be generalized. This has lead to the introduction of, among others, the general class of multistep methods. A linear k-step method may be written

$$\sum_{j=-k+1}^{1} [\alpha_j \underset{\sim}{y}_{n+j} - h\beta_j \dot{\underset{\sim}{y}}_{n+j}] = \underset{\sim}{0} \tag{3.1}$$

where α_j and β_j are constants, and $\dot{\underset{\sim}{y}}_{n+j} = \underset{\sim}{f}(\underset{\sim}{y}_{n+j}, t_{n+j})$.

It was concluded in the previous section that to achieve efficient numerical integration of stiff equations it is necessary to use methods with large stability regions. Since this property is not shared by classical explicit and most implicit methods, an extensive search began for new methods with superior stability properties, as well as higher accuracies. Recently, a class of methods, the composite multistep methods, was introduced. This class subsumes the class of linear multistep methods in that, a member of the former class is composed of one or more of the latter. A particular subclass of the class of composite multstep methods—the cyclic composite methods—forms the basis for the new set of computational methods employed in the program STINT. The development of these methods is described next.

Consider the differential equation (1.2) which is, for the reader's convenience rewritten as

$$\dot{\underset{\sim}{y}}(t) = \underset{\sim}{f}(\underset{\sim}{y}(t),t),\ \underset{\sim}{y}(t_0) \text{ given.} \tag{3.2}$$

Suppose that (3.2) possesses a unique solution $\underset{\sim}{y}(t)$. Now, set $t_n = t_0 + nh$ $(n = 0, 1, 2, \ldots)$ and let $\underset{\sim}{y}_n$ denote the approximation of $\underset{\sim}{y}(t_n)$ defined for $n = m\ell + j$ $(j = 1,\ldots,\ell$ and $m = 0, 1, 2,\ldots)$ by the set of ℓ linear multistep formulas

$$\sum_{j=-k+1}^{\ell}[\alpha_{ij}\underset{\sim}{y}_{m\ell+j} - h\beta_{ij}\dot{\underset{\sim}{y}}_{m\ell+j}] = \underset{\sim}{0},\ i = 1,\ldots,\ell, \tag{3.3}$$

where $\dot{\underset{\sim}{y}}_n = \underset{\sim}{f}(\underset{\sim}{y}_n,t_n)$. This set, solved simultaneously for a block of solution values $\underset{\sim}{y}_{m\ell+i}$, defines a composite multistep method.

Next consider the expression

$$L_i[\underset{\sim}{y}(t_{m\ell+\hat{i}}),h] = \sum_{j=-k+1}^{\ell}[\alpha_{ij}\underset{\sim}{y}(t_{m\ell+j}) - h\beta_{ij}\dot{\underset{\sim}{y}}(t_{m\ell+j})], \tag{3.4}$$

i.e., the amount by which the exact solution $\underset{\sim}{y}(t)$ fails to satisfy the i-th constituent formula of (3.2). Under the assumption that $\underset{\sim}{y}(t)$ possesses a convergent Taylor series for $t \in [t_{m\ell-k+1}, t_{m\ell+\ell}]$, we can write (3.4) as

$$L_i[\underset{\sim}{y}(t_{m\ell+i}),h] = \sum_{j=0}^{\infty}\gamma_{ij}\underset{\sim}{y}^{(j)}(t_{m\ell+\hat{i}})\ h^j, \tag{3.5}$$

where

$$\gamma_{i0} = \sum_{\nu=-k+1}^{\ell}\alpha_{i\nu} \tag{3.6a}$$

and

$$\gamma_{ij} = \sum_{\nu=-k+1}^{\ell}[\frac{1}{j!}(\nu - \hat{i})^j\alpha_{i\nu} - \frac{1}{(j-1)!}(\nu - \hat{i})^{j-1}\beta_{i\nu}],\ j > 0. \tag{3.6b}$$

Let p_i be the largest integer such that $\gamma_{ij} = 0$ for $j = 0,\ldots,p_i$, where γ_{ij} is given by (3.6). If $p = \text{MIN}_{1\le i\le\ell}\{p_i\}$, then the composite multistep method (3.3) is said to be of order p. Note: If a composite multistep method of order p is used to integrate an equation whose solution is or can be expressed as a polynomial of degree no greater than p, the numerical solution will be identical with the exact solution, except for round-off errors. The order of the method can thus be considered as the first (possibly crude) measure of its accuracy.

The composite multistep methods which satisfy the constraining relations

$$\left.\begin{aligned}\alpha_{ij} &= 0\\ \beta_{ij} &= 0\end{aligned}\right\} \quad \ell \geq j > i \geq 1. \tag{3.7}$$

are said to be cyclic. This subclass of the class of composite multistep methods is important because the simultaneous solution of the set of formulas (3.3) can now be carried out efficiently by solving the formulas cyclically—first, that for i = 1; second, that for i = 2; and so on. Thus, the effort expended in solving for ℓ solution values $\underset{\sim}{y}_{m\ell+1}$ through $\underset{\sim}{y}_{m\ell+\ell}$ is no greater than that required when using the same fixed-step multistep method ℓ times. Yet, by using a cycle of multistep formulas—not all the same, there is created the possibility for stability properties superior to those corresponding to the "best" multistep methods.

While each constituent formula of a general composite multistep method contains $2(\ell + k)$ parameters (i.e., undetermined coefficients), the number of parameters in the i-th constituent formula of a cyclic method is, in light of (3.7), $2(k + i)$. This number can be further reduced if we require each constituent formula to be a k-step multistep method, in other words, if we demand that each formula span only $(k + 1)$ points. This will be the case, if the constraints

$$\left.\begin{aligned}\alpha_{ij} &= 0\\ \beta_{ij} &= 0\end{aligned}\right\} \quad -k + i > j \tag{3.8}$$

are satisfied. Then, there remain $2(k + 1)$ parameters to be further specified. If we demand that, for accuracy, each of the constituent formulas be of order $p_i = k$, then in view of (3.6), $(k + 1)$ additional constraints on the $2(k + 1)$ remaining parameters will be imposed. Once these order constraints are satisfied, each constituent formula will still contain $(k + 1)$ parameters. These can be used to enhance the cyclic method's stability properties.

In our discussion of stability properties for a "good" stiff integration method we required, when such a method is applied to solve (1.2) with $\underset{\sim}{f}(\underset{\sim}{y}(t),t) = qy$, that $y_n \to 0$ as $n \to \infty$ for every point $\lambda = qh$ in some large region in the left half of the complex λ-plane. Now, when $\lambda \to \infty$, a point which should be a part of any large stability region, y_n might cease to decay, i.e., $|y_{n+1}/y_n| \to 1$ as $\lambda \to \infty$. It can be shown, that to achieve $|y_{n+1}/y_n| \leq \rho < 1$ as $\lambda \to \infty$ it is sufficient for the parameters β_{ij} of (3.3) to satisfy the following constraints:

$$\beta_{ij} = 0,\ j \leq 0,$$

$$\beta_{ii} \neq 0, \tag{3.9}$$

When this constraint is satisfied, each constituent formula then contains MIN{i, k + 1} parameters. Since each formula can be scaled—multiplied by a constant—without effecting the order or stability of the method, one parameter should be viewed as a scale factor. There then remain MIN{i - 1, k} arbitrary parameters in the i-th constituent formula.

Before proceeding further with our description of the development of the new set of methods, we recapitulate what has been done so far. We started from a general composite multistep method and chose the coefficients α_{ij} and β_{ij} so as to specify a method which has a number of desirable properties, such as: computational ease (conditions (3.7) and (3.8)), accuracy (condition (3.6)), and error decay (condition (3.9)). The remaining undetermined coefficients were used to accomplish the following task: Establish a set of cyclic methods which through order six exhibits improved stability properties, order-by-order, over the backward differentiation methods and which exhibits stiff stability properties at order(s) exceeding six. The chosen measure of goodness was the angle $\alpha \in (0, \pi/2]$, determining the set

$$S_\alpha = \{z : |\text{ang}(-z)| < \alpha,\ z \neq 0\},$$

such that for any complex constant $hq \in S_\alpha$ all solutions of the cyclic method applied to solve (1.2) with $\underset{\sim}{f}(\underset{\sim}{y}(t),t) = qy$, tend to zero as $n \to \infty$.

The search for these new methods was carried out in two stages. During the first stage, a cyclic method of cycle length ℓ whose each constituent formula was of order k and whose coefficients satisfied conditions (3.8) and (3.9) was synthesized. In the second stage, the stability property—as characterized by the value of α—of the new cyclic composite method was numerically evaluated. If the value of α was found to be the largest achievable for those values of k and ℓ and superior to the corresponding value for k-th order backward differentiation method, the search was halted. Otherwise, a new synthesis was—possibly with a different ℓ—was performed and α measured again.

The synthesizing process was as follows. First, a set of k + 1 linearly independent multisteps methods, each of order k and whose coefficients satisfied conditions (3.7) and (3.8), was determined. Then, for each i = 1,..., ℓ, a linear

combination of these methods was formed to realize the i-th formula of the method. The value of ℓ was kept as small as possible by doing the search with ℓ incremented by 1 from 2, until the maximized value of α exceeded that for the backward differentiation method of the same order.

To illustrate this process, consider the $k + 1 = 4$ linearly independent multistep methods at order $k = 3$, subject to (3.7) and (3.8), given as:

$$\begin{aligned}
\text{F1:}\quad & -2\underset{\sim}{y}_{n+i-3} + 9\underset{\sim}{y}_{n+i-2} - 18\underset{\sim}{y}_{n+i-1} + 11\underset{\sim}{y}_{n+i} - h6\dot{\underset{\sim}{y}}_{n+i} = \underset{\sim}{0} \\
\text{F2:}\quad & \underset{\sim}{y}_{n+i-3} - 6\underset{\sim}{y}_{n+i-2} + 3\underset{\sim}{y}_{n+i-1} + 2\underset{\sim}{y}_{n+i} - h6\dot{\underset{\sim}{y}}_{n+i-1} = \underset{\sim}{0} \\
\text{F3:}\quad & -2\underset{\sim}{y}_{n+i-3} - 3\underset{\sim}{y}_{n+i-2} + 6\underset{\sim}{y}_{n+i-1} - \underset{\sim}{y}_{n+i} - h6\dot{\underset{\sim}{y}}_{n+i-2} = \underset{\sim}{0} \\
\text{F4:}\quad & -11\underset{\sim}{y}_{n+1-3} + 18\underset{\sim}{y}_{n+i-2} - 9\underset{\sim}{y}_{n+i-1} + 2\underset{\sim}{y}_{n+i} - h6\dot{\underset{\sim}{y}}_{n+i-3} = \underset{\sim}{0}.
\end{aligned} \tag{3.10}$$

From this "arsenal" of multistep formulas, the new cyclic composite method of order 3 was obtained as follows: Each constituent formula was obtained as a linear combination of F1 through F4; thus,

$$C_{i1}F1 + C_{i2}F2 + C_{i3}F3 + C_{i4}F4. \tag{3.11}$$

Obviously, to satisfy (3.9) $C_{ij} \equiv 0$ for $j > i$. The value of α was otherwise maximized at $\ell = 3$ with $(i = 1)$ $C_{11} = 1$, with $(i = 2)$ $C_{21} = 1$ and $C_{22} = 0$, and with $(i = 3)$ $C_{31} = 1/3$, $C_{32} = -2/3$, and $C_{33} = -2/3$. The third order cyclic composite method thus realized may be written as

$$\begin{aligned}
-2\underset{\sim}{y}_{m\ell-2} + 9\underset{\sim}{y}_{m\ell-1} - 18\underset{\sim}{y}_{m\ell} + 11\underset{\sim}{y}_{m\ell+1} - h6\dot{\underset{\sim}{y}}_{m\ell+1} &= \underset{\sim}{0} \\
-2\underset{\sim}{y}_{m\ell-1} + 9\underset{\sim}{y}_{m\ell} - 18\underset{\sim}{y}_{m\ell+1} + 11\underset{\sim}{y}_{m\ell+2} - h6\dot{\underset{\sim}{y}}_{m\ell+2} &= \underset{\sim}{0} \\
9\underset{\sim}{y}_{m\ell+1} - 12\underset{\sim}{y}_{m\ell+2} + 3\underset{\sim}{y}_{m\ell+3} - h(-4\dot{\underset{\sim}{y}}_{m\ell+1} - 4\dot{\underset{\sim}{y}}_{m\ell+2} + 2\dot{\underset{\sim}{y}}_{m\ell+3}) &= \underset{\sim}{0}.
\end{aligned} \tag{3.12}$$

Since the maximized angle α for (3.12) was found to be greater then that corresponding to the third order backward differentiation method, the cyclic method (3.12) was accepted as the new third order method. The cyclic methods of orders three through seven, derived in this fashion are presented in Table 1. Note, that for orders $p = 3$ and 4 the new methods consists of three constituent formulas;

for p = 5, 6, and 7, each method consists of four constituent formulas. The values of the angle α for the new methods as well as for the backward differentiation methods are given in Table 2. The reason why the methods of orders 1 and 2 and their values of α are omitted from the Tables 1 and 2 is explained next.

$p = k = 3$

j \ i	1	2	3	
-2	.2	0	0	α_{ij}
-1	9	.2	0	
0	.18	9	0	
1	11	.18	9	
2	0	11	.12	
3	0	0	3	
-2	0	0	0	β_{ij}
-1	0	0	0	
0	0	0	0	
1	6	0	.4	
2	0	6	.4	
3	0	0	2	

$p = k = 4$

j \ i	1	2	3	
-3	3	0	0	α_{ij}
-2	.16	3	0	
-1	36	.16	11	
0	.48	36	.48	
1	25	.48	216	
2	0	25	.272	
3	0	0	93	
-3	0	0	0	β_{ij}
-2	0	0	0	
-1	0	0	0	
0	0	0	0	
1	12	0	.60	
2	0	12	.48	
3	0	0	48	

$p = k = 5$

j \ i	1	2	3	4	
-4	.12	0	0	0	α_{ij}
-3	75	.12	0	0	
-2	.200	75	.118	0	
-1	300	.200	735	.133	
0	.300	300	.1940	780	
1	137	.300	2980	.1680	
2	0	137	.3030	5470	
3	0	0	1373	.5595	
4	0	0	0	1158	
-4	0	0	0	0	β_{ij}
-3	0	0	0	0	
-2	0	0	0	0	
-1	0	0	0	0	
0	0	0	0	0	
1	60	0	.60	30	
2	0	60	0	.1860	
3	0	0	600	.1530	
4	0	0	0	600	

$p = k = 6$

j \ i	1	2	3	4	
-5	10	0	0	0	α_{ij}
-4	.72	202	0	0	
-3	225	.1455	195	0	
-2	.400	4550	.1399	285	
-1	450	.8100	4340	.2039	
0	.360	9150	.7540	6225	
1	147	.7277	8905	.10360	
2	0	2930	.7445	18455	
3	0	0	2944	.14865	
4	0	0	0	2299	
-5	0	0	0	0	β_{ij}
-4	0	0	0	0	
-3	0	0	0	0	
-2	0	0	0	0	
-1	0	0	0	0	
0	0	0	0	0	
1	60	.60	.420	180	
2	0	1200	.60	.4080	
3	0	0	1200	.4680	
4	0	0	0	1200	

$p = k = 7$

j \ i	1	2	3	4	
-6	.60	0	0	0	α_{ij}
-5	490	.60	0	0	
-4	.1764	490	.210	0	
-3	3675	.1764	1722	.774	
-2	.4900	3675	.6235	6349	
-1	4410	.4900	13100	.22988	
0	.2940	4410	.17650	48160	
1	1089	.2940	17710	.66290	
2	0	1089	.11297	68159	
3	0	0	2860	.42364	
4	0	0	0	9748	
-6	0	0	0	0	β_{ij}
-5	0	0	0	0	
-4	0	0	0	0	
-3	0	0	0	0	
-2	0	0	0	0	
-1	0	0	0	0	
0	0	0	0	0	
1	420	0	.600	840	
2	0	420	.1860	.2100	
3	0	0	1200	.8400	
4	0	0	0	4200	

Table 1. Cyclic Methods: Values of coefficients.

p	α	
	New Method	BDF
3	89.43°	86.03°
4	80.88°	73.35°
5	77.48°	51.84°
6	63.25°	17.84°
7	33.53°	*

*The order seven backward differentiation method does not exhibit stiff stability.

Table 2. Comparison of α-values.

The backward differentiation methods of orders 1 and 2 are, respectively,

$$-\underset{\sim}{y}_n + \underset{\sim}{y}_{n+1} - h\dot{\underset{\sim}{y}}_{n+1} = \underset{\sim}{0} \tag{3.13}$$

and

$$\underset{\sim}{y}_{n-1} - 4\underset{\sim}{y}_n + 3\underset{\sim}{y}_{n+1} - h2\dot{\underset{\sim}{y}}_{n+1} = \underset{\sim}{0} \tag{3.14}$$

The value of α for both of these methods attains its maximum—$\alpha = \pi/2$; therefore, no further improvement in α can take place. Thus, the methods (3.13) and (3.14) are taken to be—with $\ell = 1$—the order 1 and order 2 composite cyclic methods, respectively.

The better stability properties of the new cyclic methods over those of the backward differentiation methods, as attested to by the greater value of α for the former, should permit a numerical integration algorithm based on the cyclic methods to increase the step-size more rapidly than an algorithm with a similar structure which is based on the backward differentiation methods. As indicated in Section 7, the performance data do show improvements in efficiency for stiff problems wherein the eigenvalues of the Jacobian matrix are complex and relatively close to the imaginary axis.

Since the goal of the research which has lead to the new cyclic methods was to develop methods with significantly superior stability properties over those enjoyed by the backward differentiation methods, as measured by the angle α, the

new cyclic methods may not necessarily be the best. It is entirely possible, an even very likely, in view of the fact that the parameters C_{ij} in (3.11) were "manually" set in an interactive computer search for the maximum α, that better and simpler cyclic composite methods exist. An interested reader is invited to try the development of such methods on his own. For those, who are satisfied with the methods obtained thus far, in what follows, we describe how they are implemented in a numerical integration algorithm.

3.2 INTEGRATION ALGORITHM

A program embodying the set of seven cyclic composite multistep methods presented in the previous sub-section will have to use particular techniques for starting the process, for solving a chosen cyclic method, for determing acceptance of a solution, and for changing the step-size and/or order as necessary. It may be said that the choice of these techniques is at least as important as the choice of the method itself. The description of these techniques and how they interact constitutes the integration algorithm. It is convenient to divide the algorithm into segments within which one or several important procedures take place. The integration algorithm described herein is modeled after that due to Gear and later improved by Hindmarsh. The principle segments of this algorithm are: (i) initialization, (ii) solution, (iii) error control, and (iv) step-size and order control. Their interaction is as shown in the block diagram in Figure 2 and as summarized, except for initialization, next.

The past solution values $\underset{\sim}{y}_{m\ell-k+1},\ldots,\underset{\sim}{y}_{m\ell}$—separated by step-size h—are assumed to be known. Then, on entry to the algorithm, the index i is set to 1 and

$$\sum_{j=-k+i}^{i}[\alpha_{ij}\underset{\sim}{y}_{m\ell+j} - h\beta_{ij}\dot{\underset{\sim}{y}}_{m\ell+j}] = \underset{\sim}{0} \qquad (3.15)$$

with

$$\dot{\underset{\sim}{y}}_{m\ell+i} = \underset{\sim}{f}(\underset{\sim}{y}_{m\ell+i}, t_{m\ell+i}),$$

where α_{ij} and β_{ij} are as given in Table 1, is solved for $\underset{\sim}{y}_{m\ell+i}$ and $\dot{\underset{\sim}{y}}_{m\ell+i}$. After the second and each subsequent step —there are ℓ steps in a block— an error test is performed and, if the estimated error is within the pre-specified, per-step error tolerance, the step is accepted; otherwise it is rejected. After all ℓ steps are accepted or when a step is rejected, control passes to the step-size and order control segment. There, the largest step-sizes, consistent with the allowed error tolerance, at the current order and (except when a step was rejected) at orders one lower and one higher are established. If a step was rejected, the entire block is repeated using the new step-size at the current order. Otherwise, the largest step-size and the corresponding order to be used when the algorithm is entered next time are recorded and exit from the algorithm occurs.

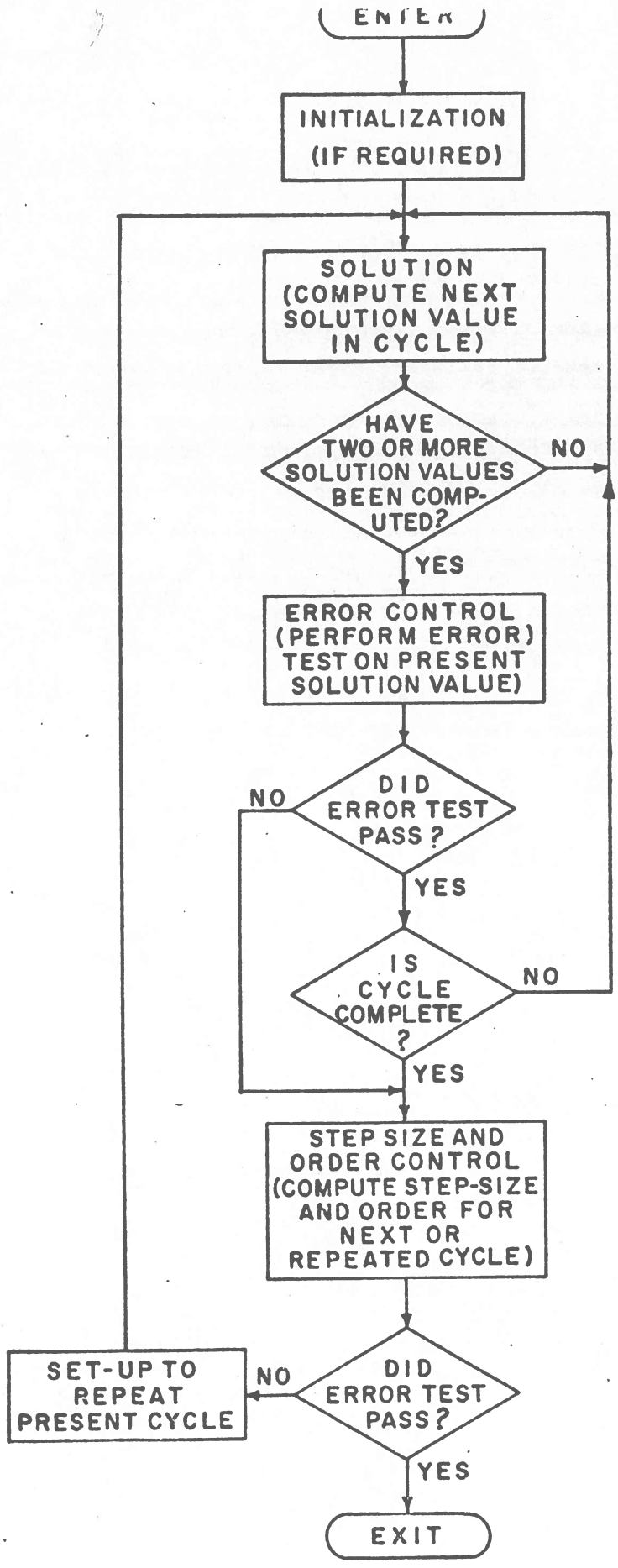


FIGURE 2 BLOCK DIAGRAM OF ALGORITHM

A detailed description of the four principle segments of the algorithm is given in the following.

Solution Segment

Each constituent formula of a cyclic method is an implicit, vector algebraic equation. It is solved by a modified Newton method which is described next.

For each i, equation (3.15) may be written as

$$y_{m\ell+i} - h\frac{\beta_{ii}}{\alpha_{ii}} f(y_{m\ell+i}, t_{m\ell+i}) = -\frac{1}{\alpha_{ii}} \sum_{j=-k+i}^{i-1} [\alpha_{ij} y_{m\ell+j} - h\beta_{ij} \dot{y}_{m\ell+j}], \qquad (3.16)$$

where the right hand side is a scaled sum of known quantities and, hence, constant. Rewriting (3.16) as

$$H(y_{m\ell+i}) \equiv G(y_{m\ell+i}) - C = 0 \qquad (3.17)$$

where

$$G(y_{m\ell+i}) = y_{m\ell+i} - h\frac{\beta_{ii}}{\alpha_{ii}} f(y_{m\ell+i}, t_{m\ell+i}),$$

and

$$C = -\frac{1}{\alpha_{ii}} \sum_{j=-k+i}^{i-1} [\alpha_{ij} y_{m\ell+j} - h\beta_{ij} \dot{y}_{m\ell+j}],$$

and assuming that $y_{m\ell+i}^{(\nu)}$ is an estimate of the solution, the next estimate $y_{m\ell+i}^{(\nu+1)}$ obtained by the Newton method is

$$y_{m\ell+i}^{(\nu+1)} = y_{m\ell+i}^{(\nu)} - W_\nu^{-1} H(y_{m\ell+i}^{(\nu)}), \nu = 0, 1, 2, \ldots, \qquad (3.18)$$

where

$$W_\nu \equiv \frac{\partial H}{\partial y}\Big|_{y = y_{m\ell+i}^{(\nu)}} = I - h\frac{\beta_{ii}}{\alpha_{ii}} J_\nu,$$

with I being the unit matrix and J_ν being the Jacobian matrix of (3.1) evaluated at $(y_{m\ell+i}^{(\nu)}, t_{m\ell+i})$. Evaluation of J_ν—hence, of W_ν and, more importantly, of W_ν^{-1}—may be a difficult as well as time consuming task. We may avoid such evaluations during all but the first iteration step by letting $W_\nu^{-1} \equiv W_0^{-1} \equiv W^{-1}$. This leads to the simplified Newton method which after substitution of (3.17) into (3.18), after

re-arrangement of the resulting terms, and after dropping of the subscript mℓ+i, yields

$$[\underset{\sim}{I} - h\frac{\beta_{ii}}{\alpha_{ii}}\underset{\sim}{J}]\,\underset{\sim}{y}^{(\nu+1)} = h\frac{\beta_{ii}}{\alpha_{ii}}[\underset{\sim}{f}(\underset{\sim}{y}^{(\nu)},t) - \underset{\sim}{J}\underset{\sim}{y}^{(\nu)}] + \underset{\sim}{C}. \qquad (3.19)$$

Now, replace ν by $\nu - 1$ in (3.19); then subract this result from (3.19). After re-arranging terms, we get

$$\underset{\sim}{y}^{(\nu+1)} = \underset{\sim}{y}^{(\nu)} + \frac{\beta_{ii}}{\alpha_{ii}}\underset{\sim}{W}^{-1}[hf(\underset{\sim}{y}^{(\nu)},t) - \underset{\sim}{d}^{(\nu)}], \qquad (3.20)$$

where

$$\underset{\sim}{d}^{(\nu)} \equiv h\underset{\sim}{f}(\underset{\sim}{y}^{(\nu-1)},t) + h\,\underset{\sim}{J}(\underset{\sim}{y}^{(\nu)} - \underset{\sim}{y}^{(\nu-1)}).$$

Equation (3.20) defines a recurrence relation for $\underset{\sim}{y}^{(\nu)}$, $\nu = 1, 2, \ldots$; a similar relation is easily shown to exist for $\underset{\sim}{d}^{(\nu)}$, namely

$$\underset{\sim}{d}^{(\nu+1)} = \underset{\sim}{d}^{(\nu)} + \underset{\sim}{W}^{-1}[h\underset{\sim}{f}(\underset{\sim}{y}^{(\nu)},t) - \underset{\sim}{d}^{(\nu)}]. \qquad (3.21)$$

Observe, that the "correction" terms in (3.20) and (3.21) differ only by a scaling factor β_{ii}/α_{ii}. Note also, that, by accumulating the correction term of (3.21) during the iteration process, we are able to recover the starting values $\underset{\sim}{y}^{(0)}$ and $\underset{\sim}{d}^{(0)}$ whenever desired.

The starting values for the recursion processes (3.20) and (3.21) respectively—the values $\underset{\sim}{y}^{(0)}_{m\ell+i}$ and $\underset{\sim}{d}^{(0)}_{m\ell+i}$—are obtained from the following expressions:

$$\underset{\sim}{y}^{(0)}_{m\ell+i} = -\frac{1}{\alpha^{*}_{ii}}\sum_{j=-k+i}^{i-1}[\alpha^{*}_{ij}\underset{\sim}{y}_{m\ell+j} - h\beta^{*}_{ij}\dot{\underset{\sim}{y}}_{m\ell+j}] \qquad (3.22)$$

and

$$\underset{\sim}{d}^{(0)}_{m\ell+i} = \frac{1}{\alpha^{*}_{ii}\beta_{ii}}\sum_{j=-k+i}^{i-1}[(\alpha^{*}_{ii}\alpha_{ij} - \alpha^{*}_{ij}\alpha_{ii})\underset{\sim}{y}_{m\ell+j} - h(\alpha^{*}_{ii}\beta_{ij} - \beta^{*}_{ij}\alpha_{ii})\dot{\underset{\sim}{y}}_{m\ell+j}]. \qquad (3.23)$$

Note: The expresions for $\underset{\sim}{y}^{(0)}_{m\ell+i}$ and $\underset{\sim}{d}^{(0)}_{m\ell+i}$ are linear combinations of earlier, known quantities. Moreover, if $\underset{\sim}{y}^{(0)}_{m\ell+i}$ satisfies (3.16), then $\underset{\sim}{y}^{(0)}_{m\ell+i} = \underset{\sim}{y}^{(1)}_{m\ell+i} = \underset{\sim}{y}_{m\ell+i}$, and $\underset{\sim}{d}^{(0)}_{m\ell+i} = h\underset{\sim}{f}(\underset{\sim}{y}^{(0)}_{m\ell+i}, t_{m\ell+i}) = h\dot{\underset{\sim}{y}}_{m\ell+i}$. Therefore, as $\underset{\sim}{y}^{(0)}_{m\ell+i}$ is the predicted

value of of $y_{\sim m\ell+i}$, $d^{(0)}_{\sim m\ell+i}$ can be viewed as the predicted value of $h\dot{y}_{\sim m\ell+i}$. Similarly, just as $y^{(\nu)}_{m\ell+i}$ is considered the ν-th iterate for $y_{\sim m\ell+i}$, so too can $d^{(\nu)}_{\sim m\ell+i}$ be considered the ν-th iterate for $h\dot{y}_{\sim m\ell+i}$.

For simplicity, the predictor formulas for $y^{(0)}_{\sim m\ell+i}$ are the same for each i; i.e.,

$$\left.\begin{aligned} \alpha^*_{ij} &= \alpha^*_{i-1,\,j-1} \qquad i = 2,\ldots,\ell \\ \beta^*_{ij} &= \beta^*_{i-1,\,j-1} \qquad j = -k+i,\ldots,i. \end{aligned}\right\} \tag{3.24}$$

For i = 1, the predictor for $y^{(0)}_{\sim m\ell+1}$ is required to be of order ℓ and to have the form

$$y^{(0)}_{\sim m\ell+1} = \sum_{j=-k+1}^{0} \alpha^*_{1j} y_{\sim m\ell+j} + h\beta^*_{10}\dot{y}_{\sim m\ell}. \tag{3.25}$$

Under these conditions the coefficients α^*_{1j} and β^*_{1j} are completely specified. (Note that $\beta^*_{1j} = 0$ if $j \neq 0$.) Once they are known, they can be used together with the coefficients of the k-th order cyclic composite method, to obtain the coefficients for $d^{(0)}_{\sim m\ell+i}$ as specified by (3.23). The coefficients α^*_{1j} and β^*_{10} for orders k = 1 thru 7 are given in Table 3.

The simplified Newton method described thus far requires one evaluation of the iteration matrix—$\underset{\sim}{W}^{-1}$, the inverse of the Jacobian matrix of the algebraic equation (3.16)—per integration step. Since this evaluation requires very many arithmetic operations for large systems, it is important to consider just how $\underset{\sim}{W}$ changes from one integration step to the next and then whether $\underset{\sim}{W}^{-1}$ might be evaluated less often. We do this by observing the changes in its constitutents: (i) $\underset{\sim}{J}$, the Jacobian matrix of the differential equations, (ii) h, the step-size, and (iii) β_{ii}/α_{ii}, the scale factor for the computational method in use. For i = $1,\ldots,\ell$ the step-size h is fixed and the changes in β_{ii}/α_{ii} are small. Assuming the changes in $\underset{\sim}{J}$ during these ℓ steps are also small, the convergence properties of the simplified Newton method will not be substantially effected, if the iteration matrix is evaluated only once—at the beginning of a block of ℓ fixed step-size integration steps. When block-to-block changes in the iteration matrix are considered, it has been observed that monitoring the variation in the factor $h\,\beta_{ii}/\alpha_{ii}$ since the last evaluation of the iteraton matrix is of importance. Numerous observations of the changes in this factor and of those in the value of the iteration matrix, together with various reported experiences, motivated a further

relaxation of the criterion for evaluation of the iteration matrix. The new criterion is: (i) for changes in h β_{ii}/α_{ii} less than 40% since the last evaluation of the iteration matrix, no evaluation of $\underset{\sim}{W}^{-1}$ is performed, (ii) for changes are between 40% and 80%, $\underset{\sim}{W}^{-1}$ is evaluated without a concurrent evaluation of $\underset{\sim}{J}$, and (iii) for changes greater than 80%, both $\underset{\sim}{J}$ and $\underset{\sim}{W}^{-1}$ are evaluated. The simplifed Newton method with the iteration matrix evaluation criterion so relaxed is called the modified Newton method. Note: Since the variation in β_{ii}/α_{ii} for i = 1,...,ℓ is always less than 20%, for simplicity, the change in the value of h β_{ii}/α_{ii} is tested for only once in the block of (ℓ) steps—at its beginning.

k	$\beta^*_{1,0}$	$\alpha^*_{1,-6}$	$\alpha^*_{1,-5}$	$\alpha^*_{1,-4}$	$\alpha^*_{1,-3}$	$\alpha^*_{1,-2}$	$\alpha^*_{1,-1}$	$\alpha^*_{1,0}$
1	1							1
2	2						1	0
3	3					$-\frac{1}{2}$	$\frac{6}{2}$	$-\frac{3}{2}$
4	4				$\frac{1}{3}$	$-\frac{6}{3}$	$\frac{18}{3}$	$-\frac{10}{3}$
5	5			$-\frac{3}{12}$	$\frac{20}{12}$	$-\frac{60}{12}$	$\frac{120}{12}$	$-\frac{65}{12}$
6	6		$\frac{2}{10}$	$-\frac{15}{10}$	$\frac{50}{10}$	$-\frac{100}{10}$	$\frac{150}{10}$	$-\frac{77}{10}$
7	7	$-\frac{10}{60}$	$\frac{84}{60}$	$-\frac{315}{60}$	$\frac{700}{60}$	$-\frac{1050}{60}$	$\frac{1260}{60}$	$-\frac{609}{60}$

Table 3. Coefficients for the $y^{(0)}_{mℓ+1}$ predictor.

The test for convergence of the modified Newton method used for solving (3.16) employs a look-ahead feature. Let

$$\delta^{(\nu)}_{mℓ+i} = \| \underset{\sim}{W}^{-1}[h\underset{\sim}{f}(y^{(\nu)}_{mℓ+i}, t_{mℓ+i}) - \underset{\sim}{d}^{(\nu)}_{mℓ+i}] \|_w, \tag{3.26}$$

where $\underset{\sim}{W}^{-1}$ is the iteration matrix of the modified Newton method, where $y^{(\nu)}_{mℓ+i}$ and $\underset{\sim}{d}^{(\nu)}_{mℓ+i}$ are, respectively, the ν-th iterates for $y_{mℓ+i}$ and $h\dot{y}_{mℓ+i}$, and where $\|\cdot\|_w$ denotes a weighted RMS norm.

Furthermore, let

$$C_\nu = \text{MAX}\{0.9C_{\nu-1}, \frac{\delta^{(\nu)}_{m\ell+i}}{\delta^{(\nu-1)}_{m\ell+i}}\}, \quad \nu > 1, \tag{3.27a}$$

be the modified convergence rate between the $(\nu - 1)$-st and ν-th iterations. Set

$$\beta = \frac{\varepsilon}{2p + 4}, \tag{3.27b}$$

where ε is the pre-specified, per-step error tolerance, and p is the order of the method. Now, convergence is assumed to have occured if

$$\delta^{(\nu)}_{m\ell+i} \times \text{MIN}\{1, 2\ C_\nu\} \leq \beta \tag{3.28a}$$

Note: C_0 is the last value of C_ν from the previous step.

When (3.28) becomes satisfied, it is assumed that convergence has been or would be achieved after the next iteration. In either case, the iteration process is terminated. Note: The look-ahead feature as implemented assumes linear convergence which is, for the modified Newton method, indeed the case. The factor 2 is applied to insure that on the next iteration, if performed, the inequality $\delta^{(\nu)}_{m\ell+i} \leq \beta$ would be much more than satisfied. By not performing that next iteration the number of function evaluations is reduced. Note: The look ahead feature is rendered somewhat conservative by allowing the convergence rate to decrease by a factor of at most 9/10 on any one iteration. Furthermore, the use of β rather than ε guarantees that the algebraic equations will be solved to somewhat greater accuracy than just the pre-specified error tolerance.

Other heuristically established features of the solution segment are: No more than three iterations of the modified Newton method are allowed. If convergence is not achieved by the completion of the third iteration, it is assumed that convergence will not occur. At this point, the iteration matrix is reevaluated, if it has not already been evaluated for the current block of steps, the zero-th iterates $\underset{\sim}{y}^{(0)}_{m\ell+i}$ and $\underset{\sim}{d}^{(0)}_{m\ell+i}$ are recovered, and the modified Newton method is invoked again in an effort to establish the solution value. If it has been previously evaluated for the current block of steps, the step-size is decreased by a factor of 3/10, provided it exceeds a least step-size, and the iteration matrix (without the concurrent reevaluation of the Jacobian matrix of the differential equations) is reevaluated. In the case that the step-size does not exceed a least step-size, the order is reduced to two, if it is greater than two, and to one, if it is equal to two. In all these cases, the solution process for the

entire block is begun again. If the block cannot be repeated—the step-size and order are as small as possible—then an exit from the algorithm, with an appropriate indication, takes place.

There is no theoretical justification for the limit of three on the number of iterations. However, test results indicate that: (i) if four or more iterations are allowed, the number of times the convergence test has failed is not appreciably less, and (ii) if no more than two iterations are permitted, the number of times the convergence criterion is not satisfied is considerably greater. The consequence is a significant increase—in the second case—in the number of times the iteration matrix is reevaluated, and —in both cases—in the number of function evaluations.

When the solution has converged, control is transfered to the error control segment which is described next.

Error Control Segment

Whether or not the solution generated in the previous segment will be acceptable to the user is determined in this segment. As this solution is only an approximation to the exact solution of the ODE (which, unfortunately, is not at our disposal), the need arises for an estimate of the error of this approximation from the approximate values available. Ideally, we would like to check on the quality of the approximation after every integration step. However, the necessary solution values for obtaining an error estimate after the first step in a block of ℓ steps are not always available. Thus, an error test is performed only after the second and each subsequent step. There exist, then, two distinct stages in the testing process. In the first stage we consider acceptance of the solution after the second integration step. During the second stage, the solution generated at each subsequent integration step is tested. The classical concept of the local (stepwise) error is readily applicable to the second stage. The appropriate concept for the first stage, that of blockwise local error, is developed next.

Consider the cyclic composite multistep method of order p and the local initial value problem

$$\underset{\sim}{\dot{z}}(t) = \underset{\sim}{f}(\underset{\sim}{z}(t),t) \tag{3.29}$$

with

$$\underset{\sim}{z}(t_{m\ell}) = \underset{\sim}{y}_{m\ell}. \tag{3.30}$$

By reference to (3.4), under the assumption that $\underset{\sim}{z}(t)$ is sufficiently smooth, it is found that

$$L_i[\underset{\sim}{z}(t_{m\ell+\hat{i}}),h] = \gamma_{i,p+1}\underset{\sim}{z}^{(p+1)}(t_{m\ell+\hat{i}})h^{p+1} + \mathcal{O}[h^{p+2}], \tag{3.31}$$

where the $\gamma_{i,p+1}$ is given by (3.6b). Note: p_i rather than p should appear in (3.31), but the orders of the constituent formulas under consideration are all equal; that is $p_i = p$ for all i. Hence, for all i, the $\gamma_{i,p+1}$ do not depend upon $\hat{i}$.

The $L_i[\underset{\sim}{z}(t_{m\ell+\hat{i}}),h]$'s can be catenated to yield

$$\underset{\sim}{L}[\underset{\sim}{z}(t_{m\ell+\hat{i}}),h] = (\underset{\sim}{\gamma}_{p+1}\otimes \underset{\sim}{I})\underset{\sim}{z}^{(p+1)}(t_{m\ell+\hat{i}})h^{p+1} + \mathcal{O}[h^{p+2}], \tag{3.32}$$

where $\underset{\sim}{\gamma}_{p+1} = [\gamma_{1,p+1},\ldots,\gamma_{\ell,p+1}]^T$ and $\otimes$ denotes the Kronecker product. As $L_i[\underset{\sim}{z}(t_{m\ell+\hat{i}}),h]$ is the local discretization error for the i-th constituent formula of the composite multistep method, $\underset{\sim}{L}[\underset{\sim}{z}(t_{m\ell+\hat{i}}),h]$ is that, in a blockwise sense, for the method as whole. Under the simplifying assumption that the stored values $\underset{\sim}{y}_{m\ell+j}$, $j = -k+1,\ldots,0$ coincide with the exact values $\underset{\sim}{z}(t_{m\ell+j})$ of the local initial value problem, the blockwise local error—i.e., the local error accrued in stepping from $t_{m\ell}$ to $t_{m\ell+\ell}$ step - by - step—is related to the blockwise local discretization error as

$$\begin{bmatrix} \underset{\sim}{z}(t_{m\ell+\ell}) \\ \vdots \\ \underset{\sim}{z}(t_{m\ell+1}) \end{bmatrix} - \begin{bmatrix} \underset{\sim}{y}_{m\ell+\ell} \\ \vdots \\ \underset{\sim}{y}_{m\ell+1} \end{bmatrix} = (\underset{\sim}{\delta}_{p+1}\otimes \underset{\sim}{I})\underset{\sim}{z}^{(p+1)}(t_{m\ell+\hat{i}})h^{p+1} + \mathcal{O}[h^{p+2}], \tag{3.33}$$

with $\underset{\sim}{\delta}_{p+1} = \underset{\sim}{A}^{-1}\underset{\sim}{\gamma}_{p+1}$, where the matrix $\underset{\sim}{A}^{-1}$ is given as

$$\underset{\sim}{A}^{-1} = \begin{bmatrix} \alpha_{1\ell} & \cdots & \alpha_{11} \\ \vdots & & \vdots \\ \alpha_{\ell\ell} & \cdots & \alpha_{\ell 1} \end{bmatrix}^{-1}$$

Note: Although the $y_{m\ell+j}$ and $z_{m\ell+j}$, $j = 1,\ldots,\ell$, appear in reverse order in (3.33), the vector δ_{p+1} follows conventional ordering. Now, to use $(\delta_{p+1} \otimes I)\, z^{(p+1)}(t_{m\ell+\hat{i}})h^{p+1}$, the principal part of the blockwise local discretization error, for error estimation, it is necessary to approximate $z^{(p+1)}(t_{m\ell+\hat{i}})h^{p+1}$ by solution values which are available. The approximation chosen is $\nabla^{p+1}y_{m\ell+i}$, where the notation $\nabla^i y_j$ means the backward difference of order i formed from y_j, $y_{j-1},\ldots,y_{j-i}$. Note, $\nabla^{p+1}y_{m\ell+\hat{i}}$ approximates $\nabla^{p+1}z(t_{m\ell+\hat{i}}) = z^{(p+1)}(\tau)h^{p+1}$ for some $\tau\varepsilon[t_{m\ell+\hat{i}-p-1}, t_{m\ell+\hat{i}}]$, an interval containing $t_{m\ell+\hat{i}}$.

The first phase of the error estimation is as follows: After the first two integration steps, $\nabla^{p+1}y_{m\ell+2}$ is evaluated.* (Note that the value of $\hat{i}$ is set to 2.) The solution values $y_{m\ell+1}$ and $y_{m\ell+2}$ obtained in the two steps are then accepted, if

$$\| \nabla^{p+1} y_{m\ell+2} \|_w \leq \frac{\varepsilon}{\text{MAX}\{\delta_{\ell,p+1}, \delta_{\ell-1,p+1}\}} = \frac{\varepsilon}{\delta_{\ell-1,p+1}}, \tag{3.34}$$

where $\| \cdot \|_w$ denotes the same weighted RMS norm found in (3.26).

Next, consider the local initial value problem

$$\dot{z}(t) = f(z(t),t) \tag{3.35}$$

with

$$z(t_{m\ell+i-1}) = y_{m\ell+i-1}, \quad i = 3,\ldots,\ell. \tag{3.36}$$

As before, if $z(t)$ is sufficiently smooth

$$L_i[z(t_{m\ell+\hat{i}}),h] = \gamma_{i,p+1} z^{(p+1)}(t_{m\ell+\hat{i}})h^{p+1} + \bigcirc[h^{p+2}], \tag{3.37}$$

Under the simplifying assumption that the values $y_{m\ell+j}$, for $j = -k + i,\ldots, i - 1$, coincide with the exact values of $z(t_{m\ell+j})$, the local error in stepping from $t_{m\ell+i-1}$ to $t_{m\ell+i}$ is

$$z(t_{m\ell+\hat{i}}) - y_{m\ell+i} = \frac{\gamma_{i,p+1}}{\alpha_{ii}} z^{(p+1)}(t_{m\ell+\hat{i}})h^{p+1} + \bigcirc[h^{p+2}]. \tag{3.38}$$

---

*For reasons, and by means explained in the subsection on the step-size and order control segment, the cycle lengths of the order 1 and 2 methods have been increased from 1 to 3.

As previously, $\underset{\sim}{z}^{(p+1)}(t_{m\ell+\hat{i}})h^{p+1}$ is to be approximated by $\nabla^{p+1}\underset{\sim}{y}_{m+\hat{i}}$, in particular with $\hat{i} = i$.

The second phase of the error control criterion is: For each of the remaining $\ell-2$ steps evaluate $\nabla^{p+1}\underset{\sim}{y}_{m\ell+i}$ and accept $\underset{\sim}{y}_{m\ell+i}$ as the solution value, if

$$\| \nabla^{p+1}\underset{\sim}{y}_{m\ell+i} \|_w \leq \frac{\varepsilon}{\gamma_{i,p+1}/\alpha_{ii}}, \quad i = 3,\ldots,\ell \tag{3.39}$$

is satisfied.

The values $1/\delta_{\ell-1,p+1}$ and $\alpha_{ii}/\gamma_{i,p+1}$ rounded up to five significant digits are displayed in Table 4.

p	1	2	3	4	5	6	7
$1/\delta_{\ell-1,p+1}$	1.0	1.9286	2.7816	3.5674	4.295	4.9065	5.6066
$\alpha_{33}/\gamma_{3,p+1}$	2.0	4.5	6.0	9.3	13.869	17.435	15.921
$\alpha_{44}/\gamma_{4,p+1}$					9.6904	9.472	14.781

Table 4: Reciprocal values of the constants $\delta_{\ell-1,p+1}$, $\gamma_{3,p+1}/\alpha_{33}$, and $\gamma_{4,p+1}/\alpha_{44}$.

If a step is rejected—(3.34) or (3.39) is not satisfied—then the following, heuristically established, procedures are followed: (i) if the error test has failed fewer than three times, then the step-size is reduced, provided it exceeds a least step-size. After the first error test failure, the new step-size is determined in the step-size and order control segment. After the second error test failure, the step-size is further reduced by a factor of 2. In both cases, the iteration matrix is ordered to be evaluated without concurrent evaluation of its constituent matrix $\underset{\sim}{J}$. When the step-size does not exceed a least step-size, the order is set to two, if it exceeds two; otherwise, it is set to one. In all these cases, the entire cycle of ℓ-steps is then repeated. If it cannot be repeated—the step-size equals its lower bound and the order equals one—then the steps computed thus far are accepted and an appropriate indication to this effect is given. (ii) After three successive error test failures, if the step-size still exceeds a least step-size, it is further reduced by a factor of 10 and the cycle is repeated under conditions essentially as on initial entry to the algorithm; in fact, the numerical integration is restarted.

After all ℓ steps have been completed—a block of ℓ new solution values accepted—it is necessary to establish the step-size and order to be used for the next block. This is accomplished in the step-size and order control segment.

Step-size and Order Control Segment

In this segment a new step-size $\hat{h}$ and/or $\hat{p}$ for the next cycle or, in the case that the error test failed, a step-size $\tilde{h}$ for the repetition of the same cycle are determined. The new step-size $\hat{h}$ and order $\hat{p}$ for the next cycle are set as follows: For $j \in \mathcal{O} = \{p-1, p, p+1\} \cap \{1,\ldots,7\}$ compute

$$\eta_j = \text{MIN}\left\{\bar{\eta}_{p,j},\ \sigma_{pj}\left[\frac{\varepsilon}{\delta_{\ell-1,j+1}\,\|\,\nabla^{j+1} y_{m\ell+\ell}\,\|_w}\right]^{\frac{1}{j+1}}\right\} \tag{3.40}$$

where the constant σ_{pj} satisfies $0 < \sigma_{pj} \leq 1$, and where the value $\bar{\eta}_{pj}$ is displayed in Table 5. (This value—a step-size limiter—is explained subsequently). Let $\eta = 1$ and $\hat{p} = p$ if $\text{MAX}_{j\in\mathcal{O}}\{\eta_j\} < 1.1$, otherwise set $\eta = \text{MAX}_{j\in\mathcal{O}}\{\eta_j\}$ and $\hat{p}$ equal to the least index j such that $\eta_j = \eta$. Then, set $\hat{h} = \eta h$.

In the event that the error test failed after the i-th itegration step ($i = 2, 3,\ldots,\ell$), the step-size $\tilde{h}$ for the repetition of the same cycle is given by

$$h = \sigma_{pp}\left[\frac{\varepsilon}{\delta_{\ell-1,p+1}\,\|\,\nabla^{p+1} y_{m\ell+i}\,\|_w}\right]^{\frac{1}{p+1}},\quad i = 2 \tag{3.41a}$$

and

$$\tilde{h} = \sigma_{pp}\left[\frac{\varepsilon}{(\gamma_{i,p+1}/\alpha_{ii})\,\|\,\nabla^{p+1} y_{m\ell+i}\,\|_w}\right]^{\frac{1}{p+1}},\quad i = 3,\ldots,\ell, \tag{3.41b}$$

where p is the order of the rejected cycle. Note: If $\nabla^{p+1} y_{m\ell+i}$ were constant, then a repeated cycle with the new step-size $\tilde{h}$ given by (3.41a) or (3.41b) with $\sigma_{pp} = 1$ would just satisfy the error test. As $\nabla^{p+1} y_{m\ell+i}$ is not usually constant, a slightly smaller step-size is selected by setting the factor σ_{pp} to a value less than unity. Similarly, consider the expression (3.40) for the new step-size $\hat{h}$, where, for the moment $\bar{\eta}_{pj}$ is set to infinity. If this step-size were used and if $\nabla^{\hat{p}+1} y_{m\ell+\ell}$ and $\nabla^{\hat{p}+1} y_{m\ell+\ell+2}$ were equal then the error test at the second integration step of the next cycle would just be satisfied. As these differences are not usually constants, again, a slightly smaller step-size is used. It has been found that $\sigma_{pj} = 1/1.2$ for $j \in \mathcal{O}$ and $p \in \{1,\ldots,7\}$ is a good choice.

Observe, also, that the step-size and/or the order are permitted to change only if a step-size increase of not less than 10% is indicated. The rationale is: A step-size change, as an increase, should be sufficient to justify the considerable effort required to interpolate established solution values so as to obtain the starting values for the next cycle on the new mesh which corresponds to the new step-size.

As just noted, when a step-size change is indicated, the next set of starting values—the past solution values—at the new mesh points must be obtained by interpolation. To keep the interpolation errors reasonably small the following condition has been imposed: The next set of starting solution values must correspond to values lying within the interval $[t_{m\ell-p+1}, t_{m\ell+\ell}]$, that is, within the interval containing all solution values needed for computation of the cycle just accepted—the past values—as well as the solution values generated within the cycle—the current values. This imposes the inequality

$$(\hat{p} - 1)\hat{h} \leq (\ell + p - 1)h \tag{3.42}$$

which must be satisfied with the new step-size $\hat{h}$. The largest $\hat{h}$ which satisfies (3.42) is

$$\hat{h}_{MAX} = \frac{\ell + p - 1}{\hat{p} - 1} h = \bar{\eta}_{p\hat{p}} h. \tag{3.43}$$

The values of $\bar{\eta}_{p\hat{p}}$ are given in Table 5. They follow from (3.43) when the dependency of ℓ on p is taken into account. The value of ℓ for p equal to one and two is not the one specified previously; this is discussed next.

$\bar{\eta}_{p\hat{p}}$	$\hat{p}$		
	p-1	p	p+1
1	-	∞	3
2	∞	4	2
3	5	5/2	5/3
p 4	3	2	3/2
5	8/3	2	8/5
6	9/4	9/5	3/2
7	2	5/3	-

Table 5: Values of Bound $\bar{\eta}_{p\hat{p}}$ on Step-size Increase.

As the step-size and order control procedure requires $p + 3 = k + 3$ solution values to evaluate $\nabla^{j+1} y_{m\ell+\ell}$ with $j = p + 1$, the cycle length ℓ—the number of steps with fixed h—must equal or exceed three (for methods of all orders, including those of orders one and two), if the solution values are to be only those associated with the current cycle. To satisfy this condition, the length of the cycle for the methods of orders one and two was increased from one to three by letting each constituent formula in these expanded cycles be the same as in (3.13) and (3.14), respectively.

Since the stability properties indicated in Section 3.1 were predicted upon a fixed step-size throughout the entire integration process, by varying the step-size in the course of integration, we should expect these properties to change. It has been observed that, even under a very mild variation of the step-size, the stability properties of the backward differentiation methods deteriorate rapidly—especially at high orders. Since it is felt that the same is true for the cyclic composite multistep methods, entry to the step-size and order control segment to determine the step-size and/or order for the next cycle is precluded (i) if there was a convergence or error test failure which required a step-size reduction during the current cycle, and (ii) every other successful cycle of order four or higher.

Initialization Segment

The integration process is started at order one with given initial conditions y_0, t_0, and initial step-size h_0. After evaluation of $d_0 = hy_0 = hf(y_0,t_0)$, the error control and the step-size and order control procedures, as previously described, maintain the proper sequencing of events.

The realization of this basic integration algorithm as a computer program is described in the next section.

4. PROGRAM CODE REALIZATION

Since the integration algorithm's structure as described above is oriented towards evaluation of a block of solution values, the algorithm is realized as a subroutine program which can be called repeatedly to establish a numerical solution over the desired range of integration. After computing a block of ℓ steps with a fixed step-size h, control is returned to the user who may inspect the solution and various internally established quantities to be used—unless altered—when the subroutine is executed next time. We repeat: The user may alter various quantities and parameters as he sees fit before computing the next solution block. While the need to repeatedly call the subroutine may become bothersome for the casual user, especially, if he is interested in the solution values at only a relatively few points, the sophisticated user may greatly profit from the ability to examine and possibly alter the course of the solution after each block. Since it was assumed that the majority of users of this program would be of latter type, only limited checks of input arguments as protection against accidental misuse of the code were incorporated into the program code.

The various segments of the integration algorithm as described in sub-section 3.2 were programmed in a straightforward manner, except, as noted in the following paragraphs. (Each distinct topic commences from the mark ■ at the beginning of a paragraph.)

■ As the error control segment is dependent on the backward differences $\nabla^{p+1} y_{m\ell+i}$, $i = 2,\ldots,\ell$, and as the step-size and order control segment is dependent on the backward differences $\nabla^{j} y_{m\ell+\ell}$, $j = p, p+1, p+2$, it is advantageous to store and work with the backward differences of the dependent variable in all instances, except, in the case of equations (3.20) and (3.21). It can be seen from the form of these equations that it is simpler to work with the ordinate values of the dependent variable. However, the defining expressions (3.22) and (3.23) for the zero-th iterates $y^{(0)}_{m\ell+i}$ and $d^{(0)}_{m\ell+i}$ can be easily reformulated in terms of the backward differences $\nabla^{j} y_{m\ell+i-1}$, $j = 0, 1,\ldots,k-1$. For example, for $k = 3$, $y^{(0)}_{m\ell+i}$ and $d^{(0)}_{m\ell+i}$ can be expressed as

$$y^{(0)}_{m\ell+i} = y_{m\ell+i-1} - 2\nabla y_{m\ell+i-1} - \frac{1}{2}\nabla^2 y_{m\ell+i-1} + 3h\dot{y}_{m\ell+i-1},\ i = 1, 2, 3, \tag{4.1}$$

and

$$d^{(0)}_{m\ell+i} = -\frac{9}{2}\nabla y_{m\ell+i-1} - \frac{5}{4}\nabla^2 y_{m\ell+i-1} + \frac{11}{2}h\dot{y}_{m\ell+i-1},\ i = 1, 2, \tag{4.2a}$$

$$d^{(0)}_{\sim m\ell+3} = -\frac{15}{2}\nabla y_{\sim m\ell+2} - \frac{3}{4}\nabla^2 y_{\sim m\ell+2} + 2h\dot{y}_{\sim m\ell+1} + \frac{13}{2}h\dot{y}_{\sim m\ell+2}. \qquad (4.2b)$$

In general, the k-th order zero-th iterate $y^{(0)}_{\sim m\ell+i}$ formulated in terms of backward differences can be expressed as

$$y^{(0)}_{\sim m\ell+i} = y_{\sim m\ell+i-1} + kh\dot{y}_{\sim m\ell+i-1} + \sum_{j=1}^{k-1} \frac{(j-k)}{j}\nabla^j y_{\sim m\ell+i-1}, \quad i = 1,\ldots,\ell. \qquad (4.3)$$

However, a similar expression (with the coefficients $\alpha^{(k)}_{ij}$ and $\beta^{(k)}_{ij}$ given explicitly) of the k-th order zero-th iterate $d^{(0)}_{\sim m\ell+i}$ written as

$$d^{(0)}_{\sim m\ell+i} = \sum_{j=1}^{k-1} \alpha^{(k)}_{ij}\nabla^j y_{\sim m\ell+i-1} + h\sum_{j=1}^{i} \beta^{(k)}_{ij}\dot{y}_{\sim m\ell+i-j}, \quad i = 1,\ldots,\ell, \qquad (4.4)$$

does not exist. For this reason, the values of $\alpha^{(k)}_{ij}$ and $\beta^{(k)}_{ij}$, given in Table 6, were pre-computed and stored in the array labeled B in the program, for which 82 double precision storage locations were allocated.

It is of interest to observe that should the i-th cyclic method be replaced by a different one, but of the same order, only the coefficients determining the $d^{(0)}_{\sim m\ell+i}$ need to be changed. The zero-th iterate $y^{(0)}_{m\ell+i}$ remains as in (4.3).

When a change in the step-size and/or the order for a new cycle is indicated, a set of starting values for that cycle, corresponding to a mesh with the new step-size $\hat{h}$, must be established. By limiting the step-size increase to the value given in Table 5, the new starting values are constrained to lie in the interval $[t_{m\ell+1-p}, t_{m\ell+\ell}]$, where p is the old order. There are then $\ell + p - 1$ solution values, each separated by the old step-size h, from which to choose $p + 1$ interpolation nodes as a base for the p-th order interpolating polynomial. In order to keep the interpolation errors small, and for the sake of simplicity, two interpolation bases were selected. The first one consists of the set of points $\{y_{m\ell+\ell-p-1}, \ldots, y_{m\ell+\ell-1}\}$, or equivalently, the values $\nabla^j y_{m\ell+\ell-1}$, $j = 0, 1,\ldots,p$, while the second is composed of the points $\{y_{m\ell-p+1}, \ldots, y_{m\ell+1}\}$, or equivalently, the values $\nabla^j y_{m\ell+1}$. Designating by $\hat{p}$ the new order, the criterion for deciding which of these two bases to use is as follows: if $\hat{t}_{m\ell+\ell-j} = t_{m\ell+\ell} - j\hat{h}$, $j = 1,\ldots,\hat{p} - 1$ is such that $\hat{t}_{m\ell+\ell-j} \in (t_{m\ell+\ell-p-1}, t_{m\ell+\ell})$, use the first base; otherwise, use the second base. The $\hat{p} - 1$ new starting values are obtained by evaluating

$$\hat{y}_{(m+1)\ell-j} \equiv \hat{y}_{m\ell+\ell-j} = y_{m\ell+\ell-1} + s\nabla y_{m\ell+\ell-1} + \ldots + \frac{s(s+1)\ldots(s+p-1)}{p!}\nabla^p y_{m\ell+\ell-1}, \quad j = 1,\ldots,n \text{ and } n \leq \hat{p} - 1 \qquad (4.5a)$$

k	i	$\alpha_{i1}^{(k)}$	$\alpha_{i2}^{(k)}$	$\alpha_{i3}^{(k)}$	$\alpha_{i4}^{(k)}$	$\alpha_{i5}^{(k)}$	$\alpha_{i6}^{(k)}$	$\beta_{i1}^{(k)}$	$\beta_{i2}^{(k)}$	$\beta_{i3}^{(k)}$	$\beta_{i4}^{(k)}$
1	1,2,3							1			
2	1,2,3	-2						3			
3	1,2	$-\frac{9}{2}$	$-\frac{5}{4}$					$\frac{11}{2}$			
	3	$-\frac{15}{2}$	$-\frac{3}{4}$					$\frac{13}{2}$	2		
4	1,2	$-\frac{22}{3}$	$-\frac{8}{3}$	$-\frac{17}{18}$				$\frac{25}{3}$			
	3	-9	$-\frac{9}{4}$	$-\frac{7}{8}$				$\frac{35}{4}$	$\frac{5}{4}$		
5	1,2	$-\frac{125}{12}$	$-\frac{101}{24}$	$-\frac{71}{36}$	$-\frac{37}{48}$			$\frac{137}{12}$			
	3	$-\frac{253}{24}$	$-\frac{1001}{240}$	$-\frac{707}{360}$	$-\frac{123}{160}$			$\frac{1373}{120}$	$\frac{1}{10}$		
	4	$-\frac{57}{4}$	$-\frac{25}{8}$	$-\frac{17}{10}$	$-\frac{169}{240}$			$\frac{61}{5}$	$\frac{31}{10}$	$-\frac{1}{20}$	
6	1,2	$-\frac{137}{10}$	$-\frac{117}{20}$	$-\frac{46}{15}$	$-\frac{191}{120}$	$-\frac{197}{300}$		$\frac{147}{10}$			
	3	$-\frac{353}{25}$	$-\frac{571}{100}$	$-\frac{1819}{600}$	$-\frac{79}{50}$	$-\frac{3919}{6000}$		$\frac{1477}{100}$	$\frac{7}{20}$		
	4	$-\frac{3527}{200}$	$-\frac{1889}{400}$	$-\frac{1609}{600}$	$-\frac{3527}{2400}$	$-\frac{931}{1500}$		$\frac{3079}{200}$	$\frac{17}{5}$	$-\frac{3}{20}$	
7	1,2	$-\frac{343}{20}$	$-\frac{303}{40}$	$-\frac{253}{60}$	$-\frac{589}{240}$	$-\frac{101}{75}$	$-\frac{23}{40}$	$\frac{363}{20}$			
	3	$-\frac{266}{15}$	$-\frac{221}{30}$	$-\frac{749}{180}$	$-\frac{73}{30}$	$-\frac{803}{600}$	$-\frac{103}{180}$	$\frac{547}{30}$	$\frac{1}{2}$		
	4	$-\frac{1316}{75}$	$-\frac{1151}{150}$	$-\frac{3689}{900}$	$-\frac{121}{50}$	$-\frac{4003}{3000}$	$-\frac{257}{450}$	$\frac{2737}{150}$	$\frac{1}{2}$	$-\frac{1}{5}$	

Table 6. The values of the coefficients $\alpha_{ij}^{(k)}$ and $\beta_{ij}^{(k)}$.

where

$$s = \frac{\hat{t}_{m\ell+\ell-j} - t_{m\ell+\ell-1}}{h},$$

when using the first base, and by evaluating

$$\hat{y}_{(m\ell+1)\ell-j} \equiv \hat{y}_{m\ell+\ell-j} = y_{m\ell+1} + s\nabla y_{m\ell+1} + \ldots + \frac{s(s+1)\ldots(s+p-1)}{p!}\nabla^p y_{m\ell+1}, \tag{4.5b}$$

where

$$s = \frac{\hat{t}_{m\ell+\ell-j} - t_{m\ell+1}}{h},$$

when using the second base. These values, together with $\hat{y}_{(m\ell+1)\ell} \equiv \hat{y}_{m\ell+\ell}$, form the set of $\hat{p}$ starting values necessary for computing the $(m+1)$-st block of order $\hat{p}$. Finally, this set is reformulated in terms of the $\nabla^j y_{m\ell+\ell}$, $j = 0, 1, \ldots, \hat{p} - 1$ backward differences. Observe that when $\hat{p} \leq p$, the new starting values are, at least locally, accurate up to the order $\hat{p} + 1$ in h. Although this is not true when $\hat{p} > p$, the algorithm has been found, even in this case, to perform satisfactorily.

If it is decided that the block which is to be computed next, should use the same step-size and order as used in the previous block, the starting values for the next block are immediately available. In terms of backward differences, these are the p values $y_{m\ell+\ell}, \nabla y_{m\ell+\ell}, \ldots, \nabla^{p-1} y_{m\ell+\ell}$. Moreover, from the last error test, the value $\nabla^p y_{m\ell+\ell}$ is also available. As will be shown in a subsequent paragraph, this backward difference can be of practical value when interpolation following an error test failure is required. To distinguish between the case when $\nabla^{\hat{p}} y_{(m\ell+1)\ell}$ is available and when it is not, in the program, we set the parameter IDEL = $p+1$ in the former case and IDEL = 0 in the latter.

When the corrector converges, the backward differences at the new point $y_{m\ell+i}$ are obtained from the simple recursion

$$\nabla^j y_{m\ell+i} = \nabla^{j-1} y_{m\ell+i} - \nabla^{j-1} y_{m\ell+i-1}, \quad i = 1, \ldots, \ell \text{ and } j = 1, \ldots \hat{p}. \tag{4.6}$$

Now, for the error test, $\nabla^{\hat{p}+1} y_{m\ell+i}$ must be established. As discussed above, $\nabla^{\hat{p}} y_{m\ell}$ is not always available. Hence, the error test is performed only for $i = 2, \ldots, \ell$, for which cases $\nabla^{\hat{p}+1} y_{m\ell+i}$ is easily found from

$$\nabla^{\hat{p}+1} y_{m\ell+i} = \nabla^{\hat{p}} y_{m\ell+i} - \nabla^{\hat{p}} y_{m\ell+i-1}. \tag{4.7}$$

The reciprocal values of the constants $\delta_{\ell-1,\ p+1}$ for p = 1,...,7 which appear in the error test (3.34) are stored in the array PERROR. Similarly, the reciprocal values of $\gamma_{3,\ p+1}/\alpha_{33}$ and $\gamma_{4,\ p+1}/\alpha_{44}$, which appear in the error test (3.39), together with the reciprocal values of $\gamma_{1,\ p+1}/\alpha_{11}$, and $\gamma_{2,\ p+1}/\alpha_{22}$ are stored in the array PC. The reason for adding the last mentioned values will become clear when the convergence criterion is discussed subsequently in greater detail.

■ When the error test fails for the first time, the step-size for the repetition of the entire cycle is determined in the step-size and order control segment. Let ˜ denote the starting values, on the mesh corresponding to the step-size $\tilde{h}$, to be established and ∧ the starting values established prior to the error test failure. Note: These latter values were established by the p-th order interpolation process, where p is the order of the last accepted block. Now, if the step-size h corresponding to this block did not change—$\hat{h} \equiv h$ and $\hat{p} \equiv p$—the $\hat{p} + 1$ values are available—each (at least) locally accurate to (error of) order $\hat{h}^{\hat{p}+1}$—to determine the $\hat{p}$-th order interpolating polynomial which interpolates the values on the new mesh with a step-size $\tilde{h}$ accurate to order $\tilde{h}^{\tilde{p}+1}$ with $\tilde{p} = \hat{p}$. The new starting values are derived from

$$\tilde{y}_{m\ell-j} = \hat{y}_{m\ell} + s\nabla\hat{y}_{m\ell} + \ldots + \frac{s(s+1)\ldots(s+\tilde{p}-1)}{\tilde{p}!}\nabla^{\tilde{p}}\hat{y}_{m\ell},\ j = 1,\ldots,\tilde{p}-1, \tag{4.8}$$

where

$$s = \frac{t_{m\ell} - j\tilde{h} - t_{m\ell}}{\hat{h}}.$$

Together with $\tilde{y}_{m\ell} \equiv \hat{y}_{m\ell}$, these values are easily reformulated in the form of backward differences $\nabla^j y_{m\ell}$, $j = 0,\ldots,\tilde{p}-1$.

Next, suppose that the order decreased from the one corresponding to the last accepted block, that is, $\hat{h} > h$ and $\hat{p} = p - 1$. The starting values had been obtained (at least) locally accurate to order $\hat{h}^{\hat{p}+2}$. The new starting values corresponding to the step-size $\tilde{h}$ should be established with local accuracy to order $\tilde{h}^{\tilde{p}+1}$ ($\tilde{p} = \hat{p}$). This would be possible if we were able to construct the $\hat{p}$-th order interpolating polynomial. However, since only $\hat{p}$ solution points are available, we have to add the value $\hat{h}\dot{\hat{y}}_{m\ell}$ to this set so that an interpolating polynomial of that order can be constructed. Similarly, when $\hat{h} > h$ and $\hat{p} = p$ or $\hat{p} = p+1$, by adding the value $\hat{h}\,\hat{\hat{y}}_{m\ell}$ to the set of $\hat{p}$ points, we can construct the $\hat{p}$-th order interpolating polynomial which interpolates the new starting values corresponding to the step-size $\tilde{h}$ to order $\tilde{h}^{\tilde{p}+1}$ with $\tilde{p} = \hat{p}$. Observe, however, that when

$\hat{p} = p + 1$, the starting values corresponding to the step-size $\hat{h}$ are known to be locally accurate only to order $\hat{h}^{\hat{p}}$. Thus, there are rational reasons—however, not supported by experience—to say: It would be inappropriate to interpolate for the new starting values corresponding to the step-size $\tilde{h}$ using an interpolating polynomial of order higher than $\hat{p} - 1$. It is easily seen that such a $(\hat{p} - 1)$-st order interpolating polynomial, such as (4.8), could be constructed using the $\hat{p}$ starting values which correspond to the step-size $\hat{h}$.*

Having added the value $\hat{h}\hat{\dot{y}}_{m\ell}$ to the set of $\hat{p}$ starting points for these cases having $\hat{h} \neq h$, the $\hat{p}$-th order interpolating polynomial (4.8) can be used after establishing the $\hat{p}$-th order backward difference $\nabla^{\hat{p}}\hat{y}_{m\ell}$. That can be accomplished with

$$\nabla^{\hat{p}}\hat{y}_{m\ell} \approx \hat{p}\{\hat{h}\hat{\dot{y}}_{m\ell} - \sum_{\nu=1}^{\hat{p}-1} \frac{1}{\nu} \nabla^{\nu}\hat{y}_{m\ell}\}. \tag{4.9}$$

Once the difference $\nabla^{\hat{p}}y_{m\ell}$ is obtained from (4.9), it is stored and the program variable IDEL is set to equal $\hat{p} + 1$. The values $\tilde{y}_{m\ell-j}$, $j = 1,\ldots, \tilde{p} - 1$—obtained from (4.8)—together with $\tilde{y}_{m\ell} \equiv \hat{y}_{m\ell}$ are then reformulated as the backward differences $\nabla^{j}\tilde{y}_{m\ell}$, $j = 0, 1,\ldots,\tilde{p} - 1$, in order to ready the algorithm for the repetition of the cycle just rejected.

When the error test fails a second time, the step-size is further reduced by a factor 1/2. However, having earlier stored the difference $\nabla^{\hat{p}}y_{m\ell}$, the new starting values corresponding to this step-size can be obtained by directly evaluating (4.8).

■ Evaluation of the equations (3.20) and (3.21) is referred to, in the program, as the corrector iteration. As the values of β_{ii}/α_{ii} for all constituent formulas of the new set of methods are pre-computed and stored in the array C, at the outset of the corrector iteration, the value of β_{ii}/α_{ii} coresponding to the constituent formula in use is selected. Furthermore, the storage locations for the accumulation of correction values are cleared. Then, the derivative—equivalently, the function—is evaluated by calling the subroutine DIFFUN. Next, a check is made to determine whether an, if any, evaluation of the iteration matrix was ordered. When the value of the parameter IWEVAL equals +1, evaluation of the iteration matrix with a concurrent evaluation of the Jacobian matrix $\underset{\sim}{J}$ is indicated. If an analytic expression for the Jacobian is available, which is indicated by setting the method flag MF to 1, a call to the subroutine PEDERV provides such an evaluation. If this expression is not available, indicated by setting MF to 2, the Jacobian will be computed numerically. This process requires

*This course of action can be pursued by inserting the following line of code immediately after card STNT4050 in STINT (see the listing in the Appendix):

IF(NQ.GT.NQOLD) IDEL=NQ

a careful choice of increments for finite differencing. The increment δ_j which is added to the j-th component of $\underset{\sim}{y}$, j = 1,...,N, to estimate $\partial \underset{\sim}{f}/\partial y_j$ is given by

$$\delta_j = \text{MAX}\{\sqrt{u}w_j, \delta_0\} \tag{4.10}$$

where u is the machine round-off unit (UROUND), w_j is the weight (to be described) for the j-th component of $\| \cdot \|_W$, and $\delta_0 = 10^3 \times |h| \times u \times \| \underset{\sim}{f} \|$. This choice for δ_0 satisfies the demand that the relative error in the iteration matrix be about 0.1% or less. The Jacobian matrix evaluated with MF either 1 or 2 is stored in the array RJ. The evaluation of the iteration matrix $\underset{\sim}{W}^{-1} = \{\underset{\sim}{I} - h\beta_{ii}/\alpha_{ii}\,\underset{\sim}{J}\}^{-1}$ is next performed. Note: When the value of IWEVAL is greater than +1, the evaluation of the Jacobian matrix is by-passed and the iteration matrix is evaluated using an old value of $\underset{\sim}{J}$. The evaluation of the iteration matrix is carried out in two steps. During the first step, the value of $I - h\beta_{ii}/\alpha_{ii}\,\underset{\sim}{J}$ is established. This value is stored in the array RW. During the second step, the inverse of the matrix $\underset{\sim}{W}$ is obtained. Actually, rather than finding an explicit inverse of the matrix $\underset{\sim}{W}$, an LU factorization is obtained, with the factors stored in place of W in the array RW. The product $\underset{\sim}{W}^{-1}[h\underset{\sim}{f}(\underset{\sim}{y}^{(\nu)}(t), t) - \underset{\sim}{d}^{(\nu)}]$, which is needed for computation of the new estimates $\underset{\sim}{y}^{(\nu+1)}$ and $\underset{\sim}{d}^{(\nu+1)}$, simply requires the solution of two triangular systems. The LU decomposition of the matrix $\underset{\sim}{W}$ is carried out by calling the subroutine DEC, while the triangular systems are solved by execution of the subroutine SOL. Each time the iteration matrix is evaluated, that is, its LU decomposition is obtained, the parameter IWEAL is set to -i, where i is the index of the constituent formula which is being used. At the same time, the parameter RC is set to 1. This parameter is used for monitoring the change in the factor $h[\beta_{ii}/\alpha_{ii}]_{avg}$ since the last evaluation of the iteration matrix. The values $[\beta_{ii}/\alpha_{ii}]_{avg} = 1/\ell \sum_{i=1}^{\ell} \beta_{ii}/\alpha_{ii}$ for orders 1 thru 7—accurate to 5 significant digits—are stored in the array PD. If reevaluation of the iteration matrix was not requested, that is, the value of IWEVAL was non-positive the correction term $\underset{\sim}{W}^{-1}[h\underset{\sim}{f}(\underset{\sim}{y}^{(\nu)}(t), t) - \underset{\sim}{d}^{(\nu)}]$ is evaluated, at once, by calling the subroutine SOL. The new estimates $\underset{\sim}{y}^{(\nu+1)}$ and $\underset{\sim}{d}^{(\nu+1)}$ follow directly. Note: In order to recover the zero-th iterated $\underset{\sim}{y}^{(0)}$ and $\underset{\sim}{d}^{(0)}$, the correction factors are accumulated in a storage area set aside for this purpose.

The convergence bound β which appears in the convergence criterion (3.28) is not that given by (3.27b). Since the step-wise error criterion, such as (3.39), tolerates errors $\varepsilon\,\alpha_{ii}/\gamma_{i,p+1}$ the convergence criterion (3.28) should tolerate these errors as well. Thus, it is this error bound which is then decreased by the empirical factor 1/2p+4. Thus, $\beta_{new} = \varepsilon\,\alpha_{ii}/(2p+4)\,\gamma_{i,p+1}$ is the convergence bound used in the program.

5. USAGE OF THE PROGRAM

To use the program STINT, the user must provide: (i) routines which define the problem, that is, subroutines DIFFUN and PEDERV, (ii) other routines which are called by the program, that is, subroutines DEC and SOL, and (iii) a calling program which makes calls to the program STINT. The typical calling program must set the initial solution values and parameters, include calls to the program STINT, and provide the sought output. Although the program STINT is strictly block-oriented—that is, a block of solution values is computed at each call—by the simple addition of an interpolation routine coupled with a few control statements in the calling program, the whole program can be made to behave like an interval-oriented integrator. Examples of both, blockwise as well as interval-oriented integration, are given in the sub-section 5.2. In the following sub-section general explanations regarding the usage of program STINT are given.

5.1 General Explanations

The program STINT integrates a set of N first order ordinary differential equations. A block of three or four solution points, each separated by a step-size H, is computed at each call. The program requires four routines which must be named: DIFFUN, PEDERV, DEC, and SOL. Their calling sequences and functions are described below.

■ Subroutine DIFFUN (N, T, Y, DY) evaluates the derivatives (the functions) of the dependent variables Y(I), I = 1, ... , N at T and stores the results in array DY.

■ Subroutine PEDERV (N, T, Y, RJ, NO) evaluates the partial derivative of $\underset{\sim}{f}(\underset{\sim}{y}(t),t)$ with respect to $\underset{\sim}{y}$ (i.e., the Jacobian matrix) at t=T and $\underset{\sim}{y}=\underset{\sim}{Y}$. It must form an array RJ containing the elements of that matrix according to

$$RJ_{j+(k-1)\times N\emptyset} = \frac{\partial f_j}{\partial y_k}, \qquad 1 \leq j,k \leq N.$$

Note: NØ is the value of N on the first call. If the Jacobian matrix is obtained by differencing, a dummy version of PEDERV must be supplied. One such dummy version could be

```
SUBROUTINE PEDERV (N, T, Y, RJ, NO)
RETURN
END
```

■ Subroutine DEC (N, NO, A, IPIV, IER) performs an LU decomposition of the matrix A. The pivots generated during the decomposition are stored in the array IPIV. If the decomposition is successful, IER should be set to 0, otherwise to +1.

■ Subroutines SOL (N, NO, A, B, IPIV) solves the linear algebraic system $\underset{\sim}{A}\underset{\sim}{x} = \underset{\sim}{b}$, where $\underset{\sim}{b}$ = B and where the matrix A has been already processed by DEC.

The library routines required by the program are DFLOAT, DABS, DMIN1, DMAX1, DSQRT.

The temporary storage space needed by the program must be provided by the user in the arrays IPIV, RJ, RW, and SAVE. The array IPIV holds a vector of the same name. The arrays RJ and RW are used to hold matrices of the same names. The array SAVE is partitioned as follows:

SAVE (J, I) $1 \leq J \leq 8$, $1 \leq I \leq N$ is used to save Y(I, J, 1) in case a step is rejected and, hence, the whole cycle has to be repeated.

SAVE (9, I) $1 \leq I \leq N$ is used to save DY(I, 1) in case a step is rejected, and, hence, the whole cycle has to be repeated.

SAVE (N9 + I, 1) is used to store temporarily the derivatives as they are computed during the solution iteration process. It is accessed as a complete array by SAVE (N9P1, 1). In addition, it is used to hold the (NQ + 1)-st backward difference for the error test. (N9 = N * 9 and N9P1 = N9 + 1).

SAVE (N10 + I, 1) is used to hold the correction terms generated during the solution iteration process (N10 = N * 10).

SAVE (N11 + I, 1) is used to hold the derivatives evaluated at Y(I) + γ and T, where γ is the increment used in the numerical differencing scheme invoked when MF = 2. (N11 = N * 11).

SAVE (N12 + I, 1) is used to hold the derivatives evaluated at Y(I) and T in order to obtain approximate values of the partial derivatives. (N12 = NO * 12).

The calling sequence to program STINT is as follows:

CALL STINT (N, T, Y, DY, SAVE, H, HNEXT, HMIN, HMAX, EPS, YMAX, KFLAG, KNEXT, JSTART, MAXORD, RW, RJ, MF, IPIV).

The arguments in this sequence have the following meaning:

N - the number of first order ordinary differential equations to be integrated. It can be decreased on subsequent calls if the number of active equations decreases, with the first ones being those retained. But, it must not be increased without setting JSTART = 0.

T - the independent variable. On return to the calling program T corresponds to the value of the independent variable for the most forward solution point obtained thus far. On the first call T should be set to the initial value of the independent variable.

Y - an N x 8 x 4 array containing the dependent variables and their backward differences computed during the current cycle. On each call up to four solution points are obtained. The most forward solution point $y_{m\ell}$, m = 0, 1, ... , at the beginning and at the end of each cycle is in Y(I, 1, 1). The solution point next to the most forward point is returned in Y(I, 1, KFLAG). The most backward solution point in the new block is returned in Y(I, 1, 2). The vector of the initial values y_0 is to be supplied on the first call at Y(I, 1, 1), I = 1, ... , N. Y(I, J+1, K) contains the J-th backward difference of the I-th dependent variable for K = 1, ... , KFLAG. If it is desired to interpolate to non-mesh points, these values can be used as follows: If the current step-size is H and the value needed is at T-E, $0 < |E| < |H|$, form S = E/H and then compute

$$Y(I) = \sum_{J=0}^{NQ} Y(I,J+1,K) * B(J),$$

where K, which corresponds to T, is the first mesh point beyond the point T - E, and B(J + 1) = B(J) * (J - S)/(J + 1) with B(0) = 1.

DY - an N x 4 array. DY(I, K), $1 \leq I \leq N$, $1 \leq K \leq$ KFLAG, contains for K=1, ... , KFLAG the computed derivative of the I-th dependent variable scaled by H.

SAVE - a block of a least ~~12~~ 13 * N double precision floating point locations.

H - the step-size used for the block just completed.

HNEXT - the step-size for the next block. On the first call (JSTART=0) the user must specify an initial step-size. A good estimate of its magnitude is given by 0.2*(EPS/ABS(&))**0.5, where & is the largest eigenvalue of the Jacobian matrix evaluated at the initial values of T and Y. The sign of the initial HNEXT should be positive (negative) if the final time is greater (less) than the initial time. If the initial step-size choice does not cause an error greater than EPS, in the RMS NORM, it will be accepted; otherwise, its magnitude will be decreased until an error less than EPS is achieved. The subroutine automatically adjusts the step-size after the initial and subsequent calls for the step-size of largest possible magnitude. The magnitude may be adjusted down on any subsequent call. Note: A magnitude adjustment up or a sign change will be ignored.

HMIN - |HMIN| is the minimum step-size magnitude to be allowed for the next integration cycle. (On the first call (JSTART=0), |HMIN| should be chosen significantly smaller than |HNEXT| so as to avoid start-up problems if the error criterion is not met with the user specified HNEXT.) Note: The

sign of HMIN is ignored. HMIN may be changed on subsequent calls.

HMAX - |HMAX| is the maximum step-size magnitude to be allowed for the next integration cycle. Note: If|HMAX| is less than|HMIN|, then the subroutine functions as though |HMAX| equals |HMIN|. HMAX may be changed on subsequent calls.

EPS - an estimate of the local discretization error, i.e., error committed on taking a partial block or a single step with the method, starting with stored data which are regarded as exact. This has to be distinguished from the global or true error which is the error in any given computed solution value as a result of the accumulation of local errors from all steps taken to reach the given point. Since it is virtually always the latter error the user wants to control, some experimentation on the user's part is necessary to obtain a value of EPS which will essentially bound the global error. It has been observed that EPS should be not larger than 10^{-2}, i.e., local accuracy of at least two significant digits should be demanded.

YMAX - an array of N locations. It determines the nature—mixture of relative and absolute, unless altered by the user—of the error control. After each return to the calling program YMAX(I) contains the maximum of unity and each |Y(I, 1, K)|, K = 1, ... , KFLAG. On the first call, YMAX(I) should be set to the maximum of unity and the magnitude of th I-th component of y_0.

KFLAG - a completion code. If KFLAG is greater than 0, then KFLAG points have been computed. If KFLAG is -2, -3, or -4, than 2, 3 or 4 points, respectively, have been computed with H = HMIN, but the requested error was not achieved. Other values KFLAG can assume are as follows:

-5 - The requested error was smaller than can be handled for this problem.

-6 - Convergence could not be achieved for H $\geq$ HMIN.

-7 - The maximum order specified was too large.

KNEXT- After the initial call (JSTART=0), the value of KNEXT is the number of points to be computed during the next cycle. The value is supplied for information only. The user cannot control the number of points by setting KNEXT.

JSTART- an input indicator with the following meanings:

0 - initialization call.

$\neq$ 0 - continue from the last step.

On return JSTART is set to NQ, the maximum order of the backward

difference available in the Y array. This also corresponds to the order of the method used to compute the just completed block of points.

MAXORD - the maximum order ($1 \leq$ MAXORD ≤ 7) that may be used. Note: If MAXFORD is reset between cycles to a value less than the order determined for the next cycle, then the order may for several cycles exceed MAXORD. However, it cannot exceed the above determined value and, once the order is less than or equal to MAXORD, it cannot then exceed MAXORD.

RJ - a block of at least NØ**2 double precision floating point locations, which contains an estimate of the Jacobian of the differential equation.

RW - a block of at least NØ**2 double precision floating point locations.

IPIV - a block of at least NØ integers.

MF - method flag. It determines the mode by which the partial derivatives are obtained. The values which it can assume are:

1 - Analytic expressions for the partial derivatives are supplied by the user in the subroutine PEDERV.

2 - The partial derivatives are obtained by numerical differencing.

The storage locactions SAVE(N9P1,1) thru SAVE (N13, 1) are for local working storage within STINT. (N9P1 = 9 * N + 1, N13 = 13 * N) Hence, these 4*N storage locations can be used, between the return from STINT and a new call to it, for other purposes.

Statistical quantities which may be useful in determining STINT's efficiency for a particular problem are gathered in the common block STAT. Its contents are as follows:

NSTEP - The cummulative number of successive steps taken so far.

NFE - The cummulative number of calls to the routine DIFFUN.

NJE - The cummulative number of Jacobian evaluations.

NINVS - The cummulative number of LU decomposition.

Moreover, on return from STINT the varialbe H contains the value of step-size successfully used for the last block, and the value of JSTART corresponds to the corresponding order.

Prior to using the program STINT, the constant UROUND — the unit roundoff of the machine — should be set* to conform to the particular machine which will

*See card STNT2200 in the listing of STINT.

execute this program. This unit is defined as the smallest positive number u such that 1. + u ≠ 1. on the machine.

5.2 Examples of Use

Two examples showing how one might use the program STINT are given below. For the first example, the end point of the integration is given and the solution is computed and printed block-by-block, at whatever time points the program finds convenient. On the other hand, for the second problem we give a set of specific time points at which we wish to compute and print the solution.

Problem I

Consider the linear, mildly stiff system of ordinary differential equations with constant coefficients

$$\dot{\underset{\sim}{y}}(t)=\underset{\sim}{A}\underset{\sim}{y}(t) = \begin{bmatrix} -5 & -5 & -45.5 & 44.5 \\ -5 & -5 & -44.5 & 45.5 \\ 45.5 & 44.5 & -5 & 5 \\ -44.5 & -45.5 & 5 & -5 \end{bmatrix} \underset{\sim}{y}(t),\ \underset{\sim}{y}(t_0) = \underset{\sim}{y}_0 = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 1 \end{bmatrix},$$

whose solution is

$$\underset{\sim}{y}(t) = \underset{\sim}{U}\underset{\sim}{g}(t) = \frac{1}{2} \begin{bmatrix} -1 & 1 & 1 & 1 \\ 1 & -1 & 1 & 1 \\ 1 & 1 & -1 & 1 \\ 1 & 1 & 1 & -1 \end{bmatrix} \begin{bmatrix} \sin t \\ \cos t \\ e^{-10t} \cos 90\, t \\ e^{-10t} \sin 90\, t \end{bmatrix}.$$

It can be readily verified that the eigenvalues of $\underset{\sim}{A}$ are $\lambda_i = \{ \pm i, -10 \pm 90\, i\}$. The ratio $\{\max|\lambda_i|\} / \{\min_{\lambda_i \neq 0}|\lambda_i|\}$ often taken as a rough measure of stiffeness is seen to be ~ 90. Suppose we wish to integrate from $t_0 = 0$ to $t_f = 15$ with 3 significant digits of local accuracy (in an absolute sense for components that are smaller than one and remain so and in a relative sense for all components bigger than one). Furthermore, because of the problem's simplicity, the Jacobian matrix will be supplied, and, because there are large (in magnitude) complex eigenvalues near the imaginary

axis and small (in magnitude) imaginary eigenvalues λ_i, the maximum order will be limited to 4. It is desired to print all solution values, block-by-block, on the mesh automatically selected by the program. (We will here also print the global error for each component.) For statistical purposes, the order and step-size used for each block as well as the number of steps, function evaluations, and Jacobian evaluations required through the current block are to be printed at the end of each block. The final error statistics and the accumulated number of steps, function evaluations, Jacobian evaluations, and LU decompositions are to be also provided.

The following calling program combined with the program STINT and its required subroutines accomplishes the tasks set forth.

```
C*                                                                      *MAIN001
C*****        EXAMPLE OF A CALLING PROGRAM FOR PROBLEM I.          ********MAIN002
C*                                                                      *MAIN003
      DOUBLE PRECISION YDOT, YI, YMAX, YO, T, TF, TI, TT, HI, HMAX, HMINMAIN004
     1                 , HO, SAVE, RJ, RW, ERO, ERORM, EPS, DMAX1, DABS,MAIN005
     2                 HNEXT                                           MAIN006
      DIMENSION YDOT(4,4), YI(4), YMAX(4), YO(4,8,4), SAVE(52), RJ(16), MAIN007
     1          RW(16), ERO(4), ERORM(4), IPIV(4)                      MAIN008
      COMMON /STAT/ NSTEP,NFE,NJE,NINVS                               MAIN009
      DATA LIN/5/,LOUT/6/                                             MAIN010
C*                                                                      *MAIN011
C*       SET THE INPUT PARAMETERS :                                     *MAIN012
C*                                                                      *MAIN013
C*       NP       PROBLEM IDENTIFIER                                    *MAIN014
C*       NC       NUMBER OF TIMES THE SAME PROBLEM IS SOLVED WITH       *MAIN015
C*                DIFFERENT VALUE OF EPS                                *MAIN016
C*       N        NUMBER OF DIFFERENTIAL EQUATIONS TO BE SOLVED         *MAIN017
C*       YI       THE INITIAL VALUES OF THE DEPENDENT VARIABLE          *MAIN018
C*       TI       THE INITIAL VALUE OF THE INDEPENDENT VARIABLE         *MAIN019
C*       TF       THE FINAL VALUE OF THE INDEPENDENT VARIABLE           *MAIN020
C*       HI       THE INITIAL STEP-SIZE                                 *MAIN021
C*       MAXORD   THE MAXIMUM ORDER ALLOWED (1.GE.MAXORD.LE.7)          *MAIN022
C*       MF       METHOD FLAG DETERMINING THE MODE OF THE JACOBIAN      *MAIN023
C*                MATRIX EVALUATION                                     *MAIN024
C*       EPS      USER'S SPECIFIED ERROR PER STEP                       *MAIN025
C*                                                                      *MAIN026
      READ(LIN,10) NP                                                  MAIN027
      READ(LIN,10) NC                                                  MAIN028
      READ(LIN,10) N                                                   MAIN029
   10 FORMAT(I2)                                                       MAIN030
      READ(LIN,20) (YI(I),I=1,N)                                       MAIN031
      READ(LIN,20) TI, TF, HI                                          MAIN032
      DO 1000 K=1,NC                                                   MAIN033
      READ(LIN,10) MAXORD                                              MAIN034
      READ(LIN,10) MF                                                  MAIN035
      READ(LIN,20) EPS                                                 MAIN036
   20 FORMAT(3D22.15)                                                  MAIN037
      WRITE(LOUT,30) NP, MAXORD, MF, EPS                               MAIN038
   30 FORMAT('1',10X,13H PROBLEM NO. ,I2//                             MAIN039
     1            11X,18H MAXIMUM ORDER =  ,I2//                      MAIN040
     2            11X,18H METHOD FLAG   =  ,I2//                      MAIN041
     3            11X,18H ERROR PER STEP = ,D10.3////                 MAIN042
     4            5X,4HTIME,15X,29HS O L U T I O N  /  E R R O R,32X, MAIN043
     5            7H H-USED,3X,5HORDER,2X,6H#STEPS,2X,5H#FE'S,2X,     MAIN044
     6            11H#JACS/#LU'S//)                                   MAIN045
C*                                                                      *MAIN046
C*       SET THE NORMALIZING VECTOR YMAX                                *MAIN047
C*       CLEAR THE MAXIMUM ERROR ARRAY - ERORM                          *MAIN048
C*                                                                      *MAIN049
      DO 40 I=1,N                                                      MAIN050
        YMAX(I) = DMAX1(DABS(YI(I)),1.D0)                              MAIN051
        ERORM(I) = 0.D0                                                MAIN052
C*                                                                      *MAIN053
   40 YO(I,1,1)=YI(I)                                                  MAIN054
C*                                                                      *MAIN055
C*       THE FOLLOWING PARAMETERS ARE USED FOR STATISTICAL PURPOSES :   *MAIN056
C*                                                                      *MAIN057
C*       NSTEP    NUMBER OF ACCEPTED STEPS                              *MAIN058
C*       NFE      NUMBER OF FUNCTION EVALUATIONS                        *MAIN059
C*       NJE      NUMBER OF JACOBIAN EVALUATIONS                        *MAIN060
```

```
C*       NINVS   NUMBER OF LU DECOMPOSITIONS OF THE ITERATION MATRIX    *MAIN0610
C*                                                                      *MAIN0620
C*                                                                       MAIN0630
      NSTEP=0                                                           MAIN0640
      NFE=0                                                             MAIN0650
      NJE=0                                                             MAIN0660
      NINVS=0                                                          *MAIN0670
C*                                                                      MAIN0680
      HMAX = TF - TI                                                    MAIN0690
      T =TI                                                             MAIN0700
      HNEXT=HI                                                          MAIN0710
      HMIN=HI*1.D-7                                                     MAIN0720
      JSTART=0                                                         *MAIN0730
C*                                                                      MAIN0740
   50 CALL STINT (N, T, Y0, YDOT, SAVE, H0, HNEXT, HMIN, HMAX, EPS,     MAIN0750
     1            YMAX, KFLAG, KNEXT, JSTART, MAXORD, RW, RJ, MF, IPIV) *MAIN0760
C*                                                                      MAIN0770
      IF(KFLAG.GE.0) GO TO 200                                          MAIN0780
      WRITE(LOUT,60) KFLAG                                              MAIN0790
   60 FORMAT('-',20H COMPLETION CODE IS ,I4)                            MAIN0800
      GO TO 800                                                        *MAIN0810
C*                                                                     *MAIN0820
C*       OUTPUT THE (KFLAG-1) SOLUTION POINTS                          *MAIN0830
C*                                                                      MAIN0840
  200 DO 360 J=2,KFLAG                                                  MAIN0850
        TT = T-H0*DFLOAT(KFLAG+1-J)                                     MAIN0860
        CALL ERROR(TT, Y0(1,1,J), N, ERO)                               MAIN0870
        WRITE(LOUT,320) TT, (Y0(I,1,J),I=1,N), (ERO(I),I=1,N)           MAIN0880
  320   FORMAT( 4H T =,D11.4,4D16.6/15X,4D16.6/)                        MAIN0890
        DO 340 I=1,N                                                    MAIN0900
  340     ERORM(I)=DMAX1(ERORM(I),DABS(ERO(I)))                         MAIN0910
  360   CONTINUE                                                        *MAIN0920
C*                                                                     *MAIN0930
C*       OUTPUT THE MOST FORWARD SOLUTION POINT                        *MAIN0940
C*                                                                      MAIN0950
      CALL ERROR(T, Y0, N, ERO)                                         MAIN0960
      WRITE(LOUT,380) T, (Y0(I,1,1),I=1,N), H0, JSTART, NSTEP, NFE, NJE,MAIN0970
     1                NINVS,(ERO(I),I=1,N),HNEXT                        MAIN0980
  380 FORMAT( 4H T =,D11.4,4D16.6,D14.3,2X,I4,4X,I4,3X,I4,4X,I4,2H /,I4/MAIN0990
     1        15X,4D16.6,D14.3/)                                        MAIN1000
      DO 400 I=1,N                                                      MAIN1010
  400 ERORM(I)=DMAX1(ERORM(I),DABS(ERO(I)))                             MAIN1020
      IF((T-TF)*HNEXT.GT.0.D0) GO TO 800                                MAIN1030
      GO TO 50                                                         *MAIN1040
C*                                                                     *MAIN1050
C*       REGARDLESS OF WHETHER OR NOT THE PROBLEM HAS BEEN SOLVED TO   *MAIN1060
C*       COMPLETION, PRINT OUT STATISTICS                              *MAIN1070
C*                                                                      MAIN1080
  800 WRITE(LOUT,810)                                                   MAIN1090
  810 FORMAT('-',15HSOLUTION ERRORS)                                    MAIN1100
      WRITE(LOUT,820) (ERORM(I),I=1,N)                                  MAIN1110
  820 FORMAT('-',10X,23H MAXIMUM ABSOLUTE ERROR//11X,4D22.8)            MAIN1120
      DO 830 I=1,N                                                      MAIN1130
  830 ERORM(I)=ERORM(I)/YMAX(I)                                         MAIN1140
      WRITE(LOUT,840) (ERORM(I),I=1,N)                                  MAIN1150
  840 FORMAT('-',10X,23H MAXIMUM RELATIVE ERROR//11X,4D22.8)            MAIN1160
      WRITE(LOUT,850) NSTEP, NFE, NJE, NINVS                            MAIN1170
  850 FORMAT('-',22H THE SOLUTION REQUIRED,I5,6H STEPS/                 MAIN1180
     1       23X,I5,21H FUNCTION EVALUATIONS/                           MAIN1190
     2       23X,I5,21H JACOBIAN EVALUATIONS/                           MAIN1200
     3       23X,I5,18H LU DECOMPOSITIONS)
```

```
1000 CONTINUE                                                    MAIN1210
     STOP                                                        MAIN1220
     END                                                         MAIN1230
```

```
      SUBROUTINE DIFFUN (N, T, Y, YDOT)
      DOUBLE PRECISION Y,YDOT,T
      DIMENSION Y(N), YDOT(N)
      YDOT(1)=-5.D0*(Y(1)+Y(2))-45.5D0*Y(3)+44.5D0*Y(4)
      YDOT(2)=-5.D0*(Y(1)+Y(2))-44.5D0*Y(3)+45.5D0*Y(4)
      YDOT(3)=45.5D0*Y(1)+44.5D0*Y(2)-5.D0*(Y(3)-Y(4))
      YDOT(4)=-44.5D0*Y(1)-45.5D0*Y(2)+5.D0*(Y(3)-Y(4))
      RETURN
      END


      SUBROUTINE PEDERV(N, T, Y, P1, ND)
      DOUBLE PRECISION P0,T,Y,P1
      DIMENSION P0(4,4),P1(16),Y(ND)
      DO 10 I=1,2
      DO 10 J=1,2
   10 P0(I,J)=-5.D0
      DO 20 M=1,2
   20 P0(2+M,2+M)=-5.D0
      P0(1,3)=-45.5D0
      P0(1,4)=44.5D0
      P0(2,3)=-44.5D0
      P0(2,4)=45.5D0
      P0(3,1)=P0(2,4)
      P0(3,2)=P0(1,4)
      P0(3,4)=5.D0
      P0(4,1)=P0(2,3)
      P0(4,2)=P0(1,3)
      P0(4,3)=P0(3,4)
      DO 30 I=1,4
      DO 30 J=1,4
   30 P1(J+(I-1)*4)=P0(J,I)
      RETURN
      END


      SUBROUTINE ERROR(T0, Y0, N, ERO)
      DOUBLE PRECISION T0, Y0, ERO, TEMP, W1, W2, W3, W4, DSIN, DCOS
      DOUBLE PRECISION DEXP
      DIMENSION Y0(N), ERO(N)
      W1=DSIN(T0)*5.D-1
      W2=DCOS(T0)*5.D-1
      ERO(1)=-W1+W2
      ERO(2)= W1-W2
      ERO(3)= W1+W2
      ERO(4)= W1+W2
      IF(T0.GE.1.5D1) GO TO 10
      TEMP=DEXP(-1.D1*T0)*5.D-1
      W3=TEMP*DCOS(9.D1*T0)
      W4=TEMP*DSIN(9.D1*T0)
      ERO(1)=ERO(1)+W3+W4
      ERO(2)=ERO(2)+W3+W4
      ERO(3)=ERO(3)-W3+W4
      ERO(4)=ERO(4)+W3-W4
   10 DO 20 I=1,N
   20 ERO(I)=ERO(I)-Y0(I)
      RETURN
      END
```

PROBLEM NO. 10

MAXIMUM ORDER = 4

METHOD FLAG = 1

ERROR PER STEP = 0.100D-02

When run on an IBM 370/155 the output of this program was as follows:

TIME	SOLUTION / ERROR				H-USED	ORDER	#STEPS	#FE'S	#JACS/#LU'S
= 0.5000D-07	0.100000D 01	0.202499D-05	0.252501D-05	0.999998D 00					
	0.612554D-11	0.612439D-11	-0.387433D-11	0.387555D-11					
= 0.1000D-06	0.100000D 01	0.404996D-05	0.505002D-05	0.999995D 00					
	0.122513D-10	0.122488D-10	-0.774859D-11	0.775106D-11					
= 0.1500D-06	0.100001D 01	0.607493D-05	0.757505D-05	0.999993D 00	0.500D-07	1	3	4	1 / 1
	0.183775D-10	0.183733D-10	-0.116228D-10	0.116265D-10	0.412D-03				
= 0.4117D-03	0.101542D 01	0.158291D-01	0.212829D-01	0.979129D 00					
	0.427136D-03	0.427051D-03	-0.234676D-03	0.234761D-03					
= 0.8233D-03	0.102997D 01	0.307940D-01	0.430161D-01	0.957807D 00					
	0.867164D-03	0.866995D-03	-0.435528D-03	0.435698D-03					
= 0.1235D-02	0.104365D 01	0.448888D-01	0.651734D-01	0.936060D 00	0.412D-03	1	6	10	2 / 2
	0.131812D-02	0.131787D-02	-0.602105D-03	0.602360D-03	0.123D-02				
= 0.2469D-02	0.108025D 01	0.827242D-01	0.133410D 00	0.869056D 00					
	0.184307D-02	0.184274D-02	-0.457076D-03	0.457416D-03					
= 0.3704D-02	0.110905D 01	0.112765D 00	0.204271D 00	0.799426D 00					
	0.204789D-02	0.204753D-02	-0.476476D-04	0.480168D-04					
= 0.4939D-02	0.112974D 01	0.134691D 00	0.276991D 00	0.727936D 00	0.123D-02	2	9	16	3 / 3
	0.208518D-02	0.208481D-02	0.449472D-03	-0.449093D-03	0.170D-02				
= 0.6640D-02	0.114484D 01	0.151498D 00	0.378879D 00	0.627738D 00					
	0.187970D-02	0.187933D-02	0.906025D-03	-0.905651D-03					
= 0.8342D-02	0.114436D 01	0.152739D 00	0.480414D 00	0.527892D 00					
	0.157535D-02	0.157499D-02	0.119677D-02	-0.119640D-02					
= 0.1004D-01	0.112907D 01	0.139159D 00	0.579124D 00	0.430868D 00	0.170D-02	3	12	21	3 / 3
	0.948169D-03	0.947832D-03	0.145705D-02	-0.145670D-02	0.236D-02				
= 0.1241D-01	0.108408D 01	0.965600D-01	0.707683D 00	0.304646D 00					
	0.333522D-03	0.333196D-03	0.160022D-02	-0.159988D-02					
= 0.1477D-01	0.101456D 01	0.294370D-01	0.821468D 00	0.193192D 00					
	-0.390090D-04	-0.393334D-04	0.156848D-02	-0.156814D-02					
= 0.1713D-01	0.924854D 00	-0.578645D-01	0.916090D 00	0.100897D 00	0.236D-02	4	15	25	3 / 4
	-0.331024D-03	-0.331348D-03	0.142184D-02	-0.142150D-02	0.236D-02				
= 0.1950D-01	0.819888D 00	-0.160424D 00	0.988200D 00	0.311070D-01					
	-0.575714D-03	-0.576037D-03	0.123757D-02	-0.123722D-02					

T = 0.2186D-01	0.7049930 00	-0.2729070 00	0.1035650 01	-0.1403320-01					
	-0.7668700-03	-0.7671910-03	0.1045460-02	-0.1045120-02					
T = 0.2423D-01	0.5856540 00	-0.3898280 00	0.1057560 01	-0.3363130-01	0.2360-02	4	18	28	3 / 4
	-0.8576280-03	-0.8579480-03	0.8210760-03	-0.8207340-03	0.3600-02				
T = 0.2782D-01	0.4078170 00	-0.5639770 00	0.1042700 01	-0.1526340-01					
	-0.7833790-03	-0.7836970-03	0.5487670-03	-0.5484260-03					
T = 0.3142D-01	0.2500170 00	-0.7180760 00	0.9751440 00	0.5577550-01					
	-0.5479210-03	-0.5482370-03	0.4279470-03	-0.4276050-03					
T = 0.3501D-01	0.1264550 00	-0.8379250 00	0.8653680 00	0.1690260 00	0.3600-02	4	21	34	3 / 5
	0.3695920-04	0.3664450-04	0.6838660-03	-0.6835230-03	0.3600-02				
T = 0.3861D-01	0.4763050-01	-0.9130230 00	0.7277240 00	0.3101310 00					
	0.3531510-03	0.3528370-03	0.1144030-02	-0.1143680-02					
T = 0.4221D-01	0.1849120-01	-0.9384240 00	0.5734450 00	0.4628580 00					
	0.1398950-03	0.1395830-03	0.1583170-02	-0.1582820-02					
T = 0.1157D 02	0.7019530 00	-0.7018700 00	-0.1467530 00	-0.1466770 00					
	-0.1063310-01	0.1055000-01	-0.1829140-02	-0.1906030-02					
T = 0.1183D 02	0.7162630 00	-0.7162920 00	0.3843070-01	0.3836020-01					
	-0.9961880-02	0.9990770-02	-0.4684540-02	-0.4614090-02					
T = 0.1209D 02	0.6827240 00	-0.6827070 00	0.2211370 00	0.2212230 00	0.2590 00	4	540	601	5 / 14
	-0.8719240-02	0.8702140-02	-0.7221080-02	-0.7403630-02	0.2590 00				
T = 0.1235D 02	0.6033250 00	-0.6032980 00	0.3892290 00	0.3892180 00					
	-0.6731860-02	0.6704110-02	-0.9655630-02	-0.9644580-02					
T = 0.1261D 02	0.4832770 00	-0.4832940 00	0.5311790 00	0.5311760 00					
	-0.4028930-02	0.4045600-02	-0.1125520-01	-0.1125190-01					
T = 0.1287D 02	0.3307920 00	-0.3307620 00	0.6378110 00	0.6377040 00	0.2590 00	4	543	606	5 / 14
	-0.9684050-03	0.9378720-03	-0.1233830-01	-0.1223160-01	0.2590 00				
T = 0.1313D 02	0.1559740 00	-0.1559910 00	0.7015920 00	0.7015800 00					
	0.2346460-02	-0.2331540-02	-0.1243750-01	-0.1242530-01					
T = 0.1339D 02	-0.2937170-01	0.2938130-01	0.7182580 00	0.7182670 00					
	0.5594680-02	-0.5604230-02	-0.1155060-01	-0.1156060-01					

T = 0.1364D 02	-0.213028D 00	0.212965D 00	0.686860D 00	0.686908D 00	0.259D 00	4	546	610	5 / 14
	0.874319D-02	-0.868051D-02	-0.990512D-02	-0.995299D-02	0.291D 00				
T = 0.1394D 02	-0.401470D 00	0.401443D 00	0.596987D 00	0.597017D 00					
	0.115714D-01	-0.115453D-01	-0.708957D-02	-0.711923D-02					
T = 0.1423D 02	-0.556150D 00	0.556157D 00	0.456672D 00	0.456673D 00					
	0.134140D-01	-0.134209D-01	-0.341962D-02	-0.342070D-02					
T = 0.1452D 02	-0.664379D 00	0.664365D 00	0.277666D 00	0.277664D 00	0.291D 00	4	549	616	5 / 14
	0.144296D-01	-0.144157D-01	0.840096D-03	0.841744D-03	0.291D 00				
T = 0.1481D 02	-0.716592D 00	0.716595D 00	0.750784D-01	0.750863D-01					
	0.140650D-01	-0.140677D-01	0.526842D-02	0.526056D-02					
T = 0.1510D 02	-0.708215D 00	0.708221D 00	-0.133959D 00	-0.133961D 00					
	0.121669D-01	-0.121726D-01	0.939258D-02	0.939462D-02					
T = 0.1539D 02	-0.640266D 00	0.640257D 00	-0.332163D 00	-0.332165D 00	0.291D 00	4	552	622	5 / 14
	0.920776D-02	-0.919943D-02	0.131549D-01	0.131574D-01	0.291D 00				

SOLUTION ERRORS

MAXIMUM ABSOLUTE ERROR

0.14429557D-01 0.14415688D-01 0.13154902D-01 0.13157372D-01

MAXIMUM RELATIVE ERROR

0.12604040D-01 0.14415688D-01 0.12438895D-01 0.13157372D-01

THE SOLUTION REQUIRED 552 STEPS
622 FUNCTION EVALUATIONS
5 JACOBIAN EVALUATIONS
14 LU DECOMPOSITIONS

Problem II

Consider the linear, time-varying, system of ordinary differential equations

$$\dot{\underset{\sim}{y}}(t) = \underset{\sim}{V}\,\underset{\sim}{B}\,\underset{\sim}{V}^{T}\,\underset{\sim}{y}(t) + \underset{\sim}{V}\,\underset{\sim}{z}(t),$$

where

$$\underset{\sim}{V} = \frac{1}{3}\begin{bmatrix} 2 & 2 & 1 \\ -2 & 1 & 2 \\ 1 & -2 & 2 \end{bmatrix}, \quad \underset{\sim}{B} = \begin{bmatrix} -100-\sin t, & -1000, & 0 \\ 1000, & -100-\sin t, & 0 \\ 0, & 0, & -1 \end{bmatrix}, \quad \underset{\sim}{z}(t) = \begin{bmatrix} 0 \\ 0 \\ \dfrac{t}{(t+1)^2} \end{bmatrix},$$

with

$$\underset{\sim}{y}(t_0) = \underset{\sim}{y}_0 = \frac{1}{3}\,[2 + 2e,\ 4 - 2e,\ 4 + e]^{T},$$

whose solution is

$$\underset{\sim}{g}(t) = \underset{\sim}{V}\,\underset{\sim}{g}(t), \quad \text{where } \underset{\sim}{g}(t) = \begin{bmatrix} e^{-100t + \cos t}\cos 1000t \\ e^{-100t + \cos t}\sin 1000t \\ e^{-t} + \dfrac{1}{1+t} \end{bmatrix}.$$

The eigenvalues of the Jacobian matrix are seen to be $\lambda_i(t) = \{-1, -(100 + \sin t) \pm 1000_i\}$. The stiffness ratio is roughly 10^3. We wish to integrate from $t_0 = 0$ to $t_f = 100$ with 3 significant digits of local accuracy as in Problem 1. The Jacobian matrix is to be obtained by numerical differencing and, because there are large (in magnitude) complex eigenvalues $\lambda_i(t)$ near the imaginary axis, the maximum order is to be limited to 4. Furthermore, solution values are to be printed at the set of points $\mathcal{T} = \{100/2^k,\ k = 0, 1, \ldots, 20\}$. Also to be printed are the global error at the output points and the statistics of step-size and order and of the number of steps, function evaluations, Jacobian evaluations, and LU decompositions needed for integration up to the specified output point.

The following calling program combined with the program STINT and its required subroutines yields the desired outputs.

```
C*                                                                      *MAINC
C*******     EXAMPLE OF A CALLING PROGRAM FOR PROBLEM II     **********MAINC
C*                                                                      *MAINC
      DOUBLE PRECISION YDOT, YI, YMAX, YOUT, Y0, T, TF, TI, TOUT, TOUTP,MAINC
     1                 TS, HI, HMAX, HMIN, H0, SAVE, RJ, RW, ERO, ERORM,MAINC
     2                 EPS, D, S, DABS, DFLOAT, DMAX1, HNEXT             MAINC
      DIMENSION YDOT(3,4), YI(3), YMAX(3), YOUT(3), Y0(3,8,4), SAVE(39),MAINC
     1          RJ(9), RW(9), ERO(3), ERORM(3), IPIV(3)                  MAINC
      COMMON /STAT/ NSTEP,NFE,NJE,NINVS                                 MAINC
      DATA LIN/5/,LOUT/6/                                               MAINC
C*                                                                      *MAINC
C*       SET THE INPUT PARAMETERS :                                     *MAINC
C*                                                                      *MAINC
C*       NP       PROBLEM IDENTIFIER                                    *MAINC
C*       NC       NUMBER OF TIMES THE SAME PROBLEM IS SOLVED WITH       *MAINC
C*                DIFFERENT VALUE OF EPS                                *MAINC
C*       IOP      THE NUMBER OF GEOMETRICALLY SPACED PRINTED OUTPUT     *MAINC
C*                POINTS FROM TI THROUGH TF                             *MAINC
C*       N        NUMBER OF DIFFERENTIAL EQUATIONS TO BE SOLVED         *MAINC
C*       YI       THE INITIAL VALUES OF THE DEPENDENT VARIABLE          *MAINC
C*       TI       THE INITIAL VALUE OF THE INDEPENDENT VARIABLE         *MAINC
C*       TF       THE FINAL VALUE OF THE INDEPENDENT VARIABLE           *MAINC
C*       HI       THE INITIAL STEP-SIZE                                 *MAINC
C*       MAXORD   THE MAXIMUM ORDER ALLOWED (1.GE.MAXORD.LE.7)          *MAINC
C*       MF       METHOD FLAG DETERMINING THE MODE OF THE JACOBIAN      *MAINC
C*                MATRIX EVALUATION                                     *MAINC
C*       EPS      USER'S SPECIFIED ERROR PER STEP                       *MAINC
C*                                                                      *MAINC
      READ(LIN,10) NP                                                   MAINC
      READ(LIN,10) NC                                                   MAINC
      READ(LIN,10) IOP                                                  MAINC
      READ(LIN,10) N                                                    MAINC
   10 FORMAT(I2)                                                        MAINC
      READ(LIN,20) (YI(I),I=1,N)                                        MAINC
      READ(LIN,20) TI, TF, HI                                           MAINC
      DO 1000 K=1,NC                                                    MAINC
      READ(LIN,10) MAXORD                                               MAINC
      READ(LIN,10) MF                                                   MAINC
      READ(LIN,20) EPS                                                  MAINC
   20 FORMAT(3D22.15)                                                   MAINC
      WRITE(LOUT,30) NP, MAXORD, MF, EPS                                MAINC
   30 FORMAT('1',10X,13H PROBLEM NO. ,I2//                              MAINC
     1           11X,18H MAXIMUM ORDER =  ,I2//                         MAINC
     2           11X,18H METHOD FLAG   =  ,I2//                         MAINC
     3           11X,18H ERROR PER STEP = ,D10.3////                    MAINC
     4           5X,4HTIME,15X,29HS O L U T I O N  /  E R R O R,32X,    MAINC
     5           7H H-USED,3X,5HORDER,2X,6H#STEPS,2X,5H#FE'S,2X,        MAINC
     6           11H#JACS/#LU'S//)                                      MAINC
C*                                                                      *MAINC
C*       SET THE NORMALIZING VECTOR YMAX                                *MAINC
C*       CLEAR THE MAXIMUM ERROR ARRAY - ERORM                          *MAINC
C*                                                                      *MAINC
      DO 40 I=1,N                                                       MAINC
        YMAX(I) = DMAX1(DABS(YI(I)),1.D0)                               MAINC
        ERORM(I) = 0.D0                                                 MAINC
C*                                                                      *MAINC
   40 Y0(I,1,1)=YI(I)                                                   MAINC
C*                                                                      *MAINC
C*       THE FOLLOWING PARAMETERS ARE USED FOR STATISTICAL PURPOSES :   *MAINC
C*                                                                      *MAINC
```

```
C*        NSTEP   NUMBER OF ACCEPTED STEPS                              *MAIN0610
C*        NFE     NUMBER OF FUNCTION EVALUATIONS                        *MAIN0620
C*        NJE     NUMBER OF JACOBIAN EVALUATIONS                        *MAIN0630
C*        NINVS   NUMBER OF LU DECOMPOSITIONS OF THE ITERATION MATRIX   *MAIN0640
C*                                                                      *MAIN0650
      NSTEP=0                                                           MAIN0660
      NFE=0                                                             MAIN0670
      NJE=0                                                             MAIN0680
      NINVS=0                                                           MAIN0690
C*                                                                      *MAIN0700
C*        SET THE INDEPENDENT VARIABLE FOR THE FIRST OUTPUT POINT AT    *MAIN0710
C*        TOUT. THE VALUE OF THE INDEPENDENT VARIABLE FOR THE LAST      *MAIN0720
C*        OUTPUT POINT IS AT TOUTP. SET THE MAXIMUM STEP-SIZE HMAX TO   *MAIN0730
C*        EQUAL THE DISTANCE BETWEEN THE NEXT AND THE LAST OUTPUT       *MAIN0740
C*        POINTS MULTIPLIED BY 10.                                      *MAIN0750
C*                                                                      *MAIN0760
      TOUT =TI+(TF-TI)/2.D0**IOP                                        MAIN0770
      TOUTP =TI                                                         MAIN0780
      HMAX =(TOUT-TOUTP)*1.D1                                           MAIN0790
      T =TI                                                             MAIN0800
      HNEXT=HI                                                          MAIN0810
      HMIN=HI*1.D-7                                                     MAIN0820
      JSTART=0                                                          MAIN0830
C*                                                                      *MAIN0840
   50 CALL STINT (N, T, Y0, YDOT, SAVE, H0, HNEXT, HMIN, HMAX, EPS,     MAIN0850
     1             YMAX, KFLAG, KNEXT, JSTART, MAXORD, RW, RJ, MF, IPIV) MAIN0860
C*                                                                      *MAIN0870
      IF(KFLAG.GE.0) GO TO 200                                          MAIN0880
      WRITE(LOUT,60) KFLAG                                              MAIN0890
   60 FORMAT('-',20H COMPLETION CODE IS ,I4)                            MAIN0900
      GO TO 800                                                         MAIN0910
C*                                                                      *MAIN0920
C*        CHECK WHETHER ONE OF THE NEWLY COMPUTED SOLUTION POINTS       *MAIN0930
C*        REACHED BEYOND THE OUTPUT POINT CORRESPONDING TO TOUT.        *MAIN0940
C*                                                                      *MAIN0950
  200 KFP1 = KFLAG + 1                                                  MAIN0960
  220 IND = KFLAG + 3                                                   MAIN0970
      DO 300 I=1,KFP1                                                   MAIN0980
        IND = IND - 1                                                   MAIN0990
        TS = TOUT-T+H0*DFLOAT(I-1)                                      MAIN1000
        IF (TS.LT.0.D0) GO TO 300                                       MAIN1010
        IF (I-2) 50, 400, 420                                           MAIN1020
  300 CONTINUE                                                          MAIN1030
C*                                                                      *MAIN1040
C*        THE NEWLY COMPUTED SOLUTION POINTS REACHED BEYOND TOUT.       *MAIN1050
C*        THE OUTPUT VALUE AT TOUT IS OBTAINED BY INTERPOLATION WITH    *MAIN1060
C*        BACKWARD DIFFERENCES CORRESPONDING TO THE COMPUTED POINT       *MAIN1070
C*        Y0(I,J,IND), J=1,...,NQ.                                      *MAIN1080
C*                                                                      *MAIN1090
  400 IND = 1                                                           MAIN1100
  420 S = (TS-H0)/H0                                                    MAIN1110
      DO 440 I=1,N                                                      MAIN1120
        D = 1.D0                                                        MAIN1130
        YOUT(I) = Y0(I,1,IND)                                           MAIN1140
        DO 440 J=1,JSTART                                               MAIN1150
          D = D*(S+DFLOAT(J-1))/DFLOAT(J)                               MAIN1160
  440 YOUT(I) = YOUT(I)+D*Y0(I,J+1,IND)                                 MAIN1170
C*                                                                      *MAIN1180
C*        PRINT OUT THE VALUE (TOUT,YOUT) , THE ABSOLUTE ERROR, AND THE *MAIN1190
C*        STATISTICS. DETERMINE THE MAXIMUM ERROR COMMITTED SO FAR.     *MAIN1200
```

```
C*                                                                      *MAIN12
      CALL ERROR(TOUT, YOUT, N, ERO)                                     MAIN12
      WRITE(LOUT,500) TOUT, (YOUT(I),I=1,N), HO, JSTART, NSTEP, NFE,     MAIN12
     1                NJE, NINVS, (ERO(I),I=1,N)                         MAIN12
  500 FORMAT( 4H T =,D11.4,3D16.6,16X,D14.3,2X,I4,4X,I4,3X,I4,4X,I4,2H /MAIN12
     1        ,I4/15X,3D16.6/)                                           MAIN12
      DO 520 I=1,N                                                       MAIN12
  520 ERORM(I)=DMAX1(DABS(ERO(I)),ERORM(I))                              MAIN12
C*                                                                      *MAIN12
C*       SET THE NEW OUTPUT POINT AND CHECK WHETHER ITS VALUE CAN BE    *MAIN13
C*       OBTAINED BY INTERPOLATION USING THE NEWLY COMPUTED BLOCK OF    *MAIN13
C*       SOLUTION POINTS. IF NOT, SET NEW VALUE OF HMAX.                *MAIN13
C*                                                                      *MAIN13
      TOUTP = TOUT                                                       MAIN13
      TOUT = TI+(TOUT-TI)*2.D0                                           MAIN13
      IF((TOUT-TF)*HNEXT.GT.0.D0) GO TO 800                              MAIN13
      IF((TOUT-T)*HNEXT.LE.0.D0) GO TO 220                               MAIN13
      HMAX = (TOUT-TOUTP)*1.D1                                           MAIN13
      GO TO 50                                                           MAIN13
C*                                                                      *MAIN14
C*       REGARDLESS OF WHETHER OR NOT THE PROBLEM HAS BEEN SOLVED TO    *MAIN14
C*       COMPLETION, PRINT OUT STATISTICS                               *MAIN14
C*                                                                      *MAIN14
  800 WRITE(LOUT,810)                                                    MAIN14
  810 FORMAT('-',15HSOLUTION ERRORS)                                     MAIN14
      WRITE(LOUT,820) (ERORM(I),I=1,N)                                   MAIN14
  820 FORMAT('-',10X,23H MAXIMUM ABSOLUTE ERROR//11X,3D22.8)             MAIN14
      DO 830 I=1,N                                                       MAIN14
  830 ERORM(I)=ERORM(I)/YMAX(I)                                          MAIN14
      WRITE(LOUT,840) (ERORM(I),I=1,N)                                   MAIN15
  840 FORMAT('-',10X,23H MAXIMUM RELATIVE ERROR//11X,3D22.8)             MAIN15
      WRITE(LOUT,850) NSTEP, NFE, NJE, NINVS                             MAIN15
  850 FORMAT('-',22H THE SOLUTION REQUIRED,I5,6H STEPS/                  MAIN15
     1           23X,I5,21H FUNCTION EVALUATIONS/                        MAIN15
     2           23X,I5,21H JACOBIAN EVALUATIONS/                        MAIN15
     3           23X,I5,18H LU DECOMPOSITIONS)                           MAIN15
 1000 CONTINUE                                                           MAIN15
      STOP                                                               MAIN15
      END                                                                MAIN15
```

```
      SUBROUTINE DIFFUN (N, T, Y, YDOT)
      DOUBLE PRECISION T, Y, YDOT, A, DSIN, S1, S2, S3, S4
      DIMENSION Y(N), YDOT(N)
      A=-1.D2-DSIN(T)
      S1=(-2.D0*A+5.998D3)
      S2=(-2.D0*A-6.002D3)
      S3=(5.D0*A-4.D0)
      S4=T/((T+1.D0)**2*3.D3)
      YDOT(1)=((8.D0*A-1.D0)*Y(1)+S2*Y(2)+S1*Y(3))/9.D0+S4
      YDOT(2)=(S1*Y(1)+S3*Y(2)+(-4.D0*A-3.004D3)*Y(3))/9.D0+2.D0*S4
      YDOT(3)=(S2*Y(1)+(-4.D0*A+2.996D3)*Y(2)+S3*Y(3))/9.D0+2.D0*S4
      RETURN
      END


      SUBROUTINE PEDERV (N, T, Y, P, NP)
      RETURN
      END


      SUBROUTINE ERROR (T, Y, N, ERR)
      DOUBLE PRECISION T, Y, ERR, X, E1, Y1, Y2, Y3
      DOUBLE PRECISION DSIN, DCOS, DEXP
      DIMENSION Y(N), ERR(N)
      X=1.D2*T-DCOS(T)
      IF(X.GT.1.7D2) GO TO 1
      E1=DEXP(-X)
      X=1.D3*T
      Y1=E1*DCOS(X)
      Y2=E1*DSIN(X)
      GO TO 2
    1 Y1=0.D0
      Y2=0.D0
    2 Y3=DEXP(-T)+1.D0/(T+1.D3)
      ERR(1)=(2.D0*(Y1+Y2)+Y3)/3.D3
      ERR(2)=(2.D0*(Y3-Y1)+Y2)/3.D6
      ERR(3)=(2.D0*(Y3-Y2)+Y1)/3.D6
      DO 3 I=1,N
    3 ERR(I)=ERR(I)-Y(I)
      RETURN
      END
```

PROBLEM NO. 8

MAXIMUM ORDER = 4

METHOD FLAG = 2

ERROR PER STEP = 0.100D-02

When run on an IBM 370/155 the output of this program was as follows:

TIME	S O L U T I O N / E R R O R			H-USED	ORDER	#STEPS	#FE'S	#JACS/#LU'S
0.9537D-04	0.262180D 01	-0.366585D 00	0.205540D 01	0.765D-04	2	9	26	3 / 3
	0.255868D-02	-0.157765D-02	0.298303D-03					
0.1907D-03	0.274596D 01	-0.242542D 00	0.186899D 01	0.765D-04	2	9	26	3 / 3
	0.335694D-02	-0.155426D-02	-0.124215D-03					
0.3815D-03	0.293101D 01	0.388627D-01	0.149449D 01	0.112D-03	2	12	32	3 / 4
	0.377441D-02	-0.311316D-03	-0.157589D-02					
0.7629D-03	0.303787D 01	0.695734D 00	0.783043D 00	0.191D-03	3	15	38	3 / 5
	0.225302D-02	0.309730D-02	-0.422382D-02					
0.1526D-02	0.229609D 01	0.203533D 01	-0.187942D 00	0.211D-03	4	18	43	3 / 5
	-0.641873D-02	0.320084D-02	0.351349D-05					
0.3052D-02	-0.546151D 00	0.272274D 01	0.541197D 00	0.339D-03	4	24	52	3 / 6
	0.427630D-03	-0.338145D-02	0.316763D-02					
0.6104D-02	0.147750D 01	0.262902D 00	0.198012D 01	0.339D-03	4	33	70	3 / 6
	-0.223400D-01	0.593648D-02	0.523352D-02					
0.1221D-01	0.100666D 01	0.709386D 00	0.175099D 01	0.339D-03	4	51	98	3 / 6
	-0.356037D-01	0.133701D-01	0.443178D-02					
0.2441D-01	0.692161D 00	0.110771D 01	0.147429D 01	0.439D-03	4	84	134	3 / 6
	-0.266108D-01	0.230656D-01	-0.976019D-02					
0.4883D-01	0.617883D 00	0.124977D 01	0.129997D 01	0.613D-03	4	129	179	3 / 7
	0.561431D-02	0.121379D-01	-0.149450D-01					
0.9766D-01	0.607111D 00	0.121854D 01	0.120490D 01	0.976D-03	4	192	242	3 / 8
	-0.124075D-02	-0.645516D-02	0.707553D-02					
0.1953D 00	0.553657D 00	0.110731D 01	0.110731D 01	0.308D-01	1	222	280	5 / 13
	-0.597146D-03	-0.119429D-02	-0.119424D-02					
0.3906D 00	0.466073D 00	0.932146D 00	0.932146D 00	0.923D-01	2	225	287	6 / 14
	-0.828001D-03	-0.165601D-02	-0.165602D-02					
0.7813D 00	0.340149D 00	0.680298D 00	0.680298D 00	0.157D 00	3	228	290	6 / 15
	-0.403404D-03	-0.806808D-03	-0.806807D-03					
0.1563D 01	0.201024D 00	0.402048D 00	0.402048D 00	0.225D 00	3	231	293	6 / 16
	-0.107213D-02	-0.214426D-02	-0.214426D-02					
0.3125D 01	0.955591D-01	0.191118D 00	0.191118D 00	0.343D 00	4	237	300	6 / 16
	-0.105366D-03	-0.210706D-03	-0.210602D-03					

T = 0.6250D 01	0.459675D-01	0.919354D-01	0.919364D-01	0.686D 00	4	243	310	7 / 17
	0.652967D-03	0.130557D-02	0.130460D-02					
T = 0.1250D 02	0.250157D-01	0.5003170-01	0.500324D-01	0.351D 01	2	249	324	9 / 19
	-0.323123D-03	-0.646527D-03	-0.647170D-03					
T = 0.2500D 02	0.129959D-01	0.259918D-01	0.259918D-01	0.661D 01	3	252	328	9 / 20
	-0.175412D-03	-0.350819D-03	-0.350818D-03					
T = 0.5000D 02	0.6859050-02	0.1371810-01	0.1371810-01	0.113D 02	2	255	336	10 / 21
	-0.323103D-03	-0.646207D-03	-0.646203D-03					
T = 0.1000D 03	0.332964D-02	0.665928D-02	0.665928D-02	0.181D 02	1	258	343	11 / 22
	-0.293088D-04	-0.586176D-04	-0.586176D-04					

SOLUTION ERRORS

MAXIMUM ABSOLUTE ERROR

|0.356037160-01	|0.230656690-01	|0.149450410-01

MAXIMUM RELATIVE ERROR

0.117050730-01	0.800274410-02	0.588460410-02

THE SOLUTION REQUIRED 258 STEPS
343 FUNCTION EVALUATIONS
11 JACOBIAN EVALUATIONS
22 LU DECOMPOSITIONS

It is recognized that special situations may require that the user make some modifications to the program STINT to further increase its effectiveness for his particular problem. Some of the modifications which might be appropriate are outlined in the next section.

6. PROGRAM MODIFICATIONS

Although the program STINT has been found to solve accurately and efficiently a wide class of problems, situations may arise, when some modification of the program might enhance its performance for a particular problem or a family of problems. All such modifications require some familiarity with the internal structure of the program which may be best attained by reading sections 2 thru 4.

The following list of some modifications is provided so that a user can choose or extend the most appropriate one(s) for his problem. All changes should be done with great caution, especially the ones which replace heuristically established features in the program.

(1) As discussed earlier, the nature of the error control is determined by values contained in the array YMAX. This array must be first initialized in the calling program; then it is updated within the program STINT (statement 1420) each time a new block of solution values is accepted. Some alternatives to the existing error control might be (i) a purely relative error control or (ii) a mixture of relative and absolute error control of the type

$$|\text{estimated local error}| \leq \text{relative error} * |\text{solution}| + \text{absolute error}.$$

To achieve either of these or any other type of error control, YMAX must be set by the user in the calling program. Note: YMAX must be always positive.

(2) The choice of the maximum order MAXORD effects the efficiency of the program STINT. At low accuracy requirements — large EPS values, such as 0.01 — efficiency is often improved if MAXORD is set to a relatively small value such as 4. However, MAXORD should increase with increasing stringency in accuracy. Furthermore, if an estimate of the position of the largest (in magnitude) complex eigenvalue is available, it should be used in conjunction with the Table 2 to loosely bound the value of MAXORD, as was done in the examples. Note that the existing dimensions of the arrays SAVE, Y, and DY assume MAXORD = 7, its maximum value. If the storage space is at a premium, and if MAXORD is reduced from its maximum value, the dimensions of these arrays might be also reduced.

(3) For some probelms the modified Newton method is too powerful

and, consequently, too expensive. It can be replaced by, among others, (i) a functional (Picard) iteration, or (ii) a modified Newton method using a digonal approximation to the Jacobian. Comments on these two follow: (i) A functional iteration ensues when the iteration matrix $\underset{\sim}{W}^{-1}$ is replaced by the unit matrix of the same size. Consequently, no inversion (or LU decomposition) is required; this may reduce considerably the computation cost. On the other hand, for convergence of this type of iteration, the step-size h must be small. Therefore, it might take a much longer time to integrate to completion and, thus the reduction in the cost by not performing the inversion might never materialize. This is especially true for probelms with large stiffness ratios. On the other hand, for problems which are mildly stiff or not stiff at all, the running time might be decreased.

(ii) A considerably less costly modified Newton method which still retains some of the convergence power of that used is one which approximates the Jacobian matrix $\underset{\sim}{J}$ by an appropriately chosen diagonal matrix. In this case, the iteration matrix $\underset{\sim}{W}^{-1}$ becomes $\underset{\sim}{W}^{-1} = (\underset{\sim}{I} - h \frac{\beta_{ii}}{\alpha_{ii}} \underset{\sim}{D})^{-1}$, which is easy to compute directly, thus avoiding, again, an LU decomposition. One choice for the diagonal elements of $\underset{\sim}{D}$ might be the directional derivative of $\underset{\sim}{f}$.

(4) There are a number of heuristically established parameters whose final value was obtained largely after "fine tuning" based on observed and various reported experiences. These are, among others:

- The number of corrector iterations for a single step is set to 3 in the loop DO 780.
- If convergence was not achieved after 3 iterations and the iteration matrix has been freshly evaluated, the step-size is reduced, below statement 800, by a multiplicative factor of 3/10.
- The number of error test failures for any single step is accumulated in the variable IFAIL. If there are more than two consecutive error test failures in one block the integration process is restarted. The test for this is performed below statement 1030.
- After the second error test failure, the step-size is reduced by a multiplicative factor of 1/2. This is done below statement 1030.
- For the restart below statement 1080, the step-size is further reduced, if possible, by a multiplicative factor of 1/10.

- In choosing new values of the step-size, a factor of 1.2 for all three orders p - 1, p, and p + 1, respectively, is used to make the choice somewhat conservative. This factor appears in statement 1200, and below statements 1220 and 1260.
- The entry to the step-size and order control segment is precluded every other block for blocks of order 4 or higher. This is accomplished by the use of the switch variable ISW1 which gets set above statement 1200.
- The entry to the step-size and order control segment is precluded, furthermore, if convergence or error test problems during the current block were experienced. This is done by setting the switch variable ISW2 to 1 after such test failures.
- The convergence bound constant BND is smaller than the error test constant EPS by a factor 2(p+2). The inverse of this factor is set in the variable ENQUP below statement 260.
- The estimated convergence rate CRATE is allowed to decrease by a factor at most of 0.9 on any one iteration, and is used only with a multiplicative factor of 2 in the convergence test. This makes the look ahead feature somewhat conservative. Both factors appear below statement 760.
- There are several criteria for updating the matrices $\underset{\sim}{W}$ and $\underset{\sim}{J}$. If the changes in $h[\beta_{ii}/\alpha_{ii}]_{avg}$ are less than 40% since the last evaluation of the iteration matrix $\underset{\sim}{W}^{-1}$, no reevaluation of $\underset{\sim}{W}^{-1}$ takes place. If these changes are greater than 40% but less than 80%. $\underset{\sim}{W}^{-1}$ is evaluated without concurrent evaluation of $\underset{\sim}{J}$. This is indicated by setting the variable IWEVAL to +2. For changes greater than 80%, both $\underset{\sim}{J}$ and $\underset{\sim}{W}^{-1}$ are evaluated. To indicate this request, IWEVAL is set to +1. These criteria are tested and appropriate values of IWEVAL set below statement 300. In addition, after the first or the second error test failure the value of IWEVAL is set to +2, indicating a request for evaluation of $\underset{\sim}{W}^{-1}$ without concurrent evaluation of $\underset{\sim}{J}$. The same is requested after a convergence test failure, if the matrix $\underset{\sim}{W}^{-1}$ was freshly evaluated, Otherwise, evaluation of both, $\underset{\sim}{J}$ and $\underset{\sim}{W}^{-1}$ is indicated by setting IWEVAL to +1. Each time the iteration matrix $\underset{\sim}{W}^{-1}$ is evaluated, IWEVAL is set to -i, i being the index of the constituent formula which is being solved at the time of evaluation.
- The reevaluation criteria described above are by-passed after a second error test failure during one block and until the time when the value

of the independent variable at the beginning of a new cycle is greater than its value at the time the second error test failure took place.
Also: This by pass is removed following a convergence failure.

7. TESTING

During the development stage of the program STINT a set of problems was created and the program was "tuned" so, as to achieve a high degree of accuracy and efficiency for that particular set of problems. The testing with this (somewhat limited) set of problems verified at least one objective of the program design, that of relatively high efficiency for problems with eigenvalues near the imaginary axis. However, such a problem characteristic as the position of the eigenvalues of the Jacobian matrix is often not known or cannot be estimated a priori. Therefore, for a program which is to have wide utilization, one would require that it perform well on a wider variety of stiff problems.

In order to obtain a more thorough assessment of the qualities of STINT, it was submitted to the testing facility at the University of Toronto. The procedure for testing was the same as that described in the paper by W.H. Enright, T.E. Hull, and B. Lindberg, "Comparing Numerical Methods for Stiff Systems of ODE's", which appeared in BIT, vol. 15 (1975), pp. 10-48. The problem set they use excerises many aspects of a program. There are 25 problems which are divided into various categories according to certain properties they possess, such as linearity, position of eigenvalues, tightness of coupling, etc. The statistics generated from the testing are very useful for the determination of the problem domain over which a program performs well and to indicate where potential improvements could be realized. All problems are solved for three different local accuracies, that of 10^{-2}, 10^{-4}, and 10^{-6}, with absolute error control (i.e., YMAX (I) = 1) in each case. Besides collecting the number of function, Jacobian, and LU decomposition evaluations needed for problem completion, the running time and the reliability of the local and the global error control are also recorded. An analysis of these data for STINT yielded the following conclusions.

For linear problems with real eigenvalues the program STINT was found to be both, accurate and efficient. The problems used varied in size from N = 4 to N = 10 with stiffness ratios from 200 to 10^5. Neither the degree of coupling nor the gap in the distribution of the eigenvalues had any profound effect on the efficiency or accuracy of STINT. The estimates of the local error for problems of this type were shown to be realiable.

The next group of problems are linear and have some complex eigenvalues. As expected, STINT performed efficiently and accurately, particularly on problems with complex eigenvalues near the imaginary axis. However, for problems for which a large number of step-size changes, both increases and decreases, are needed, the number

of LU decompositions, and consequently, the running time were rather large. It was also observed that the realiability of the local error estimate for this group of problems was somewhat diminished.

Problems with non-linear coupling formed another class of test problems. The triangular systems with real eigenvalues had a stiffness ratio of about 100 and no large gap in their eigenvalue distributions. The performance of the program over this class was very good. This is particularly true for problems with coupling from transient components to smooth components. When the direction of the coupling was reversed and made progressively tighter, the realiability of the local error estimate decreased somewhat.

The fourth class consisted of non-linear problems with real eigenvalues. It was observed that a rather large number of LU decompositions was required, particularly when higher accuracies were requested. The realiability of the local error estimate decreased as some of the eigenvalues moved very close to the origin. This then resulted in a larger global error than was expected. On the other hand, for highly non-linear, strongly stable problems the program performed accurately and efficiently.

Five non-linear problems with some non-real eigenvalues formed the last class of test problems. Similar conclusions to the ones made for the fourth group of problems could be drawn also for this class. In particular, it was noticed that the global error for problems whose eigenvalues moved to the right-half plane was about two orders of magnitude bigger that the global error for strongly stable problems.

Summarizing the overall results, STINT's performance appears to justify its inclusion into a program library of any institution, where a wide variety of stiff problems must often be solved. Of secondary importance might be its built-in flexibility for various modifications required to accomodate it to special problem features.

BIBLIOGRAPHY

Enright, W. H, Hull T. E., and Lindberg B., Comparing Numerical Methods for Stiff Systems of ODE's, *BIT 15* (1975), 10-48.

Gear, C. W., *Numerical Initial Value Problems in Ordinary Differential Equations*, Prentice-Hall, Englewood Cliffs, NJ, 1971.

Hindmarsh, A. C., *GEAR: Ordinary Differential Equation System Solver*, Report UCID-30001 Rev. 3, Lawrence Livermore Laboratory, Livermore, CA, 1974.

Sloate, H. M. and Bickart, T. A., A-Stable Composite Multistep Methods, *J. ACM 20* (1973), 7-26.

Shampine, L. F. and Gordon, M. K., "Local error and variable order Adams codes," *Appl. Math. and Comp 1* (1975), 47-66.

Tendler, J. M., *A Stiffly Stable Integration Process Using Cyclic Composite Methods*, Phd Dissertation, Syracuse University, Syracuse, NY, 1973.

APPENDIX

LISTING OF THE PROGRAM

S T I N T

```
      SUBROUTINE STINT (N, T, Y, DY, SAVE, H, HNEXT, HMIN, HMAX, EPS,    STNT0010
     +                  YMAX, KFLAG, KNEXT, JSTART, MAXORD, RW, RJ, MF, STNT0020
     +                  IPIV)                                           STNT0030
                                                                         STNT0040
C*********************************************************************STNT0050
C                                                                       *STNT0060
C     THIS PROGRAM INTEGRATES A SET OF N FIRST ORDER ORDINARY           *STNT0070
C   DIFFERENTIAL EQUATIONS.  A BLOCK OF THREE OR FOUR SOLUTION POINTS,  *STNT0080
C   EACH SEPARATED BY A STEP-SIZE H, IS COMPUTED AT EACH CALL. THE      *STNT0090
C   STEP-SIZE MAGNITUDE MAY BE SPECIFIED BY THE USER AT EACH CALL;      *STNT0100
C   ALTERNATIVELY, IT MAY BE INCREASED OR DECREASED BY STINT WITHIN     *STNT0110
C   THE RANGE |HMIN| TO |HMAX|,                                         *STNT0120
C   IN ORDER TO ACHIEVE AS LARGE A STEP AS POSSIBLE, WHILE NOT          *STNT0130
C   COMMITING A SINGLE STEP ERROR WHICH IS LARGER THAN EPS IN THE       *STNT0140
C   RMS NORM, WHERE EACH COMPONENT OF THE ERROR VECTOR IS DIVIDED BY    *STNT0150
C   THE CORRESPONDING COMPONENT OF YMAX.                                *STNT0160
C                                                                       *STNT0170
C     THE PROGRAM REQUIRES 4 SUBROUTINES NAMED:                         *STNT0180
C                                                                       *STNT0190
C          DIFFUN(N, T, Y, DY)                                          *STNT0200
C          PEDERV(N, T, Y, RJ, NO)                                      *STNT0210
C          DEC(N, NO, A, IPIV, IER)                                     *STNT0220
C          SOL(N, NO, A, B, IPIV)                                       *STNT0230
C                                                                       *STNT0240
C     DIFFUN EVALUATES THE DERIVATIVES OF THE DEPENDENT VARIABLE Y(I)   *STNT0250
C   FOR I=1,...,N AND T AND STORES THE RESULT IN THE ARRAY DY.          *STNT0260
C                                                                       *STNT0270
C     PEDERV COMPUTES THE PARTIAL DERIVATIVES OF THE DIFFERENTIAL       *STNT0280
C   EQUATIONS AT THE VALUES Y(I) FOR I=1,...,N AND T AND STORES THE     *STNT0290
C   RESULT IN ARRAY RJ.  THUS, RJ(J+(K-1)*NO) IS THE PARTIAL OF DY(J)   *STNT0300
C   WITH RESPECT TO Y(K) FOR J,K=1,...,N EVALUATED AT Y(I) FOR I=1,..   *STNT0310
C   ..,N AND T.  NOTE: NO IS THE VALUE OF N ON THE FIRST CALL.  IF THE  *STNT0320
C   ANALYTIC EXPRESSIONS FOR THE PARTIAL DERIVATIVES ARE NOT            *STNT0330
C   AVAILABLE, THEIR APPROXIMATE VALUES CAN BE OBTAINED BY NUMERICAL    *STNT0340
C   DIFFERENCING.  (SEE PARAMETER MF.)  IN THIS CASE SUBROUTINE PEDERV  *STNT0350
C   MAY SIMPLY BE:                                                      *STNT0360
C                                                                       *STNT0370
C          SUBROUTINE PEDERV(N, T, Y, RJ, NO)                           *STNT0380
C          RETURN                                                       *STNT0390
C          END                                                          *STNT0400
C                                                                       *STNT0410
C     DEC PERFORMS AN LU DECOMPOSITION OF A MATRIX A. IF THE            *STNT0420
C   DECOMPOSITION IS SUCCESSFUL IER SHOULD BE SET TO 0, OTHERWISE IT    *STNT0430
C   SHOULD BE SET TO +1.                                                *STNT0440
C                                                                       *STNT0450
C     SOL SOLVES THE LINEAR ALGEBRAIC SYSTEM A*X=B, FOR WHICH THE       *STNT0460
C   MATRIX A WAS PROCESSED BY DEC.                                      *STNT0470
C                                                                       *STNT0480
C     THIS PROGRAM USES DOUBLE PRECISION FOR ALL FLOATING POINT         *STNT0490
C   VARIABLES, EXCEPT FOR THOSE STARTING WITH P, WHICH ARE IN SINGLE    *STNT0500
C   PRECISION.                                                          *STNT0510
C                                                                       *STNT0520
C     TEMPORARY STORAGE SPACE IS PROVIDED (BY THE USER) IN THE          *STNT0530
C   ARRAYS IPIV, RJ, RW, AND SAVE. THE ARRAY IPIV HOLDS A VECTOR OF     *STNT0540
C   THE SAME NAME. THE ARRAYS RJ AND RW ARE USED TO HOLD MATRICES OF    *STNT0550
C   THE SAME NAMES.  THE ARRAY SAVE IS PARTITIONED AS FOLLOWS:          *STNT0560
C                                                                       *STNT0570
C     SAVE(J,I)      1.LE.J.LE.8 AND 1.LE.I.LE.N IS USED TO SAVE        *STNT0580
C                    Y(I,J) IN CASE A STEP (AND HENCE THE WHOLE CYCLE)  *STNT0590
C                    HAS TO BE REPEATED.                                *STNT0600
```

```
*     SAVE(9,I)       1.LE.I.LE.N IS USED TO STORE THE DERIVATIVE OF THE *STNT0610
*                     I-TH DEPENDENT VARIABLE SCALED BY H.                *STNT0620
*     SAVE(N9+I,1)    IS USED TO STORE THE DERIVATIVES AS THEY ARE        *STNT0630
*                     COMPUTED BY DIFFUN FOR THE CORRECTOR. IT IS ALSO    *STNT0640
*                     ACCESSED AS A COMPLETE ARRAY SAVE(N9P1,1).          *STNT0650
*                     IN ADDITION IT IS USED IN THE ERROR CONTROL TEST.   *STNT0660
*                     (N9 = NO * 9  AND  N9P1 = N9 + 1.)                  *STNT0670
*     SAVE(N10+I,1)   IS USED TO HOLD THE CORRECTION TERMS FOR THE ENTIRE *STNT0680
*                     CORRECTOR ITERATION IN THE CASE, THE CORRECTOR HAS  *STNT0690
*                     TO BE REPEATED. (N10 = NO * 10.)                    *STNT0700
*     SAVE(N11+I,1)   IS USED TO HOLD THE DERIVATIVES EVALUATED AT        *STNT0710
*                     Y(I)+∂ AND T, WHERE ∂ IS THE INCREMENT USED IN THE  *STNT0720
*                     NUMERICAL DIFFERENCING SCHEME INVOKED, IN ORDER TO  *STNT0730
*                     OBTAIN APPROXIMATE VALUES OF THE PARTIAL            *STNT0740
*                     DERIVATIVES. (N11 = NO * 11.)                       *STNT0750
*     SAVE(N12+I,1)   IS USED TO HOLD THE DERIVATIVES EVALUATED AT Y(I)   *STNT0760
*                     AND T IN ORDER TO OBTAIN APPROXIMATE VALUES OF THE  *STNT0770
*                     PARTIAL DERIVATIVES. (N12 = NO * 12.)               *STNT0780
*                                                                         *STNT0790
*   THE PARAMETERS IN THE CALLING SEQUENCE HAVE THE FOLLOWING MEANING:    *STNT0800
*                                                                         *STNT0810
*     N         THE NUMBER OF FIRST ORDER DIFFERENTIAL EQUATIONS TO BE    *STNT0820
*               INTEGRATED.  N MAY BE DECREASED ON SUBSEQUENT CALLS IF    *STNT0830
*               THE NUMBER OF ACTIVE EQUATIONS DECREASES, WITH THE        *STNT0840
*               FIRST ONES BEING THOSE RETAINED.  BUT, IT MUST            *STNT0850
*               NOT BE INCREASED WITHOUT CALLING WITH JSTART = 0.         *STNT0860
*     T         THE INDEPENDENT VARIABLE.  ON ENTRY T IS THE CURRENT      *STNT0870
*               SETTING OF THE INDEPENDENT VARIABLE.  ON RETURN TO THE    *STNT0880
*               CALLING PROGRAM, T CORRESPONDS TO THE SETTING OF THE      *STNT0890
*               INDEPENDENT VARIABLE FOR THE MOST FORWARD POINT OBTAINED  *STNT0900
*               THUS FAR.                                                 *STNT0910
*     Y         AN N BY 8 BY 4 ARRAY CONTAINING THE DEPENDENT VARIABLES   *STNT0920
*               AND THEIR BACKWARD DIFFERENCES.  ON EACH CALL UP TO FOUR  *STNT0930
*               SOLUTION POINTS ARE OBTAINED. THE MOST FORWARD POINT IS   *STNT0940
*               ALWAYS AT Y(I,1,1). THE POINT NEXT TO THE MOST FORWARD    *STNT0950
*               POINT IS RETURNED IN Y(I,1,KFLAG). THE MOST BACKWARD      *STNT0960
*               POINT IN THE NEW BLOCK IS RETURNED IN Y(I,1,2). ONLY THE  *STNT0970
*               INITIAL SOLUTION VALUES, ENTERED IN Y(I,1,1) FOR          *STNT0980
*               I=1,...,N, NEED TO BE SUPPLIED ON THE FIRST               *STNT0990
*               CALL (JSTART = 0).                                        *STNT1000
*               Y(I,J+1,K) CONTAINS THE J-TH BACKWARD DIFFERENCE OF THE   *STNT1010
*               I-TH DEPENDENT VARIABLE (FOR K=1,...,KFLAG).              *STNT1020
*               IF IT IS DESIRED TO INTERPOLATE TO NON-MESH POINTS        *STNT1030
*               THESE VALUES CAN BE USED : IF THE CURRENT STEP-SIZE IS    *STNT1040
*               H AND THE VALUE AT T-E (0<|E|<|H|) IS NEEDED, FORM S=E/H  *STNT1050
*               AND THEN COMPUTE                                          *STNT1060
*                                    NQ                                   *STNT1070
*                     Y(I)(T-E) = SUM Y(I,J+1,K)*B(J)                     *STNT1080
*                                   J=0                                   *STNT1090
*               WHERE K, WHICH CORRESPONDS TO T, IS THE FIRST MESH POINT  *STNT1100
*               BEYOND THE POINT T-E, AND B(J+1) = B(J)*(J-S)/(J+1)       *STNT1110
*               WITH B(0) = 1.                                            *STNT1120
*     DY        AN N BY 4 ARRAY.  DY(I,K), 1.LE.I.LE.N, 1.LE.K.LE.KFLAG,  *STNT1130
*               CONTAINS THE DERIVATIVE OF THE I-TH DEPENDENT VARIABLE    *STNT1140
*               SCALED BY H.  THE DY(I,1) ARRAY NEED NOT BE SUPPLIED AT   *STNT1150
*               THE FIRST CALL (JSTART = 0).                              *STNT1160
*     SAVE      A BLOCK OF AT LEAST 13*NO DOUBLE PRECISION FLOATING POINT *STNT1170
*               LOCATIONS.                                                *STNT1180
*     H         THE STEP-SIZE USED FOR THE JUST COMPLETED BLOCK.          *STNT1190
*     HNEXT     THE STEP-SIZE FOR THE NEXT BLOCK.  ON THE                 *STNT1200
```

```
*            FIRST CALL (JSTART=0) THE USER MUST SPECIFY AN INITIAL   *STNT1210
*            STEP-SIZE.  A GOOD ESTIMATE OF ITS MAGNITUDE IS GIVEN    *STNT1220
*            BY    H=.2*(EPS/ABS(&))**.5  , WHERE & IS THE LARGEST    *STNT1230
*            EIGENVALUE OF THE JACOBIAN EVALUATED AT THE INITIAL      *STNT1240
*            VALUES OF T AND Y.  THE SIGN OF THE INITIAL HNEXT        *STNT1250
*            SHOULD BE POSITIVE (NEGATIVE) IF THE FINAL TIME IS       *STNT1260
*            GREATER (LESS) THAN THE INITIAL TIME.  IF THE INITIAL    *STNT1270
*            STEP-SIZE CHOICE DOES NOT CAUSE AN ERROR GREATER THAN    *STNT1280
*            EPS, IN THE RMS NORM, IT WILL BE ACCEPTED; OTHERWISE,    *STNT1290
*            ITS MAGNITUDE WILL BE DECREASED UNTIL AN ERROR LESS      *STNT1300
*            THAN EPS IS ACHIEVED.  THE SUBROUTINE AUTOMATICALLY      *STNT1310
*            ADJUSTS THE STEP-SIZE AFTER THE INITIAL AND SUBSEQUENT   *STNT1320
*            CALLS FOR THE STEP-SIZE OF LARGEST POSSIBLE MAGNITUDE.   *STNT1330
*            THE MAGNITUDE MAY BE ADJUSTED DOWN ON ANY SUBSEQUENT     *STNT1340
*            CALL.  NOTE: A MAGNITUDE ADJUSTMENT UP OR A SIGN CHANGE  *STNT1350
*            WILL BE IGNORED.                                         *STNT1360
*    HMIN    |HMIN| IS THE MINIMUM STEP-SIZE MAGNITUDE TO BE ALLOWED  *STNT1370
*            FOR THE NEXT INTEGRATION CYCLE.  (ON THE FIRST CALL      *STNT1380
*            (JSTART=0), |HMIN| SHOULD BE CHOSEN SIGNIFICANTLY       *STNT1390
*            SMALLER THAN |HNEXT| SO AS TO AVOID START-UP PROBLEMS IF*STNT1400
*            THE ERROR CRITERION IS NOT MET WITH THE USER SPECIFIED   *STNT1410
*            HNEXT.)  NOTE: THE SIGN OF HMIN IS IGNORED.  HMIN MAY BE*STNT1420
*            CHANGED ON SUBSEQUENT CALLS.                             *STNT1430
*    HMAX    |HMAX| IS THE MAXIMUM STEP-SIZE MAGNITUDE TO BE          *STNT1440
*            ALLOWED FOR THE NEXT INTEGRATION CYCLE.  NOTE: IF |HMAX| *STNT1450
*            IS LESS THAN |HMIN|, THEN THE SUBROUTINE FUNCTIONS AS    *STNT1460
*            THOUGH |HMAX| EQUALS |HMIN|.  HMAX MAY BE CHANGED ON     *STNT1470
*            SUBSEQUENT CALLS.                                        *STNT1480
*    EPS     THE ERROR TEST CONSTANT.  THE SINGLE STEP ERROR ESTIMATE *STNT1490
*            FOR Y, COMPUTED AS A WEIGHTED RMS NORM, MUST NOT EXCEED  *STNT1500
*            EPS.  THE WEIGHT FOR THE I-TH ELEMENT IN THE ERROR       *STNT1510
*            ESTIMATE IS 1/YMAX(I).  (SEE PARAMETER YMAX.)  THE       *STNT1520
*            STEP-SIZE AND/OR ORDER ARE ADJUSTED TO ACHIEVE THIS.     *STNT1530
*    YMAX    AN ARRAY OF NQ LOCATIONS, WITH--FOR I=1,...,N--YMAX(I)   *STNT1540
*            BEING THE MAXIMUM OF UNITY AND THE MAXIMUM VALUE OF      *STNT1550
*            |Y(I)| SEEN THUS FAR.  ON THE FIRST CALL IT SHOULD BE    *STNT1560
*            SET TO THE MAXIMUM OF UNITY AND THE INITIAL VALUE OF     *STNT1570
*            |Y(I)|.                                                  *STNT1580
*    KFLAG   A COMPLETION CODE. IF KFLAG IS GREATER THAN 0,THEN       *STNT1590
*            KFLAG POINTS HAVE BEEN COMPUTED. IF KFLAG IS             *STNT1600
*            -2, -3, OR -4 THEN 2, 3, OR 4 MESH POINTS, RESPECTIVELY,*STNT1610
*            HAVE BEEN COMPUTED WITH |H|=|HMIN|, BUT THE REQUESTED    *STNT1620
*            ERROR WAS  NOT ACHIEVED.  OTHER VALUES KFLAG CAN ASSUME  *STNT1630
*            ARE AS FOLLOWS:                                          *STNT1640
*              -5  THE REQUESTED ERROR WAS SMALLER THAN CAN BE HANDLED*STNT1650
*                  FOR THIS PROBLEM.                                  *STNT1660
*              -6  CORRECTOR CONVERGENCE COULD NOT BE ACHIEVED FOR    *STNT1670
*                  DABS(H).GT.DABS(HMIN).                             *STNT1680
*              -7  THE MAXIMUM ORDER SPECIFIED WAS TOO LARGE.         *STNT1690
*    KNEXT   AFTER THE INITIAL CALL (JSTART=0), THE VALUE OF KNEXT IS *STNT1700
*            THE NUMBER OF POINTS TO BE COMPUTED DURING THE NEXT      *STNT1710
*            CYCLE.  THE VALUE IS SUPPLIED FOR INFORMATION ONLY.  THE*STNT1720
*            USER CANNOT CONTROL THE NUMBER OF POINTS BY SETTING      *STNT1730
*            KNEXT.                                                   *STNT1740
*    JSTART  AN INPUT INDICATOR WITH THE FOLLOWING MEANINGS:          *STNT1750
*              0  INIALIZATION CALL.  (JSTART MUST BE SET TO 0 ON THE*STNT1760
*                 FIRST CALL.)                                       *STNT1770
*             >0  CONTINUE FROM THE LAST STEP.                       *STNT1780
*            ON RETURN JSTART IS SET TO NQ, THE MAXIMUM BACKWARD      *STNT1790
*            DIFFERENCE AVAILABLE IN THE Y ARRAY.  (THIS ALSO CORRES-*STNT1800
```

```
C*              PONDS TO THE ORDER OF THE METHOD USED TO COMPUTE THE     *STNT181
C*              JUST COMPLETED BLOCK OF POINTS.)                         *STNT182
C*     MAXORD   THE MAXIMUM ORDER (1.LE.MAXORD.LE.7) THAT MAY BE USED.   *STNT183
C*              NOTE: IF MAXORD IS RESET BETWEEN CYCLES TO A VALUE LESS  *STNT184
C*              THAN THE ORDER DETERMINED FOR THE NEXT CYCLE, THEN THE   *STNT185
C*              ORDER MAY FOR SEVERAL CYCLES EXCEED MAXORD.  HOWEVER,    *STNT186
C*              IT CANNOT EXCEED THE ABOVE DETERMINED VALUE AND, ONCE    *STNT187
C*              THE ORDER IS LESS THAN OR EQUAL TO MAXORD, IT CANNOT     *STNT188
C*              THEN EXCEED MAXORD.                                      *STNT189
C*     RJ       A BLOCK OF AT LEAST NQ**2 DOUBLE PRECISION FLOATING POINT*STNT190
C*              LOCATIONS, WHICH CONTAINS AN ESTIMATE OF THE JACOBIAN    *STNT191
C*              OF THE DIFFERENTIAL EQUATION.                            *STNT192
C*     RW       A BLOCK OF AT LEAST NQ**2 DOUBLE PRECISION FLOATING POINT*STNT193
C*              LOCATIONS.                                               *STNT194
C*     IPIV     A BLOCK OF AT LEAST NQ INTEGERS USED TO HOLD PIVOT DATA  *STNT195
C*              GENERATED DURING AN LU DECOMPOSITION.                    *STNT196
C*     MF       METHOD FLAG . IT DETERMINES THE MODE BY WHICH THE PARTIAL*STNT197
C*              DERIVATIVES ARE OBTAINED WITH THE FOLLOWING MEANING:     *STNT198
C*              1  ANALYTIC EXPRESSIONS FOR THE PARTIAL DERIVATIVES ARE  *STNT199
C*                 SUPPLIED BY THE USER IN THE SUBROUTINE PEDERV;        *STNT200
C*              2  THE ANALYTIC EXPRESSIONS ARE NOT AVAILABLE.           *STNT201
C*                 APPROXIMATE VALUES OF THE PARTIAL DERIVATIVES ARE     *STNT202
C*                 OBTAINED BY NUMERICAL DIFFERENCING.                   *STNT203
C*                                                                       *STNT204
C*************************************************************************STNT205
C                                                                         STNT206
      IMPLICIT REAL*8 (A-H,Q-Z)                                           STNT207
      DIMENSION Y(N,8,4), DY(N,4), SAVE(9,1), YMAX(1), RW(1), IPIV(1),    STNT208
     +          INDEX(7,2), B(82), C(16), PERROR(9), PC(16), PD(7),       STNT209
     +          B1(48), B2(34), RJ(1)                                     STNT210
      EQUIVALENCE(B(1),B1(1)),(B(49),B2(1))                               STNT211
      COMMON /STAT/ NSTEP,NFE,NJE,NINVS                                   STNT212
C                                                                         STNT213
C*************************************************************************STNT214
C*   THE CONSTANT UROUND SHOULD BE SET EQUAL TO THE UNIT ROUND-OFF       *STNT215
C*   FOR THE MACHINE.  THE CONSTANT SQRTUR SHOULD BE SET EQUAL TO THE    *STNT216
C*   SQUARE ROOT OF UROUND.                                              *STNT217
C*************************************************************************STNT218
C                                                                         STNT219
      DATA UROUND/4.D-16/, SQRTUR/2.D-8/                                  STNT220
C                                                                         STNT221
C*************************************************************************STNT222
C*   THE ARRAY INDEX HOLDS POINTERS AND CONSTANTS FOR THE VARIOUS        *STNT223
C*   ORDER METHODS.  FOR NQ=1,...,7 THE ENTRIES ARE AS FOLLOWS...        *STNT224
C*    INDEX(NQ,1)   BASE INDEX FOR B ARRAY (H*DY PREDICTOR).             *STNT225
C*    INDEX(NQ,2)   BASE INDEX FOR C ARRAY (CORRECTOR).                  *STNT226
C*************************************************************************STNT227
C                                                                         STNT228
      DATA INDEX/ 1, 2, 4, 11, 20, 38, 59,                                STNT229
     +            1, 2, 3,  5,  7, 10, 14/                              STNT230
C                                                                         STNT231
C*************************************************************************STNT232
C*   THE COEFFICIENTS APPEARING IN THE NEXT DATA STATEMENTS FOR THE      *STNT233
C*   B ARRAY SHOULD BE DEFINED TO THE MAXIMUM ACCURACY PERMITTED BY      *STNT234
C*   THE MACHINE. THEY ARE, IN THE ORDER SPECIFIED,...                   *STNT235
C*    1                                                                  *STNT236
C*    -2, 3                                                              *STNT237
C*    -9/2, -5/4, 11/2                                                   *STNT238
C*    -15/2, -3/4, 13/2, 2                                               *STNT239
C*    -22/3, -8/3, -17/18, 25/3                                          *STNT240
```

```
C*     -9, -9/4, -7/8, 35/4, 5/4                                          *STNT241
C*     -125/12, -101/24, -71/36, -37/48, 137/12                           *STNT242
C*     -253/24, -1001/240, -707/360, -123/160, 1373/120, 1/10             *STNT243
C*     -57/4, -25/8, -17/10, -169/240, 61/5, -1/20, 31/10                 *STNT244
C*     -137/10, -117/20, -46/15, -191/120, -197/300, 147/10               *STNT245
C*     -353/25, -571/100, -1819/600, -79/50, -3919/6000, 1477/100, 7/20   *STNT246
C*     -3529/200, -1889/400, -1609/600, -3527/2400, -931/1500, 3079/200,  *STNT247
C*         -3/20, 17/5                                                    *STNT248
C*     -343/20, -303/40, -253/60, -589/240, -101/75, -23/40, 363/20       *STNT249
C*     -266/15, -221/30, -749/180, -73/30, -803/600, -103/180, 547/30,    *STNT250
C*         1/2                                                            *STNT251
C*     -1316/75, -1151/150, -3689/900, -121/50, -4003/3000, -257/450,     *STNT252
C*        2737/150, -1/5, 1/2                                             *STNT253
C*********************************************************************** STNT254
C                                                                         STNT255
      DATA B1/                                                            STNT256
     +         1.0D0, -2.0D0, 3.0D0, -4.5D0, -1.25D0, 5.5D0, -7.5D0,      STNT257
     +         -.75D0, 6.5D0, 2.0D0, -7.3333333333333333D0,               STNT258
     +         -2.6666666666666667D0, -.9444444444444444D0,               STNT259
     +         8.3333333333333333D0, -9.0D0, -2.25D0, -.875D0, 8.75D0,    STNT260
     +         1.25D0, -10.416666666666667D0, -4.208333333333333D0,       STNT261
     +         -1.972222222222222D0, -.770833333333333D0,                 STNT262
     +         11.416666666666667D0, -10.541666666666667D0,               STNT263
     +         -4.170833333333333D0, -1.963888888888889D0, -.76875D0,     STNT264
     +         11.441666666666667D0, 0.1D0, -14.25D0, -3.125D0, -1.7D0,   STNT265
     +         -.7041666666666667D0, 12.2D0, -.05D0, 3.1D0, -13.7D0,      STNT266
     +         -5.85D0, -3.066666666666667D0, -1.591666666666667D0,       STNT267
     +         -.6566666666666667D0, 14.7D0, -14.12D0, -5.71D0,           STNT268
     +         -3.031666666666667D0, -1.58D0, -.6531666666666667D0/       STNT269
      DATA B2/                                                            STNT270
     +         14.77D0, 0.35D0, -17.645D0, -4.7225D0,                     STNT271
     +         -2.681666666666667D0, -1.469583333333333D0,                STNT272
     +         -.6206666666666667D0, 15.395D0, -.15D0, 3.4D0, -17.15D0,   STNT273
     +         -7.575D0, -4.216666666666667D0, -2.454166666666667D0,      STNT274
     +         -1.346666666666667D0, -.575D0, 18.15D0,                    STNT275
     +         -17.733333333333333D0, -7.366666666666667D0,               STNT276
     +         -4.161111111111111D0, -2.433333333333333D0,                STNT277
     +         -1.338333333333333D0, -.5722222222222222D0,                STNT278
     +         18.233333333333333D0, 0.5D0, -17.546666666666667D0,        STNT279
     +         -7.673333333333333D0, -4.098888888888889D0, -2.42D0,       STNT280
     +         -1.334333333333333D0, -.5711111111111111D0,                STNT281
     +         18.246666666666667D0, -0.2D0, 0.5D0/                       STNT282
C                                                                         STNT283
C*********************************************************************** STNT284
C*    THE COEFFICIENTS APPEARING IN THE NEXT DATA STATEMENTS FOR THE     *STNT285
C*    C ARRAY SHOULD BE DEFINED TO THE MAXIMUM ACCURACY PERMITTED BY     *STNT286
C*    THE MACHINE.  THEY ARE, IN THE ORDER SPECIFIED,...                 *STNT287
C*     -1                                                                *STNT288
C*     -2/3                                                              *STNT289
C*     -6/11, -2/3                                                       *STNT290
C*     -12/25, -16/31                                                    *STNT291
C*     -60/137, -600/1373, -100/193                                      *STNT292
C*     -20/49, -120/293, -75/184, -1200/2299                             *STNT293
C*     -140/363, -60/143, -1050/2437                                     *STNT294
C*********************************************************************** STNT295
C                                                                         STNT296
      DATA C/ -1.0D0, -.6666666666666667D0, -.5454545454545455D0,         STNT297
     +         -.6666666666666667D0, -.48D0, -.5161290322580645D0,        STNT298
     +         -.4379562043795620D0, -.4369992716678805D0,                STNT299
     +         -.5181347150259067D0, -.4081632653061224D0,                STNT300
```

```
     +          -.40955631399317406D0, -.40760869565217391D0,                STNT30
     +          -.52196607220530666D0, -.38567493112947659D0,                STNT30
     +          -.41958041958041958D0, -.43085761181780879D0/                STNT30
C                                                                             STNT30
C*************************************************************************STNT30
C*   THE COEFFICIENTS IN THE PERROR ARRAY ARE USED IN THE ERROR TEST,   *STNT30
C*   THE FIRST TIME IT IS PERFORMED, AS WELL AS IN THE STEP-SIZE/ORDER  *STNT30
C*   SELECTION SEGMENT. PERROR(I+1) = 1/D(I+1) , I=1,...,7, WHERE       *STNT30
C*   D(I+1) IS THE DISCRETIZATION ERROR CONSTANT CORRESPONDING TO THE   *STNT30
C*   SECOND PASS OF THE INTEGRATION CYCLE OF ORDER I. PERROR(1) AND     *STNT31
C*   PERROR(9) ARE DEFINED SOLELY FOR PROGRAMMING EASE; THEY ARE NOT    *STNT31
C*   USED.                                                              *STNT31
C*************************************************************************STNT31
C                                                                             STNT31
      DATA PERROR/  1.0, 1.0, 1.92857, 2.78161, 3.56735,                     STNT31
     +              4.29497, 4.9065, 5.6066, 1.0/                            STNT31
C                                                                             STNT31
C*************************************************************************STNT31
C*   THE COEFFICIENTS IN THE ARRAY PC ARE USED BOTH IN THE CONVERGENCE  *STNT31
C*   AND ERROR TESTS. THEY ARE THE RECIPROCAL VALUES OF THE             *STNT32
C*   DISCRETIZATION ERROR CONSTANTS FOR EQUATIONS CONSTITUTING THE      *STNT32
C*   METHODS OF ORDER 1 THRU 7.                                         *STNT32
C*************************************************************************STNT32
C                                                                             STNT32
      DATA PC/  2.0, 4.5, 7.3333, 6.0, 10.4167, 9.3, 13.7, 13.8687,          STNT32
     +          9.6904, 17.15, 16.9504, 17.4349, 9.472, 20.7429, 15.921,STNT32
     +          14.7809/                                                     STNT32
C                                                                             STNT32
C*************************************************************************STNT32
C*   THE COEFFICIENTS APPEARING IN THE ARRAY PD ARE USED IN THE TESTING*STNT33
C*   OF THE 'OUTDATENESS' OF THE ARRAY RW. THE PD(I) ELEMENT CONTAINS   *STNT33
C*   THE AVERAGE VALUE OF THE COEFFICIENTS IN ARRAY C CORRESPONDING TO  *STNT33
C*   ORDER I.                                                           *STNT33
C*************************************************************************STNT33
C                                                                             STNT33
      DATA PD/  1.0, 0.6667, 0.6061, 0.4981, 0.4644, 0.4368, 0.412/          STNT33
C                                                                             STNT33
C*************************************************************************STNT33
C*************************************************************************STNT33
C                                                                             STNT34
      KFLAG = 0                                                               STNT34
      IFAIL = 0                                                               STNT34
      IF(JSTART.NE.0) GO TO 80                                                STNT34
C*                                                                         *STNT34
C*       INITIALIZATION --- FIRST CALL                                     *STNT34
C*                                                                         *STNT34
   20 NO = N                                                                  STNT34
      FN=DFLOAT(N)                                                            STNT34
      N9 = NO * 9                                                             STNT34
      N9P1 = N9 + 1                                                           STNT35
      N10 = N9 + NO                                                           STNT35
      N11 = N10 + NO                                                          STNT35
      N11P1 = N11 + 1                                                         STNT35
      N12 = N11 + NO                                                          STNT35
      N12P1 = N12 + 1                                                         STNT35
      NSQ = NO*NO                                                             STNT35
      IWEVAL=+1                                                               STNT35
      TDL=T                                                                   STNT35
      H=DMAX1(DABS(HMIN),DMIN1(DABS(HNEXT),DABS(HMAX)))                       STNT35
      IF(HNEXT.LT.0.D0) H=-H                                                  STNT36
```

```
 30 NQOLD= 0                                                              STNT3610
    ISW1=0                                                                STNT3620
    ISW2=0                                                                STNT3630
    CRATE=1.D0                                                            STNT3640
    CALL DIFFUN (N, T, Y, DY)                                             STNT3650
    NFE=NFE+1                                                             STNT3660
    DO 40 I=1,N                                                           STNT3670
 40   DY(I,1) = H*DY(I,1)                                                 STNT3680
    NQ = 1                                                                STNT3690
    IST = 1                                                               STNT3700
    IDEL = 0                                                              STNT3710
    GO TO 180                                                             STNT3720
C                                                                        *STNT3730
C         CONTINUE WITH THE STEP-SIZE H                                  *STNT3740
C                                                                        *STNT3750
 80 HNEW=DMAX1(DABS(HMIN),DMIN1(DABS(HNEW),DABS(HNEXT),DABS(HMAX)))       STNT3760
    IF(H.LT.0.D0) HNEW=-HNEW                                              STNT3770
    IF(H.NE.HNEW) GO TO 100                                               STNT3780
    IST = NQP1                                                            STNT3790
    IDEL = NQP1                                                           STNT3800
    ISW2=0                                                                STNT3810
    GO TO 180                                                             STNT3820
*                                                                        *STNT3830
*         NEW STEP-SIZE --- INTERPOLATE FOR NEW POINTS                   *STNT3840
*                                                                        *STNT3850
100 RATIO=HNEW/H                                                          STNT3860
    H = RATIO*H                                                           STNT3870
    RC=RC*RATIO*DBLE(PD(NQ)/PDOLD)                                        STNT3880
    PDOLD = PD(NQ)                                                        STNT3890
    IF (NQ.EQ.1) GO TO 150                                                STNT3900
    IEQ = NEQ                                                             STNT3910
    D = 0.0D0                                                             STNT3920
    DO 140 J=2,NQ                                                         STNT3930
       D = D + RATIO                                                      STNT3940
       IF (D.GT.DFLOAT(NEQ+NQP1-IEQ)) IEQ=2                               STNT3950
       D1 = DFLOAT(NEQ-IEQ-1) - D                                         STNT3960
      DO 140 I=1,N                                                        STNT3970
        D2 = 1.0D0                                                        STNT3980
        D3 = 0.0D0                                                        STNT3990
        DO 120 J1=2,NQP1                                                  STNT4000
          D2 = D2*(DFLOAT(J1) + D1)/DFLOAT(J1-1)                          STNT4010
120       D3 = D3 + D2*Y(I,J1,IEQ)                                        STNT4020
140     Y(I,J,1) = D3 + Y(I,1,IEQ)                                        STNT4030
150 IST = NQ                                                              STNT4040
    IDEL = 0                                                              STNT4050
    DO 160 I=1,N                                                          STNT4060
160   DY(I,1) = DY(I,1)*RATIO                                             STNT4070
    IRET = 3                                                              STNT4080
    GO TO 4000                                                            STNT4090
C*                                                                       *STNT4100
C*         INITIALIZE SAVE ARRAY                                         *STNT4110
C*                                                                       *STNT4120
180 DO 200 I=1,N                                                          STNT4130
      SAVE(9,I) = DY(I,1)                                                 STNT4140
      DO 200 J=1,IST                                                      STNT4150
200     SAVE(J,I) = Y(I,J,1)                                              STNT4160
    NQST = NQ                                                             STNT4170
    RATIO = 1.0D0                                                         STNT4180
    TOLD = T                                                              STNT4190
    HOLD = H                                                              STNT4200
```

```
  220 IF ((NQ.EQ.NQOLD).AND.(FN.EQ.DFLOAT(N))) GO TO 300
      IF ((NQ.EQ.NQOLD).AND.(FN.NE.DFLOAT(N))) GO TO 280
      IF (MAXORD.LT.8) GO TO 260
      KFLAG = -7
      GO TO 4185
C*
C*          SET APPROPRIATE PARAMETERS AND CONSTANTS
C*          FOR NEW CYCLE OF ORDER NQ
C*
  260 INDB = INDEX(NQ,1)
      INDC = INDEX(NQ,2)
      NEQ = 3+NQ/5
      JSTART = NQ
      NQOLD = NQ
      NQM1=NQ-1
      NQP1 = NQ+1
      Q = DFLOAT (NQ)
      ENQDWN = 0.5D0/Q
      ENQSAM = 0.5D0/(Q+1.0D0)
      ENQUP = 0.5D0/(Q+2.0D0)
  280 FN = DFLOAT(N)
      EDOWN = FN*(DBLE(PERROR(NQ))*EPS)**2
      EUP = FN*(DBLE(PERROR(NQ+2))*EPS)**2
      ES = FN*(DBLE(PERROR(NQP1))*EPS)**2
      IF(EDOWN.GT.0.D0) GO TO 300
      KFLAG = -5
      GO TO 4185
C*
C*          CHECK FOR REEVALUATION OF JACOBIAN
C*
  300 IF(IWEVAL.GT.0) GO TO 320
      IF((TDL-TOLD)*H.GT.0.D0) GO TO 320
      IF(DABS(RC-1.D0).GE.4.D-1) IWEVAL = +2
      IF(DABS(RC-1.D0).GE.8.D-1) IWEVAL = +1
  320 INDBB = INDB
      INDCC = INDC
      DO 1000 IEQ=1,NEQ
        IST = MOD (IEQ, NEQ) + 1
        T = T + H
        BND=FN*(DBLE(PC(INDCC))*ENQUP*EPS)**2
        E=ES
        IF(IEQ.GT.2) E=FN*(DBLE(PC(INDCC))*EPS)**2
C*
C*          PREDICT Y AND DY FOR THE NEXT MESH POINT
C*
        DO 460 I=1,N
          D = DY(I,IEQ)
          D1 = Y(I,1,IEQ) + Q*D
          D2 = B(INDBB+NQM1)*D
          IF (NQ-2) 440, 400, 340
  340     IF (IEQ-3) 400, 380, 360
  360     D2 = D2 + B(INDBB+NQP1)*DY(I,3)
  380     D2 = D2 + B(INDBB+NQ)*DY(I,2)
  400     DO 420 J=2,NQ
            D = Y(I,J,IEQ)
            D3 = DFLOAT (J-1)
            D1 = D1 + D*(D3-Q)/D3
  420       D2 = D2 + D*B(INDBB+J-2)
  440     Y(I,1,IST) = D1
  460     DY(I,IST) = D2
```

```
      IF(NQ.LE.2) GO TO 470                                              STNT4810
      IF(IEQ.GT.1) INDBB = INDBB + NQ + IEQ - 2                          STNT4820
*                                                                        *STNT4830
*       ITERATE THE CORRECTOR UP TO THREE TIMES. ACCUMULATE THE          *STNT4840
*       CORRECTION TERMS IN SAVE(N10+I,1) FOR REDOING THE ENTIRE         *STNT4850
*       CORRECTOR IF CONVERGENCE IS NOT ACHIEVED                         *STNT4860
*                                                                        *STNT4870
  470 D1 = C(INDCC)                                                      STNT4880
  500 DO 510 I=1,N                                                       STNT4890
  510   SAVE(N10+I,1) = 0.D0                                             STNT4900
      DO 780 J=1,3                                                       STNT4910
        CALL DIFFUN (N, T, Y(1,1,IST), SAVE(N9P1,1))                     STNT4920
        NFE=NFE+1                                                        STNT4930
        IF(IWEVAL-1) 680, 52), 620                                       STNT4940
  520   IND = IEQ                                                        STNT4950
        IF(IEQ.EQ.2) IND = 1                                             STNT4960
        GO TO (610, 530), MF                                             STNT4970
  530   CALL DIFFUN (N, T-H*DFLOAT(1+IEQ-IND), Y(1,1,IND),               STNT4980
     +    SAVE(N12P1,1))                                                 STNT4990
        NFE=NFE+1                                                        STNT5000
        D = 0.D0                                                         STNT5010
        DO 540 I=1,N                                                     STNT5020
  540     D = D + SAVE(N12+I,1)**2                                       STNT5030
        D = DABS(H)*1.D3*UROUND*DSQRT(D)                                 STNT5040
        NJE = NJE + 1                                                    STNT5050
        J0 = 0                                                           STNT5060
        DO 560 J1=1,N                                                    STNT5070
          DI = SQRTUR*YMAX(J1)                                           STNT5080
          DI = DMAX1(DI,D)                                               STNT5090
          YJ1 = Y(J1,1,IND)                                              STNT5100
          Y(J1,1,IND) = Y(J1,1,IND) + DI                                 STNT5110
          CALL DIFFUN (N, T-H*DFLOAT(1+IEQ-IND), Y(1,1,IND),             STNT5120
     +      SAVE(N11P1,1))                                               STNT5130
          NFE = NFE + 1                                                  STNT5140
          DO 550 I=1,N                                                   STNT5150
  550       RJ(I+J0) = (SAVE(N11+I,1) - SAVE(N12+I,1))/DI                STNT5160
          J0 = J0 + NQ                                                   STNT5170
          Y(J1,1,IND) = YJ1                                              STNT5180
  560     CONTINUE                                                       STNT5190
        GO TO 620                                                        STNT5200
  610   CALL PEDERV (N, T-H*DFLOAT(1+IEQ-IND), Y(1,1,IND), RJ, NQ)       STNT5210
        NJE=NJE+1                                                        STNT5220
  620   D = D1*H                                                         STNT5230
        DO 640 I=1,NSQ                                                   STNT5240
  640     RW(I) = RJ(I)*D                                                STNT5250
        J1 = 1                                                           STNT5260
        DO 660 I=1,N                                                     STNT5270
          RW(J1) = 1.D0 + RW(J1)                                         STNT5280
  660     J1 = J1 + NQ + 1                                               STNT5290
        CALL DEC (N, NQ, RW, IPIV, IER)                                  STNT5300
        NINVS=NINVS+1                                                    STNT5310
        IWEVAL = -IEQ                                                    STNT5320
        RC=1.D0                                                          STNT5330
        PDOLD=PD(NQ)                                                     STNT5340
        IF(IER.NE.0) GO TO 800                                           STNT5350
  680   DO 700 I=1,N                                                     STNT5360
  700     SAVE(N9+I,1) = DY(I,IST) - H*SAVE(N9+I,1)                      STNT5370
        CALL SOL (N, NQ, RW, SAVE(N9P1,1), IPIV)                         STNT5380
        D2 = 0.0D0                                                       STNT5390
        J3 = N9                                                          STNT5400
```

```
            DO 760 I=1,N                                                        STNT541
              J3 = J3 + 1                                                       STNT542
              J2 = J3 + NO                                                      STNT543
              SAVE(J2,1) = SAVE(J2,1) + SAVE(J3,1)                              STNT544
              Y(I,1,IST) = Y(I,1,IST) + D1*SAVE(J3,1)                           STNT545
              DY(I,IST) = DY(I,IST) - SAVE(J3,1)                                STNT546
  760         D2 = D2 + (SAVE(J3,1)/YMAX(I))**2                                 STNT547
            IF(J.NE.1) CRATE = DMAX1(CRATE*9.D-1,D2/D3)                         STNT548
            IF (D2*DMIN1(1.D0,2.D0*CRATE).GT.BND) GO TO 770                     STNT549
            GO TO 940                                                           STNT550
  770       D3 = D2                                                           STNT551
  780       CONTINUE                                                          STNT552
C*                                                                            *STNT553
C*          CORRECTOR FAILED TO CONVERGE IN THREE ITERATIONS.                 *STNT554
C*          IF JACOBIAN WAS REEVALUATED DURING THIS CYCLE, STEP-SIZE IS       *STNT555
C*          REDUCED TO 3/10 OF H. OTHERWISE, JACOBIAN IS REEVALUATED          *STNT556
C*                                                                            *STNT557
          TOL = TOLD                                                          STNT558
          IF(IWEVAL.EQ.0) GO TO 900                                           STNT559
  800     IF(DABS(H).LE.(1.00D100*DABS(HMIN))) IF(NQ-2) 4180, 4120, 4140      STNT560
          RATIO = RATIO*0.300                                                 STNT561
          IRET = 1                                                            STNT562
          ISW1=0                                                              STNT563
          ISW2=1                                                              STNT564
          IWEVAL = +2                                                         STNT565
          GO TO 3000                                                          STNT566
  900     DO 920 I=1,N                                                        STNT567
            D = SAVE(N10+I,1)                                                 STNT568
            Y(I,1,IST) = Y(I,1,IST) - D1*D                                    STNT569
  920       DY(I,IST) = DY(I,IST) + D                                         STNT570
          IWEVAL = +1                                                         STNT571
          GO TO 500                                                           STNT572
C*                                                                            *STNT573
C*          CORRECTOR CONVERGED. THE BACKWARD DIFFERENCES OF ORDER            *STNT574
C*          ONE THROUGH NQ AT THE NEW MESH POINT ARE COMPUTED                 *STNT575
C*                                                                            *STNT576
  940     DO 980 I=1,N                                                        STNT577
            DO 960 J=1,NQ                                                     STNT578
  960         Y(I,J+1,IST) = Y(I,J,IST) - Y(I,J,IEQ)                          STNT579
          IF(IEQ.EQ.1) GO TO 980                                              STNT580
C*                                                                            *STNT581
C*          THE (NQ+1)-ST BACKWARD DIFFERENCE FOR ALL BUT THE FIRST           *STNT582
C*          MESH POINT IS ESTABLISHED                                         *STNT583
C*                                                                            *STNT584
          SAVE(N9+I,1) = Y(I,NQP1,IST) - Y(I,NQP1,IEQ)                        STNT585
  980     CONTINUE                                                            STNT586
          IF (IEQ.EQ.NEQ) GO TO 1015                                          STNT587
          IF (NQ.LE.2) GO TO 1010                                             STNT588
          IF ((IEQ.GT.1). OR .(NQ.EQ.6)) INDCC = INDCC + 1                    STNT589
 1010     IF(IEQ.EQ.1) GO TO 1000                                             STNT590
C*                                                                            *STNT591
C*          ERROR TEST FOR ALL BUT THE FIRST MESH POINT IS PERFORMED          *STNT592
C*                                                                            *STNT593
 1015     D = 0.D0                                                            STNT594
          DO 1020 I=1,N                                                       STNT595
 1020       D = D + (SAVE(N9+I,1)/YMAX(I))**2                                 STNT596
          IF(D.GT.E) GO TO 1030                                               STNT597
 1000     CONTINUE                                                            STNT598
        GO TO 1160                                                            STNT599
C*                                                                            *STNT600
```

```
C*         THE ERROR CRITERION WAS NOT MET                                    *STNT6010
C*                                                                            *STNT6020
 1030 IFAIL=IFAIL+1                                                            STNT6030
      IF(IFAIL.GT.2) GO TO 1080                                                STNT6040
      IF(DABS(H).LE.(DABS(HMIN)*1.00001D0)) IF(NQ-2) 1140, 4120, 4140          STNT6050
      IWEVAL = +2                                                              STNT6060
      IF(IFAIL.EQ.1) GO TO 1200                                                STNT6070
      TDL=T                                                                    STNT6080
      RATIO = RATIO*5.D-1                                                      STNT6090
      IRET = 1                                                                 STNT6100
      ISW1=0                                                                   STNT6110
      ISW2=1                                                                   STNT6120
      GO TO 3000                                                               STNT6130
C*                                                                             *STNT6140
C*         START OVER WITH ORDER 1 METHOD                                      *STNT6150
C*                                                                             *STNT6160
 1080 IFAIL = 0                                                                STNT6170
      DO 1090 I=1,N                                                            STNT6180
 1090   Y(I,1,1) = SAVE(1,I)                                                   STNT6190
      T = TOLD                                                                 STNT6200
      D=DMAX1(1.D-1,DABS(HMIN/H))                                              STNT6210
      H = H*D                                                                  STNT6220
      GO TO 30                                                                 STNT6230
C*                                                                             *STNT6240
C*         A NEW BLOCK OF SOLUTION POINTS HAS BEEN ACCEPTED                    *STNT6250
C*                                                                             *STNT6260
 1140 IWEVAL = 0                                                               STNT6270
      NSTEP = NSTEP + IEQ                                                      STNT6280
      KFLAG = -IEQ                                                             STNT6290
      RETURN                                                                   STNT6300
 1160 IWEVAL = 0                                                               STNT6310
      E = ES                                                                   STNT6320
      KFLAG = NEQ                                                              STNT6330
      NSTEP = NSTEP + KFLAG                                                    STNT6340
      HNEW=H                                                                   STNT6350
C*                                                                             *STNT6360
C*         CHECK FOR CONTINUATION WITH THE SAME H AND NQ                       *STNT6370
C*                                                                             *STNT6380
      IF(ISW2.EQ.1) GO TO 1420                                                 STNT6390
      IF(NQ.GT.3) ISW1=1-ISW1                                                  STNT6400
      IF(ISW1.EQ.1) GO TO 1420                                                 STNT6410
C*                                                                             *STNT6420
C*         NEW STEP-SIZE AND/OR ORDER SELECTION                                *STNT6430
C*                                                                             *STNT6440
 1200 RRSAME=1.2D0*(D/E)**ENQSAM                                               STNT6450
      IF(IFAIL.NE.0) GO TO 1380                                                STNT6460
      RMAX = 1.D-4                                                             STNT6470
      DF=DFLOAT(NEQ+NQM1)                                                      STNT6480
      IF(NQ.NE.1) RMAX=(Q-1.D0)/DF                                             STNT6490
      RRSAME=DMAX1(RRSAME,RMAX)                                                STNT6500
      RRUP = 1.D20                                                             STNT6510
      RRDOWN = 1.D20                                                           STNT6520
      IF(NQ.GE.MAXORD) GO TO 1240                                              STNT6530
      D = 0.0D0                                                                STNT6540
      DO 1220 I=1,N                                                            STNT6550
        D1 = Y(I,NQP1,NEQ) - Y(I,NQP1,NEQ-1)                                   STNT6560
 1220   D = D + ((SAVE(N9+I,1) - D1)/YMAX(I))**2                               STNT6570
      RRUP = 1.2D0*(D/EUP)**ENQUP                                              STNT6580
      RMAX=Q/DF                                                                STNT6590
      RRUP=DMAX1(RRUP,RMAX)                                                    STNT6600
```

```
 1240 IF(NQ.EQ.1) GO TO 128)                                              STNT6
      D = 0.0D0                                                           STNT6
      DO 1260 I=1,N                                                       STNT6
 1260   D = D + (Y(I,NQP1,1)/YMAX(I))**2                                  STNT6
      RRDOWN = 1.2D0*(D/EDOWN)**ENQDWN                                    STNT6
      RMAX = 1.D-4                                                        STNT6
      IF(NQ.NE.2) RMAX=(Q-2.D0)/DF                                        STNT6
      RRDOWN=DMAX1(RRDOWN,RMAX)                                           STNT6
 1280 IF (RRSAME.GT.RRUP) IF (RRUP-RRDOWN) 1340, 1300, 1300               STNT6
      IF (RRSAME.LE.RRDOWN) GO TO 1320                                    STNT6
 1300 NEWQ = NQM1                                                         STNT6
      D=1.D0/RRDOWN                                                       STNT6
      GO TO 1360                                                          STNT6
 1320 NEWQ = NQ                                                           STNT6
      D=1.D0/RRSAME                                                       STNT6
      GO TO 1360                                                          STNT6
 1340 NEWQ = NQP1                                                         STNT6
      D=1.D0/RRUP                                                         STNT6
 1360 IF (D-1.1D0) 1420, 1400, 1400                                       STNT6
 1380 RATIO=RATIO/RRSAME                                                  STNT6
      IRET = 1                                                            STNT6
      ISW1=0                                                              STNT6
      ISW2=1                                                              STNT6
      GO TO 3000                                                          STNT6
 1400 HNEW = H*D                                                          STNT6
      NQ = NEWQ                                                           STNT6
 1420 DO 1460 I=1,N                                                       STNT6
        D=YMAX(I)                                                         STNT6
        DO 1440 J=1,NEQ                                                   STNT6
 1440     D=DMAX1(D,DABS(Y(I,1,J)))                                       STNT6
 1460   YMAX(I)=D                                                        STNT6
      HNEXT = HNEW                                                        STNT6
      KNEXT = 3+NQ/5                                                      STNT6
      RETURN                                                              STNT6
C*                                                                       *STNT6
C*        THIS SECTION IS USED WHEN STEP-SIZE OR ORDER IS CHANGED        *STNT6
C*        DURING THE CYCLE.  STARTING VALUES ARE RETRIEVED FROM THE      *STNT6
C*        SAVE ARRAY.                                                    *STNT6
C*                                                                       *STNT6
 3000 RATIO = DMAX1 (DABS(HMIN/HOLD), DMIN1(RATIO, 1.0D0))                STNT7
      T = TOLD                                                            STNT7
      IF (RATIO.LT.1.0D0) IF (IDEL) 3030, 3030, 3080                      STNT7
      DO 3020 I=1,N                                                       STNT7
        DY(I,1) = SAVE(9,I)                                               STNT7
        DO 3020 J=1,NQ                                                    STNT7
 3020     Y(I,J,1) = SAVE(J,I)                                            STNT7
      GO TO (320,260), IRET                                               STNT7
C*                                                                       *STNT7
C*        THE (NQST+1)-ST ORDER BACKWARD DIFFERENCE IS ESTABLISHED       *STNT7
C*                                                                       *STNT7
 3030 IDEL = NQST+1                                                       STNT7
      IF ((NQST.LT.2) .OR. (NQ.LT.2)) GO TO 3080                          STNT7
      C = DFLOAT (NQST)                                                   STNT7
      DO 3060 I=1,N                                                       STNT7
        D1 = SAVE(9,I)                                                    STNT7
        DO 3040 J=2,NQST                                                  STNT7
 3040     D1 = D1 - SAVE(J,I)/DFLOAT(J-1)                                 STNT7
 3060   SAVE(IDEL,I) = D*D1                                              STNT7
C*                                                                       *STNT7
C*        INTERPOLATE FOR NEW POINTS                                     *STNT7
```

```
C*                                                                    *STNT7210
 3080 DO 3140 I=1,N                                                    STNT7220
        DY(I,1) = RATIO*SAVE(9,I)                                      STNT7230
        IF (NQ.LT.2) GO TO 3140                                        STNT7240
        D1 = 0.0D0                                                     STNT7250
        DO 3120 J=2,NQ                                                 STNT7260
          D1 = D1 + RATIO                                              STNT7270
          D2 = 1.0D0                                                   STNT7280
          D = 0.0D0                                                    STNT7290
          DO 3100 J1=2,IDEL                                            STNT7300
            D2 = D2*(DFLOAT(J1-2) - D1)/DFLOAT(J1-1)                   STNT7310
 3100       D = D + D2*SAVE(J1,I)                                      STNT7320
 3120     Y(I,J,1) = D + SAVE(1,I)                                     STNT7330
 3140   Y(I,1,1) = SAVE(1,I)                                           STNT7340
      H = HOLD*RATIO                                                   STNT7350
C*                                                                    *STNT7360
C*          FORM THE BACKWARD DIFFERENCES                             *STNT7370
C*                                                                    *STNT7380
 4000 IF (NQ.LT.2) GO TO 4040                                          STNT7390
      NQM1 = NQ-1                                                      STNT7400
      DO 4020 I=1,N                                                    STNT7410
        DO 4020 J=1,NQM1                                               STNT7420
          J0=J + 1                                                     STNT7430
          DO 4020 J1=J0,NQ                                             STNT7440
            J2 = NQ-J1+J+1                                             STNT7450
 4020       Y(I,J2,1) = Y(I,J2-1,1) - Y(I,J2,1)                        STNT7460
 4040 GO TO (320,260,180), IRET                                        STNT7470
C*                                                                    *STNT7480
C*          CONVERGENCE OR ERROR PROBLEMS                             *STNT7490
C*                                                                    *STNT7500
 4120 NQ = 1                                                           STNT7510
      GO TO 4160                                                       STNT7520
 4140 NQ = 2                                                           STNT7530
 4160 IFAIL = 0                                                        STNT7540
      IRET = 2                                                         STNT7550
      IWEVAL = +2                                                      STNT7560
      GO TO 3000                                                       STNT7570
 4180 KFLAG = -6                                                       STNT7580
 4185 J1 = NQST + 1                                                    STNT7590
      DO 4190 I=1,N                                                    STNT7600
        DY(I,1) = SAVE(9,I)                                            STNT7610
        DO 4190 J=1,J1                                                 STNT7620
 4190     Y(I,J,1) = SAVE(J,I)                                         STNT7630
      H = HOLD                                                         STNT7640
      T = TOLD                                                         STNT7650
      JSTART = NQST                                                    STNT7660
      RETURN                                                           STNT7670
C                                                                      STNT7680
C*********************************************************************STNT7690
C*                                                                    *STNT7700
C*                  --- END OF SUBROUTINE STINT ---                   *STNT7710
C*                                                                    *STNT7720
C*********************************************************************STNT7730
C                                                                      STNT7740
      END                                                              STNT7750
```