Automated Heuristic Stability Analysis for Nonlinear Equations

L. D. Cloutman
L. W. Fullerton
AUTOMATED HEURISTIC STABILITY ANALYSIS
FOR NONLINEAR EQUATIONS

by
L. D. Cloutman and L. W. Fullerton

ABSTRACT
The modified equation method of heuristic stability analysis has proved to be a useful tool for the prediction of instabilities of nonlinear finite difference equations that are used in numerical fluid dynamics. The need to calculate and manipulate multi-dimensional Taylor series expansions is a serious disadvantage of this technique, and for many problems of interest, it is difficult to obtain a reliable result by hand. We have, therefore, written general purpose programs to do the algebra by computer, for both the series expansions and elimination of time derivatives from the truncation error terms of the modified equation. We discuss some important features of the procedure and present examples of how the results may be used to design and improve difference methods.

I. INTRODUCTION

Heuristic stability analysis (e.g., Hirt) consists of examining the lowest order truncation errors of a finite difference equation (FDE). These errors are obtained from Taylor series expansions, sometimes multi-dimensional, of the solution of the FDE about a suitably chosen point. Often simple examination of the expansion can reveal undesirable properties of the FDE, such as zeroth or negative order errors and diffusional instabilities. In principle, these expansions can also be used to help design difference methods by eliminating inaccurate or unstable forms before performing a series of numerical tests. Heuristic analysis also has been useful in predicting some of the stability requirements of nonlinear finite difference methods used for numerical fluid dynamics calculations. In particular, Rivard et al. have recently used such truncation error expansions (TEE's) as the basis of a technique to stabilize and improve the accuracy of the ICE algorithm originally described by Harlow and Amsden. Warning and Hyett discuss a procedure for analyzing linear problems using a program written in FORMAC, but they did not treat nonlinear equations.

The massive amount of algebra involved in carrying out the expansions and time derivatives eliminations for many problems of interest is a hindrance to applying the heuristic technique. Indeed, even relatively simple FDE's may be impractical to analyze by hand, because one cannot be sure there are no blunders in the derived result. We have, therefore, implemented the heuristic technique in an algebraic computer language, and this implementation is discussed in the next section. In Sec. III, we give several examples which illustrate how the results of our program may be used.

II. METHODOLOGY

In order to illustrate the heuristic technique, we first carry out an analysis of a typical FDE from the field of numerical fluid dynamics. The one-dimensional continuity equation in Cartesian coordinates is

\[ \frac{\partial p}{\partial t} + \frac{\partial pu}{\partial x} = \frac{3}{\partial x} \left( \frac{\partial p}{\partial x} \right), \]  

(1)
where \( p \) is the fluid density, \( u \) is the velocity, and \( \xi \) is an artificial mass diffusion coefficient that may be needed for stability. For the ICE method, we approximate Eq. (1) by

\[
\frac{\rho_{i}^{n+1} - \rho_{i}^{n}}{\Delta t} + \frac{\partial}{\partial x} \left[ \left( \rho_{i+1}^{n+1} + \rho_{i}^{n+1} \right) u_{i+\frac{1}{2}}^{n+1} - \left( \rho_{i-1}^{n+1} + \rho_{i}^{n+1} \right) u_{i-\frac{1}{2}}^{n+1} \right] + \rho_{i+1}^{n} u_{i+\frac{1}{2}}^{n} - \left( \rho_{i+1}^{n} + \rho_{i-1}^{n+1} \right) u_{i-\frac{1}{2}}^{n} = \frac{1}{2} \left[ \xi_{i+\frac{1}{2}}^{n} \left( \rho_{i+1}^{n} - \rho_{i}^{n} \right) - \xi_{i-\frac{1}{2}}^{n} \left( \rho_{i}^{n} - \rho_{i-1}^{n-1} \right) \right],
\] (2)

where a superscript denotes the time level and a subscript denotes the mesh cell number. Figure 1 shows the kind of staggered grid used by ICE. The time centering parameter \( \theta \) assumes values between zero and unity. We now choose a point, say time level \( n \) and cell center \( i \), about which to expand the dependent variables. Next we calculate the truncated Taylor series expansion

\[
y_{i+k}^{n+1} = \sum_{m=0}^{N} \frac{1}{m!} \left( h \frac{\partial}{\partial t} + k \frac{\partial}{\partial x} \right)^{m} y,
\] (3)

where \( y \) is either \( \rho \) or \( u \) in our example. Because we want truncation errors through \( O(\Delta t) \) and \( O(\Delta x^2) \) in the final result, we must, in this case, keep terms in Eq. (3) through \( O(\Delta t^2) \) and \( O(\Delta x^4) \). When we substitute Eq. (3) for each of the variables in Eq. (2) and drop high-order terms, we obtain the original differential equation plus extra terms that we call truncation errors:

\[
\begin{align*}
\frac{3 \rho_{i}^{n}}{\Delta t} + \frac{3 \rho_{i+1}^{n}}{\Delta x} = & \frac{3}{2} \left( \xi_{i+\frac{1}{2}}^{n} \frac{\partial \rho_{i}^{n}}{\partial x} \right) - \frac{\Delta t}{2} \left[ \frac{3 \rho_{i}^{n}}{\Delta t^2} + 20 \left( \frac{\partial u_{i}^{n}}{\partial t} \frac{\partial}{\partial x} \right) \right] \right)
\end{align*}
\]

(4)

This result is called the modified equation. This expansion procedure, we see, is simple, well defined and very tedious. It is, therefore, ideally suited for implementation in an algebraic language. We chose to code the heuristic algorithm in ALTRAN, \( 5,6 \) because ALTRAN is designed for massive algebraic operations on rational polynomial expressions. Moreover, it contains a number of routines which manipulate truncated power series efficiently. The list of the expansion code is given in Appendix A. The algorithm could be implemented in a number of other algebraic languages including MACSYMA, REDUCE, and FORMAC, provided they are available on a sufficiently large computer.

The most important consideration in designing this code was to minimize the work space (i.e., core) needed. Even though we use the LCM version of ALTRAN, which has 131 000 words of workspace, the explosive growth of intermediate terms can cause memory overflow even for fairly simple difference equations unless care is taken to make the most efficient use of the memory. Running time is usually no problem on the CDC 7600 although the efficient use of memory also tends to reduce run times.

The program uses indeterminant arrays to represent dependent variables and their partial deriv-
atives. For example, \( u_{I,J} \) \( \frac{\partial}{\partial x} \frac{\partial}{\partial t} \) is represented by the array element \( U(I,J) \). The code is set up to handle four such variables: \( P, T, \rho, \) and \( U \). More variables can be added to the layout if needed, although they would increase memory requirements. The maximum order of the expansions is set by the integer variable \( ORD \), currently set to a value of six. The maximum value of \( I \) or \( J \) is set by the integer variable \( N \), also currently set equal to six. If higher order derivatives or expansions are needed at any point in the calculation, \( N \) and/or \( ORD \) must be increased, with a corresponding increase in memory requirements and running time. In practice, however, even large, high-order problems are practical on the LASL 7600's.

The Taylor series expansions are done by the LONG ALGEBRAIC ALTRAN PROCEDURE \( TE \), which is invoked as a function. Suppose we choose \( (i, \delta r, n, \delta t) \) as the point about which we want to perform the expansions. A single call to \( TE \) can expand a product of up to four variables. The calling sequence

\[
\text{TE}(f_1, a_1, b_1, f_2, a_2, b_2, f_3, a_3, b_3, f_4, a_4, b_4)
\]

expands

\[
(f_1)^{a_1+b_1} (f_2)^{a_2+b_2} (f_3)^{a_3+b_3} (f_4)^{a_4+b_4}
\]

to order \( ORD \) in \( \delta r, \delta t, \delta r^2, \delta t^2 \). For example, \( u_{i+1,j} \) is represented by \( TE(U, 1/2, 1) \). It is more efficient to compute products with a single call than to make separate calls and multiply the results. That is, use

\[
\text{TE}(RHO(0,1) = \frac{\partial\rho}{\partial t} = -\frac{3u}{\delta x} + \frac{3u}{\delta x} + \frac{\partial^2\rho}{\delta x^2}
\]

both \( \delta r \) and \( \delta t \). For example, \( u_{i+1,j} \) is represented by \( TE(U, -1/2, 1) \). It is more efficient to compute products with a single call than to make separate calls and multiply the results. That is, use

\[
\text{TE}(RHO(0, 1, U, 1/2, 0) = n+1 \text{ if } a_4 = 0, \text{ not } \text{TE}(RHO, 0, 1) \times \text{TE}(U, 1/2, 0).
\]

The first method computes only terms of order \( ORD \). The latter method expands each variable to order \( ORD \), and the multiplication generates many terms through order \( 2 \times ORD \) that are eventually discarded.

Since there is no simple way to specify the difference equation on data cards, all input data is specified in executable ALTRAN statements in a special section of the program. \( RORD \) and \( TORD \) are the maximum orders of \( \delta r \) and \( \delta t \), respectively, to be retained in the final result. \( DERMOD \) is the left-hand side of the modified equation, and it will be explained in more detail in the example. \( DE \) is the differential equation, and \( FDE \) is the finite difference equation expressed in terms of \( TE \). The listing of the code in Appendix A contains Eq. (2) as an example. Note that \( DE \) and \( FDE \) are always written in the form such that they are equal to zero.
expanded about time \( n \) and space point \( i \) is

\[
\frac{\partial T}{\partial t} = K \left[ \frac{\partial^2 T}{\partial x^2} - \frac{\delta t}{2} \frac{\partial^2 T}{\partial t^2} + \frac{\delta x^2 K}{6} \frac{\partial^4 T}{\partial x^4} + O(\delta t^2, \delta x^4) \right].
\]  

(8)

We will keep error terms of order \( \delta t \) and \( \delta x^2 \). Begin the elimination of \( \partial^2 T / \partial t^2 \) by differentiating Eq. (8) with respect to \( t \),

\[
\frac{\partial^2 T}{\partial t^2} = K \left[ \frac{\partial^3 T}{\partial x^2 \partial t} - \frac{\delta t K}{2} \frac{\partial^3 T}{\partial t^3} + \frac{\delta x^2 K}{6} \frac{\partial^5 T}{\partial x^4 \partial t} \right].
\]

(9)

Substitute Eq. (9) into Eq. (8) and discard high-order terms:

\[
\frac{\partial T}{\partial t} = K \left[ \frac{\partial^2 T}{\partial x^2} - \frac{\delta t K}{2} \frac{\partial^2 T}{\partial x^2 \partial t^2} + \frac{\delta x^2 K}{6} \frac{\partial^4 T}{\partial x^4} \right].
\]

(10)

Note that we have lowered the order of time derivative in the error terms by one. Now we can differentiate Eq. (8) with respect to \( x \) to obtain

\[
\frac{\partial^3 T}{\partial x^2 \partial t} = K \left[ \frac{\partial^4 T}{\partial x^4} - \frac{\delta t}{2} \frac{\partial^3 T}{\partial x^4 \partial t^2} + \frac{\delta x^2 K}{6} \frac{\partial^5 T}{\partial x^4 \partial t} \right],
\]

(11)

which we substitute into Eq. (10):

\[
\frac{\partial T}{\partial t} = K \left[ \frac{\partial^2 T}{\partial x^2} + K \delta x^2 \right] \left[ \frac{1}{3} - \frac{K \delta t}{6 \delta x^2} \right] \frac{\partial^4 T}{\partial x^4}.
\]

(12)

It is obvious from this trivial example that the elimination of time derivatives from the truncation error terms of the modified equation is, in general, a very messy algebraic problem for the general case of coupled nonlinear partial differential equations.

The code and flow charts listed in Appendixes C and D describe a first attempt to solve this problem. Although this program is capable of handling very large problems in a reasonable amount of central processor time, a clever programmer should be able to improve its efficiency. For this and other reasons to be discussed later, this code should be considered a usable but unpolished tool.

The elimination code reads its input from cards punched either by itself or the expansion code. The elimination code only makes a single pass at eliminating the time derivatives, lowering the order of the time derivatives by at most one per run. Thus, our simple example would require two runs. The first run would read cards punched by the expansion code, and the next run (and all subsequent runs if necessary) would read the cards punched by the expansion code on the previous run. This multiple run procedure is inefficient in terms of the human intervention and turn around time involved, and we intend to eventually combine the expansion and elimination codes into a single completely automated code.

The elimination code can also handle simple systems of equations. It can read a second modified equation and substitute derivatives of the first, or primary, modified equation into the second, or secondary, modified equation. Our limited experience with systems of modified equations suggests that improving the efficiency of workspace utilization should receive high priority in the list of improvements to this code. The memory problem is not serious with the LCM version of ALTRAN available on the CROS operating system, where 131,000 decimal words of workspace are available, but it is likely to be quite limiting at installations with smaller workspaces. Some steps for reducing memory requirements and the number of runs are described in Appendix C.

III. APPLICATIONS

Truncation error expansions may be employed in three ways. First, they indicate the order and accuracy of FDE's, and so they may be used to help choose the best form for a particular problem. Second, they may be used to find stability conditions for some problems. And finally, they may be employed as the basis of a new method for stabilizing some finite difference algorithms. In this section we discuss examples of each of these applications. We emphasize that although most of our examples are relatively simple and could be done by hand, the ALTRAN programs are powerful tools that can do and have done expansions much too large and complicated to do reliably by hand in a reasonable amount of time.

a. Comparison of Errors of Difference Equations

The TEE's easily indicate some undesirable properties of FDE's, such as zeroth-or negative-
order errors. Such information is quite useful, for it may rule out use of a particular FDE before it is coded and subjected to numerical tests. But beyond such simple observations, FDE's are not easily compared. The next example illustrates the type of analysis frequently necessary to determine which one of several FDE's is more accurate. Consider the one-dimensional diffusion problem in spherical coordinates

\[
\frac{\partial T}{\partial t} = \phi \frac{1}{r^2} \frac{\partial}{\partial r} \left( r^2 \frac{\partial T}{\partial r} \right) \quad \text{for} \quad 0 \leq t \leq \infty, \quad 0 \leq r \leq \pi, \quad (13)
\]

\[
T(r,0) = \frac{\sin r}{r},
\]

\[
T(\pi,t) = 0,
\]

and

\[
\frac{\partial T}{\partial t} (0,t) = 0,
\]

where \( \phi \) is a constant. The analytic solution is

\[
T(r,t) = \exp(-\phi t) \frac{\sin(r)}{r}. \quad (14)
\]

Now consider the explicit FDE

\[
\frac{T_i^{n+1} - T_i^n}{\Delta t} = \frac{\phi}{V_i} \left[ \frac{r_i^{2}(T_{i+1}^n - T_{i}^n)}{r_{i+1} - r_i} - \frac{r_i^{2}(T_{i-1}^n - T_{i+1}^n)}{r_{i+1} - r_i} \right]. \quad (15)
\]

The computing mesh is illustrated in Fig. 1. We compare the accuracy of two different definitions of \( V_i \) in Eq. (15):

\[
V_i = (r_{i+\frac{1}{2}}^3 - r_{i-\frac{1}{2}}^3)/3 \quad (16a)
\]

and

\[
V_i = r_i^2 (r_{i+\frac{1}{2}} - r_{i-\frac{1}{2}}). \quad (16b)
\]

Note that the cells are spherical shells, and \( V_i \) is the volume of one steradian of the ith cell.

Heuristically we expect Eq. (16a) to be more accurate than Eq. (16b) near the origin, because the former volume elements exactly fill space. The latter volume elements are all smaller than the former for the same set of mesh points, and the effect is most pronounced at small \( r \). Both volume elements give conservative FDE's, but they conserve different amounts of the conserved quantity. For constant \( T \), volume elements in Eq. (16a) lead to conservation of the correct amount of the conserved quantity

\[
4\pi \int_0^{\pi/2} T r^2 \, dr, \quad \text{but Eq. (16b) conserves the wrong amount.}
\]

We can use the expansions to determine which volume element is more accurate. The TEE's for Eq. (15) with Eqs. (16a) and (16b), respectively, are equivalent to

\[
\frac{\partial T}{\partial t} = \phi \frac{1}{r^2} \frac{\partial}{\partial r} \left( r^2 \frac{\partial T}{\partial r} \right)
\]

\[- \left[ \frac{\phi^2 \delta t}{2} - \frac{\phi \delta r^2}{12} \right] \left[ \frac{3 \delta T}{r^2} + \frac{4 \delta^3 T}{r^4} \right] \quad (17a)
\]

\[- \left[ \frac{\phi^2 \delta t}{2} - \frac{\phi \delta r^2}{12} \right] \left[ \frac{3 \delta T}{r^2} + \frac{4 \delta^3 T}{r^4} \right] + \frac{\delta r^2}{6 r^2} \left[ \frac{3 \delta T}{r^2} - \frac{1}{r} \frac{\partial^2 T}{\partial r^2} \right] + O(\delta^2 \delta r^4)
\]

and

\[
\frac{\partial T}{\partial t} = \phi \frac{1}{r^2} \frac{\partial}{\partial r} \left( r^2 \frac{\partial T}{\partial r} \right) - \left[ \frac{\phi^2 \delta t}{2} - \frac{\phi \delta r^2}{12} \right] \left[ \frac{3 \delta T}{r^2} \right] + \frac{\delta r^2}{6 r^2} \left[ \frac{3 \delta T}{r^2} - \frac{1}{r} \frac{\partial^2 T}{\partial r^2} \right] \quad (17b)
\]

for a uniform mesh.

At first glance, Eq. (17b) appears better than Eq. (17a) because the coefficient of \( \frac{\partial T}{\partial r} \) in Eq. (17a) is proportional to \( 1/r^3 \). Furthermore, unlike Eq. (16a), Eq. (16b) leads to a difference scheme which is exact for a solution \( T \), linear in \( r \). Thus, our earlier arguments about volume elements in Eq. (16a) being better appear to be wrong. However, as we shall show, our superficial examination of Eqs. (17a) and (17b) is at fault.

Currently, there is no general procedure for
choosing the more accurate of several FDE's, based on Taylor series expansions. But we now present a procedure which works many problems, and we hope it will provide a basis for an even more general procedure. The cursory examination above is misleading, because \( \frac{dT}{dr} = 0 \) at the origin and because some error terms partially cancel each other. We expand \( T \) in Taylor series about \( r = 0 \) for some \( \eta, 0 < \eta < r_{5/2} \), and a time \( \tau, t_n < \tau < t_{n+1} \):

\[
T(\eta, \tau) = \sum_{i=0}^{\infty} \frac{\partial^{(i)} T(0, \tau)}{\partial r^{(i)}} \frac{n^i}{i!}.
\]  

(18)

After differentiating Eq. (18) and substituting into the space errors of Eqs. (17a) and (17b), we find

\[
\frac{\phi \delta r^2}{12} \left[ \frac{\partial^4 T}{\partial r^4} + \frac{4}{r} \frac{\partial^3 T}{\partial r^3} + \frac{2}{r^2} \frac{\partial^2 T}{\partial r^2} - \frac{2}{r^3} \frac{\partial T}{\partial r} \right]_{r=\eta, \tau = \tau} = \frac{\phi \delta r^2}{12} \left[ -\frac{2}{\eta^3} \frac{\partial T(O, \tau)}{\partial r} + \frac{5}{\eta} \frac{\partial^3 T(O, \tau)}{\partial r^3} + O(\eta^6) \right]_{r=\eta} \]  

(19a)

and

\[
\frac{\phi \delta r^2}{12} \left[ \frac{\partial^4 T}{\partial r^4} + \frac{4}{r} \frac{\partial^3 T}{\partial r^3} + \frac{3}{r^2} \frac{\partial^2 T}{\partial r^2} \right]_{r=\eta} = \frac{\phi \delta r^2}{12} \left[ -\frac{3}{2\eta^2} \frac{\partial^2 T(O, \tau)}{\partial r^2} + \frac{7}{\eta} \frac{\partial^3 T(O, \tau)}{\partial r^3} \right]_{r=\eta} + O(\eta^6) \]  

(19b)

Because \( \frac{\partial T(O, \tau)}{\partial r} = 0 \) for most physical problems, the \( 1/\eta^2 \) error in Eq. (19b) dominates all others in Eqs. (19), and so Eq. (16a) actually leads to errors smaller than Eq. (16b) near the origin.

The boundary conditions are imposed by

\[
T_{n+1} = T_{n+1}^{T_1} = T_{n+1}^{T_2}
\]  

(20)

and either

\[
T_{n+1} = -T_n + 2T_b
\]  

(21a)

or

\[
T_{n+1} = -2T_n + \frac{1}{3} T_{n-1} + \frac{8}{3} T_b,
\]  

(21b)

where \( T_b = 0 \) is the boundary value. For boundary conditions in Eqs. (21a) and (21b), respectively, the right side of Eq. (15) is equivalent to

\[
\phi \left( \frac{3}{4} \frac{1}{r^2} \frac{\partial T}{\partial r} \left( r^2 \frac{\partial T}{\partial r} - \frac{3}{r^2} \frac{\partial^2 T}{\partial r^2} + \frac{2}{r^3} \frac{\partial^3 T}{\partial r^3} \right) + O(\delta r^2) \right)
\]  

(22a)

and

\[
\phi \left( \frac{1}{2} \frac{\partial T}{\partial r} \left( r^2 \frac{\partial T}{\partial r} - \frac{3}{r^2} \frac{\partial^2 T}{\partial r^2} + \frac{3}{r^3} \frac{\partial^3 T}{\partial r^3} + O(\delta r^2) \right) \right).
\]  

(22b)

Each equation is valid for both volume elements in Eqs. (16a) and (16b). Note that the simpler Eq. (21a) has a large zeroth-order in the diffusion term. Therefore, we expect the first-order boundary conditions in Eq. (21b) to be more accurate in the outer part of the mesh where the boundary treatment dominates the accuracy of the solution.

In order to substantiate our deductions based on TEE's, we numerically solved Eq. (15) using several combinations of Eqs. (16) and (21). Figure 2 shows the relative errors as a function of \( r \) at time \( t = 0.23687 \) for several of these calculations. We see that the best accuracy obtains from volume element in Eq. (16a) and boundary condition in Eq. (21b) as predicted.

b. Truncation Error Cancellation Algorithms

The second application of TEE's is important in the field of numerical fluid dynamics. A number of instabilities that arise in such calculations are due to diffusional truncation errors with negative diffusion coefficients. An obvious application of TEE's is to find stability conditions for numerical algorithms that are subject to diffusion instabilities. On a higher level, these expansions can be used as the basis of new method for stabilizing the FDE's as reported by Rivard et al.² Both of these uses are illustrated with a one-dimensional
version of the ICE method\textsuperscript{3} that requires much less artificial diffusion to obtain stability than many other methods. Again, we emphasize that our simple example is chosen for clarity of presentation, and the programs are useful for much more complicated FDE's.

We describe the truncation error cancellation (TEC) technique in detail only for the continuity equation (1), but the same procedure is applied to the momentum and energy equations, as well. It is possible, however, to improve the algorithm by applying the procedure only to one or two of the equations. We use the FDE given by Eq. (2). The truncation error expansion is given in Eq. (4), but the time derivatives must be converted to space derivatives by using the continuity and momentum modified equations. We obtain for the diffusional errors

$$\zeta \frac{\partial^2 \rho}{\partial x^2} = \left[ (26-1) \frac{\delta t}{2} (u^2 + c^2) - \frac{\delta x^2}{4} \frac{\partial u}{\partial x} \right] \frac{\partial^2 \rho}{\partial x^2}.$$  

(23)

where $\zeta$ is the diffusion coefficient of the truncation errors and $c$ is the local sound speed. We have neglected the $\frac{\partial^2 \zeta}{\partial x^2}$ term in Eq. (4); as we shall see, it is a higher order term in the TEC algorithm. If $\zeta = 0$ in Eq. (1), the FDE is unstable whenever $\zeta < 0$. In the original version of ICE, a constant global artificial mass diffusion coefficient $\zeta > 0$ is used to stabilize the algorithm. It is necessary to choose $\zeta$ large enough that $\zeta + \xi > 0$ for all cells at every time step, and so a large amount of global diffusion is needed to stabilize many problems. Because the diffusion term is explicit, a necessary condition for stability is

$$\zeta \delta t < \frac{1}{2} \delta x^2.$$  

(24)

Artificial viscosity plays a similar role in the momentum equation and imposes a separate requirement analogous to Eq. (24). Although these artificial diffusion parameters stabilize the algorithm, they decrease the accuracy of the solution and introduce time step limits that can be so small as to preclude the solution of some problems.

The basic idea of the TEC algorithm is to replace the artificial diffusion parameter with a variable $\xi(x,t)$ which is chosen so that it locally cancels the destabilizing effects of diffusion truncation errors. Consequently, much less diffusion is needed for stability (often several orders of magnitude less in parts of the mesh), and so accuracy is improved and diffusional time step limits are relaxed.

The first step in deriving a TEC scheme is to evaluate algebraically the diffusion coefficient $\zeta$. Expansion yields a result of the form

$$\frac{\partial \rho}{\partial t} + \frac{\partial (\rho u)}{\partial x} = \frac{\partial}{\partial x} \left( \zeta \frac{\partial \rho}{\partial x} \right) + \zeta \frac{\partial}{\partial x} \left( \frac{\partial u}{\partial x} \right) = \frac{\partial}{\partial x} \left( \zeta \frac{\partial \rho}{\partial x} \right).$$  

(25)

The algorithm for carrying out the expansion gives the nonconservative form $\zeta \frac{\partial^2 \rho}{\partial x^2}$, but we convert it to the conservation form $\frac{\partial^2 \rho}{\partial x^2}$ the right-hand side of Eq. (25). In some cases, usually in the momentum equation, $\frac{\partial \rho}{\partial x}$ will contribute additional diffusional
errors that should be included in TEC as discussed by Rivard et al. In our continuity equation, however, \( \frac{\partial}{\partial x} \) does not produce additional diffusion, errors, and the \( \frac{\partial}{\partial x} \) truncation error is neglected. In order to obtain an improved FDE, Eq. (23) is differenced to yield

\[
\zeta_{1-2} = (2\beta - 1) \frac{\delta t}{2} \left[ (u_{1-2}^n)^2 + \frac{1}{2} (c^2 + c_{1-2}^2) \right] - \frac{\delta x}{8} (u_{1+4}^n - u_{1-3}^n).
\]

Next we choose

\[
\zeta_{1-2}^n = \begin{cases} 
-(1+\beta) \zeta_{1-2} & \text{if } \zeta_{1-2} \leq 0 \\
(1-\beta) \zeta_{1-2} & \text{if } \zeta_{1-2} \geq 0
\end{cases}
\]

which is then incorporated in the finite difference form of Eq. (1). The constant \( \beta \), \( 0 \leq \beta \leq 1 \), is a free parameter that determines the degree to which the diffusional truncation errors are cancelled. If \( \beta \) is too small, the FDE's will have so little diffusion that dispersively generated ripples destroy accuracy. If, on the other hand, \( \beta \) is too large, unnecessary artificial diffusion reduces the accuracy of the solution. The optimum value of \( \beta \) is problem dependent and must be found by trial and error. In practice, \( \beta = 1 \) is frequently an adequate value.

Although the derivation of the diffusion errors for the TEC scheme requires extra work, the modified FDE's yield substantially better solutions. TEC has been installed in several programs, and the scheme works well except in problems with very strong shocks where higher order errors are significant. We now briefly compare several TEC and non-TEC solutions in order to show the advantage that may be expected from using TEC.

Consider Fig. 3, which shows the run of density for three one-dimensional shock tube calculations, as well as the analytic solution. The initial condition is a 5:1 pressure and density jump at cell 90. All solutions coincide at the left and right bound-
the conventional method was unstable with less viscosity. The two velocity solutions are similar, although the TEC solution shows more shear in the vortex, and the weak Helmholtz instability in the upper right quadrant is somewhat stronger, an indication that viscous forces are relatively small in this problem. The isotherms, however, are much different. The TEC solution shows steeper gradients across the vortex, because there is less diffusion and less viscous heating in the energy equation.

Experience indicates that TEC is quite general in its range of applicability and that it provides significant improvement in the accuracy of numerical fluid dynamics calculations. The use of ALTRAN to compute the TEE's is proving to be extremely helpful.

IV. SUMMARY

We have shown that the Taylor series expansions needed for Hirt's heuristic stability analysis can be easily generated by a program written in a computer language such as ALTRAN. The truncation error expansions have proved quite useful in choosing optimum finite difference equations, in deriving some necessary conditions for stability, and assisting in the design of truncation error cancellation algorithms. The ability to derive the truncation errors automatically is essential for all but the simplest difference equations. The extension of these codes to include more dimensions is straightforward, and present computers are adequate to handle many problems of interest. We expect the use of such algebraic computations to increase and become a much more important part of numerical analysis as algebraic systems become more common on large computers and as potential users become familiar with the language and come to appreciate the potential of algebraic systems for accurately and quickly solving massive problems.

REFERENCES


APPENDIX A
THE EXPANSION CODE LISTING

This appendix gives instructions for running the code that computes the Taylor series expansions, a listing of the code, and a sample problem. This particular problem was run on a CDC 7600 under the CROS operating system using the LCM version of ALTRAN.

Lines 19 through 27 provide the input for this run. RORD and TORD are the maximum orders of the expansions in \( \delta r \) (denoted by DR) and \( \delta t \) (denoted by DT), respectively. This run expands the difference equation (2) for the differential equation (1) using the subscript notation for derivatives and the expansion procedure TE described in the text. DERMOD is the time derivative we want to eliminate using the second code, and it is not limited to a first derivative in time. For example, DERMOD = RHO(1,2) would be appropriate for \( \rho_{xtt} = A\rho + B\rho_{x} \). DE is the differential equation, where we have represented \( \xi \) by T in this run. Note that we have shifted the term on the right-hand side of equation (1) over to the left-hand side so DE = 0. This must always be done for both DE and the finite difference equation FDE. In FDE we have represented 0 by GI. Note that we have not followed our own advice in the text concerning the efficient use of TE. This problem is small enough to easily run on the LASL LCM version of ALTRAN, but we would have to be more careful with memory utilization with the SCM version or with larger problems. It may be necessary to break large problems into pieces and run them separately. For example, the diffusion term could be deleted from DE and FDE and then computed by itself on a second run.

Most of the output is intermediate results that are sometimes useful if the run terminates abnormally. The final results are printed after the message "CONSTRUCT THE MODIFIED EQUATIONS." The modified equation is given by DERMOD = NUMER/DENOM, and the output beginning with RORD is punched from logical unit 25 by the computer as input for the time derivative elimination code.

In lines 38 and 39, the code checks for the possible existence of errors of order \( \delta r^{-1} \) and/or \( \delta t^{-1} \) and prints a warning message if appropriate. Some difference equations, such as equations (15) and (16a), will trigger a fictitious warning. However, the truncated power series package cannot handle an error of negative order, and the code will terminate abnormally after the warning message is printed. One example is the Lax method for the diffusion equation:

\[
\begin{align*}
DE &= \frac{\partial T}{\partial t} - D \frac{\partial^2 T}{\partial x^2} \\
&= T(0,1) - DIF*T(2,0) \quad (A1)
\end{align*}
\]

and

\[
\begin{align*}
FDE &= \left[ T_{i+1}^{n+1} - (T_i^n + T_{i-1}^n)/2 \right]/\delta t \\
&\quad - D\left[ T_{i+1}^n - 2T_i^n + T_{i-1}^n \right]/\delta x^2 \quad (A2) \\
&= \frac{(TE(T,i,0) - (TE(T,i,0) + TE(T,-1,0))/2)/DT}
\end{align*}
\]

Lines 33 and 34 contain a possible trap for the unwary user. The use of relations such as \( RP = R + DR/2 \) for equations such as Eq. (16a) can simplify the input phase. The user may find other useful substitutions, and these were left in the code as examples of substitutions that we found useful in our test runs. These statements must be removed or replaced before RM and RP can be used for another purpose. A similar situation exists for line 55, where F1 and F2 are used as ratios of widths of adjacent cells for cases where \( \delta r \) is not constant. That is, FDE may be a (at most) four-point difference scheme over three cells of widths DR, F1*DR, and F2*DR, with the order being chosen by the user.
PROCEDURE MAIN  # TRUNCATION ERRORS OF DIFFERENCE EQUATIONS.
EXTERNAL INTEGER ORD=6
INTEGER M; N=O,M
INTEGER RNORM, TOPD
LONG ALGEBRAIC (M; N, T; N, P; (M, O; N); XP(N), T(P; M, P; N); XP(N),
P; M, THETA; N, RP; M, RM; P; (M, O; N); DIF; M, LAM; M, FI; M, F2; M,
T; M, U; (M, N; R; N); XP(N), RHO; (M, N; R; N); XP(N)) ARRAY DETPS, FDETPS,
TFPL, CONTPS
EXTERNAL ALGEBRAIC DR=DR, DT=DT, LAM2=LAM
LONG ALGEBRAIC FNE, DE, DERMOD, NUMFR, DENOM
LONG ALGEBRAIC ARRAY MODEO
ALTRAN INTEGER TPSORD
ALTRAN SHORT INTEGER ARRAY XP
ALTRAN ALGEBRAIC TE, TPSEL
ALTRAN ALGEBRAIC ARRAY TPS, TPSMUL, TPSRSS, TRRSRS, TETPS, TPSCHO

# -- -- -- -- INSERT INPUT IN THIS INITIALIZATION BLOCK -- -- -- --

ORD = 2  # TRND = 1
DERMOD = RHO(P; 1)
DE = RHO(P; 1) + RHO(1, P) * U(0, P) + RHO(0, P) * U(1, P) - T(0, P) * RHO(2, P) =
T(1, P) * RHO(1, P)
FDE = (TE(RHO, 1, 1) + TE(RHO, 0, 1)) *
TE(U, 1/2, P) = (TE(RHO, 1, 1) + TE(RHO, 1, 1)) * TE(U, 1/2, 1) / (2 * DR) +
(1 + 1) * ((TE(RHO, 1, 0) + RHO(0, P)) * TE(U, 1/2, 0) - (TE(RHO, 1, 0) +
RHO(0, P)) * TE(U, 1/2, 0) / (2 * DR) - (TE(U, 1/2, 0) + (TE(RHO, 1, 0) -
RHO(0, P)) - TE(T, 1/2, 0) * (RHO(0, P) - TE(RHO, 1, 1))) / DR**2

WRITE DERMOD, DE, FDE, "END PHASE ONE"

# -- -- -- --

FDE = FDE (RP, RM = R*DR/2, R=DR/2)
DF = DE (RP, RM = R*DR/2, R=DR/2)
FDF = FNE (NR, DT = LAM*DR, LAM*DT)

# CHECK FOR TRUNCATION ERRORS OF NEGATIVE ORDER
NUMER = ANUM (FDF, DENOM)
IF (DEG(DENOM, LAM), GT, 0) WRITE FDF, "MAY ABORT DUE TO NEGATIVE ORDER ERROR"
NUMER = 0; DENOM = 0

# CONVERT DE AND FDE TO TRUNCATED POWER SERIES
DETPS = TPS ( DE(DR, DT = DR*LAM, DT*LAM), LAM, ORD)
FDETPS = TPS (FDE, LAM, ORD)
FDE = 0

WRITE DETPS, FDETPS

# BEGIN REDUCTION OF ERRORS
FDETPS = TPS (TPSDEV(FDETPS, LAM), LAM, IMAX(RORD, TORD))
FDETPS = TPSCHOP (FDETPS, RORD, TORD)
DETPS = TPSCHOP (DETPS, RORD, TORD)
WRITE FDETPS

cntps = ARRSBS (FDETPS, (F1, F2), (1, 1))
TER = FDETPS - TPS (TPSDEV(DETPS, LAM), LAM, TPSORD(FDETPS))
WRITE FDETPS, CNTPS, "TER WITH ALL TIME DERIVATIVES", TER

# COMPUTE AND PUNCH MODIFIED EQUATION
WRITE "CONSTRUCT THE MODIFIED EQUATION"
MODER = TPS (DF, DERMOD, DER (DE, DERMOD))
IF (MODER(1) = 0) RETURN, "INCORRECT DERMOD"
NUMER = ANUM ((MODER(1)*DERMOD-DE-TPSDEV(TER, 1))/MODER(1), DENOM)
WRITE RORD, TORD, NUMER, DENOM
WRITE (25) RORD, TORD, DERMOD, NUMER, DENOM

NAME/EXTNAME USE TYPE STRUC PREC CLASS SCOPE DB LAY ADDR

CONTPS VAR ALG A L *001
DETPS VAR ALG A L *001
DIF IND ALG L *001
DR IND ALG L *001
NT IND ALG L *001
FDETPS VAR ALG A L *001
F1 IND ALG L *001
F2 IND ALG L *001
G1 IND ALG L *001
G2 IND ALG L *001
LAM IND ALG L *001
PHI IND ALG L *001
RM IND ALG L *001
RP IND ALG L *001
R IND ALG L *001
TER VAR ALG A L *001
THETA IND ALG L *001
TIM IND ALG L *001
P IND ALG A L *001
T IND ALG A L *001
U IND ALG A L *001
RHO IND ALG A L *001
ANUM/S9ANUM PROC ALG L S X
ARRSBS PROC ALG A L S X
DIF PROC ALG L S X
DDT PROC INT S X
DENOM VAR ALG L
DERMOD VAR ALG L
DE VAR ALG L
FDE VAR ALG L
IMAX/SQIMAX  PROC INT  L  S  X  
LAM2   VAR ALG   S  X  
MAIN   PROC       L  S  X  
MODEQ  VAR ALG   A  L  
M   VAR INT       L  
NIUMER VAR ALG       L  
N   VAR INT       S  X  
ORD   VAR INT       S  X  
ORDO  VAR INT       S  X  
TETPS  PROC ALG   A  L  S  X  
TE    PROC ALG   L  S  X  
TORO  VAR INT       L  S  X  
TPSCHOP PROC ALG   A  L  S  X  
TPSEVL PROC ALG   L  S  X  
TPSMUL PROC ALG   A  L  S  X  
TPSORD PROC INT   L  S  X  
TPSSRS PROC ALG   A  L  S  X  
TPS   PROC ALG   L  S  X  
XP    PROC INT   A  S  X  

ALTRAN VERSION 1 LEVEL 9

1  PROCEDURE TE (A,AX,AT, B,BX,RT, C,CX,CT, D,DX,DT)  
2    # 2-D TAYLOR SERIES EXPANSION OF THE PRODUCT A*B*C*D  
3  
4  VALUE A,AX,AT, B,BX,RT, C,CX,CT, D,DX,DT  
5  LONG ALGEBRAIC ARRAY A,B,C,D  
6  LONG ALGEBRAIC AX,AT, BX,RT, CX,CT, DX,DT  
7  ALTRAN ALGEBRAIC ARRAY TETPS  
8  ALTRAN ALGEBRAIC TPSFV  
9  
10 RETURN ( TPSEVL(TETPS(A,AX,AT, B,BX,RT, C,CX,CT, D,DX,DT), 1) )  
11  
12 END  

<table>
<thead>
<tr>
<th>NAME/EXTNAME</th>
<th>USE TYPE STRUC PPFC CLASS SCOPE DB LAY ADDR</th>
</tr>
</thead>
<tbody>
<tr>
<td>AT</td>
<td>VAR ALG L V</td>
</tr>
<tr>
<td>AX</td>
<td>VAR ALG L V</td>
</tr>
<tr>
<td>A</td>
<td>VAR ALG A L V</td>
</tr>
<tr>
<td>RT</td>
<td>VAR ALG L V</td>
</tr>
<tr>
<td>RX</td>
<td>VAR ALG L V</td>
</tr>
<tr>
<td>B</td>
<td>VAR ALG A L V</td>
</tr>
<tr>
<td>CT</td>
<td>VAR ALG L V</td>
</tr>
<tr>
<td>CX</td>
<td>VAR ALG L V</td>
</tr>
<tr>
<td>C</td>
<td>VAR ALG A L V</td>
</tr>
<tr>
<td>DT</td>
<td>VAR ALG L V</td>
</tr>
<tr>
<td>DX</td>
<td>VAR ALG L V</td>
</tr>
<tr>
<td>TETPS</td>
<td>PROC ALG A L S X</td>
</tr>
<tr>
<td>TE</td>
<td>PROC L S X</td>
</tr>
<tr>
<td>TPSEVL</td>
<td>PROC ALG L S X</td>
</tr>
<tr>
<td>1</td>
<td>CONS INT S</td>
</tr>
</tbody>
</table>

ALTRAN VERSION 1 LEVEL 9

1
2
PROCEDURE TETPS (A, AX, AT, B, RX, BT, C, CX, CT, D, DX, DT)
3
4 # 2.0 TPS TAYLOR SERIES OF THE PRODUCT A*B*C*D
5
6 VALUE A, AX, AT, B, RX, BT, C, CX, CT, D, DX, DT
7 LONG ALGEBRAIC ARRAY A, B, C, D
8 ALTRAN ALGEBRAIC ARRAY TETPS, TAYLOR, TPSMUL
9 IF (NULL(R)) RETURN (TAYLOR(A, AX, AT))
10 RETURN (TPSMUL(TAYLOR(A, AX, AT), TETPS(R, RX, BT, C, CX, CT, D, DX, DT)))
11 END

<table>
<thead>
<tr>
<th>NAME/EXTNAME</th>
<th>USE TYPE STRUC PPFC CLASS SCOPE DB LAY ADDR</th>
</tr>
</thead>
<tbody>
<tr>
<td>AT</td>
<td>VAR ALG L V</td>
</tr>
<tr>
<td>AX</td>
<td>VAR ALG L V</td>
</tr>
<tr>
<td>A</td>
<td>VAR ALG A L V</td>
</tr>
<tr>
<td>RT</td>
<td>VAR ALG L V</td>
</tr>
<tr>
<td>RX</td>
<td>VAR ALG L V</td>
</tr>
<tr>
<td>B</td>
<td>VAR ALG A L V</td>
</tr>
<tr>
<td>CT</td>
<td>VAR ALG L V</td>
</tr>
<tr>
<td>CX</td>
<td>VAR ALG A L V</td>
</tr>
<tr>
<td>C</td>
<td>VAR ALG A L V</td>
</tr>
<tr>
<td>DT</td>
<td>VAR ALG L V</td>
</tr>
<tr>
<td>DX</td>
<td>VAR ALG L V</td>
</tr>
</tbody>
</table>
**ALTRAN VERSION 1 LEVEL 9**

1. PROCEDURE TAYLOR (F, A, R)
2. # 2=0 TPS TAYLOR SERIES OF THE VARIABLE F
3. VALUE F, A, R
4. EXTERNAL ALGEBRAIC DDR, DDT
5. EXTERNAL INTEGER ORD
6. INTEGER I, J
7. LONG ALGEBRAIC A, R
8. LONG ALGEBRAIC ARRAY F
9. LONG ALGEBRAIC ARRAY(R;ORD) TAY=(F(0,A), ORD=0)
10. INTEGER ARRAY(M;10) FACT=(1,1,2,6,24,120,720,5040,40320,362880,3628800)
11. INTEGER ARRAY(M;10) COF=(1,1;1)
12. IF (A,EQ,P) DO # DIFF W,R,T, T
13. IF (B,EQ,P) RETURN (TAY)
14. IF I=1,ORD
15. TAY(I) = (R*DDT)**I*F(0,I)/FACT(I)
16. END
17. RETURN (TAY)
18. END
19. IF (B,EQ,P) DO # DIFF W,R,T, R
20. IF I=1,ORD
21. TAY(I) = (A*DDR)**I*F(I,P)/FACT(I)
22. END
23. RETURN (TAY)
24. END
25. DO I=1,ORD # DIFF W,R,T, R AND T
26. DO J=I,I-1 : COF(J)=COF(J)+COF(J-1) ; DDEND
27. DO J=0,I
28. TAY(I) = TAY(I) + COF(J)*(A*DDR)**J*(R*DDT)**(I-J)*F(J,I-J)
29. END
30. TAY(I) = TAY(I)/FACT(I)
31. END
32. RETURN (TAY)
33. END
PROCEDURE TPSCHOP (A, RORD, TORD)

# CHOP THE P-D TPS TO ORDER RORD IN DR AND TO ORDER TORD IN DT

WALUFA, RORD, TORD
EXTERNAL ALGEBRAIC DDR, DDT, LAM2
LONG ALGEBRAIC ARRAY A
INTEGER I, RORD, TORD, ORD=TPSORDER(A)
ALTRAN ALGEBRAIC ARRAY TPS
ALTRAN ALGEBRAIC TPSEVI
ALTRAN SHORT INTEGER TPSORD

DO I=0, ORD
   A(I) = TPSEVL (TPS(A(I), DDR, RORD), DDR)
   A(I) = TPSEVL (TPS(A(I), DDT, TORD), DDT)
END
RETURN (A)
END
<table>
<thead>
<tr>
<th>NAME/EXTNAME</th>
<th>USE</th>
<th>TYPE</th>
<th>STRUC</th>
<th>PREC</th>
<th>CLASS</th>
<th>SCOPE</th>
<th>DB</th>
<th>LAY</th>
<th>ADDR</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>VAR</td>
<td>ALG</td>
<td>A</td>
<td>L</td>
<td>V</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DDR</td>
<td>VAR</td>
<td>ALG</td>
<td>S</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DDT</td>
<td>VAR</td>
<td>ALG</td>
<td>S</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>VAR</td>
<td>INT</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LAMP</td>
<td>VAR</td>
<td>ALG</td>
<td>S</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ORD</td>
<td>VAR</td>
<td>INT</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RORD</td>
<td>VAR</td>
<td>INT</td>
<td>V</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TORD</td>
<td>VAR</td>
<td>INT</td>
<td>V</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TPSCHOP</td>
<td>PROC</td>
<td>L</td>
<td>S</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TPSEVL</td>
<td>PROC</td>
<td>ALG</td>
<td>L</td>
<td>S</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TPSORD</td>
<td>PROC</td>
<td>INT</td>
<td>S</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TPS</td>
<td>PROC</td>
<td>ALG</td>
<td>A</td>
<td>L</td>
<td>S</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>CONS</td>
<td>INT</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>S</td>
<td></td>
</tr>
</tbody>
</table>

ALTRAN VERSION 1 LEVEL 0

1
2
3
PROCEDURE ARRSBS (A, LHS, RHS)
4
5      # SUBSTITUTE THE LIST RHS FOR THE LIST LHS IN THE 1-D ARRAY A
6
7      VALUE A, LHS, RHS
8      LONG ALGEBRAIC ARRAY A, LHS, RHS
9      INTEGER ARRAY DB=DRINFO(A)
10     INTEGER I
11
12     DO I=OR(1, A), DR(I, 1)
13     A(I) = A(I)(LHS=RHS)
14     DOEND
15     RETURN (A)
16     END

<table>
<thead>
<tr>
<th>NAME/EXTNAME</th>
<th>USE</th>
<th>TYPE</th>
<th>STPUC</th>
<th>PREC</th>
<th>CLASS</th>
<th>SCOPE</th>
<th>DB</th>
<th>LAY</th>
<th>ADDR</th>
</tr>
</thead>
<tbody>
<tr>
<td>ARRSBS</td>
<td>PROC</td>
<td>L</td>
<td>S</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A</td>
<td>VAR</td>
<td>ALG</td>
<td>A</td>
<td>L</td>
<td>V</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DRINFO/S9DBIN</td>
<td>PROC</td>
<td>INT</td>
<td>A</td>
<td>S</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DB</td>
<td>VAR</td>
<td>INT</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>VAR</td>
<td>INT</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LHS</td>
<td>VAR</td>
<td>ALG</td>
<td>A</td>
<td>L</td>
<td>V</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RHS</td>
<td>VAR</td>
<td>ALG</td>
<td>A</td>
<td>L</td>
<td>V</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>CONS</td>
<td>INT</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>S</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>CONS</td>
<td>INT</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>S</td>
<td></td>
</tr>
</tbody>
</table>
PROCEDURE XP (N)
VALUE N
INTEGER I, J, N,
INTEGER ARRAY(P,N,N) EXP=1
DO I=0,N
DO J=0,N=1
EXP(I,J) = 7
GOEND
DOEND
RETURN (EXP)
END

NAME/EXTNAME USE TYPE STRUC PREC CLASS SCOPE DB LAY ADDR

EXP VAR INT A
I VAR INT
J VAR INT
N VAR INT V
XP PROC L S X
0=0@1
data
0 CONS INT 5
1 CONS INT 5
7 CONS INT 5

# DERH0D
RHD(0,1)
# DE
- ( T(R,A)*RHD(2,2) + T(1,2)*RHD(1,8) + U(R,A)*RHD(1,0) + U(1,0)*RHD(0,0) + RHD(0,1) )
# FOD
( 6*DR**18*DT*G1*(U(5,1)*RHD(6,8) + 3*DR**1P*DT*G1*U(6,0)*RHD(5,1) + DR**10*T(b,A)*RHD(b,8) + 6*DR**18*U(5,8)*RHD(6,0) + 3*DR**18*U(b,A)*RHD(5,8) + 6*DR**18*DT*G1*U(3,3)*RHD(6,8) + 18*DR**18*DT*G1*U(4,2)*RHD(5,1) + 90*DR**18*DT*G1*U(5,1)*
RHD(4,2) + 18*DR**18*DT*G1*U(3,3)*RHD(3,3) + 24*DR**18*DT*G1*U(3,2)*RHD(3,2) + 36*DR**18*DT*G1*U(4,2)*RHD(5,1) +
18*DR**18*DT*G1*U(4,2)*RHD(5,0) + 90*DR**18*DT*G1*U(5,1)*RHD(4,2) + 18*DR**18*DT*G1*U(5,1)*RHD(4,1) + 30*DR**18*DT*G1*U(5,1)*
G1*U(b,A)*RHD(3,2) + 48*DR**18*DT*G1*U(3,1)*RHD(6,8) + 36*DR**18*DT*G1*U(4,2)*RHD(5,1) + 36*DR**18*DT*G1*U(4,1)*RHD(5,0) +
18*DR**18*DT*G1*U(5,1)*RHD(4,1) + 18*DR**18*DT*G1*U(5,1)*RHD(4,0) + 6*DR**18*DT*G1*U(5,2)*RHD(3,1) - 120*DR**18*DT(4,1)*
RHD(6,8) + 72*DR**18*DT(5,1)*RHD(5,0) + 30*DR**18*DT(6,8) + 48*DR**18*DT(3,0)*RHD(6,8) + 36*DR**18*DT(4,0)*RHD(5,0) +
18*DR**18*DT(5,0)*RHD(4,0) + 6*DR**18*DT(6,0)*RHD(3,0) + 96*DR**18*DT*G1*U(1,5)*RHD(5,8) + 72*DR**18*DT*G1*U(2,4)*RHD(5,1) +

RHO(0,4) + 1382080D*T*5*G1+U(1,2)*RHO(0,3) + 1382080D*T*5*G1+U(1,3)*RHO(0,2) + 691200D*T*5*G1+U(1,4)*RHO(0,1) + 1382080D*T*5*G1+U(1,5)*RHO(0,0) + 2384080D*T*5*RHO(0,6) + 691200D*T*4*G1+U(0,0)*RHO(1,6) + 2768000D*T*4*G1+U(0,1)*RHO(1,5) + 4147200D*T*4*G1+U(0,2)*RHO(1,4) + 2768000D*T*4*G1+U(0,3)*RHO(1,3) + 691200D*T*4*G1+U(0,4)*RHO(1,2) + 691200D*T*4*G1+U(1,0)*RHO(1,1) + 691200D*T*4*G1+U(1,1)*RHO(1,0) + 691200D*T*4*G1+U(1,2)*RHO(0,0) + 691200D*T*4*G1+U(1,3)*RHO(0,1) + 691200D*T*4*G1+U(1,4)*RHO(0,2) + 691200D*T*4*G1+U(1,5)*RHO(0,3) + 691200D*T*4*G1+U(0,0)*RHO(1,0) + 691200D*T*4*G1+U(0,1)*RHO(1,1) + 691200D*T*4*G1+U(0,2)*RHO(1,2) + 691200D*T*4*G1+U(0,3)*RHO(1,3) + 691200D*T*4*G1+U(0,4)*RHO(1,4) + 691200D*T*4*G1+U(0,5)*RHO(1,5) + 691200D*T*4*G1+U(0,6)*RHO(1,6)

# END PHASE ONE

# DETPS

# DETPS

# DETPS

# DETPS
96*DT**6*G1(U(0,2)) = RHD(1,4) + 128*DT**6*G1(U(0,3)) = RHD(1,3) + 96*DT**6*G1(U(0,4)) = RHD(1,2) + 384*DT**6*G1(U(0,5)) = RHD(1,1) + 64*DT**6*G1(U(0,6)) = RHD(1,0) + 64*DT**6*G1(U(1,0)) = RHD(0,6) + 384*DT**6*G1(U(1,1)) = RHD(0,5) + 96*DT**6*G1(U(1,2)) = RHD(0,4) + 128*DT**6*G1(U(1,3)) = RHD(0,3) + 96*DT**6*G1(U(1,4)) = RHD(0,2) + 384*DT**6*G1(U(1,5)) = RHD(0,1) / 40000

# FOETPS

\( T(0,0) \cdot \text{RHD}(2,0) + T(1,0) \cdot \text{RHD}(1,0) = U(0,0) \cdot \text{RHD}(0,0) = \text{RHD}(0,1) \),

\( DT = \{ 2*G1(U(0,0)) \cdot \text{RHD}(1,1) + 2*G1(U(1,0)) \cdot \text{RHD}(0,1) + 2*G1(U(0,1)) \cdot \text{RHD}(1,0) + 2*G1(U(1,1)) \cdot \text{RHD}(0,0) + \text{RHD}(0,2) \} / 2 \),

\( DR**2 = \{ 2*T(0,0) \cdot \text{RHD}(4,0) + 4*T(1,0) \cdot \text{RHD}(3,0) + 3*T(2,0) \cdot \text{RHD}(2,0) + T(3,0) \cdot \text{RHD}(1,0) = U(0,0) \cdot \text{RHD}(3,0) = 6*U(1,0) \cdot \text{RHD}(2,0) + 3*U(2,0) \cdot \text{RHD}(1,0) = U(3,0) \cdot \text{RHD}(0,0) \} / 24 \)

# FOETPS

\( \{ 2*G1(U(0,0)) \cdot \text{RHD}(1,1) + 2*G1(U(0,1)) \cdot \text{RHD}(0,1) + 2*G1(U(0,2)) + 2*G1(U(1,1)) \cdot \text{RHD}(0,0) + \text{RHD}(0,2) \} / 2 \),

\( 37801 = DR**2 = \{ 2*T(0,0) \cdot \text{RHD}(4,0) + 4*T(1,0) \cdot \text{RHD}(3,0) + 3*T(2,0) \cdot \text{RHD}(2,0) + T(3,0) \cdot \text{RHD}(1,0) = U(0,0) \cdot \text{RHD}(3,0) = 6*U(1,0) \cdot \text{RHD}(2,0) + 3*U(2,0) \cdot \text{RHD}(1,0) = U(3,0) \cdot \text{RHD}(0,0) \} / 24 \)

# CONTPS

\( \{ 2*G1(U(0,0)) \cdot \text{RHD}(1,1) + 2*G1(U(0,1)) \cdot \text{RHD}(0,1) + 2*G1(U(0,2)) + 2*G1(U(1,1)) \cdot \text{RHD}(0,0) + \text{RHD}(0,2) \} / 2 \),

\( DR**2 = \{ 2*T(0,0) \cdot \text{RHD}(4,0) + 4*T(1,0) \cdot \text{RHD}(3,0) + 3*T(2,0) \cdot \text{RHD}(2,0) + T(3,0) \cdot \text{RHD}(1,0) = U(0,0) \cdot \text{RHD}(3,0) = 6*U(1,0) \cdot \text{RHD}(2,0) + 3*U(2,0) \cdot \text{RHD}(1,0) = U(3,0) \cdot \text{RHD}(0,0) \} / 24 \)

# WITH ALL TIME DERIVATIVES

# TER

\( \{ 0 \),

\( 32*DT = \{ 2*G1(U(0,0)) \cdot \text{RHD}(1,1) + 2*G1(U(1,0)) \cdot \text{RHD}(0,1) + 2*G1(U(1,1)) \cdot \text{RHD}(0,0) + \text{RHD}(0,2) \} / 2 \),

\( 37801 = DR**2 = \{ 2*T(0,0) \cdot \text{RHD}(4,0) + 4*T(1,0) \cdot \text{RHD}(3,0) + 3*T(2,0) \cdot \text{RHD}(2,0) + T(3,0) \cdot \text{RHD}(1,0) = U(0,0) \cdot \text{RHD}(3,0) = 6*U(1,0) \cdot \text{RHD}(2,0) + 3*U(2,0) \cdot \text{RHD}(1,0) = U(3,0) \cdot \text{RHD}(0,0) \} / 24 \)

# CONSTRUCT THE MODIFIED EQUATION

# RORD

2

# TNID

1

# NUMER

\( 2*DR**2*T(0,0) = \text{RHD}(4,0) + 4*DR**2*T(1,0) \cdot \text{RHD}(3,0) + 3*DR**2*T(2,0) \cdot \text{RHD}(2,0) + \text{RHD}(1,0) + 4*DR**2*T(3,0) \cdot \text{RHD}(1,0) + 4*DR**2*T(4,0) \cdot \text{RHD}(0,0) + \text{RHD}(0,2) + 3*DR**2*T(0,1) \cdot \text{RHD}(4,0) + 4*DR**2*T(1,1) \cdot \text{RHD}(3,0) + 3*DR**2*T(2,1) \cdot \text{RHD}(2,0) + \text{RHD}(1,0) + 4*DR**2*T(3,1) \cdot \text{RHD}(1,0) + 4*DR**2*T(4,1) \cdot \text{RHD}(0,0) + \text{RHD}(0,2) + 24*T(0,0) \cdot \text{RHD}(2,0) + 24*T(1,0) \cdot \text{RHD}(1,0) - 24*U(0,0) \cdot \text{RHD}(0,0) \)
# DENOM

24

*** NORMAL RETURN FROM MAIN PROCEDURE

*** RUN STATISTICS

 14,264 SECONDS ELAPSED
 131,070 WORDS IN WORKSPACE
   14 DIGITS IN SHORT INTEGERS
   28 DIGITS IN LONG INTEGERS
   0 GARBAGE COLLECTIONS
 94,557 WORDS OF WORKSPACE NEVER USED

$EJ$
APPENDIX B
FLOW CHARTS FOR THE TRUNCATION ERROR EXPANSION PROGRAM

START

MAIN

Declarations, initialize maximum order of the expansions, initialize maximum exponents by invoking PROCEDURE XP.

Lines 2-15

Set RORD, TORD, DERMOD, DE and FDE with user-supplied ALTRAN statements.

Lines 19-27

Optional substitutions: RP = R + DR/2, RM = R - DR/2.

Lines 33-34

Substitute δr = λ δr, δt = λ δt.

Line 35

Check for possible errors of order λ⁻¹, print warning message if appropriate, and set NUMER = DENOM = 0 for efficiency.

Lines 38-40

Convert DE and FDE to truncated power series in λ.

Lines 43-44

END

Print and punch RORD, TORD, DERMOD, NUMER, and DENOM.

Lines 65-66

Split the terms of MODEQ equal to DERMOD into their numerator NUMER and denominator DENOM.

Line 63

Set the truncated power series for the modified equation, MODEQ, equal to DE expanded in DERMOD. Check to make sure DERMOD appears to first order in DE.

Lines 61-62

Split the terms of MODEQ equal to DERMOD into their numerator NUMER and denominator DENOM.

Line 63

Compute truncation error TER as modified equation FDETPS minus differential equation DETPS.

Line 56

Use array substitution procedure ARRSBS to set F1 and F2 to unity for the special case of a uniform mesh (optional).

Line 55

Use procedure TPSCHØP to discard terms of FDETPS and DETPS that are of higher order than RORD in DR and TORD in DT.

Lines 51, 52

Discard terms of FDETPS that are of higher order than max (RORD, TORD) in λ.

Line 50
Calls TETPS to do the expansion as a TFS, then sums the terms of the TPS to return an algebraic form of the expansion.

Is the expansion for only a single variable?

Call TETPS to get TPS expansion of all but first variable; call TAYLOR to get TPS expansion of first variable; call TPSMUL to multiply the two TPS's together.

Initialize: Taylor series terms TAY are set to case of A=B=0; FACT = table of factorials; COF is set for the case A=B=0.

Is the expansion for only a single variable?

Call TAYLOR to get TPS expansion of that one variable.

Do a 1-d Taylor series in r.

Do a 1-d Taylor series expansion in time.

A = 0 ?

B = 0 ?

RETURN

RETURN

RETURN

RETURN
START TPSCHQP

Initialize; find order of TPS A.

Line 8

For each array element of A, cut out terms of order greater than RORD in φt and then TORD in φt.

Lines 13-16

RETURN Line 18

START ARRSBS

Initialize DB to find DB(1,0) & DB(1,1) the lower & upper indices of array elements of A.

Line 7

Substitute the array RHS for the array LHS in each element of A.

Lines 10-12

RETURN Line 14

START XP

Computes maximum exponents for each element of the indeterminant arrays in the layout in MAIN. Begins by filling XP with ones.

Line 4

For sufficiently small i+j, the array element is set to 7.

Lines 6-10

RETURN Line 12
APPENDIX C

INSTRUCTIONS AND LISTING FOR THE TIME DERIVATIVE ELIMINATION PROGRAM

This appendix describes the current form of the code that eliminates time derivatives from the modified equation. This program is continuing to evolve, and our goal is to eventually combine this code with the expansion code to form a completely automated package that we will describe in a future report. However, this first generation program is useful enough to justify its inclusion in this report.

Input for this program is punched by either itself or the expansion program. If only one equation is being manipulated, there must be a data card setting SDER to zero. If there is a system of two equations, only the first equation read in (the primary equation) is differentiated. However, both the primary and secondary (the second equation read in) equations have derivatives of DERMOD eliminated. For the secondary equation, SDER, SNUM, and SDEN are the analogs of DERMOD, NUCER, and DENOM for the primary equation. RORD and TORD are the same for both equations.

A problem is begun by running the expansion code and using its punched output as input for the elimination code. Each run of the elimination code will reduce the order of time derivatives present by at most one. If a given run does not successfully eliminate all the time derivatives, its punched output is used as input for the next run. The optimum strategy for handling systems of equations has not been worked out.

The listings include the setup statements and results from a sample expansion run, a complete listing and first run of the sample problem, and the results of the second elimination run. The input and results for the expansion run are given below.

```
18    RORD = 7    1    TORD = 1
20    CEw = w * T(0,1)
21    DIF = T(T,1) = DIF * T(2,P) = 2 * DIF * T(1,0) / R
22    FOE = T(T,T,T) = T(T,0) / DT = DIF * (R**2*2*(T(T,T,T) + T(0,0))) / R
24    (R**2*R**2)

# CONSTRUCT THE MODIFIED EQUATION

# RORD

2

# TORD

1

# NUMER

DR**2*R**2*T(3,0)*DIF + 4*DR**2*R**2*T(2,0)*DIF + 3*DR**2*R**2*T(2,0)*DIF = 6*DT*R**2*2*T(0,0) + 12*R**2*T(2,0)*DIF + 24*R*T(1,0)*DIF

# DENOM

12*R**2
```

The remainder of this appendix is a listing of the time derivative elimination program and the output of the two runs needed to complete this sample problem.
PROCEDURE MAIN # PROGRAM TO READ MODIFIED EQUATION AND ELIMINATE T DERIVS

EXTERNAL INTEGER N1=7, N2=7
INTEGER M=31, MM=7
LONG ALGEBRAIC (P1M1, P1M2, P1M3, P1M4, P1M5, G1M1, G2M1, LAM1M, F1M1,
    F2M, DTMZ, (P1M1, P1M2)SIMH, (P1M1, P1M2)SHHH, RH<0,1,1,1,2)SIMH, TMMH,
    (P1M1, P1M2)SHHM) DERMOD, NUMEP, DENOM, SECOND, SNUM, SDER, SDEN
EXTERNAL LONG ALGEBRAIC LAM1LAM, SER, TIMZ=TI
EXTERNAL LONG ALGEBRAIC ARRAY R1=RHO, P1=P1, T1=T1, U1=U1
INTEGER I, J, ROPO, TOMP, IT, IR, ISR, IST, NT
INTEGER ARRAY (P1M2) TSRO
LONG ALGEBRAIC ARRAY (Q1N1, Q1N2) DERM
LONG ALGEBRAIC ARRAY SIMH
LONG ALGEBRAIC ALTRAN TDEP, ROER
ALGEBRAIC ARRAY ALTRAN TPS
ALGEBRAIC ALTRAN TOSVEI
REAL NFTA, ETIM

# - - - - - - - - - - - - - - -
READ ROPO, TOMP, TORMD, NUMER, DENOM, SDER
WRITE "INITIALIZATION", ROPO, TOMP, TORMD, NUMER, DENOM, SDER
SNUM=0
SDEN=1
IF (SDER, NE, 0) GO
READ SNUM, SDEN
WRITE SNUM, SDEN
COLO

# - - - - - - - - - - - - - - -
# SET UP THE SUBSTITUTION MATRIX
DO T = 0, N1
    IR = I
    DO J = 0, N2
        IT = J
        IF (DERMOD, NE, P1M0(I, J)) GO TO A1
        SUB = QM0
        GO TO A1
A1: CONTINUE
    IF (DERMOD, NE, U1(I, J)) GO TO A2
    SUB = U
    GO TO A3
A2: CONTINUE
    IF (DERMOD, NE, P(I, J)) GO TO A3
    SUB = P
    GO TO A3
A3: CONTINUE
49 IF (DERMON.\textlt;F,T,I,J) GO TO A4
50 SUB = T\textlt;F
51 GO TO B1
52 A4\textlt;CONTINUE
53 DO\textlt;END
54 DO\textlt;END
55 WRITE DERMON, "TILFGAL DFMON, ABORTING"
56 GO TO ST
57 B1\textlt;CONTINUE
58 WRITE IR, IT, SUB
59 IF I = IR, N1
60 IF J = IT, N2
61 NUR\textlt;V(I-IR, J=IT) = NUR(I,J)
62 NUR\textlt;V(I,J) = 0
63 NUR\textlt;V(I-IR, J+IT) = NUR(I,J)
64 NUR(I,J) = 0
65 DO\textlt;END
66 DO\textlt;END
67 ISR = N1 - IR
68 IST = N2 - IT
69 WRITE SUB, NUR\textlt;V, TSW, IST
70 DFLTA = TIME(TII)F; WRITE DFLTA,FTIME
71 # = = = = = = = = = = = = =
72 # CALCULATE HIGHEST ORDER DERIVATIVES NEEDED
73 #
74 IF = 0
75 IT = 0
76 IF J = 0, IST
77 ISR\textlt;V(J) = 0
78 IF I = ISR, 0, -1
79 NT = IMAX( IMAX( DEG(NUMER, SUB(I,J)), DEG(DENOM, SUB(I,J))),
80 IMAX(DEN(SNUM, SUB(I,J)), OEG(SDEN, SUB(I,J))))
81 IF (NT GT 0) DO
82 IR = I
83 IT = J
84 ISR\textlt;V(J) = I
85 GO TO NMO
86 DO\textlt;END
87 NMO\textlt;CONTINUE
88 DO\textlt;END; NT = DEG(NUMER* SUB(I,J)) + DEG(DENOM* SUB(I,J))
89 OEG(SDEN, SUB(I,J))
90 IF (IR GT 0 OR IT GT 0 OR NT GT 0) GO TO 95
91 WRITE "NO TIME DERIVATIVES FOUND THAT CAN BE ELIMINATED"
92 GO TO ST
93 OS\textlt;CONTINUE
94 ISR = IR
95 IST = IT
96 WRITE "HIGHEST ORDER OF DERIVATIVE TO BE COMPUTED", ISR, IST, ISRM
97 DFLTA = TIME(TII)F; WRITE DFLTA,FTIME
98 # = = = = = = = = = = = = =
# CREATE HIGHER ORDER DERIVATIVES

DERIV(U, N) = NUMER / DENOM
WRITE DERIV(U, N)

# PURE TIME DERIVATIVE OF ORDER IT

IF IT = 0, IST
   NUMER = NUM(DERIV(U, IT), DENOM)
   DERIV(U, IT) = (TOPR(NUMER)*DENOM - TOPR(DENOM)*NUMER) / DENOM**2
   WRITE "PURE TIME DERIVATIVE", IT, NUMER, DENOM, DERIV(U, IT)
END

WRITE "SPACE DERIVATIVES"
IF (TSRM(IT), GT, P) DO IT = 1, TSRM(IT)
   NUMER = NUM(DERIV(U = 1, IT), DENOM)
   DERIV(U, IT) = (PDER(NUMER)*DENOM - PDER(DENOM)*NUMER) / DENOM**2
   WRITE IT, NUMER, DENOM, DERIV(U, IT)
END

END

WRITE DERTV
DELTA=TIME(FTIME) WRITE DELTA,FTIME
NUMER=
DENOM=

# ELIMINATE TIME DERIVATIVES FROM THE PRIMARY MODIFIED EQUATION

IF J = 0, IST
   IF I = 0, TSRM(J)
      DERIV(V, O) = DERIV(V, O) (SUH(I, J) = DERIV(I, J))
      DERIV(V, O) = TPSVL(TPS(DERIV(V, O) (DP, DT = LAM*DP, LAM*DT), LAM, THD(DP, THD)))
      DERIV(V, O) = TPSVL(TPS(DERIV(V, O) (DP = LAM*DP), LAM, THD), LAM, THD))
      END
   END
   NUMER = ANUM (DERIV(V, O), DENOM)
   WRITE RORD, TORD, DERMON, NUMER, DENOM
   WRITE (PS) RORD, TORD, DERMON, NUMER, DENOM, SDER
   DELTA=TIME(ETIME), WRITE DELTA,FTIME
   NUMER=
   DENOM=
   IF (SDER, FO, 0) GO TO ST
   END

# ELIMINATE TIME DERIVATIVES FROM THE SECONDARY MODIFIED EQUATION

WRITE "SECONDARY EQUATION", SDER, SNWM, SDEN
SECOND = SNWM / SDEN
IF J = 0, IST
   IF I = 0, TSRM(J)
      SECOND = SECOND (SUH(I, J) = DERIV(I, J))
   END
   END

# ELIMINATE TIME DERIVATIVES FROM THE SECONDARY MODIFIED EQUATION

WRITE "SECONDARY EQUATION", SDER, SNWM, SDEN
SECOND = SNWM / SDEN
IF J = 0, IST
   IF I = 0, TSRM(J)
      SECOND = SECOND (SUH(I, J) = DERIV(I, J))
   END
   END
<table>
<thead>
<tr>
<th>NAME/EXTNAME</th>
<th>USF</th>
<th>TYPE</th>
<th>STORC</th>
<th>PREC</th>
<th>CLASS</th>
<th>SCOPE</th>
<th>DR</th>
<th>LAY</th>
<th>ADIR</th>
</tr>
</thead>
<tbody>
<tr>
<td>TSNM</td>
<td>VAR</td>
<td>INT</td>
<td>A</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DERIV</td>
<td>VAR</td>
<td>ALG</td>
<td>A</td>
<td>L</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DENOM</td>
<td>VAR</td>
<td>ALG</td>
<td>A</td>
<td>L</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DERMN</td>
<td>VAR</td>
<td>ALG</td>
<td>A</td>
<td>I</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DIF</td>
<td>IND</td>
<td>ALG</td>
<td>A</td>
<td>L</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DP</td>
<td>IND</td>
<td>ALG</td>
<td>A</td>
<td>L</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DT</td>
<td>IND</td>
<td>ALG</td>
<td>A</td>
<td>L</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>F1</td>
<td>IND</td>
<td>ALG</td>
<td>A</td>
<td>L</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>F2</td>
<td>IND</td>
<td>ALG</td>
<td>A</td>
<td>L</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>G1</td>
<td>IND</td>
<td>ALG</td>
<td>A</td>
<td>L</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>G2</td>
<td>IND</td>
<td>ALG</td>
<td>A</td>
<td>L</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LAM</td>
<td>IND</td>
<td>ALG</td>
<td>A</td>
<td>L</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NUMER</td>
<td>VAR</td>
<td>ALG</td>
<td>A</td>
<td>L</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RM</td>
<td>IND</td>
<td>ALG</td>
<td>A</td>
<td>L</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RP</td>
<td>IND</td>
<td>ALG</td>
<td>A</td>
<td>L</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SDEN</td>
<td>VAR</td>
<td>ALG</td>
<td>A</td>
<td>L</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SDFR</td>
<td>VAR</td>
<td>ALG</td>
<td>A</td>
<td>L</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SECOND</td>
<td>VAR</td>
<td>ALG</td>
<td>A</td>
<td>L</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SNUM</td>
<td>VAR</td>
<td>ALG</td>
<td>A</td>
<td>L</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TI</td>
<td>TND</td>
<td>ALG</td>
<td>A</td>
<td>L</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>U</td>
<td>TND</td>
<td>ALG</td>
<td>A</td>
<td>L</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>P</td>
<td>TND</td>
<td>ALG</td>
<td>A</td>
<td>L</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RHO</td>
<td>TND</td>
<td>ALG</td>
<td>A</td>
<td>L</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>T</td>
<td>TND</td>
<td>ALG</td>
<td>A</td>
<td>L</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ANUM/SNUM</td>
<td>PROC</td>
<td>ALG</td>
<td>A</td>
<td>L</td>
<td>S</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DEG/SQDEG</td>
<td>PROC</td>
<td>INT</td>
<td>A</td>
<td>L</td>
<td>S</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DELTA</td>
<td>VAR</td>
<td>REAL</td>
<td>A</td>
<td>L</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TIME</td>
<td>VAR</td>
<td>REAL</td>
<td>A</td>
<td>L</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>IMAX/SQIMAX</td>
<td>PROC</td>
<td>INT</td>
<td>A</td>
<td>L</td>
<td>S</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>IR</td>
<td>VAR</td>
<td>INT</td>
<td>A</td>
<td>L</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TSR</td>
<td>VAR</td>
<td>INT</td>
<td>A</td>
<td>L</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>IST</td>
<td>VAR</td>
<td>INT</td>
<td>A</td>
<td>L</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>IT</td>
<td>VAR</td>
<td>INT</td>
<td>A</td>
<td>L</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>I</td>
<td>VAR</td>
<td>INT</td>
<td>A</td>
<td>L</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
2. LISTING OF THE ELIMINATION PROGRAM AND FIRST RUN OF THE SAMPLE PROBLEM

ALTRAN VERSION 1 LEVEL 9

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29

PROCEDURE DER (A) = TIME DERIVATIVE OF AN ALGEBRATIC WITH DENOMINATOR = 1

EXTERNAL INTEGER N, NP
EXTERNAL LONG ALGEBRATIC LAM, S, TIM
EXTERNAL LONG ALGEBRATIC ARRAY P1, P1, I1, U1

VALUE A
LONG ALGEBRATIC A, DER
INTEGER I, J

# DIFFERENTIATE WITH RESPECT TO TIME (TIM)

DER = DIFF (A, TIM)

# CHAIN RULE FOR IMPLICIT DIFFERENTIATION OF DEPENDENT VARIABLES

DO I = N1, N2, -1
IF (A, NP, A(I1, I2), T1(I, N2), P(I1, NP), U1(I, NP) = 0, 0, 0, 0) GO TO KICKOUT
DO J = N2-1, N-1
DER = DER + DIFF(A, P(I1, J)) * P1(I1, J+1) + DIFF(A, P(I1, I)) * P1(I1, J+1)*
DIFF (A, U1(I, J)) * U1(I, J+1) + DIFF (A, T1(I, J)) * T1(I, J+1)

DOEND

END

RETURN(DER)

KICKOUT; WRITE "ERROR IN DER = N IS TOO SMALL", A, DER, N1, N2, I, J

END

<table>
<thead>
<tr>
<th>NAME/EXTNAME</th>
<th>USE</th>
<th>TYPE</th>
<th>STRUC-PHFC</th>
<th>CLASS</th>
<th>SCOPE</th>
<th>OR</th>
<th>LAY</th>
<th>ADDR</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>VAR</td>
<td>ALG</td>
<td>L</td>
<td>V</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DER</td>
<td>VAR</td>
<td>ALG</td>
<td>L</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DIFF/A9DIFF</td>
<td>PROC</td>
<td>ALG</td>
<td>L</td>
<td>S</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>T</td>
<td>VAR</td>
<td>INT</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>J</td>
<td>VAR</td>
<td>INT</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LAM</td>
<td>VAR</td>
<td>ALG</td>
<td>L</td>
<td>S</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>N1</td>
<td>VAR</td>
<td>INT</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>N2</td>
<td>VAR</td>
<td>INT</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>P1</td>
<td>VAR</td>
<td>ALG</td>
<td>A</td>
<td>L</td>
<td>S</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>P2</td>
<td>VAR</td>
<td>ALG</td>
<td>A</td>
<td>L</td>
<td>S</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>S</td>
<td>VAR</td>
<td>ALG</td>
<td>L</td>
<td>S</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TDFR</td>
<td>PROC</td>
<td>ALG</td>
<td>L</td>
<td>S</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TIM</td>
<td>VAR</td>
<td>ALG</td>
<td>L</td>
<td>S</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
3. RESULTS OF THE SECOND RUN OF THE ELIMINATION PROGRAM

ALTRAN VERSION 1 LEVEL 9

PROCEDURE RDPF (A) # TIME DERIVATIVE OF AN ALGEBRAIC WITH DENOMINATOR = 1

EXTERNAL INTEGER N1, N2
EXTERNAL LONG ALGEBRAIC IAN, S, T1M
EXTERNAL LONG ALGEBRAIC ARRAY R1, P1, T1, U1

VALUE A
LONG ALGEBRAIC A, DER
INTEGER I, J

# DIFFERENTIATE WITH RESPECT TO R (S)

DER = DIFF (A, S)

# CHAIN RULE FOR IMPPLICIT DIFFERENTIATION OF DEPENDENT VARIABLES

DO J = N2, 1, -1
IF (A, R1, R1(I1, J), T1(I1, J), P1(I1, J), U1(I1, J) = 0, 0, 0) GO TO KICKOUT
DO I = N1, 1, -1
DER = DER + DIFF(A, R1(I, J)) * R1(I+1, J) + DTFF(A, P1(I, J)) * P1(I+1, J) +
DIFF(A, U1(I, J)) * U1(I+1, J) + DIFF(A, T1(I, J)) * T1(I+1, J)
ENDDO
ENDD

RETURN(DER)

KICKOUT: WRITE "ERROR IN RDPF = - N IS TOO SMALL", A, DER, N1, N2, I, J

NAME/EXTNAME USE TYPE STRUCT PREC CLASS RDPF DE LAY ADDR

A VAR ALG I L V
DER VAR ALG L S X
DIFF/AQDIFF PROC ALG L S X
# INITIALIZATION
# RROD
  2
# TORD
  1
# OERMOO
  T(R,1)
  # NUMER
  * ( b*D*T*R*2*T(R,2) = D*R*2*R*2*D*T(T(3,0)) - 4*D*R*2*R*D*T(T(3,0)) - 3*D*R*2*D*T(T(2,8)) - 12*R*2*D*T(T(2,8)) - 24*R*D*T
  T(1,8) )
  # DENOM
  12*R*2
  # SDER
  0
  # VR
  0
  # IT
  1

# SUB
  ( T(R,R) ,
   T(R,1) ,
   T(R,2) ,
On this page and the next, the reader should be aware that the columns, beginning with T(0,3), are to be read as one continuous run.
<table>
<thead>
<tr>
<th>T(5,7)</th>
<th>T(1,1)</th>
<th>T(4,3)</th>
</tr>
</thead>
<tbody>
<tr>
<td>T(6,1)</td>
<td>T(1,2)</td>
<td>T(4,4)</td>
</tr>
<tr>
<td>T(6,2)</td>
<td>T(1,3)</td>
<td>T(4,5)</td>
</tr>
<tr>
<td>T(6,3)</td>
<td>T(1,4)</td>
<td>T(4,6)</td>
</tr>
<tr>
<td>T(6,4)</td>
<td>T(1,5)</td>
<td>T(4,7)</td>
</tr>
<tr>
<td>T(6,5)</td>
<td>T(1,6)</td>
<td>a</td>
</tr>
<tr>
<td>T(6,6)</td>
<td>T(1,7)</td>
<td>T(5,1)</td>
</tr>
<tr>
<td>T(6,7)</td>
<td>a</td>
<td>T(5,2)</td>
</tr>
<tr>
<td>T(7,1)</td>
<td>T(2,1)</td>
<td>T(5,3)</td>
</tr>
<tr>
<td>T(7,2)</td>
<td>T(2,2)</td>
<td>T(5,4)</td>
</tr>
<tr>
<td>T(7,3)</td>
<td>T(2,3)</td>
<td>T(5,5)</td>
</tr>
<tr>
<td>T(7,4)</td>
<td>T(2,4)</td>
<td>T(5,6)</td>
</tr>
<tr>
<td>T(7,5)</td>
<td>T(2,5)</td>
<td>T(5,7)</td>
</tr>
<tr>
<td>T(7,6)</td>
<td>T(2,6)</td>
<td>a</td>
</tr>
<tr>
<td>T(7,7)</td>
<td>T(2,7)</td>
<td>T(6,1)</td>
</tr>
<tr>
<td>a</td>
<td>T(3,1)</td>
<td>T(6,2)</td>
</tr>
<tr>
<td>a</td>
<td>T(3,2)</td>
<td>T(6,3)</td>
</tr>
<tr>
<td>a</td>
<td>T(3,3)</td>
<td>T(6,4)</td>
</tr>
<tr>
<td>a</td>
<td>T(3,4)</td>
<td>T(6,5)</td>
</tr>
<tr>
<td>a</td>
<td>T(3,5)</td>
<td>T(6,6)</td>
</tr>
<tr>
<td>a</td>
<td>T(3,6)</td>
<td>T(6,7)</td>
</tr>
<tr>
<td>a</td>
<td>T(3,7)</td>
<td>a</td>
</tr>
<tr>
<td>a</td>
<td>T(4,1)</td>
<td>T(7,1)</td>
</tr>
<tr>
<td>a</td>
<td>T(4,2)</td>
<td>T(7,2)</td>
</tr>
<tr>
<td>a</td>
<td>T(4,3)</td>
<td>T(7,3)</td>
</tr>
<tr>
<td>a</td>
<td>T(4,4)</td>
<td>T(7,4)</td>
</tr>
<tr>
<td>a</td>
<td>T(4,5)</td>
<td>T(7,5)</td>
</tr>
</tbody>
</table>
T(7,4),
T(7,7),
0

# ISR
7

# IST
6

# DELTA
1.5029262875

# ETIME
1.5029262875

# MAXIMUM ORDER OF DERIVATIVE TO BE COMPUTED
# ISP

# IST
1

# ISRM
(0,0,0,0,0,0,0,0,NULL)

# DELTA
4.194992240999F-1

# ETIME
1.9724165125

# DERIV(P,R)

  = ( 6*DT*R**2*T(8,2) + 3*DR**2*R**2*DIF*T(4,R) + 3*DR**2*R*DIF*T(3,0) + 3*DR**2*DIF*T(2,0) + 12*R**2*DIF*T(4,0) + 24*R*DIF*T(1,0) ) / ( 12*R**2 )

# SPACE DERIVATIVES
# PURE TIME DERIVATIVE
# IT
1
# NUMER
- ( 6*D*T*R**2*T(0,2) + D*R**2*R**2*D*F*T(0,0) - 3*D*R**2*R*D*F*T(3,0) - 3*D*R**2*D*F*T(2,0) - 12*R**2*D*F*T(2,0) - 24*R*D*F*
  T(1,0) )
# DENOM
12*R**2
# DERIV(0,1)
- ( 6*D*T*R**2*T(0,3) + D*R**2*R**2*D*F*T(0,1) - 3*D*R**2*R*D*F*T(3,1) - 3*D*R**2*D*F*T(2,1) - 12*R**2*D*F*T(2,1) - 24*R*D*F*
  T(1,1) ) / ( 12*R**2 )
# SPACE DERIVATIVES
# DERIV
{ = (3*4); A*D*T*R**2*T(0,2) + D*R**2*R**2*D*F*T(4,0) - 3*D*R**2*R*D*F*T(3,0) - 3*D*R**2*D*F*T(2,0) - 12*R**2*D*F*T(2,0) - 24*R*D*F*
  T(1,0) ) / ( 12*R**2 ),
- ( 6*D*T*R**2*T(0,3) + D*R**2*R**2*D*F*T(4,1) - 3*D*R**2*R*D*F*T(3,1) - 3*D*R**2*D*F*T(2,1) - 12*R**2*D*F*T(2,1) - 24*R*D*F*
  T(1,1) ) / ( 12*R**2 ),
  T(0,3),
  T(0,4),
  T(0,5),
  T(0,6),
  T(0,7),
  n,
  T(1,1),
  T(1,2),
  T(1,3),
  T(1,4),
  T(1,5),
  T(1,6),
  T(1,7),
  p,
  T(2,1),
  T(2,2),
  T(2,3),
  T(2,4),
  T(2,5),
  T(2,6),
  T(2,7).}
# T余R
1
# D余RMON
T(0,1)
# NUMER
  \cdot DIF = \{ 6 \cdot DT \cdot R^2 \cdot T(2,1) + 12 \cdot DT \cdot R \cdot T(1,1) 
  + DR \cdot R^2 \cdot T(1,1) - 6 \cdot DR \cdot R \cdot T(0,0) 
  + 3 \cdot DR \cdot R^2 \cdot T(0,0) = 12 \cdot R^2 \cdot T(2,0) = 
  24 \cdot DR \cdot T(1,0) \}
# DENOM
12 \cdot R^2
# DELTA
1.84115975
# FTIME
1.9501935425F1
# DFLTA
2.9493164999E-2
# ETIME
1.9531399607E1

*** NORMAL RETURN FROM MAIN PROCEDURE ***

*** RUN STATISTICS ***

19.799 SECONDS ELAPSED
131076 WORDS IN WORKSPACE
14 DIGITS IN SHORT INTEGERS
28 DIGITS IN LONG INTEGERS
0 GARBAGE COLLECTIONS
125165 WORDS OF WORKSPACE NEVER USED

$EJ$
# INITIALIZATION
# ROWN
On this page and the next, the reader should be aware that the columns, beginning with T(1,4), are to be read as one continuous run.

<table>
<thead>
<tr>
<th>T(0,0)</th>
<th>T(0,1)</th>
<th>T(0,2)</th>
<th>T(0,3)</th>
<th>T(0,4)</th>
<th>T(0,5)</th>
<th>T(0,6)</th>
<th>T(0,7)</th>
<th>T(0,8)</th>
</tr>
</thead>
<tbody>
<tr>
<td>T(1,0)</td>
<td>T(1,1)</td>
<td>T(1,2)</td>
<td>T(1,3)</td>
<td>T(1,4)</td>
<td>T(1,5)</td>
<td>T(1,6)</td>
<td>T(1,7)</td>
<td>T(1,8)</td>
</tr>
<tr>
<td>T(2,0)</td>
<td>T(2,1)</td>
<td>T(2,2)</td>
<td>T(2,3)</td>
<td>T(2,4)</td>
<td>T(2,5)</td>
<td>T(2,6)</td>
<td>T(2,7)</td>
<td>T(2,8)</td>
</tr>
<tr>
<td>T(3,0)</td>
<td>T(3,1)</td>
<td>T(3,2)</td>
<td>T(3,3)</td>
<td>T(3,4)</td>
<td>T(3,5)</td>
<td>T(3,6)</td>
<td>T(3,7)</td>
<td>T(3,8)</td>
</tr>
<tr>
<td>T(4,0)</td>
<td>T(4,1)</td>
<td>T(4,2)</td>
<td>T(4,3)</td>
<td>T(4,4)</td>
<td>T(4,5)</td>
<td>T(4,6)</td>
<td>T(4,7)</td>
<td>T(4,8)</td>
</tr>
<tr>
<td>T(5,0)</td>
<td>T(5,1)</td>
<td>T(5,2)</td>
<td>T(5,3)</td>
<td>T(5,4)</td>
<td>T(5,5)</td>
<td>T(5,6)</td>
<td>T(5,7)</td>
<td>T(5,8)</td>
</tr>
<tr>
<td>T(6,0)</td>
<td>T(6,1)</td>
<td>T(6,2)</td>
<td>T(6,3)</td>
<td>T(6,4)</td>
<td>T(6,5)</td>
<td>T(6,6)</td>
<td>T(6,7)</td>
<td>T(6,8)</td>
</tr>
<tr>
<td>T(7,0)</td>
<td>T(7,1)</td>
<td>T(7,2)</td>
<td>T(7,3)</td>
<td>T(7,4)</td>
<td>T(7,5)</td>
<td>T(7,6)</td>
<td>T(7,7)</td>
<td>T(7,8)</td>
</tr>
<tr>
<td>T(8,0)</td>
<td>T(8,1)</td>
<td>T(8,2)</td>
<td>T(8,3)</td>
<td>T(8,4)</td>
<td>T(8,5)</td>
<td>T(8,6)</td>
<td>T(8,7)</td>
<td>T(8,8)</td>
</tr>
<tr>
<td>T(9,0)</td>
<td>T(9,1)</td>
<td>T(9,2)</td>
<td>T(9,3)</td>
<td>T(9,4)</td>
<td>T(9,5)</td>
<td>T(9,6)</td>
<td>T(9,7)</td>
<td>T(9,8)</td>
</tr>
<tr>
<td>T(10,0)</td>
<td>T(10,1)</td>
<td>T(10,2)</td>
<td>T(10,3)</td>
<td>T(10,4)</td>
<td>T(10,5)</td>
<td>T(10,6)</td>
<td>T(10,7)</td>
<td>T(10,8)</td>
</tr>
<tr>
<td>T(11,0)</td>
<td>T(11,1)</td>
<td>T(11,2)</td>
<td>T(11,3)</td>
<td>T(11,4)</td>
<td>T(11,5)</td>
<td>T(11,6)</td>
<td>T(11,7)</td>
<td>T(11,8)</td>
</tr>
<tr>
<td>T(12,0)</td>
<td>T(12,1)</td>
<td>T(12,2)</td>
<td>T(12,3)</td>
<td>T(12,4)</td>
<td>T(12,5)</td>
<td>T(12,6)</td>
<td>T(12,7)</td>
<td>T(12,8)</td>
</tr>
<tr>
<td>T(13,0)</td>
<td>T(13,1)</td>
<td>T(13,2)</td>
<td>T(13,3)</td>
<td>T(13,4)</td>
<td>T(13,5)</td>
<td>T(13,6)</td>
<td>T(13,7)</td>
<td>T(13,8)</td>
</tr>
</tbody>
</table>

In this page and the next, the reader should be aware that the columns begin with T(1,4), are to be read as one continuous run.
| $\tau(7*1)$ | $\tau(7*2)$ | $\tau(7*3)$ | $\tau(7*4)$ | $\tau(7*5)$ | $\tau(7*6)$ | $\tau(7*7)$ | $\tau(0*1)$ | $\tau(0*2)$ | $\tau(0*3)$ | $\tau(0*4)$ | $\tau(0*5)$ | $\tau(0*6)$ | $\tau(0*7)$ | $\tau(1*1)$ | $\tau(1*2)$ | $\tau(1*3)$ | $\tau(1*4)$ | $\tau(1*5)$ | $\tau(1*6)$ | $\tau(1*7)$ | $\tau(2*1)$ | $\tau(2*2)$ |
|------------|------------|------------|------------|------------|------------|------------|------------|------------|------------|------------|------------|------------|------------|------------|------------|------------|------------|------------|------------|------------|------------|------------|------------|
MAXIMUM ORDER OF DERIVATIVE TO BE COMPUTED

ISR

IST

ISRM

CFLTA

ETIME

CFRIV(0,0)

DIF = 6*DTR*2*Q(2,N) + 2*DT*Q(1,N) + D*Q(2,N) = 4*D*Q(3,N) + 2*DEL*2*Q(2,N) - DEL*2*Q(1,N)

SPACE DERIVATIVES

IA

AIMER

DIF = ( 6*Q**2*N(2,N) + 2*DT*Q(1,N) + D*Q(2,N) = 4*D*Q(3,N) + 3*DEL*2*Q(2,N) - DEL*2*Q(1,N)

CFNOM

12*R**2

CFRIV(1,0)

DIF = ( 6*x**2*N(3,N) + 3*Q**2*2*N(2,N) - 12*DT*Q(1,N) - D*Q(2,N) = 12*DEL*2*Q(4,N) + 2*DEL**2*Q(3,N) + 4*D*DEL**2*Q(4,N) - 2*DEL**2*Q(3,N) - 12*DE**2*Q(2,N) - 2*DE**2*Q(1,N) ) / (12*DE**3)

IA

AIMER

DIF = ( 6*Q**2*N(3,N) + 12*Q**2*2*N(2,N) - 12*DT*Q(1,N) - D*Q(2,N) = 12*DEL*2*Q(4,N) + 2*DEL**2*Q(3,N) + 4*D*DEL**2*Q(4,N) - 2*DEL**2*Q(3,N) - 12*DE**2*Q(2,N) - 2*DE**2*Q(1,N) )

CFNOM

12*DE**3
# CENON
120555

# CENLA
2,1373492825

# ETIMF
4,1361742837E1

# CENLA
2,86131040019F-5

# ETIMF
4,119036637E1

*** NORMAL RETURN FROM DATA PROCESSING

*** RUN STATISTICS

41.441 SECONDS ELAPSED
131970 WORDS IN WORKSPACE
16 HITS IN SUMPT INTERESE
28 HITS IN LUAL INTERESE
6 GARBAGE COLLECTIONS
124228 WORDS OF WORKSPACE NEVER USED

COMPLETE TILREALIZATION
There are three simple modifications that the user can make to improve efficiency. The first change can be made only if all truncation errors containing time derivative are first or higher order in \( \dot{t} \) or \( \ddot{t} \). In that case, we can safely eliminate the highest order errors from the modified equation before differentiating it. This greatly reduces the amount of algebra by disposing of these terms at an early stage rather than waiting until the late stages of the calculation to discard them. To do this, insert the following three statements after line 105:

```plaintext
SECOND = DERIV(0,0)
DERIV(0,0) = TPSEVL(TPS(DERIV(0,0),DT,TORD-1), DT)
DERIV(0,0) = TPSEVL(TPS(DERIV(0,0),DR,RORD-1), DR)
```

Insert

```plaintext
DERIV(0,0) = SECOND
SECOND = 0
```

after line 129.

The second modification increases the running time of the code for each run, but reduces the number of runs and therefore the amount of human intervention. This modification is recommended for users who have no difficulty getting the necessary central processor time for a single run. It consists of looping through the code repeatedly until no more eliminations can be made with the current DERM@D. After line 10 insert the following:

```plaintext
INTEGER NPASS = 1
```

After line 30 insert the following:

```plaintext
AG: CONTINUE
```

After line 141 insert the following:

```plaintext
REWIND(25)
```

Replace lines 145 through 147 with the following:

```plaintext
IF(SDER.EQ.0 .AND. I + J.NE.0)DERIV(I,J) = 0
```

After line 139 insert the following:

```plaintext
DERIV(0,0) = DERIV(0,0)(SUB(0,0) = DERIV(0,0))
```

After line 156 insert the following:

```plaintext
IF (I + J .NE. 0) DERIV(I,J) = 0
```

The third set of changes should improve the core utilization of the program enough to avoid running out of workspace if the problem is only slightly too large, and it will reduce the number of passes through the elimination loop for certain problems. After line 133, insert the following:

```plaintext
IF(SDER .EQ. 0 .AND. I + J.NE.0)DERIV(I,J) = 0
```

After line 139 insert the following:

```plaintext
DERIV(0,0) = DERIV(0,0)(SUB(0,0) = DERIV(0,0))
```

After line 156 insert the following:

```plaintext
IF (I + J .NE. 0) DERIV(I,J) = 0
```

```plaintext
WRITE (25) SNUM, SDEN
```

The third set of changes should improve the core utilization of the program enough to avoid running out of workspace if the problem is only slightly too large, and it will reduce the number of passes through the elimination loop for certain problems. After line 133, insert the following:

```plaintext
IF(SDER .EQ. 0 .AND. I + J.NE.0)DERIV(I,J) = 0
```

After line 139 insert the following:

```plaintext
DERIV(0,0) = DERIV(0,0)(SUB(0,0) = DERIV(0,0))
```

After line 156 insert the following:

```plaintext
IF (I + J .NE. 0) DERIV(I,J) = 0
```

```plaintext
WRITE (25) SNUM, SDEN
```

The third set of changes should improve the core utilization of the program enough to avoid running out of workspace if the problem is only slightly too large, and it will reduce the number of passes through the elimination loop for certain problems. After line 133, insert the following:

```plaintext
IF(SDER .EQ. 0 .AND. I + J.NE.0)DERIV(I,J) = 0
```

After line 139 insert the following:

```plaintext
DERIV(0,0) = DERIV(0,0)(SUB(0,0) = DERIV(0,0))
```

After line 156 insert the following:

```plaintext
IF (I + J .NE. 0) DERIV(I,J) = 0
```

```plaintext
WRITE (25) SNUM, SDEN
```

The third set of changes should improve the core utilization of the program enough to avoid running out of workspace if the problem is only slightly too large, and it will reduce the number of passes through the elimination loop for certain problems. After line 133, insert the following:

```plaintext
IF(SDER .EQ. 0 .AND. I + J.NE.0)DERIV(I,J) = 0
```

After line 139 insert the following:

```plaintext
DERIV(0,0) = DERIV(0,0)(SUB(0,0) = DERIV(0,0))
```

After line 156 insert the following:

```plaintext
IF (I + J .NE. 0) DERIV(I,J) = 0
```
APPENDIX D

FLOW CHARTS FOR THE TIME DERIVATIVE ELIMINATION PROGRAM

START

Declarations, set array sizes N1 and N2, and initialize maximum exponents N, MM.

Lines 3-17

MAIN

Read RORD, TORD, DERMAD, NUMER, DENOM, SDER.

Line 21

If SDER = 0 (i.e., no secondary equation), SNUM = 0 and SDEN = 1.
Otherwise, read SNUM and SDEN.

Lines 23-28

Set up the substitution matrix SUB, set DERIV equal to SUB.

Lines 33-68

Search for the highest order r and t derivatives needed; ISRM(J) is the highest order r derivative needed for t derivative of order J.

Lines 75-97

END

ST

Substitute DERIV for SUB in the secondary equation, discarding terms of too high an order.

Lines 153-163

Is there a secondary equation?

Line 147

Yes

No

Substitute DERIV for SUB in the primary equation, then eliminate terms of too high an order in DR and DT. Compute an updated NUMER and DENOM.

Lines 131-140

Compute the needed derivatives using procedures RDER and TDER and store in DERIV;

\[ \frac{\partial (DERIV)}{\partial \theta^j} = DERIV(i,j). \]

Lines 105-122
The PROCEDURE RDER uses the same algorithm as TDER to differentiate A with respect to r.