LA-6885-MS

Informal Report

C.3

UC-32

Issued: July 1977

Automated Heuristic Stability Analysis for Nonlinear Equations

L. D. Cloutman
L. W. Fullerton

los alamos
scientific laboratory
of the University of California
LOS ALAMOS, NEW MEXICO 87545

An Affirmative Action/Equal Opportunity Employer

UNITED STATES
ENERGY RESEARCH AND DEVELOPMENT ADMINISTRATION
CONTRACT W-7405-ENG. 36

Printed in the United States of America. Available from
National Technical Information Service
U.S. Department of Commerce
5285 Port Royal Road
Springfield, VA 22161
Price: Printed Copy $4.50 Microfiche $3.00

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[1]) 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.[2] 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 orginally described by Harlow and Amsden.[3] Warming and Hyett[4] 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

$$\frac{\partial \rho}{\partial t} + \frac{\partial \rho u}{\partial x} = \frac{\partial}{\partial x}\left(\xi \frac{\partial \rho}{\partial x}\right), \qquad (1)$$

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

$$\frac{\rho_i^{n+1} - \rho_i^n}{\delta t} + \frac{\theta}{2\,\delta x}\left[(\rho_{i+1}^{n+1} + \rho_i^{n+1})\, u_{i+\frac{1}{2}}^{n+1} - (\rho_i^{n+1} + \rho_{i-1}^{n+1})\, u_{i-\frac{1}{2}}^{n+1}\right] + \frac{(1-\theta)}{2\,\delta x}\left[(\rho_{i+1}^{n} + \rho_i^{n})\, u_{i+\frac{1}{2}}^{n} - (\rho_i^{n} + \rho_{i-1}^{n})\, u_{i-\frac{1}{2}}^{n}\right]$$
$$= \frac{1}{\delta x^2}\left[\xi_{i+\frac{1}{2}}(\rho_{i+1}^n - \rho_i^n) - \xi_{i-\frac{1}{2}}(\rho_i^n - \rho_{i-1}^n)\right] , \qquad (2)$$

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 θ 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

$$y_{i+k}^{n+h} = \sum_{m=0}^{M} \frac{1}{m!}\left(h\delta t \frac{\partial}{\partial t} + k\delta x \frac{\partial}{\partial x}\right)^m y \ , \qquad (3)$$

where y is either ρ 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

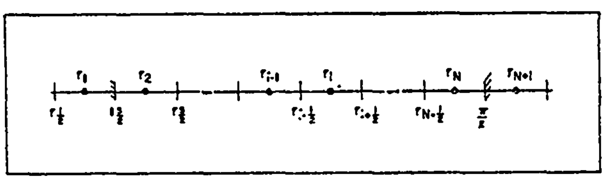


Fig. 1 Fragment of the computing mesh for the thermal diffusion example, Eq. (15). The T_i are defined on the cell centers r_i, and the $r_{i-\frac{1}{2}}$ are the cell edges. The same subscripting notation is used in the ICE difference equations, where ρ is defined at $x_i = r_i$ and u is defined on the cell edges.

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:

$$\frac{\partial \rho}{\partial t} + \frac{\partial \rho u}{\partial x} = \frac{\partial}{\partial x}\left(\xi \frac{\partial \rho}{\partial x}\right) - \frac{\delta t}{2}\left[\frac{\partial^2 \rho}{\partial t^2} + 2\theta\left(u \frac{\partial^2 \rho}{\partial t \partial x} + \frac{\partial u}{\partial t}\frac{\partial \rho}{\partial x} + \frac{\partial u}{\partial x}\frac{\partial \rho}{\partial t} + \rho \frac{\partial^2 u}{\partial t \partial x}\right)\right]$$
$$- \frac{\delta x^2}{24}\left[4u \frac{\partial^3 \rho}{\partial x^3} + 6\frac{\partial u}{\partial x}\frac{\partial^2 \rho}{\partial x^2} + 3\frac{\partial^2 u}{\partial x^2}\frac{\partial \rho}{\partial x} + \rho \frac{\partial^3 u}{\partial x^3} - 2\xi \frac{\partial^4 \rho}{\partial x^4} - 4\frac{\partial \xi}{\partial x}\frac{\partial^3 \rho}{\partial x^3} - 3\frac{\partial^2 \xi}{\partial x^2}\frac{\partial^2 \rho}{\partial x^2} - \frac{\partial^3 \xi}{\partial x^3}\frac{\partial \rho}{\partial x}\right] . \qquad (4)$$

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 derivatives. For example, $\partial^{(i+j)} u/\partial x^i \partial t^j$ 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 $TE(f_1,a_1, b_1,f_2,a_2,b_2,f_3,a_3,b_3,f_4,a_4,b_4)$ expands $(f_1)^{n+b_1}_{i+a_1}\ (f_2)^{n+b_2}_{i+a_2}\ (f_3)^{n+b_3}_{i+a_3}\ (f_4)^{n+b_4}_{i+a_4}$ to order ORD in both δr and δt. For example, $u^{n+1}_{i-\frac{1}{2}}$ 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 TE(RHO, 0, 1, U, 1/2, 0) for $\rho^{n+1}_i u^n_{i+\frac{1}{2}}$, not TE(RHO, 0, 1)* 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*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 δr and δt, 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.

We want the truncation errors to $\mathcal{O}(\delta t)$ and $\mathcal{O}(\delta r^2)$, so RORD = 2 and TORD = 1. Since the expansions are divided by δt and δr^2, they must be carried out to at least order 2 and 4 in δt and δr respectively. Therefore, ORD must be at least 4.

This example is a trivial problem -- only 14 seconds of central processor time and 37 000 words of workspace were required on a CDC 7600. Although 131 000 words of workspace are available in our version of ALTRAN, memory space, not running time, still limits the size of the largest problem that can be run. Very large problems often can be run piecemeal, however.

Appendix A consists of a complete listing of the expansion code, plus a sample problem. Appendix B contains a detailed flow chart of the ALTRAN coding, definition of all variables, and a description of the purpose and operation of every procedure.

For some purpose it is necessary to eliminate all time derivatives from the modified equation. In our example, we need $\partial\rho/\partial t$ and $\partial u/\partial t$ and their derivatives with respect to both r and t. Therefore, the modified equation is punched out in the form

$$\text{DERMOD} = \text{RHO}(0,1) = \frac{\partial\rho}{\partial t} = -\frac{\partial u}{\partial x} - \rho\frac{\partial u}{\partial x} + \xi\frac{\partial^2\rho}{\partial x^2} + \frac{\partial\xi}{\partial r}\frac{\partial\rho}{\partial r} - \text{TER}. \tag{5}$$

The time derivative elimination code then differentiates the right-hand side and eliminates the time derivative from the truncation error terms TER. It is necessary to use the modified equation for the momentum equation to eliminate the time derivatives of u. We will return this example in the next section.

A simpler example will suffice to illustrate the complexities of automating the general procedure for eliminating time derivatives. The modified equation for the difference approximation,

$$\frac{T^{n+1}_i - T^n_i}{\delta t} = \frac{K}{\delta x^2}\left(T^n_{i+1} - 2T^n_i + T^n_{i-1}\right) \tag{6}$$

to

$$\frac{\partial T}{\partial t} = K\frac{\partial^2 T}{\partial x^2}, \tag{7}$$

expanded about time n and space point i is

$$\frac{\partial T}{\partial t} = K \frac{\partial^2 T}{\partial x^2} - \frac{\delta t}{2} \frac{\partial^2 T}{\partial t^2} + \frac{\delta x^2 K}{6} \frac{\partial^4 T}{\partial x^4} + O(\delta t^2, \delta x^4) . \tag{8}$$

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

$$\frac{\partial^2 T}{\partial t^2} = K \frac{\partial^3 T}{\partial x^2 \partial t} - \frac{\delta t}{2} \frac{\partial^3 T}{\partial t^3} + \frac{\delta x^2 K}{6} \frac{\partial^5 T}{\partial x^4 \partial t} . \tag{9}$$

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

$$\frac{\partial T}{\partial t} = K \frac{\partial^2 T}{\partial x^2} - \frac{\delta t K}{2} \frac{\partial^3 T}{\partial x^2 \partial t} + \frac{\delta x^2 K}{6} \frac{\partial^4 T}{\partial x^4} . \tag{10}$$

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

$$\frac{\partial^3 T}{\partial x^2 \partial t} = K \frac{\partial^4 T}{\partial x^4} - \frac{\delta t}{2} \frac{\partial^4 T}{\partial x^2 \partial t^2} + \frac{\delta x^2 K}{6} \frac{\partial^6 T}{\partial x^6} , \tag{11}$$

which we substitute into Eq. (10):

$$\frac{\partial T}{\partial t} = K \frac{\partial^2 T}{\partial x^2} + \frac{K \delta x^2}{2} \left(\frac{1}{3} - \frac{K \delta t}{\delta x^2} \right) \frac{\partial^4 T}{\partial x^4} . \tag{12}$$

It is obvious from this trivial example that the elimination of time derivatives from the truncation error terms of the modifed 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 useable 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

$$\frac{\partial T}{\partial t} = \phi \frac{1}{r^2} \frac{\partial}{\partial r}\left(r^2 \frac{\partial T}{\partial r}\right) \text{for } 0 \leqslant t \leqslant \infty,\ 0 \leqslant r \leqslant \pi, \tag{13}$$

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

$$T(\pi,t) = 0 ,$$

and

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

where ϕ is a constant. The analytic solution is

$$T(r,t) = \exp(-\phi t) \sin(r)/r . \tag{14}$$

Now consider the explicit FDE

$$\frac{T_i^{n+1} - T_i^n}{\delta t} = \frac{\phi}{V_i}\left[\frac{r^2_{i+\frac{1}{2}}(T^n_{i+1} - T^n_i)}{r_{i+1} - r_i} - \frac{r^2_{i-\frac{1}{2}}(T^n_i - T^n_{i-1})}{r_i - r_{i-1}}\right] . \tag{15}$$

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

$$V_i = (r^3_{i+\frac{1}{2}} - r^3_{i-\frac{1}{2}})/3 \tag{16a}$$

and

$$V_i = r_i^2\,(r_{i+\frac{1}{2}} - r_{i-\frac{1}{2}}) . \tag{16b}$$

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 $4\pi \int_0^{\pi/2} T\, r^2 dr$, 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

$$\frac{\partial T}{\partial t} = \phi \frac{1}{r^2} \frac{\partial}{\partial r}\left(r^2 \frac{\partial T}{\partial r}\right) - \left[\frac{\phi^2 \delta t}{2} - \frac{\phi\, \delta r^2}{12}\right]\left[\frac{\partial^4 T}{\partial r^4} + \frac{4}{r}\frac{\partial^3 T}{\partial r^3}\right] + \frac{\phi \delta r^2}{6\, r^2}\left[\frac{\partial^2 T}{\partial r^2} - \frac{1}{r}\frac{\partial T}{\partial r}\right] + O(\delta t^2, \delta r^4) \tag{17a}$$

and

$$\frac{\partial T}{\partial t} = \phi \frac{1}{r^2} \frac{\partial}{\partial r}\left(r^2 \frac{\partial T}{\partial r}\right) - \left[\frac{\phi^2 \delta t}{2} - \frac{\phi\, \delta r^2}{12}\right]\left[\frac{\partial^4 T}{\partial r^4} + \frac{4}{r}\frac{\partial^3 T}{\partial r^3}\right] + \frac{\phi\, \delta r^2}{4\, r^2}\frac{\partial^2 T}{\partial r^2} + O(\delta t^2, \delta r^4) \tag{17b}$$

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 $\frac{\partial T}{\partial r} = 0$ at the origin and because some error terms partially cancel each other. We expand T in Taylor series about $r = 0$ for some η, $0 < \eta < r_{5/2}$, and a time τ, $t_n < \tau < t_{n+1}$:

$$T(\eta,\tau) = \sum_{i=0}^{\infty} \frac{\partial^{(i)}T(0,\tau)}{\partial r^{(i)}} \frac{\eta^i}{i!} \quad . \tag{18}$$

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

$$\frac{\phi\delta r^2}{12}\left[\frac{\partial^4 T}{\partial r^4} + \frac{4}{r}\frac{\partial^3 T}{\partial r^3} + \frac{2}{r^2}\frac{\partial^2 T}{\partial r^2} - \frac{2}{r^3}\frac{\partial T}{\partial r}\right]_{r=\eta,\ t=\tau}$$

$$= \frac{\phi\delta r^2}{12}\left[-\frac{2}{\eta^3}\frac{\partial T(0,\tau)}{\partial r} + \frac{5}{\eta}\frac{\partial^3 T(0,\tau)}{\partial r^3} + O(\eta^0)\right] \tag{19a}$$

and

$$\frac{\phi\delta r^2}{12}\left[\frac{\partial^4 T}{\partial r^4} + \frac{4}{r}\frac{\partial^3 T}{\partial r^3} + \frac{3}{r^2}\frac{\partial^2 T}{\partial r^2}\right]_{r=\eta}$$

$$= \frac{\phi\delta r^2}{12}\left[\frac{3}{2\eta^2}\frac{\partial^2 T(0,\tau)}{\partial r^2} + \frac{7}{\eta}\frac{\partial^3 T(0,\tau)}{\partial r^3} + O(\eta^0)\right] \tag{19b}$$

Because $\frac{\partial T(0,\tau)}{\partial r} = 0$ 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

$$T_1^{n+1} = T_2^{n+1} \tag{20}$$

and either

$$T_{N+1} = -T_N + 2T_b \tag{21a}$$

or

$$T_{N+1} = -2T_N + \frac{1}{3}T_{N-1} + \frac{8}{3}T_b, \tag{21b}$$

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

$$\phi\left\{\frac{3}{4}\frac{1}{r^2}\frac{\partial}{\partial r}\left(r^2\frac{\partial T}{\partial r}\right) - \frac{\delta r}{8}\left[\frac{\partial^3 T}{\partial r^3} + \frac{2}{r}\frac{\partial^2 T}{\partial r^2}\right] + O(\delta r^2)\right\} \tag{22a}$$

and

$$\phi\left\{\frac{1}{r^2}\frac{\partial}{\partial r}\left(r^2\frac{\partial T}{\partial r}\right) - \frac{\delta r}{6}\frac{\partial^3 T}{\partial r^3} + O(\delta r^2)\right\} \quad . \tag{22b}$$

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

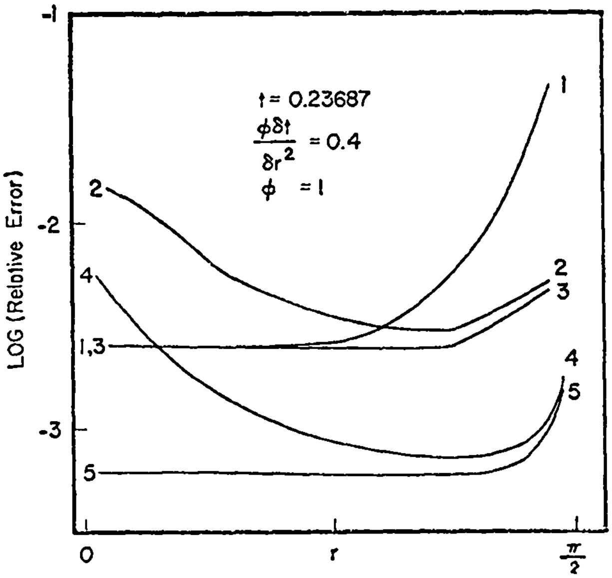


Fig. 2. Relative truncation errors vs. radius at a fixed time for five solutions to Eq. (15). Curves 1, 3 and 5 use volume elements (16a), and curves 2 and 4 use volume elements (16b). Curve 1 used boundary condition (21a). All others use boundary condition (21b). Curves 1, 2, and 3 were computed using 10 cells, and curves 4 and 5 were computed with 20 cells.

version of the ICE method[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

$$\zeta \frac{\partial^2 \rho}{\partial x^2} = \left[(2\theta-1) \frac{\delta t}{2} (u^2 + c^2) - \frac{\delta x^2}{4} \frac{\partial u}{\partial x} \right] \frac{\partial^2 \rho}{\partial x^2} , \tag{23}$$

where ζ is the diffusion coefficient of the truncation errors and c is the local sound speed. We have neglected the $\partial^2\xi/\partial x^2$ term in Eq. (4); as we shall see, it is a higher order term in the TEC algorithm. If $\xi = 0$ in Eq. (1), the FDE is unstable whenever $\zeta < 0$. In the original version of ICE, a constant global artificial mass diffusion coefficient $\xi \geqslant 0$ is used to stabilize the algorithm. It is necessary to choose ξ large enough that $\xi + \zeta \geqslant 0$ 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

$$\xi\, \delta t < \frac{1}{2}\, \delta x^2 . \tag{24}$$

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 ζ. Expansion yields a result of the form

$$\frac{\partial \rho}{\partial t} + \frac{\partial(\rho u)}{\partial x} = \frac{\partial}{\partial x}\left(\xi \frac{\partial \rho}{\partial x}\right) + \zeta \frac{\partial^2 \rho}{\partial x^2} = \frac{\partial}{\partial x}\left(\xi \frac{\partial \rho}{\partial x}\right) + \frac{\partial}{\partial x}\left(\zeta \frac{\partial \rho}{\partial x}\right) - \frac{\partial \zeta}{\partial x}\frac{\partial \rho}{\partial x} . \tag{25}$$

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 in the right-hand side of Eq. (25). In some cases, usually in the momemtum equation, $\frac{\partial \zeta}{\partial x}$ will contribute additional diffusional errors that should be included in TEC as discussed by Rivard et al. In our continuity equation, however, $\frac{\partial \zeta}{\partial x}$ does not produce additional diffusional errors, and the $\frac{\partial \zeta}{\partial x}\frac{\partial \rho}{\partial x}$ truncation error is neglected. In order to obtain an improved FDE, Eq. (23) is differenced to yield

$$\zeta_{i-\frac{1}{2}} = (2\theta-1)\,\frac{\delta t}{2}\left[(u^n_{i-\frac{1}{2}})^2 + \frac{1}{2}\,(c^2 + c^2_{i-1})\right]$$

$$- \frac{\delta x}{8}\,(u^n_{i+\frac{1}{2}} - u^n_{i-3/2}) \quad . \qquad (26)$$

Next we choose

$$\xi^n_{i-\frac{1}{2}} = \begin{cases} -(1+\beta)\ \zeta_{i-\frac{1}{2}} \text{ if } \zeta_{i-\frac{1}{2}} < 0 \\ \\ -(1-\beta)\ \zeta_{i-\frac{1}{2}} \text{ if } \zeta_{i-\frac{1}{2}} \geq 0 \end{cases}$$

which is then incorporated in the finite difference form of Eq. (1). The constant β, $0 \leq \beta \leq 1$, is a free parameter that determines the degree to which the diffusional truncation errors are cancelled. If β is too small, the FDE's will have so little diffusion that dispersively generated ripples destroy accuracy. If, on the other hand, β is too large, unnecessary artifical diffusion reduces the accuracy of the solution. The optimum value of β 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 boundaries; the solutions have been displaced vertically for clarity. The bottom curve is the analytic solution. The top solution is an artificial viscosity solution with nearly the minimum diffusion needed for stability. The right density jump is a shock wave, and the left discontinuity is a contact surface. Both discontinuities move to the right. The shock is smooth, but the contact surface has dispersively driven ripples behind it. TEC with the same viscosity μ as the conventional method, was used in the second solution from the top. The shock is unchanged, but the ripples behind the contact surface are stronly damped. The third numerical solution is also TEC run, but the viscosity is reduced by a factor of ten. The shock is significant sharper, but it shows a little overshoot. The first peak behind the contact surface is as high as in the artificial viscosity run, but the damping behind the contact is much stronger. The artificial viscosity scheme is unstable with this little viscosity.

The TEC algorithm readily generalizes to multidimensional flows. As an example, consider a Mach 0.1 wind blowing over a pair of walls as shown in Fig. 4. Shown there are the velocity vectors and isotherms of a TEC solution obtained from the two-dimensional RICE program.[7] The comparison solution with normal artificial viscosity stabilization was obtained with 100 times as much viscosity, because

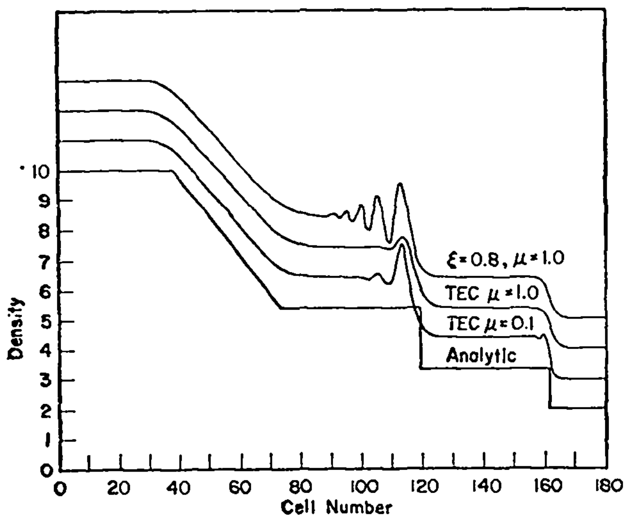


Fig. 3. One dimensional shock tube calculations. All four solutions coincide at the left and right ends, but the three numerical solutions have been displaced vertically for clarity.

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 algebraic 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.

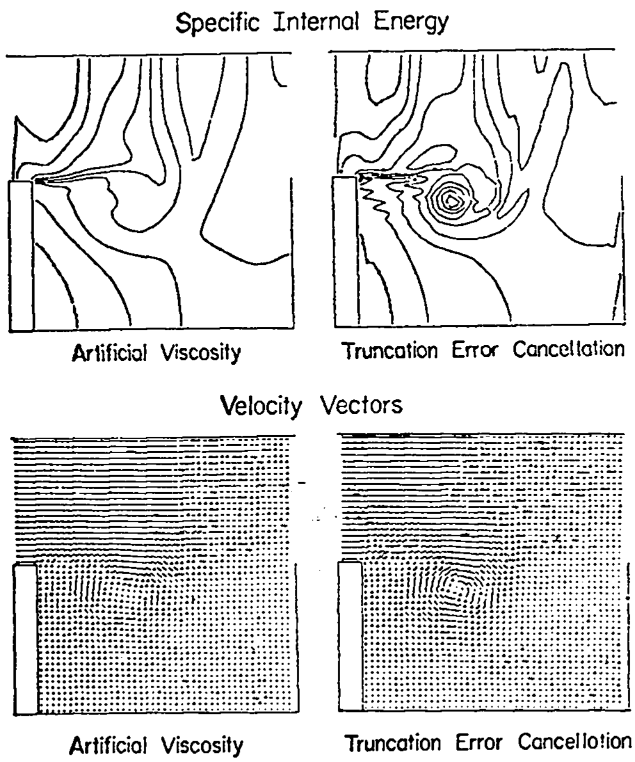


Fig. 4. A two-dimensional flow with and without TEC. The TEC run has 1% as much viscosity as the artificial viscosity run. A Mach 0.1 wind enters the mesh across the upper half of the left boundary, and it leaves across the upper half of the right boundary.

REFERENCES

1. C. W. Hirt, "Heuristic Stability Theory for Finite Difference Equations," J. Comput. Phys. 2, 339 (1968).

2. W. C. Rivard, O. A. Farmer, T. D. Butler, and P. J. O'Rourke, "A Method for Increased Accuracy in Eulerian Fluid Dynamics Calculations," Los Alamos Scientific Laboratory report LA-5426-MS (October 1973).

3. F. H. Harlow and A. A. Amsden, "A Numerical Fluid Dynamics Calculation Method for All Flow Speeds," J. Comput. Phys. 8, 197 (1971).

4. R. F. Warming and B. J. Hyett, "The Modified Equation Approach to the Stability and Accuracy Analysis of Finite-Difference Methods," J. Comput. Phys. 14, 159 (1974).

5. A. D. Hall, Comm. Assoc. Computing Mach. 14, 517 (1971).

6. W. S. Brown, "ALTRAN Users Manual, 3rd ed.," Bell Laboratories, Murray Hill, N. J. (1973).

7. W. C. Rivard, O. A. Farmer, and T. D. Butler, "RICE: A Computer Program for Multicomponent Chemically Reactive Flows at All Speeds," Los Alamos Scientific Laboratory report LA-5812 (March 1975).

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 δr (denoted by DR) and δt (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 $\rho_{xtt} = A\rho + B\rho_{xx}$. DE is the differential equation, where we have represented ξ 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 θ by G1. 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 δr^{-1} and/or δt^{-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:

$$\mathrm{DE} = \frac{\partial T}{\partial t} - D\,\frac{\partial^2 T}{\partial X^2}$$

$$= \mathrm{T(0,1)\text{-}DIF{*}T(2,0)} \tag{A1}$$

and

$$\mathrm{FDE} = [T_i^{n+1} - (T_{i+1}^n + T_{i-1}^n)/2]/\delta t$$

$$- D[T_{i+1}^n - 2T_i^{\ n} + T_{i-1}^{\ \ n}]/\delta X^2 \tag{A2}$$

= (TE(T,0,1) - (TE(T,1,0) + TE(T,-1,0))/2) / DT

-DIF*(TE(T,1,0)-2*T(0,0) + TE(T,-1,0))/DR**2

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 δr 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=31, N=ORD
    INTEGER RORD, TORD
    LONG ALGEBRAIC (DR:M, DT:M, R:M, P(0:N,0:N):XP(N), T(0:N,0:N):XP(N),
      PHI:M, THETA:M, RP:M, RM:M, G1:M, G2:M, DIF:M, LAM:M, F1:M, F2:M,
      TIM:M, U(0:N,0:N):XP(N), RHO(0:N,0:N):XP(N))  ARRAY DETPS, FDETPS,
      TER, CONTPS
    EXTERNAL ALGEBRAIC DDR=DR, DDT=DT, LAM2=LAM
    LONG ALGEBRAIC FDE, DE, DERMOD, NUMER, DENOM
    LONG ALGEBRAIC ARRAY MODEQ
    ALTRAN INTEGER TPSORD
    ALTRAN SHORT INTEGER ARRAY XP
    ALTRAN ALGEBRAIC TE, TPSEVL
    ALTRAN ALGEBRAIC ARRAY TPS, TPSMUL, TPSSBS, ARRSBS, TETPS, TPSCHOP

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

    RORD = 2   ;   TORD = 1
    DERMOD = RHO(0,1)
    DE = RHO(0,1) + RHO(1,0)*U(0,0) + RHO(0,0)*U(1,0) - T(0,0)*RHO(2,0) -
      T(1,0)*RHO(1,0)
    FDE = (TE(RHO,0,1) - RHO(0,0)) / DT + G1*((TE(RHO,1,1) + TE(RHO,0,1)) *
      TE(U,1/2,1) - (TE(RHO,0,1) + TE(RHO,-1,1)) * TE(U,-1/2,1)) / (2*DR) +
      (1-G1) * ((TE(RHO,1,0) + RHO(0,0)) * TE(U,1/2,0) - (TE(RHO,-1,0) +
      RHO(0,0)) * TE(U,-1/2,0)) / (2*DR) - (TE(T,1/2,0) * (TE(RHO,1,0) -
      RHO(0,0)) - TE(T,-1/2,0) * (RHO(0,0) - TE(RHO,-1,0))) / DR**2

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

# - - - - - - -                                              - - - - - - - -

    FDE = FDE (RP, RM = R+DR/2, R-DR/2)
    DE  =  DE (RP, RM = R+DR/2, R-DR/2)
    FDE = FDE (DR, DT = LAM*DR, LAM*DT)

#  CHECK FOR TRUNCATION ERRORS OF NEGATIVE ORDER
    NUMER = ANUM (FDE, DENOM)
    IF (DEG(DENOM,LAM).GT.0) WRITE FDE,"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 (TPSEVL(FDETPS,LAM), LAM, IMAX(RORD,TORD))
        FDETPS = TPSCHOP (FDETPS, RORD, TORD)
        DETPS = TPSCHOP (DETPS, RORD, TORD)
        WRITE FDETPS

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

   #  COMPUTE AND PUNCH MODIFIED EQUATION
        WRITE "CONSTRUCT THE MODIFIED EQUATION"
        MODEQ = TPS (DE, DERMOD, DEG (DE, DERMOD))
        IF (MODEQ(1).EQ.0) FRETURN , "INCORRECT DERMOD"
        NUMER = ANUM ((MODEQ(1)*DERMOD-DE-TPSEVL(TER,1))/MODEQ(1), DENOM)

        WRITE RORD, TORD, NUMER, DENOM
        WRITE (25) RORD, TORD, DERMOD, NUMER, DENOM

        END
```

NAME/EXTNAME	USE	TYPE	STRUC	PREC	CLASS	SCOPE	DB	LAY	ADDR
CONTPS	VAR	ALG	A	L				L*001	
DETPS	VAR	ALG	A	L				L*001	
DIF	IND	ALG						L*001	
DR	IND	ALG						L*001	
DT	IND	ALG						L*001	
FDETPS	VAR	ALG	A	L				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				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			
DDR	VAR	ALG			S	X			
DDT	VAR	ALG			S	X			
DEG/S9DEG	PROC	INT			S	X			
DENOM	VAR	ALG		L					
DERMOD	VAR	ALG		L					
DE	VAR	ALG		L					
FDE	VAR	ALG		L					

```
IMAX/S9IMAX           PROC INT         L     S     X
LAM2                  VAR  ALG               S     X
MAIN                  PROC             L     S     X
MODEQ                 VAR  ALG    A    L
M                     VAR  INT
NUMER                 VAR  ALG         L
N                     VAR  INT
ORD                   VAR  INT               S     X
RORD                  VAR  INT
TETPS                 PROC ALG    A    L     S     X
TE                    PROC ALG         L     S     X
TORD                  VAR  INT
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
TPSSBS                PROC ALG    A    L     S     X
TPS                   PROC ALG    A    L     S     X
XP                    PROC INT    A          S     X
L*001                 LAY
CONSTRUCT THE MODIFI  CONS CHAR              S
END PHASE ONE         CONS CHAR              S
INCORRECT DERMOD      CONS CHAR              S
MAY ABORT DUE TO NEG  CONS CHAR              S
TER WITH ALL TIME DE  CONS CHAR              S
0                     CONS INT               S
1                     CONS INT               S
25                    CONS INT               S
2                     CONS INT               S
31                    CONS INT               S
6                     CONS INT               S
```

ALTRAN VERSION 1 LEVEL 9

```
      PROCEDURE TE (A,AX,AT, B,BX,BT, C,CX,CT, D,DX,DT)

  # 2-D TAYLOR SERIES EXPANSION OF THE PRODUCT A*B*C*D

      VALUE A,AX,AT, B,BX,BT, C,CX,CT, D,DX,DT
      LONG ALGEBRAIC ARRAY A,B,C,D
      LONG ALGEBRAIC AX,AT, BX,BT, CX,CT, DX,DT
      ALTRAN ALGEBRAIC ARRAY TETPS
      ALTRAN ALGEBRAIC TPSEVL

      RETURN ( TPSEVL(TETPS(A,AX,AT, B,BX,BT, C,CX,CT, D,DX,DT), 1) )

      END
```

NAME/EXTNAME	USE	TYPE	STRUC	PREC	CLASS	SCOPE	DB	LAY	ADDR
AT	VAR	ALG		L		V			
AX	VAR	ALG		L		V			
A	VAR	ALG	A	L		V			
BT	VAR	ALG		L		V			
BX	VAR	ALG		L		V			
B	VAR	ALG	A	L		V			
CT	VAR	ALG		L		V			
CX	VAR	ALG		L		V			
C	VAR	ALG	A	L		V			
DT	VAR	ALG		L		V			
DX	VAR	ALG		L		V			
D	VAR	ALG	A	L		V			
TETPS	PROC	ALG	A	L	S	X			
TE	PROC			L	S	X			
TPSEVL	PROC	ALG		L	S	X			
1	CONS	INT			S				

ALTRAN VERSION 1 LEVEL 9

```
      PROCEDURE TETPS (A,AX,AT, B,BX,BT, C,CX,CT, D,DX,DT)

  # 2-D TPS TAYLOR SERIES OF THE PRODUCT A*B*C*D

      VALUE A,AX,AT, B,BX,BT, C,CX,CT, D,DX,DT
      LONG ALGEBRAIC ARRAY A,B,C,D
      LONG ALGEBRAIC AX,AT, BX,BT, CX,CT, DX,DT
      ALTRAN ALGEBRAIC ARRAY TETPS,TAYLOR,TPSMUL

      IF (NULL(B)) RETURN ( TAYLOR(A,AX,AT) )

      RETURN (TPSMUL(TAYLOR(A,AX,AT), TETPS(B,BX,BT, C,CX,CT, D,DX,DT)) )

      END
```

NAME/EXTNAME	USE	TYPE	STRUC	PREC	CLASS	SCOPE	DB	LAY	ADDR
AT	VAR	ALG		L		V			
AX	VAR	ALG		L		V			
A	VAR	ALG	A	L		V			
BT	VAR	ALG		L		V			
BX	VAR	ALG		L		V			
B	VAR	ALG	A	L		V			
CT	VAR	ALG		L		V			
CX	VAR	ALG		L		V			
C	VAR	ALG	A	L		V			
DT	VAR	ALG		L		V			
DX	VAR	ALG		L		V			

```
D                      VAR  ALG    A    L              V
NULL/S9NULL            PROC LOG                  S     X
TAYLOR                 PROC ALG    A    L        S     X
TFTPS                  PROC ALG    A    L        S     X
TPSMUL                 PROC ALG    A    L        S     X
```

ALTRAN VERSION 1 LEVEL 9

```
        PROCEDURE TAYLOR (F, A, B)

    # 2-D TPS TAYLOR SERIES OF THE VARIABLE F

        VALUE F, A, B
        EXTERNAL ALGEBRAIC DDR, DDT
        EXTERNAL INTEGER ORD
        INTEGER I, J
        LONG ALGEBRAIC A, B
        LONG ALGEBRAIC ARRAY F
        LONG ALGEBRAIC ARRAY(Ø:ORD) TAY=(F(Ø,Ø), ORD$Ø)
        INTEGER ARRAY(Ø:1Ø) FACT=(1,1,2,6,24,12Ø,72Ø,5Ø4Ø,4Ø32Ø,36288Ø,36288ØØ)
        INTEGER ARRAY(Ø:1Ø) COF=(1,1Ø$Ø)

        IF (A.EQ.Ø) DO     #  DIFF W.R.T. T
          IF (B.EQ.Ø) RETURN (TAY)
          DO I=1,ORD
            TAY(I) = (B*DDT)**I*F(Ø,I)/FACT(I)
          DOEND
          RETURN (TAY)
        DOEND

        IF (B.EQ.Ø) DO     #  DIFF W.R.T. R
          DO I=1,ORD
            TAY(I) = (A*DDR)**I*F(I,Ø)/FACT(I)
          DOEND
          RETURN (TAY)
        DOEND

        DO I=1,ORD      # DIFF W.R.T. R AND T
          DO J=I,1,-1   ;  COF(J)=COF(J)+COF(J-1)   ;  DOEND
          DO J=Ø,I
            TAY(I) = TAY(I) + COF(J)*(A*DDR)**J*(B*DDT)**(I-J)*F(J,I-J)
          DOEND
          TAY(I) = TAY(I)/FACT(I)
        DOEND
        RETURN (TAY)

        END
```

NAME/EXTNAME	USE	TYPE	STRUC	PREC	CLASS	SCOPE	DB	LAY	ADDR
TAY	VAR	ALG	A	L			D*001		
FACT	VAR	INT	A				D*002		
COF	VAR	INT	A				D*003		
A	VAR	ALG		L		V			
B	VAR	ALG		L		V			
DDR	VAR	ALG			S	X			
DDT	VAR	ALG			S	X			
F	VAR	ALG	A	L		V			
I	VAR	INT							
J	VAR	INT							
ORD	VAR	INT			S	X			
TAYLOR	PROC			L	S	X			
D*001	DB								
D*002	DB								
D*003	DB								
0	CONS	INT			S				
10	CONS	INT			S				
120	CONS	INT			S				
1	CONS	INT			S				
24	CONS	INT			S				
2	CONS	INT			S				
3628800	CONS	INT			S				
362880	CONS	INT			S				
40320	CONS	INT			S				
5040	CONS	INT			S				
6	CONS	INT			S				
720	CONS	INT			S				

ALTRAN VERSION 1 LEVEL 9

```
        PROCEDURE TPSCHOP (A, RORD, TORD)

   # CHOP THE 2-D TPS TO ORDER RORD IN DR AND TO ORDER TORD IN DT

        VALUE A, RORD, TORD
        EXTERNAL ALGEBRAIC DDR, DDT, LAM2
        LONG ALGEBRAIC ARRAY A
        INTEGER I, RORD, TORD, ORD=TPSORD(A)
        ALTRAN ALGEBRAIC ARRAY TPS
        ALTRAN ALGEBRAIC TPSEVL
        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)
        DOEND

        RETURN (A)

        END
```

NAME/EXTNAME	USE	TYPE	STRUC	PREC	CLASS	SCOPE	DB	LAY	ADDR
A	VAR	ALG	A	L		V			
DDR	VAR	ALG			S	X			
DDT	VAR	ALG			S	X			
I	VAR	INT							
LAM2	VAR	ALG			S	X			
ORD	VAR	INT							
RORD	VAR	INT				V			
TORD	VAR	INT				V			
TPSCHOP	PROC			L	S	X			
TPSEVL	PROC	ALG		L	S	X			
TPSORD	PROC	INT			S	X			
TPS	PROC	ALG	A	L	S	X			
0	CONS	INT			S				

ALTRAN VERSION 1 LEVEL 9

```
      PROCEDURE ARRSBS (A, LHS, RHS)

   # SUBSTITUTE THE LIST RHS FOR THE LIST LHS IN THE 1-D ARRAY A

      VALUE A, LHS, RHS
      LONG ALGEBRAIC ARRAY A, LHS, RHS
      INTEGER ARRAY DB=DBINFO(A)
      INTEGER I

      DO I=DB(1,0),DB(1,1)
        A(I) = A(I)(LHS=RHS)
      DOEND

      RETURN (A)
      END
```

NAME/EXTNAME	USE	TYPE	STRUC	PREC	CLASS	SCOPE	DB	LAY	ADDR
ARRSBS	PROC			L	S	X			
A	VAR	ALG	A	L		V			
DBINFO/S9DBIN	PROC	INT	A		S	X			
DB	VAR	INT	A						
I	VAR	INT							
LHS	VAR	ALG	A	L		V			
RHS	VAR	ALG	A	L		V			
0	CONS	INT			S				
1	CONS	INT			S				

ALTRAN VERSION 1 LEVEL 9

```
PROCEDURE XP (N)
VALUE N
INTEGER I, J, N
INTEGER ARRAY(0:N,0:N) EXP=1

DO I=0,N
  DO J=0,N-I
    EXP(I,J) = 7
  DOEND
DOEND

RETURN (EXP)

END
```

NAME/EXTNAME	USE	TYPE	STRUC	PREC	CLASS	SCOPE	DB	LAY	ADDR
EXP	VAR	INT	A				D*001		
I	VAR	INT							
J	VAR	INT							
N	VAR	INT				V			
XP	PROC			L	S	X			
D*001	DB								
0	CONS	INT			S				
1	CONS	INT			S				
7	CONS	INT			S				

```
# DERMOD

     RHO(0,1)
# DE

     - ( T(0,0)*RHO(2,0) + T(1,0)*RHO(1,0) - U(0,0)*RHO(1,0) - U(1,0)*RHO(0,0) - RHO(0,1) )
# FDE

     ( 6*DR**10*DT*G1*U(5,1)*RHO(6,0) + 3*DR**10*DT*G1*U(6,0)*RHO(5,1) - DR**10*T(6,0)*RHO(6,0) + 6*DR**10*U(5,0)*RHO(6,0) +
     3*DR**10*U(6,0)*RHO(5,0) + 80*DR**8*DT**3*G1*U(3,3)*RHO(6,0) + 180*DR**8*DT**3*G1*U(4,2)*RHO(5,1) + 90*DR**8*DT**3*G1*U(5,1)*
     RHO(4,2) + 10*DR**8*DT**3*G1*U(6,0)*RHO(3,3) + 240*DR**8*DT**2*G1*U(3,2)*RHO(6,0) + 360*DR**8*DT**2*G1*U(4,1)*RHO(5,1) +
     180*DR**8*DT**2*G1*U(4,2)*RHO(5,0) + 90*DR**8*DT**2*G1*U(5,0)*RHO(4,2) + 180*DR**8*DT**2*G1*U(5,1)*RHO(4,1) + 30*DR**8*DT**2*
     G1*U(6,0)*RHO(3,2) + 480*DR**8*DT*G1*U(3,1)*RHO(6,0) + 360*DR**8*DT*G1*U(4,0)*RHO(5,1) + 360*DR**8*DT*G1*U(4,1)*RHO(5,0) +
     180*DR**8*DT*G1*U(5,0)*RHO(4,1) + 180*DR**8*DT*G1*U(5,1)*RHO(4,0) + 60*DR**8*DT*G1*U(6,0)*RHO(3,1) - 120*DR**8*T(4,0)*
     RHO(6,0) - 72*DR**8*T(5,0)*RHO(5,0) - 30*DR**8*T(6,0)*RHO(4,0) + 480*DR**8*U(3,0)*RHO(6,0) + 360*DR**8*U(4,0)*RHO(5,0) +
     180*DR**8*U(5,0)*RHO(4,0) + 60*DR**8*U(6,0)*RHO(3,0) + 96*DR**6*DT**5*G1*U(1,5)*RHO(6,0) + 720*DR**6*DT**5*G1*U(2,4)*RHO(5,1) +
```

```
1200*DR**6*DT**5*G1*U(3,3)*RHD(4,2) + 600*DR**6*DT**5*G1*U(4,2)*RHD(3,3) + 90*DR**6*DT**5*G1*U(5,1)*RHD(2,4) + 3*DR**6*DT**5*
G1*U(6,0)*RHD(1,5) + 480*DR**6*DT**4*G1*U(1,4)*RHD(6,0) + 2880*DR**6*DT**4*G1*U(2,3)*RHD(5,1) + 720*DR**6*DT**4*G1*U(2,4)*
RHD(5,0) + 3600*DR**6*DT**4*G1*U(3,2)*RHD(4,2) + 2400*DR**6*DT**4*G1*U(3,3)*RHD(4,1) + 1200*DR**6*DT**4*G1*U(4,1)*RHC(3,3) +
1800*DR**6*DT**4*G1*U(4,2)*RHD(3,2) + 90*DR**6*DT**4*G1*U(5,0)*RHD(2,4) + 360*DR**6*DT**4*G1*U(5,1)*RHD(2,3) + 15*DR**6*DT**4*
G1*U(6,0)*RHD(1,4) + 1920*DR**6*DT**3*G1*U(1,3)*RHD(6,0) + 8640*DR**6*DT**3*G1*U(2,2)*RHD(5,1) + 2880*DR**6*DT**3*G1*U(2,3)*
RHD(5,0) + 7200*DR**6*DT**3*G1*U(3,1)*RHD(4,2) + 7200*DR**6*DT**3*G1*U(3,2)*RHD(4,1) + 2400*DR**6*DT**3*G1*U(3,3)*RHC(4,0) +
1200*DR**6*DT**3*G1*U(4,0)*RHD(3,3) + 3600*DR**6*DT**3*G1*U(4,1)*RHD(3,2) + 3600*DR**6*DT**3*G1*U(4,2)*RHD(3,1) +
360*DR**6*DT**3*G1*U(5,0)*RHD(2,3) + 1080*DR**6*DT**3*G1*U(5,1)*RHD(2,2) + 60*DR**6*DT**3*G1*U(6,0)*RHD(1,3) +
5760*DR**6*DT**2*G1*U(1,2)*RHD(6,0) + 17280*DR**6*DT**2*G1*U(2,1)*RHD(5,1) + 8640*DR**6*DT**2*G1*U(2,2)*RHD(5,0) +
7200*DR**6*DT**2*G1*U(3,0)*RHD(4,2) + 14400*DR**6*DT**2*G1*U(3,1)*RHD(4,1) + 7200*DR**6*DT**2*G1*U(3,2)*RHD(4,0) +
3600*DR**6*DT**2*G1*U(4,0)*RHD(3,2) + 7200*DR**6*DT**2*G1*U(4,1)*RHD(3,1) + 3600*DR**6*DT**2*G1*U(4,2)*RHD(3,0) +
1080*DR**6*DT**2*G1*U(5,0)*RHD(2,2) + 2160*DR**6*DT**2*G1*U(5,1)*RHD(2,1) + 180*DR**6*DT**2*G1*U(6,0)*RHD(1,2) +
11520*DR**6*DT*G1*U(1,1)*RHD(6,0) + 17280*DR**6*DT*G1*U(2,0)*RHD(5,1) + 17280*DR**6*DT*G1*U(2,1)*RHD(5,0) + 14400*DR**6*DT*G1*
U(3,0)*RHD(4,1) + 14400*DR**6*DT*G1*U(3,1)*RHD(4,0) + 7200*DR**6*DT*G1*U(4,0)*RHD(3,1) + 7200*DR**6*DT*G1*U(4,1)*RHD(3,0) +
2160*DR**6*DT*G1*U(5,0)*RHD(2,1) + 2160*DR**6*DT*G1*U(5,1)*RHD(2,0) + 360*DR**6*DT*G1*U(6,0)*RHD(1,1) - 5760*DR**6*T(2,0)*
RHD(6,0) - 5760*DR**6*T(3,0)*RHD(5,0) - 3600*DR**6*T(4,0)*RHD(4,0) - 1440*DR**6*T(5,0)*RHD(3,0) - 360*DR**6*T(6,0)*RHD(2,0) +
11520*DR**6*U(1,0)*RHD(6,0) + 17280*DR**6*U(2,0)*RHD(5,0) + 14400*DR**6*U(3,0)*RHD(4,0) + 7200*DR**6*U(4,0)*RHD(3,0) +
2160*DR**6*U(5,0)*RHD(2,0) + 360*DR**6*U(6,0)*RHD(1,0) + 192*DR**4*DT**7*G1*U(0,6)*RHD(5,1) + 1440*DR**4*DT**7*G1*U(1,5)*
RHD(4,2) + 2400*DR**4*DT**7*G1*U(2,4)*RHD(3,3) + 1200*DR**4*DT**7*G1*U(3,3)*RHD(2,4) + 180*DR**4*DT**7*G1*U(4,2)*RHD(1,5) +
12*DR**4*DT**7*G1*U(5,1)*RHD(0,6) + 1152*DR**4*DT**6*G1*U(0,5)*RHD(5,1) + 192*DR**4*DT**6*G1*U(0,6)*RHD(5,0) +
7200*DR**4*DT**6*G1*U(1,4)*RHD(4,2) + 2880*DR**4*DT**6*G1*U(1,5)*RHD(4,1) + 9600*DR**4*DT**6*G1*U(2,3)*RHD(3,3) +
7200*DR**4*DT**6*G1*U(2,4)*RHD(3,2) + 3600*DR**4*DT**6*G1*U(3,2)*RHD(2,4) + 4800*DR**4*DT**6*G1*U(3,3)*RHD(2,3) +
360*DR**4*DT**6*G1*U(4,1)*RHD(1,5) + 900*DR**4*DT**6*G1*U(4,2)*RHD(1,4) + 12*DR**4*DT**6*G1*U(5,0)*RHD(0,6) + 72*DR**4*DT**6*
G1*U(5,1)*RHD(0,5) + 5760*DR**4*DT**5*G1*U(0,4)*RHD(5,1) + 1152*DR**4*DT**5*G1*U(0,5)*RHD(5,0) + 28800*DR**4*DT**5*G1*U(1,3)*
RHD(4,2) + 14400*DR**4*DT**5*G1*U(1,4)*RHD(4,1) + 2880*DR**4*DT**5*G1*U(1,5)*RHD(4,0) + 28800*DR**4*DT**5*G1*U(2,2)*RHD(3,3) +
28800*DR**4*DT**5*G1*U(2,3)*RHD(3,2) + 14400*DR**4*DT**5*G1*U(2,4)*RHD(3,1) + 7200*DR**4*DT**5*G1*U(3,1)*RHD(2,4) +
14400*DR**4*DT**5*G1*U(3,2)*RHD(2,3) + 14400*DR**4*DT**5*G1*U(3,3)*RHD(2,2) + 360*DR**4*DT**5*G1*U(4,0)*RHD(1,5) +
1800*DR**4*DT**5*G1*U(4,1)*RHD(1,4) + 3600*DR**4*DT**5*G1*U(4,2)*RHD(1,3) + 72*DR**4*DT**5*G1*U(5,0)*RHD(0,5) +
360*DR**4*DT**5*G1*U(5,1)*RHD(0,4) + 23040*DR**4*DT**4*G1*U(0,3)*RHD(5,1) + 5760*DR**4*DT**4*G1*U(0,4)*RHD(5,0) +
86400*DR**4*DT**4*G1*U(1,2)*RHD(4,2) + 57600*DR**4*DT**4*G1*U(1,3)*RHD(4,1) + 14400*DR**4*DT**4*G1*U(1,4)*RHD(4,0) +
57600*DR**4*DT**4*G1*U(2,1)*RHD(3,3) + 86400*DR**4*DT**4*G1*U(2,2)*RHD(3,2) + 57600*DR**4*DT**4*G1*U(2,3)*RHD(3,1) +
14400*DR**4*DT**4*G1*U(2,4)*RHD(3,0) + 7200*DR**4*DT**4*G1*U(3,0)*RHD(2,4) + 28800*DR**4*DT**4*G1*U(3,1)*RHD(2,3) +
43200*DR**4*DT**4*G1*U(3,2)*RHD(2,2) + 28800*DR**4*DT**4*G1*U(3,3)*RHD(2,1) + 1800*DR**4*DT**4*G1*U(4,0)*RHD(1,4) +
7200*DR**4*DT**4*G1*U(4,1)*RHD(1,3) + 10800*DR**4*DT**4*G1*U(4,2)*RHD(1,2) + 360*DR**4*DT**4*G1*U(5,0)*RHD(0,4) +
1440*DR**4*DT**4*G1*U(5,1)*RHD(0,3) + 69120*DR**4*DT**3*G1*U(0,2)*RHD(5,1) + 23040*DR**4*DT**3*G1*U(0,3)*RHD(5,0) +
```

```
172800*DR**4*DT**3*G1*U(1,1)*RHD(4,2) + 172800*DR**4*DT**3*G1*U(1,2)*RHD(4,1) + 57600*DR**4*DT**3*G1*U(1,3)*RHD(4,0) +
57600*DR**4*DT**3*G1*U(2,0)*RHD(3,3) + 172800*DR**4*DT**3*G1*U(2,1)*RHD(3,2) + 172800*DR**4*DT**3*G1*U(2,2)*RHD(3,1) +
57600*DR**4*DT**3*G1*U(2,3)*RHD(3,0) + 28800*DR**4*DT**3*G1*U(3,0)*RHD(2,3) + 86400*DR**4*DT**3*G1*U(3,1)*RHD(2,2) +
86400*DR**4*DT**3*G1*U(3,2)*RHD(2,1) + 28800*DR**4*DT**3*G1*U(3,3)*RHD(2,0) + 7200*DR**4 [illegible] G1*U(4,0)*RHD(1,3) +
21600*DR**4*DT**3*G1*U(4,1)*RHD(1,2) + 21600*DR**4*DT**3*G1*U(4,2)*RHD(1,1) + 1440*DR**4*DT**3*G1*U(5,0)*RHD(0,3) +
4320*DR**4*DT**3*G1*U(5,1)*RHD(0,2) + 138240*DR**4*DT**2*G1*U(0,1)*RHD(5,1) + 69120*DR**4*DT**2*G1*U(0,2)*RHD(5,0) +
172800*DR**4*DT**2*G1*U(1,0)*RHD(4,2) + 345600*DR**4*DT**2*G1*U(1,1)*RHD(4,1) + 172800*DR**4*DT**2*G1*U(1,2)*RHD(4,0) +
172800*DR**4*DT**2*G1*U(2,0)*RHD(3,2) + 345600*DR**4*DT**2*G1*U(2,1)*RHD(3,1) + 172800*DR**4*DT**2*G1*U(2,2)*RHD(3,0) +
86400*DR**4*DT**2*G1*U(3,0)*RHD(2,2) + 172800*DR**4*DT**2*G1*U(3,1)*RHD(2,1) + 86400*DR**4*DT**2*G1*U(3,2)*RHD(2,0) +
21600*DR**4*DT**2*G1*U(4,0)*RHD(1,2) + 43200*DR**4*DT**2*G1*U(4,1)*RHD(1,1) + 21600*DR**4*DT**2*G1*U(4,2)*RHD(1,0) +
4320*DR**4*DT**2*G1*U(5,0)*RHD(0,2) + 8640*DR**4*DT**2*G1*U(5,1)*RHD(0,1) + 138240*DR**4*DT*G1*U(0,0)*RHD(5,1) +
138240*DR**4*DT*G1*U(0,1)*RHD(5,0) + 345600*DR**4*DT*G1*U(1,0)*RHD(4,1) + 345600*DR**4*DT*G1*U(1,1)*RHD(4,0) + 345600*DR**4*DT
G1*U(2,0)*RHD(3,1) + 345600*DR**4*DT*G1*U(2,1)*RHO(3,0) + 172800*DR**4*DT*G1*U(3,0)*RHD(2,1) + 172800*DR**4*DT*G1*U(3,1)*
RHD(2,0) + 43200*DR**4*DT*G1*U(4,0)*RHD(1,1) + 43200*DR**4*DT*G1*U(4,1)*RHD(1,0) + 8640*DR**4*DT*G1*U(5,0)*RHD(0,1) +
8640*DR**4*DT*G1*U(5,1)*RHD(0,0) - 46080*DR**4*T(0,0)*RHD(6,0) - 138240*DR**4*T(1,0)*RHC(5,0) - 172800*DR**4*T(2,0)*RHD(4,0) -
115200*DR**4*T(3,0)*RHD(3,0) - 43200*DR**4*T(4,0)*RHD(2,0) - 8640*DR**4*T(5,0)*RHD(1,0) + 138240*DR**4*U(0,0)*RHD(5,0) +
345600*DR**4*U(1,0)*RHD(4,0) + 345600*DR**4*U(2,0)*RHD(3,0) + 172800*DR**4*U(3,0)*RHD(2,0) + 43200*DR**4*U(4,0)*RHD(1,0) +
8640*DR**4*U(5,0)*RHD(0,0) + 640*DR**2*DT**9*G1*U(0,6)*RHD(3,3) + 1440*DR**2*DT**9*G1*U(1,5)*RHD(2,4) + 720*DR**2*DT**9*G1*
U(2,4)*RHD(1,5) + 160*DR**2*DT**9*G1*U(3,3)*RHD(0,6) + 3840*DR**2*DT**8*G1*U(0,5)*RHD(3,3) + 1920*DR**2*DT**8*G1*U(0,6)*
RHD(3,2) + 7200*DR**2*DT**8*G1*U(1,4)*RHD(2,4) + 5760*DR**2*DT**8*G1*U(1,5)*RHD(2,3) + 2880*DR**2*DT**8*G1*U(2,3)*RHD(1,5) +
3600*DR**2*DT**8*G1*U(2,4)*RHD(1,4) + 480*DR**2*DT**8*G1*U(3,2)*RHD(0,6) + 960*DR**2*DT**8*G1*U(3,3)*RHO(0,5) +
19200*DR**2*DT**7*G1*U(0,4)*RHD(3,3) + 11520*DR**2*DT**7*G1*U(0,5)*RHD(3,2) + 3840*DR**2*DT**7*G1*U(0,6)*RHD(3,1) +
28800*DR**2*DT**7*G1*U(1,3)*RHD(2,4) + 28800*DR**2*DT**7*G1*U(1,4)*RHD(2,3) + 17280*DR**2*DT**7*G1*U(1,5)*RHD(2,2) +
8640*DR**2*DT**7*G1*U(2,2)*RHD(1,5) + 14400*DR**2*DT**7*G1*U(2,3)*RHD(1,4) + 14400*DR**2*DT**7*G1*U(2,4)*RHD(1,3) +
960*DR**2*DT**7*G1*U(3,1)*RHD(0,6) + 2880*DR**2*DT**7*G1*U(3,2)*RHD(0,5) + 4800*DR**2*DT**7*G1*U(3,3)*RHD(0,4) +
76800*DR**2*DT**6*G1*U(0,3)*RHD(3,3) + 57600*DR**2*DT**6*G1*U(0,4)*RHD(3,2) + 23040*DR**2*DT**6*G1*U(0,5)*RHD(3,1) +
3840*DR**2*DT**6*G1*U(0,6)*RHD(3,0) + 86400*DR**2*DT**6*G1*U(1,2)*RHD(2,4) + 115200*DR**2*DT**6*G1*U(1,3)*RHD(2,3) +
86400*DR**2*DT**6*G1*U(1,4)*RHD(2,2) + 34560*DR**2*DT**6*G1*U(1,5)*RHD(2,1) + 17280*DR**2*DT**6*G1*U(2,1)*RHD(1,5) +
43200*DR**2*DT**6*G1*U(2,2)*RHD(1,4) + 57600*DR**2*DT**6*G1*U(2,3)*RHD(1,3) + 43200*DR**2*DT**6*G1*U(2,4)*RHD(1,2) +
960*DR**2*DT**6*G1*U(3,0)*RHD(0,6) + 5760*DR**2*DT**6*G1*U(3,1)*RHD(0,5) + 14400*DR**2*DT**6*G1*U(3,2)*RHD(0,4) +
19200*DR**2*DT**6*G1*U(3,3)*RHD(0,3) + 230400*DR**2*DT**5*G1*U(0,2)*RHD(3,3) + 230400*DR**2*DT**5*G1*U(0,3)*RHD(3,2) +
115200*DR**2*DT**5*G1*U(0,4)*RHD(3,1) + 23040*DR**2*DT**5*G1*U(0,5)*RHD(3,0) + 172800*DR**2*DT**5*G1*U(1,1)*RHD(2,4) +
345600*DR**2*DT**5*G1*U(1,2)*RHD(2,3) + 345600*DR**2*DT**5*G1*U(1,3)*RHD(2,2) + 172800*DR**2*DT**5*G1*U(1,4)*RHD(2,1) +
34560*DR**2*DT**5*G1*U(1,5)*RHD(2,0) + 17280*DR**2*DT**5*G1*U(2,0)*RHD(1,5) + 86400*DR**2*DT**5*G1*U(2,1)*RHD(1,4) +
172800*DR**2*DT**5*G1*U(2,2)*RHD(1,3) + 172800*DR**2*DT**5*G1*U(2,3)*RHD(1,2) + 86400*DR**2*DT**5*G1*U(2,4)*RHD(1,1) +
```

```
5760*DR**2*DT**5*G1*U(3,0)*RHD(0,5) + 28800*DR**2*DT**5*G1*U(3,1)*RHD(0,4) + 57600*DR**2*DT**5*G1*U(3,2)*RHD(0,3) +
57600*DR**2*DT**5*G1*U(3,3)*RHD(0,2) + 460800*DR**2*DT**4*G1*U(0,1)*RHD(3,3) + 691200*DR**2*DT**4*G1*U(0,2)*RHD(3,2) +
460800*DR**2*DT**4*G1*U(0,3)*RHD(3,1) + 115200*DR**2*DT**4*G1*U(0,4)*RHD(3,0) + 172800*DR**2*DT**4*G1*U(1,0)*RHD(2,4) +
691200*DR**2*DT**4*G1*U(1,1)*RHD(2,3) + 1036800*DR**2*DT**4*G1*U(1,2)*RHD(2,2) + 691200*DR**2*DT**4*G1*U(1,3)*RHD(2,1) +
172800*DR**2*DT**4*G1*U(1,4)*RHD(2,0) + 86400*DR**2*DT**4*G1*U(2,0)*RHD(1,4) + 345600*DR**2*DT**4*G1*U(2,1)*RHD(1,3) +
518400*DR**2*DT**4*G1*U(2,2)*RHD(1,2) + 345600*DR**2*DT**4*G1*U(2,3)*RHD(1,1) + 86400*DR**2*DT**4*G1*U(2,4)*RHD(1,0) +
28800*DR**2*DT**4*G1*U(3,0)*RHD(0,4) + 115200*DR**2*DT**4*G1*U(3,1)*RHD(0,3) + 172800*DR**2*DT**4*G1*U(3,2)*RHD(0,2) +
115200*DR**2*DT**4*G1*U(3,3)*RHD(0,1) + 460800*DR**2*DT**3*G1*U(0,0)*RHD(3,3) + 1382400*DR**2*DT**3*G1*U(0,1)*RHD(3,2) +
1382400*DR**2*DT**3*G1*U(0,2)*RHD(3,1) + 460800*DR**2*DT**3*G1*U(0,3)*RHD(3,0) + 691200*DR**2*DT**3*G1*U(1,0)*RHD(2,3) +
2073600*DR**2*DT**3*G1*U(1,1)*RHD(2,2) + 2073600*DR**2*DT**3*G1*U(1,2)*RHD(2,1) + 691200*DR**2*DT**3*G1*U(1,3)*RHD(2,0) +
345600*DR**2*DT**3*G1*U(2,0)*RHD(1,3) + 1036800*DR**2*DT**3*G1*U(2,1)*RHD(1,2) + 1036800*DR**2*DT**3*G1*U(2,2)*RHD(1,1) +
345600*DR**2*DT**3*G1*U(2,3)*RHD(1,0) + 115200*DR**2*DT**3*G1*U(3,0)*RHD(0,3) + 345600*DR**2*DT**3*G1*U(3,1)*RHD(0,2) +
345600*DR**2*DT**3*G1*U(3,2)*RHD(0,1) + 115200*DR**2*DT**3*G1*U(3,3)*RHD(0,0) + 1382400*DR**2*DT**2*G1*U(0,0)*RHD(3,2) +
2764800*DR**2*DT**2*G1*U(0,1)*RHD(3,1) + 1382400*DR**2*DT**2*G1*U(0,2)*RHD(3,0) + 2073600*DR**2*DT**2*G1*U(1,0)*RHD(2,2) +
4147200*DR**2*DT**2*G1*U(1,1)*RHD(2,1) + 2073600*DR**2*DT**2*G1*U(1,2)*RHD(2,0) + 1036800*DR**2*DT**2*G1*U(2,0)*RHD(1,2) +
2073600*DR**2*DT**2*G1*U(2,1)*RHD(1,1) + 1036800*DR**2*DT**2*G1*U(2,2)*RHD(1,0) + 345600*DR**2*DT**2*G1*U(3,0)*RHD(0,2) +
691200*DR**2*DT**2*G1*U(3,1)*RHD(0,1) + 345600*DR**2*DT**2*G1*U(3,2)*RHD(0,0) + 2764800*DR**2*DT*G1*U(0,0)*RHD(3,1) +
2764800*DR**2*DT*G1*U(0,1)*RHD(3,0) + 4147200*DR**2*DT*G1*U(1,0)*RHD(2,1) + 4147200*DR**2*DT*G1*U(1,1)*RHD(2,0) +
2073600*DR**2*DT*G1*U(2,0)*RHD(1,1) + 2073600*DR**2*DT*G1*U(2,1)*RHD(1,0) + 691200*DR**2*DT*G1*U(3,0)*RHD(0,1) +
691200*DR**2*DT*G1*U(3,1)*RHD(0,0) - 1382400*DR**2*T(0,0)*RHD(4,0) - 2764800*DR**2*T(1,0)*RHD(3,0) - 2073600*DR**2*T(2,0)*
RHD(2,0) - 691200*DR**2*T(3,0)*RHD(1,0) + 2764800*DR**2*U(0,0)*RHD(3,0) + 4147200*DR**2*U(1,0)*RHD(2,0) + 2073600*DR**2*U(2,0)*
RHD(1,0) + 691200*DR**2*U(3,0)*RHD(0,0) + 192*DT**11*G1*U(0,6)*RHD(1,5) + 192*DT**11*G1*U(1,5)*RHD(0,6) + 1152*DT**10*G1*
U(0,5)*RHD(1,5) + 960*DT**10*G1*U(0,6)*RHD(1,4) + 960*DT**10*G1*U(1,4)*RHD(0,6) + 1152*DT**10*G1*U(1,5)*RHD(0,5) +
5760*DT**9*G1*U(0,4)*RHD(1,5) + 5760*DT**9*G1*U(0,5)*RHD(1,4) + 3840*DT**9*G1*U(0,6)*RHD(1,3) + 3840*DT**9*G1*U(1,3)*RHD(0,6) +
5760*DT**9*G1*U(1,4)*RHD(0,5) + 5760*DT**9*G1*U(1,5)*RHD(0,4) + 23040*DT**8*G1*U(0,3)*RHD(1,5) + 28800*DT**8*G1*U(0,4)*
RHD(1,4) + 23040*DT**8*G1*U(0,5)*RHD(1,3) + 11520*DT**8*G1*U(0,6)*RHD(1,2) + 11520*DT**8*G1*U(1,2)*RHD(0,6) + 23040*DT**8*G1*
U(1,3)*RHD(0,5) + 28800*DT**8*G1*U(1,4)*RHD(0,4) + 23040*DT**8*G1*U(1,5)*RHD(0,3) + 69120*DT**7*G1*U(0,2)*RHD(1,5) +
115200*DT**7*G1*U(0,3)*RHD(1,4) + 115200*DT**7*G1*U(0,4)*RHD(1,3) + 69120*DT**7*G1*U(0,5)*RHD(1,2) + 23040*DT**7*G1*U(0,6)*
RHD(1,1) + 23040*DT**7*G1*U(1,1)*RHD(0,6) + 69120*DT**7*G1*U(1,2)*RHD(0,5) + 115200*DT**7*G1*U(1,3)*RHD(0,4) + 115200*DT**7*G1*
U(1,4)*RHD(0,3) + 69120*DT**7*G1*U(1,5)*RHD(0,2) + 138240*DT**6*G1*U(0,1)*RHD(1,5) + 345600*DT**6*G1*U(0,2)*RHD(1,4) +
460800*DT**6*G1*U(0,3)*RHD(1,3) + 345600*DT**6*G1*U(0,4)*RHD(1,2) + 138240*DT**6*G1*U(0,5)*RHD(1,1) + 23040*DT**6*G1*U(0,6)*
RHD(1,0) + 23040*DT**6*G1*U(1,0)*RHD(0,6) + 138240*DT**6*G1*U(1,1)*RHD(0,5) + 345600*DT**6*G1*U(1,2)*RHD(0,4) +
460800*DT**6*G1*U(1,3)*RHD(0,3) + 345600*DT**6*G1*U(1,4)*RHD(0,2) + 138240*DT**6*G1*U(1,5)*RHD(0,1) + 138240*DT**5*G1*U(0,0)*
RHD(1,5) + 691200*DT**5*G1*U(0,1)*RHD(1,4) + 1382400*DT**5*G1*U(0,2)*RHD(1,3) + 1382400*DT**5*G1*U(0,3)*RHD(1,2) +
691200*DT**5*G1*U(0,4)*RHD(1,1) + 138240*DT**5*G1*U(0,5)*RHD(1,0) + 138240*DT**5*G1*U(1,0)*RHD(0,5) + 691200*DT**5*G1*U(1,1)*
```

```
RHD(0,4) + 1382400*DT**5*G1*U(1,2)*RHD(0,3) + 1382400*DT**5*G1*U(1,3)*RHD(0,2) + 691200*DT**5*G1*U(1,4)*RHD(0,1) +
138240*DT**5*G1*U(1,5)*RHD(0,0) + 23040*DT**5*RHD(0,6) + 691200*DT**4*G1*U(0,0)*RHD(1,4) + 2764800*DT**4*G1*U(0,1)*RHD(1,3) +
4147200*DT**4*G1*U(0,2)*RHD(1,2) + 2764800*DT**4*G1*U(0,3)*RHD(1,1) + 691200*DT**4*G1*U(0,4)*RHD(1,0) + 691200*DT**4*G1*U(1,0)*
RHD(0,4) + 2764800*DT**4*G1*U(1,1)*RHD(0,3) + 4147200*DT**4*G1*U(1,2)*RHD(0,2) + 2764800*DT**4*G1*U(1,3)*RHD(0,1) +
691200*DT**4*G1*U(1,4)*RHD(0,0) + 138240*DT**4*RHD(0,5) + 2764800*DT**3*G1*U(0,0)*RHD(1,3) + 8294400*DT**3*G1*U(0,1)*RHD(1,2) +
8294400*DT**3*G1*U(0,2)*RHD(1,1) + 2764800*DT**3*G1*U(0,3)*RHD(1,0) + 2764800*DT**3*G1*U(1,0)*RHD(0,3) + 8294400*DT**3*G1*
U(1,1)*RHD(0,2) + 8294400*DT**3*G1*U(1,2)*RHD(0,1) + 2764800*DT**3*G1*U(1,3)*RHD(0,0) + 691200*DT**3*RHD(0,4) + 8294400*DT**2*
G1*U(0,0)*RHD(),2) + 16588800*DT**2*G1*U(0,1)*RHD(1,1) + 8294400*DT**2*G1*U(0,2)*RHD(1,0) + 8294400*DT**2*G1*U(1,0)*RHD(0,2) +
16588800*DT**2*G1*U(1,1)*RHD(0,1) + 8294400*DT**2*G1*U(1,2)*RHD(0,0) + 2764800*DT**2*RHD(0,3) + 16588800*DT*G1*U(0,0)*
RHD(1,1) + 16588800*DT*G1*U(0,1)*RHD(1,0) + 16588800*DT*G1*U(1,0)*RHD(0,1) + 16588800*DT*G1*U(1,1)*RHD(0,0) +
8294400*DT*RHD(0,2) - 16588800*T(0,0)*RHD(2,0) - 16588800*T(1,0)*RHD(1,0) + 16588800*U(0,0)*RHD(1,0) + 16588800*U(1,0)*
RHD(0,0) + 16588800*RHD(0,1) ) / 16588800
# END PHASE ONE
# DETPS
( - ( T(0,0)*RHD(2,0) + T(1,0)*RHD(1,0) - U(0,0)*RHD(1,0) - U(1,0)*RHD(0,0) - RHD(0,1) ) ,
  0 ,
  0 ,
  0 ,
  0 ,
  0 ,
  0 )
# 7DETPS
( - ( T(0,0)*RHD(2,0) + T(1,0)*RHD(1,0) - U(0,0)*RHD(1,0) - U(1,0)*RHD(0,0) - RHD(0,1) ) ,
DT * ( 2*G1*U(0,0)*RHD(1,1) + 2*G1*U(0,1)*RHD(1,0) + 2*G1*U(1,0)*RHD(0,1) + 2*G1*U(1,1)*RHD(0,0) + RHD(0,2) ) / 2 ,
- ( 2*DR**2*T(0,0)*RHD(4,0) + 4*DR**2*T(1,0)*RHD(3,0) + 3*DR**2*T(2,0)*RHD(2,0) + DR**2*T(3,0)*RHD(1,0) - 4*DR**2*U(0,0)*
RHD(3,0) - 6*DR**2*U(1,0)*RHD(2,0) - 3*DR**2*U(2,0)*RHD(1,0) - DR**2*U(3,0)*RHD(0,0) - 12*DT**2*G1*U(0,0)*RHD(1,2) - 24*DT**2*
G1*U(0,1)*RHD(1,1) - 12*DT**2*G1*U(0,2)*RHD(1,0) - 12*DT**2*G1*U(1,0)*RHD(0,2) - 24*DT**2*G1*U(1,1)*RHD(0,1) - 12*DT**2*G1*
U(1,2)*RHD(0,0) - 4*DT**2*RHD(0,3) ) / 24 ,
DT * ( 4*DR**2*G1*U(0,0)*RHD(3,1) + 4*DR**2*G1*U(0,1)*RHD(3,0) + 6*DR**2*G1*U(1,0)*RHD(2,1) + 6*DR**2*G1*U(1,1)*RHD(2,0) +
3*DR**2*G1*U(2,0)*RHD(1,1) + 3*DR**2*G1*U(2,1)*RHD(1,0) + DR**2*G1*U(3,0)*RHD(0,1) + DR**2*G1*U(3,1)*RHD(0,0) +
4*DT**2*G1*U(0,0)*RHD(1,3) + 12*DT**2*G1*U(0,1)*RHD(1,2) + 12*DT**2*G1*U(0,2)*RHD(1,1) + 4*DT**2*G1*U(0,3)*RHD(1,0) +
4*DT**2*G1*U(1,0)*RHD(0,3) + 12*DT**2*G1*U(1,1)*RHD(0,2) + 12*DT**2*G1*U(1,2)*RHD(0,1) + 4*DT**2*G1*U(1,3)*RHD(0,0) +
DT**2*RHD(0,4) ) / 24 ,
- ( 16*DR**4*T(0,0)*RHD(6,0) + 48*DR**4*T(1,0)*RHD(5,0) + 60*DR**4*T(2,0)*RHD(4,0) + 40*DR**4*T(3,0)*RHD(3,0) +
15*DR**4*T(4,0)*RHD(2,0) + 3*DR**4*T(5,0)*RHD(1,0) - 48*DR**4*U(0,0)*RHD(5,0) - 120*DR**4*U(1,0)*RHD(4,0) - 120*DR**4*U(2,0)*
```

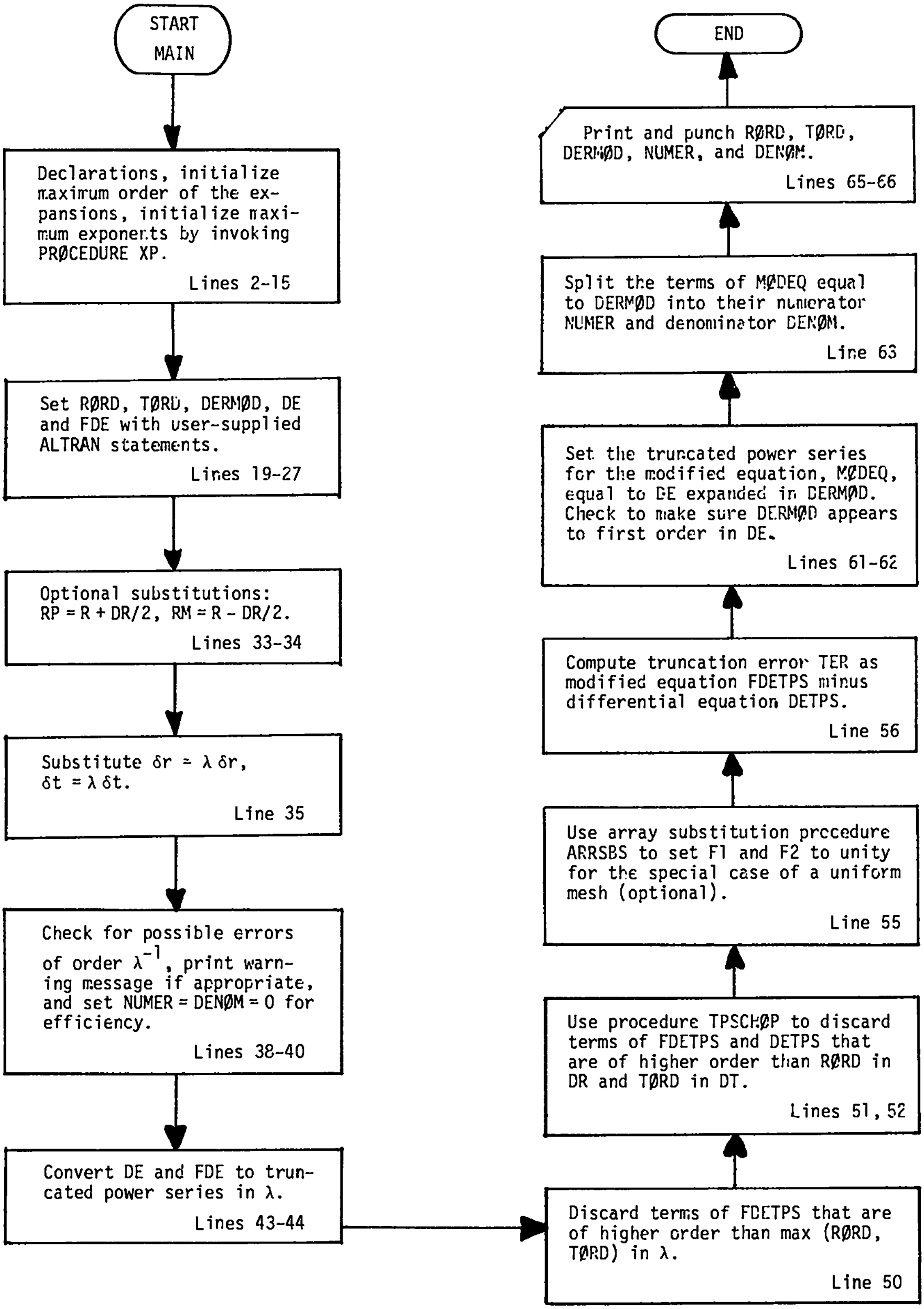
```
RHO(3,0) - 60*DR**4*U(3,0)*RHO(2,0) - 15*DR**4*U(4,0)*RHO(1,0) - 3*DR**4*U(5,0)*RHO(0,0) - 480*DR**2*DT**2*G1*U(0,0)*RHO(3,2) -
960*DR**2*DT**2*G1*U(0,1)*RHO(3,1) - 480*DR**2*DT**2*G1*U(0,2)*RHO(3,0) - 720*DR**2*DT**2*G1*U(1,0)*RHO(2,2) - 1440*DR**2*
DT**2*G1*U(1,1)*RHO(2,1) - 720*DR**2*DT**2*G1*U(1,2)*RHO(2,0) - 360*DR**2*DT**2*G1*U(2,0)*RHO(1,2) - 720*DR**2*DT**2*G1*U(2,1)*
RHO(1,1) - 360*DR**2*DT**2*G1*U(2,2)*RHO(1,0) - 120*DR**2*DT**2*G1*U(3,0)*RHO(0,2) - 240*DR**2*DT**2*G1*U(3,1)*RHO(0,1) -
120*DR**2*DT**2*G1*U(3,2)*RHO(0,0) - 240*DT**4*G1*U(0,0)*RHO(1,4) - 960*DT**4*G1*U(0,1)*RHO(1,3) - 1440*DT**4*G1*U(0,2)*
RHO(1,2) - 960*DT**4*G1*U(0,3)*RHO(1,1) - 240*DT**4*G1*U(0,4)*RHO(1,0) - 240*DT**4*G1*U(1,0)*RHO(0,4) - 960*DT**4*G1*U(1,1)*
RHO(0,3) - 1440*DT**4*G1*U(1,2)*RHO(0,2) - 960*DT**4*G1*U(1,3)*RHO(0,1) - 240*DT**4*G1*U(1,4)*RHO(0,0) - 48*DT**4*RHO(0,5) )
5760 ,
DT * ( 48*DR**4*G1*U(0,0)*RHO(5,1) + 48*DR**4*G1*U(0,1)*RHO(5,0) + 120*DR**4*G1*U(1,0)*RHO(4,1) + 120*DR**4*G1*U(1,1)*
RHO(4,0) + 120*DR**4*G1*U(2,0)*RHO(3,1) + 120*DR**4*G1*U(2,1)*RHO(3,0) + 60*DR**4*G1*U(3,0)*RHO(2,1) + 60*DR**4*G1*U(3,1)*
RHO(2,0) + 15*DR**4*G1*U(4,0)*RHO(1,1) + 15*DR**4*G1*U(4,1)*RHO(1,0) + 3*DR**4*G1*U(5,0)*RHO(0,1) + 3*DR**4*G1*U(5,1)*
RHO(0,0) + 160*DR**2*DT**2*G1*U(0,0)*RHO(3,3) + 480*DR**2*DT**2*G1*U(0,1)*RHO(3,2) + 480*DR**2*DT**2*G1*U(0,2)*RHO(3,1) +
160*DR**2*DT**2*G1*U(0,3)*RHO(3,0) + 240*DR**2*DT**2*G1*U(1,0)*RHO(2,3) + 720*DR**2*DT**2*G1*U(1,1)*RHO(2,2) + 720*DR**2*DT**2*
G1*U(1,2)*RHO(2,1) + 240*DR**2*DT**2*G1*U(1,3)*RHO(2,0) + 120*DR**2*DT**2*G1*U(2,0)*RHO(1,3) + 360*DR**2*DT**2*G1*U(2,1)*
RHO(1,2) + 360*DR**2*DT**2*G1*U(2,2)*RHO(1,1) + 120*DR**2*DT**2*G1*U(2,3)*RHO(1,0) + 40*DR**2*DT**2*G1*U(3,0)*RHO(0,3) +
120*DR**2*DT**2*G1*U(3,1)*RHO(0,2) + 120*DR**2*DT**2*G1*U(3,2)*RHO(0,1) + 40*DR**2*DT**2*G1*U(3,3)*RHO(0,0) +
48*DT**4*G1*U(0,0)*RHO(1,5) + 240*DT**4*G1*U(0,1)*RHO(1,4) + 480*DT**4*G1*U(0,2)*RHO(1,3) + 480*DT**4*G1*U(0,3)*RHO(1,2) +
240*DT**4*G1*U(0,4)*RHO(1,1) + 48*DT**4*G1*U(0,5)*RHO(1,0) + 48*DT**4*G1*U(1,0)*RHO(0,5) + 240*DT**4*G1*U(1,1)*RHO(0,4) +
480*DT**4*G1*U(1,2)*RHO(0,3) + 480*DT**4*G1*U(1,3)*RHO(0,2) + 240*DT**4*G1*U(1,4)*RHO(0,1) + 48*DT**4*G1*U(1,5)*RHO(0,0) +
8*DT**4*RHO(0,6) ) / 5760 ,
- ( 16*DR**6*T(2,0)*RHO(6,0) + 16*DR**6*T(3,0)*RHO(5,0) + 10*DR**6*T(4,0)*RHO(4,0) + 4*DR**6*T(5,0)*RHO(3,0) +
DR**6*T(6,0)*RHO(2,0) - 32*DR**6*U(1,0)*RHO(6,0) - 48*DR**6*U(2,0)*RHO(5,0) - 40*DR**6*U(3,0)*RHO(4,0) - 20*DR**6*U(4,0)*
RHO(3,0) - 6*DR**6*U(5,0)*RHO(2,0) - DR**6*U(6,0)*RHO(1,0) - 384*DR**4*DT**2*G1*U(0,1)*RHO(5,1) - 192*DR**4*DT**2*G1*U(0,2)*
RHO(5,0) - 480*DR**4*DT**2*G1*U(1,0)*RHO(4,2) - 960*DR**4*DT**2*G1*U(1,1)*RHO(4,1) - 480*DR**4*DT**2*G1*U(1,2)*RHO(4,0) -
480*DR**4*DT**2*G1*U(2,0)*RHO(3,2) - 960*DR**4*DT**2*G1*U(2,1)*RHO(3,1) - 480*DR**4*DT**2*G1*U(2,2)*RHO(3,0) - 240*DR**4*DT**2*
G1*U(3,0)*RHO(2,2) - 480*DR**4*DT**2*G1*U(3,1)*RHO(2,1) - 240*DR**4*DT**2*G1*U(3,2)*RHO(2,0) - 60*DR**4*DT**2*G1*U(4,0)*
RHO(1,2) - 120*DR**4*DT**2*G1*U(4,1)*RHO(1,1) - 60*DR**4*DT**2*G1*U(4,2)*RHO(1,0) - 12*DR**4*DT**2*G1*U(5,0)*RHO(0,2) -
24*DR**4*DT**2*G1*U(5,1)*RHO(0,1) - 1280*DR**2*DT**4*G1*U(0,1)*RHO(3,3) - 1920*DR**2*DT**4*G1*U(0,2)*RHO(3,2) - 1280*DR**2*
DT**4*G1*U(0,3)*RHO(3,1) - 320*DR**2*DT**4*G1*U(0,4)*RHO(3,0) - 480*DR**2*DT**4*G1*U(1,0)*RHO(2,4) - 1920*DR**2*DT**4*G1*
U(1,1)*RHO(2,3) - 2880*DR**2*DT**4*G1*U(1,2)*RHO(2,2) - 1920*DR**2*DT**4*G1*U(1,3)*RHO(2,1) - 480*DR**2*DT**4*G1*U(1,4)*
RHO(2,0) - 240*DR**2*DT**4*G1*U(2,0)*RHO(1,4) - 960*DR**2*DT**4*G1*U(2,1)*RHO(1,3) - 1440*DR**2*DT**4*G1*U(2,2)*RHO(1,2) -
960*DR**2*DT**4*G1*U(2,3)*RHO(1,1) - 240*DR**2*DT**4*G1*U(2,4)*RHO(1,0) - 80*DR**2*DT**4*G1*U(3,0)*RHO(0,4) - 320*DR**2*DT**4*
G1*U(3,1)*RHO(0,3) - 480*DR**2*DT**4*G1*U(3,2)*RHO(0,2) - 320*DR**2*DT**4*G1*U(3,3)*RHO(0,1) - 384*DT**6*G1*U(0,1)*RHO(1,5) -
```

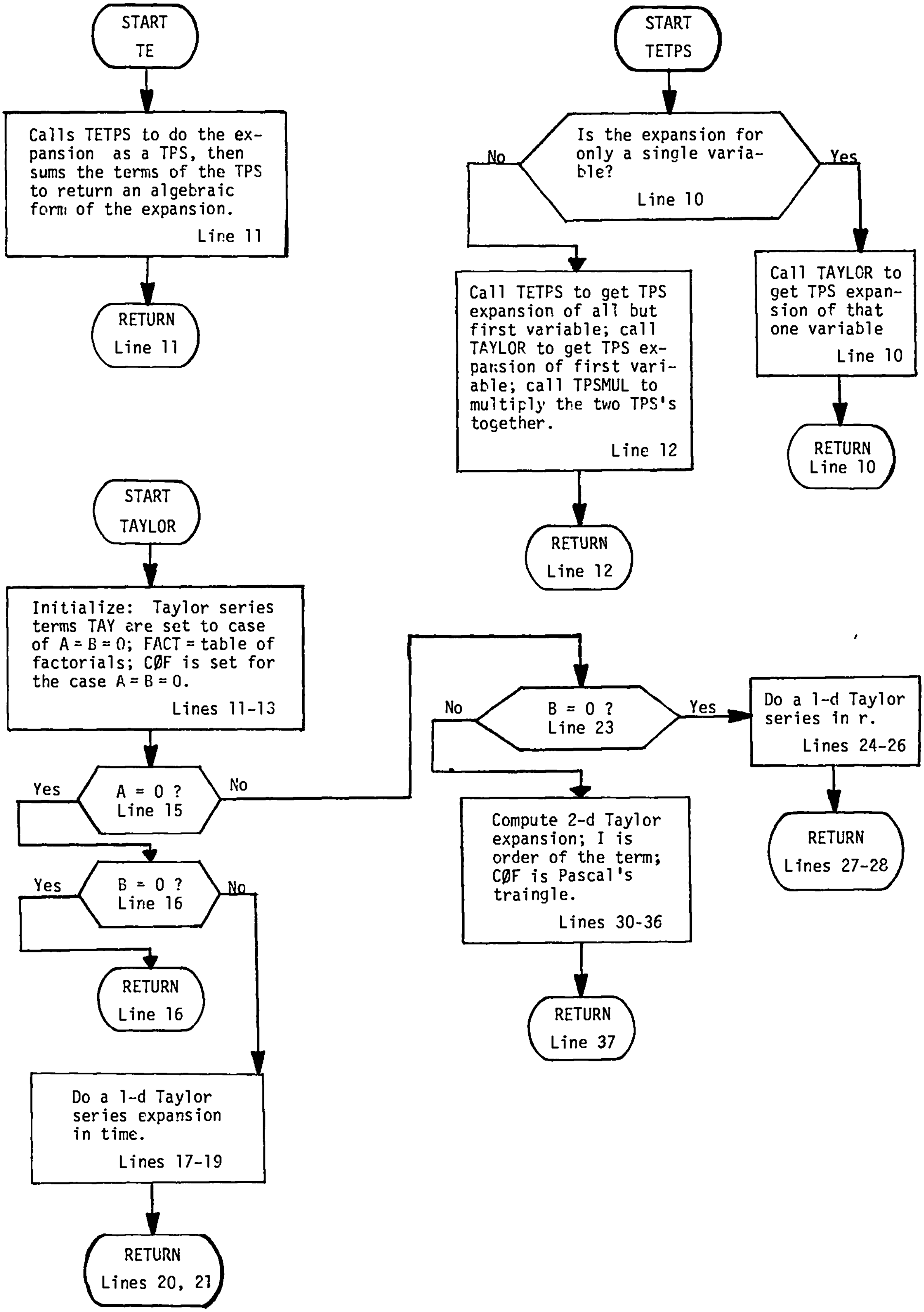
```
960*DT**6*G1*U(0,2)*RHD(1,4) - 1280*DT**6*G1*U(0,3)*RHD(1,3) - 960*DT**6*G1*U(0,4)*RHD(1,2) - 384*DT**6*G1*U(0,5)*RHD(1,1) -
64*DT**6*G1*U(0,6)*RHD(1,0) - 64*DT**6*G1*U(1,0)*RHD(0,6) - 384*DT**6*G1*U(1,1)*RHD(0,5) - 960*DT**6*G1*U(1,2)*RHD(0,4) -
1280*DT**6*G1*U(1,3)*RHD(0,3) - 960*DT**6*G1*U(1,4)*RHD(0,2) - 384*DT**6*G1*U(1,5)*RHD(0,1) ) / 46080 )
# FDETPS
  - ( T(0,0)*RHD(2,0) + T(1,0)*RHD(1,0) - U(0,0)*RHD(1,0) - U(1,0)*RHD(0,0) - RHD(0,1) ) ,
DT * ( 2*G1*U(0,0)*RHD(1,1) + 2*G1*U(0,1)*RHD(1,0) + 2*G1*U(1,0)*RHD(0,1) + 2*G1*U(1,1)*RHD(0,0) + RHD(0,2) ) / 2 ,
- DR**2 * ( 2*T(0,0)*RHD(4,0) + 4*T(1,0)*RHD(3,0) + 3*T(2,0)*RHD(2,0) + T(3,0)*RHD(1,0) - 4*U(0,0)*RHD(3,0) - 6*U(1,0)*
RHD(2,0) - 3*U(2,0)*RHD(1,0) - U(3,0)*RHD(0,0) ) / 24 )
# FDETPS
( - (29; T(0,0)*RHD(2,0) + T(1,0)*RHD(1,0) - U(0,0)*RHD(1,0) - U(1,0)*RHD(0,0) - RHD(0,1) ) ,
328; DT * ( 2*G1*U(0,0)*RHD(1,1) + 2*G1*U(0,1)*RHD(1,0) + 2*G1*U(1,0)*RHD(0,1) + 2*G1*U(1,1)*RHD(0,0) + RHD(0,2) ) / 2 ,
370; - DR**2 * ( 2*T(0,0)*RHD(4,0) + 4*T(1,0)*RHD(3,0) + 3*T(2,0)*RHD(2,0) + T(3,0)*RHD(1,0) - 4*U(0,0)*RHD(3,0) - 6*U(1,0)*
RHD(2,0) - 3*U(2,0)*RHD(1,0) - U(3,0)*RHD(0,0) ) / 24 )
# CONTPS
( - ( T(0,0)*RHD(2,0) + T(1,0)*RHD(1,0) - U(0,0)*RHD(1,0) - U(1,0)*RHD(0,0) - RHD(0,1) ) ,
DT * ( 2*G1*U(0,0)*RHD(1,1) + 2*G1*U(0,1)*RHD(1,0) + 2*G1*U(1,0)*RHD(0,1) + 2*G1*U(1,1)*RHD(0,0) + RHD(0,2) ) / 2 ,
- DR**2 * ( 2*T(0,0)*RHD(4,0) + 4*T(1,0)*RHD(3,0) + 3*T(2,0)*RHD(2,0) + T(3,0)*RHD(1,0) - 4*U(0,0)*RHD(3,0) - 6*U(1,0)*
RHD(2,0) - 3*U(2,0)*RHD(1,0) - U(3,0)*RHD(0,0) ) / 24 )
# TER WITH ALL TIME DERIVATIVES
# TER
( 0 ,
328; DT * ( 2*G1*U(0,0)*RHD(1,1) + 2*G1*U(0,1)*RHD(1,0) + 2*G1*U(1,0)*RHD(0,1) + 2*G1*U(1,1)*RHD(0,0) + RHD(0,2) ) / 2 ,
370; - DR**2 * ( 2*T(0,0)*RHD(4,0) + 4*T(1,0)*RHD(3,0) + 3*T(2,0)*RHD(2,0) + T(3,0)*RHD(1,0) - 4*U(0,0)*RHD(3,0) - 6*U(1,0)*
RHD(2,0) - 3*U(2,0)*RHD(1,0) - U(3,0)*RHD(0,0) ) / 24 )
# CONSTRUCT THE MODIFIED EQUATION
# RORD
  2
# TORD
  1
# NUMER
2*DR**2*T(0,0)*RHD(4,0) + 4*DR**2*T(1,0)*RHD(3,0) + 3*DR**2*T(2,0)*RHD(2,0) + DR**2*T(3,0)*RHD(1,0) - 4*DR**2*U(0,0)*RHD(3,0) -
6*DR**2*U(1,0)*RHD(2,0) - 3*DR**2*U(2,0)*RHD(1,0) - DR**2*U(3,0)*RHD(0,0) - 24*DT*G1*U(0,0)*RHD(1,1) - 24*DT*G1*U(0,1)*
RHD(1,0) - 24*DT*G1*U(1,0)*RHD(0,1) - 24*DT*G1*U(1,1)*RHD(0,0) - 12*DT*RHD(0,2) + 24*T(0,0)*RHD(2,0) + 24*T(1,0)*RHD(1,0) -
24*U(0,0)*RHD(1,0) - 24*U(1,0)*RHD(0,0)
```

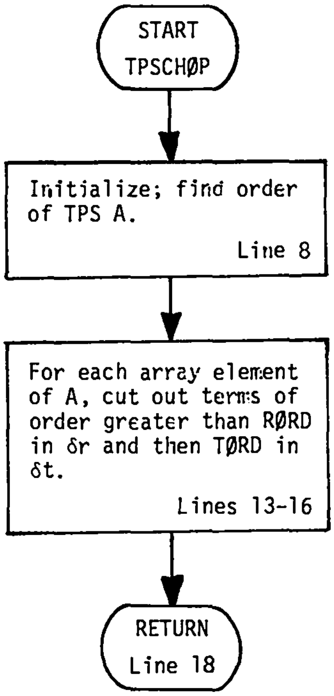
```
 # DENOM

      24

*** NORMAL RETURN FROM MAIN PROCEDURE

*** RUN STATISTICS

     14.264 SECONDS ELAPSED
     131070 WORDS IN WORKSPACE
         14 DIGITS IN SHORT INTEGERS
         28 DIGITS IN LONG  INTEGERS
          0 GARBAGE COLLECTIONS
      94557 WORDS OF WORKSPACE NEVER USED

$EJ
```

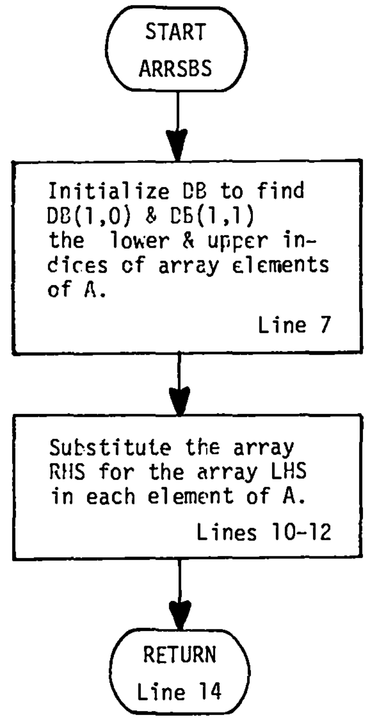
APPENDIX B

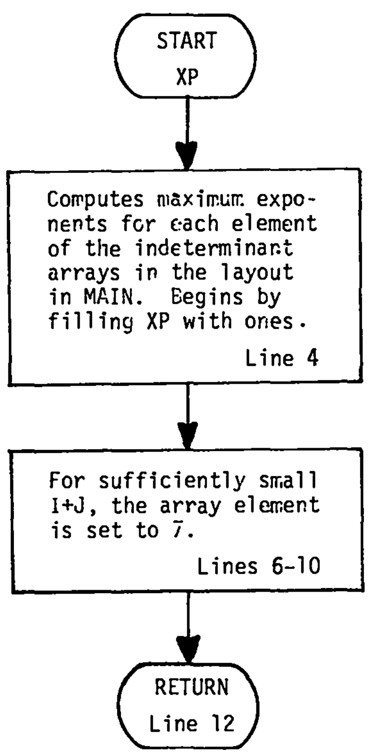
FLOW CHARTS FOR THE TRUNCATION ERROR EXPANSION PROGRAM




START TE
Calls TETPS to do the expansion as a TPS, then sums the terms of the TPS to return an algebraic form of the expansion. Line 11
RETURN Line 11
START TETPS
Is the expansion for only a single variable? Line 10
No
Yes
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. Line 12
RETURN Line 12
Call TAYLOR to get TPS expansion of that one variable Line 10
RETURN Line 10
START TAYLOR
Initialize: Taylor series terms TAY are set to case of A = B = 0; FACT = table of factorials; CØF is set for the case A = B = 0. Lines 11-13
A = 0 ? Line 15
Yes
No
B = 0 ? Line 16
Yes
No
RETURN Line 16
Do a 1-d Taylor series expansion in time. Lines 17-19
RETURN Lines 20, 21
B = 0 ? Line 23
No
Yes
Do a 1-d Taylor series in r. Lines 24-26
RETURN Lines 27-28
Compute 2-d Taylor expansion; I is order of the term; CØF is Pascal's traingle. Lines 30-36
RETURN Line 37


START
TPSCHØP
Initialize; find order of TPS A.
Line 8
For each array element of A, cut out terms of order greater than RØRD in δr and then TØRD in δt.
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, NUMER, 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.

```
      RORD = 2   ;   TORD = 1
      DERMOD = T(0,1)
      DE = T(2,1) - DIF * T(2,0) - 2 * DIF * T(1,0) / R
      FDE = (TE(T,0,1) - T(0,0)) / DT -
                 DIF * (RP**2*(TE(T,1,0)-T(0,0)) - RM**2*(T(0,0)-TE(T,-1,0))) /
        (R**2*DR**2)


# CONSTRUCT THE MODIFIED EQUATION

# RORD

      2

# TORD

      1

# NUMER

      DR**2*R**2*T(4,0)*DIF + 4*DR**2*R*T(3,0)*DIF + 3*DR**2*T(2,0)*DIF - 6*DT*R**2*T(0,2) + 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.

1. SAMPLE PROBLEM FROM THE EXPANSION PROGRAM CODE

```
ALTRAN VERSION 1 LEVEL 9


      PROCEDURE MAIN #  PROGRAM TO READ MODIFIED EQUATION AND ELIMINATE T DERIVS

      EXTERNAL INTEGER N1=7, N2=7
      INTEGER M=31, MM=7
      LONG ALGEBRAIC (DT:M, DR:M, R:M, RP:M, RM:M, G1:M, G2:M, LAM:M, F1:M,
        F2:M, DIF:M, U(0:N1,0:N2):MM, P(0:N1,0:N2):MM, RHO(0:N1,0:N2):MM,TI:M,
        T(0:N1,0:N2):MM) DERMOD, NUMER, DENOM, SECOND, SNUM, SDER, SDEN
      EXTERNAL LONG ALGEBRAIC LAN=LAM, S=R, TIM=TI
      EXTERNAL LONG ALGEBRAIC ARRAY R1=RHO, P1=P, T1=T, U1=U
      INTEGER I, J, RORD, TORD, IT, IR, ISR, IST, NT
      INTEGER ARRAY (0:N2) ISRM
      LONG ALGEBRAIC ARRAY (0:N1,0:N2) DERIV
      LONG ALGEBRAIC ARRAY SUB
      LONG ALGEBRAIC ALTRAN TDER, RDER
      ALGEBRAIC ARRAY ALTRAN TPS
      ALGEBRAIC ALTRAN TPSEVL
      REAL DELTA, ETIME

  # - - - - - - - - - - -

      READ RORD, TORD, DERMOD, NUMER, DENOM, SDER
      WRITE "INITIALIZATION", RORD, TORD, DERMOD, NUMER, DENOM, SDER
      SNUM=0
      SDEN=1
      IF (SDER.NE.0) DO
        READ SNUM, SDEN
        WRITE SNUM, SDEN
      DOEND

  # - - - - - - - - -

  #  SET UP THE SUBSTITUTION MATRIX
      DO I = 0, N1
        IR = I
        DO J = 0, N2
          IT = J
          IF (DERMOD.NE.RHO(I,J)) GO TO A1
          SUB = RHO
          GO TO B1
  A1:CONTINUE
          IF (DERMOD.NE.U(I,J)) GO TO A2
          SUB = U
          GO TO B1
  A2:CONTINUE
          IF (DERMOD.NE.P(I,J)) GO TO A3
          SUB = P
          GO TO B1
  A3:CONTINUE
```

```
            IF (DERMOD.NE.T(I,J)) GO TO A4
            SUB = T
            GO TO B1
    A4;CONTINUE
         DOEND
      DOEND
      WRITE DERMOD, "ILLEGAL DERMOD, ABORTING"
      GO TO ST
    B1;CONTINUE
      WRITE IR, IT, SUB
      DO I = IR, N1
         DO J = IT, N2
            DERIV(I-IR, J-IT) = SUB(I,J)
            DERIV(I,J) = 0
            SUB(I-IR, J-IT) = SUB(I,J)
            SUB(I,J) = 0
         DOEND
      DOEND
      ISR = N1 - IR
      IST = N2 - IT
      WRITE SUB, DERIV, ISR, IST
         DELTA=TIME(ETIME); WRITE DELTA,ETIME

    #  - - - - - - - - - - - -
    #  CALCULATE HIGHEST ORDER DERIVATIVES NEEDED

      IR = 0
      IT = 0
      DO J = 0, IST
         ISRM(J) = 0
         DO I = ISR, 0, -1
            NT = IMAX( IMAX( DEG(NUMER, SUB(I,J)), DEG(DENOM, SUB(I,J))),
               IMAX(DEG(SNUM, SUB(I,J)), DEG(SDEN, SUB(I,J))) )
            IF (NT.GT.0) DO
               IR = I
               IT = J
               ISRM(J) = I
               GO TO NMO
            DOEND
         DOEND
    NMO;CONTINUE
      DOEND; NT = DEG(NUMER, SUB(0,0)) + DEG(DENOM, SUB(0,0)) + DEG(SNUM, SUB(0,0)) +
                 DEG(SDEN, SUB(0,0))
      IF (IR.GT.0 .OR. IT.GT.0 .OR. NT.GT.0) GO TO QS
      WRITE +NO TIME DERIVATIVES FOUND THAT CAN BE ELIMINATED+
      GO TO ST
    QS;CONTINUE
      ISR = IR
      IST = IT
      WRITE "MAXIMUM ORDER OF DERIVATIVE TO BE COMPUTED", ISR, IST, ISRM

         DELTA=TIME(ETIME); WRITE DELTA,ETIME

    # - - - - - - - - - - -
```

```
# CREATE HIGHER ORDER DERIVATIVES

    DERIV(0,0) = NUMER / DENOM
    WRITE DERIV(0,0)

# PURE TIME DERIVATIVE OF ORDER IT
    DO IT = 0, IST
      IF (IT.GT.0) DO
      NUMER = ANUM(DERIV(0,IT-1), DENOM)
      DERIV(0,IT) = (TDER(NUMER)*DENOM - TDER(DENOM)*NUMER) / DENOM**2
      WRITE "PURE TIME DERIVATIVE", IT, NUMER, DENOM, DERIV(0,IT)
      DOEND

      WRITE "SPACE DERIVATIVES"
      IF (ISRM(IT).GT.0) DO IR = 1, ISRM(IT)
      NUMER = ANUM(DERIV(IR-1,IT), DENOM)
      DERIV(IR,IT) = (RDER(NUMER)*DENOM - RDER(DENOM)*NUMER) / DENOM**2
        WRITE IR, NUMER, DENOM, DERIV(IR,IT)
      DOEND
    DOEND

    WRITE DERIV
      DELTA=TIME(ETIME); WRITE DELTA,ETIME
    NUMER=0
    DENOM=0
# - - - - - - - - - -
#  ELIMINATE TIME DERIVATIVES FROM THE PRIMARY MODIFIED EQUATION

    DO J = 0, IST
      DO I = 0, ISRM(J)
        DERIV(0,0) = DERIV(0,0) (SUB(I,J) = DERIV(I,J))
        DERIV(0,0) = TPSEVL(TPS(DERIV(0,0) (DR, DT = LAM*DR, LAM*DT), LAM,
          IMAX(RORD, TORD)), 1)
        DERIV(0,0) = TPSEVL(TPS(DERIV(0,0) (DR = LAM*DR), LAM, RORD), 1)
        DERIV(0,0) = TPSEVL(TPS(DERIV(0,0) (DT = LAM*DT), LAM, TORD), 1)
      DOEND
    DOEND
    NUMER = ANUM (DERIV(0,0), DENOM)
    WRITE RORD, TORD, DERMOD, NUMER, DENOM
    WRITE (25) RORD, TORD, DERMOD, NUMER, DENOM, SDER
      DELTA=TIME(ETIME); WRITE DELTA,ETIME

    NUMER=0
    DENOM=0
    IF (SDER.EQ.0) GO TO ST

# - - - - - - - - - - - - - -
#  ELIMINATE TIME DERIVATIVES FROM THE SECONDARY MODIFIED EQUATION

    WRITE "SECONDARY EQUATION", SDER, SNUM, SDEN
    SECOND = SNUM / SDEN
    DO J = 0, IST
      DO I = 0, ISRM(J)
        SECOND = SECOND (SUB(I,J) = DERIV(I,J))
```

```
                SECOND = TPSEVL (TPS (SECOND (DR, DT = LAM*DR, LAM*DT), LAM,
                  IMAX (RORD, TORD)), 1)
                SECOND = TPSEVL (TPS (SECOND (DR = LAM*DR), LAM, RORD), 1)
                SECOND = TPSEVL (TPS (SECOND (DT = LAM*DT), LAM, TORD), 1)
              DOEND
            DOEND
            SNUM = ANUM (SECOND, SDEN)
            WRITE SNUM, SDEN
            WRITE (25) SNUM, SDEN

        ST; CONTINUE
              DELTA=TIME(ETIME); WRITE DELTA,ETIME

            END
```

NAME/EXTNAME	USE	TYPE	STRUC	PREC	CLASS	SCOPE	DR	LAY	ADDR
ISRM	VAR	INT	A				D*006		
DERIV	VAR	ALG	A	L			D*007		
DENOM	VAR	ALG		L				L*001	
DERMOD	VAR	ALG		L				L*001	
DIF	IND	ALG						L*001	
DR	IND	ALG						L*001	
DT	IND	ALG						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	
NUMER	VAR	ALG		L				L*001	
RM	IND	ALG						L*001	
RP	IND	ALG						L*001	
R	IND	ALG						L*001	
SDEN	VAR	ALG		L				L*001	
SDER	VAR	ALG		L				L*001	
SECOND	VAR	ALG		L				L*001	
SNUM	VAR	ALG		L				L*001	
TI	IND	ALG						L*001	
U	IND	ALG	A					L*001	
P	IND	ALG	A					L*001	
RHO	IND	ALG	A					L*001	
T	IND	ALG	A					L*001	
ANUM/S9ANUM	PROC	ALG		L	S	X			
DEG/S9DEG	PROC	INT			S	X			
DELTA	VAR	REAL							
ETIME	VAR	REAL							
IMAX/S9IMAX	PROC	INT		L	S	X			
IR	VAR	INT							
ISR	VAR	INT							
IST	VAR	INT							
IT	VAR	INT							
I	VAR	INT							

J	VAR	INT					
LAN	VAR	ALG		L	S	X	
MAIN	PROC			L	S	X	
MM	VAR	INT					
M	VAR	INT					
NT	VAR	INT					
N1	VAR	INT			S	X	
N2	VAR	INT			S	X	
P1	VAR	ALG	A	L	S	X	
RDER	PROC	ALG		L	S	X	
RORD	VAR	INT					
R1	VAR	ALG	A	L	S	X	
SUB	VAR	ALG	A	L			
S	VAR	ALG		L	S	X	
TDER	PROC	ALG		L	S	X	
TIME/S9CLCK	PROC	REAL		L	S	X	
TIM	VAR	ALG		L	S	X	
TORD	VAR	INT					
TPSEVL	PROC	ALG		L	S	X	
TPS	PROC	ALG	A	L	S	X	
T1	VAR	ALG	A	L	S	X	
U1	VAR	ALG	A	L	S	X	
D*006	DB						
D*007	DB						
L*001	LAY						
A1	CONS	LAB			S		328
A2	CONS	LAB			S		352
A3	CONS	LAB			S		376
A4	CONS	LAB			S		400
B1	CONS	LAB			S		422
NMD	CONS	LAB			S		727
QS	CONS	LAB			S		757
ST	CONS	LAB			S		1598
ILLEGAL DERMOD, ABOR	CONS	CHAR			S		
INITTIALIZATION	CONS	CHAR			S		
MAXIMUM ORDER OF DER	CONS	CHAR			S		
NO TIME DERIVATIVES	CONS	CHAR			S		
PURE TIME DERIVATIVE	CONS	CHAR			S		
SECONDARY EQUATION	CONS	CHAR			S		
SPACE DERIVATIVES	CONS	CHAR			S		
0	CONS	INT			S		
1	CONS	INT			S		
25	CONS	INT			S		
2	CONS	INT			S		
31	CONS	INT			S		
7	CONS	INT			S		

2. LISTING OF THE ELIMINATION PROGRAM AND FIRST RUN OF THE SAMPLE PROBLEM

ALTRAN VERSION 1 LEVEL 9

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

     EXTERNAL INTEGER N1, N2
EXTERNAL LONG ALGEBRAIC LAN, S, TIM
EXTERNAL LONG ALGEBRAIC ARRAY R1, P1, T1, U1

VALUE A
LONG ALGEBRAIC 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, 0, -1
       IF (A.NE.A(R1(I,N2), T1(I,N2), P1(I,N2), U1(I,N2) =0,0,0,0)) GO TO KICKOUT
       DO J = N2-1, 0, -1
         DER = DER + DIFF(A, R1(I,J)) * R1(I,J+1) + DIFF(A, P1(I,J)) * P1(I,J+1)+
           DIFF (A, U1(I,J)) * U1(I,J+1) + DIFF(A, T1(I,J)) * T1(I,J+1)
       DOEND
     DOEND

     RETURN(DER)

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

     END
```

NAME/EXTNAME	USE	TYPE	STRUC	PREC	CLASS	SCOPE	DB	LAY	ADDR
A	VAR	ALG		L		V			
DER	VAR	ALG		L					
DIFF/A9DIFF	PROC	ALG		L	S	X			
I	VAR	INT							
J	VAR	INT							
LAN	VAR	ALG		L	S	X			
N1	VAR	INT			S	X			
N2	VAR	INT			S	X			
P1	VAR	ALG	A	L	S	X			
R1	VAR	ALG	A	L	S	X			
S	VAR	ALG		L	S	X			
TDER	PROC			L	S	X			
TIM	VAR	ALG		L	S	X			

T1	VAR	ALG	A	L	S	X			
U1	VAR	ALG	A	L	S	X			
KICKOUT	CONS	LAB			S				225
ERROR IN TDER - - N	CONS	CHAR			S				
0	CONS	INT			S				
1	CONS	INT			S				

3. RESULTS OF THE SECOND RUN OF THE ELIMINATION PROGRAM

```
ALTRAN VERSION 1 LEVEL 9

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

     EXTERNAL INTEGER N1, N2
EXTERNAL LONG ALGEBRAIC LAN, S, TIM
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 IMPLICIT DIFFERENTIATION OF DEPENDENT VARIABLES

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

     RETURN(DER)

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

     END
```

NAME/EXTNAME	USE	TYPE	STRUC	PREC	CLASS	SCOPE	OB	LAY	ADDR
A	VAR	ALG		L		V			
DER	VAR	ALG		L					
DIFF/A9DIFF	PROC	ALG		L	S	X			

```
I                              VAR   INT
J                              VAR   INT
LAN                            VAR   ALG
N1                             VAR   INT          L      S      X
N2                             VAR   INT                 S      X
P1                             VAR   ALG                 S      X
RDER                           PROC         A     L      S      X
R1                             VAR   ALG          L      S      X
S                              VAR   ALG    A     L      S      X
TIM                            VAR   ALG          L      S      X
T1                             VAR   ALG          L      S      X
U1                             VAR   ALG    A     L      S      X
KICKOUT                        VAR   ALG    A     L      S      X
ERROR IN RDER - - N            CONS  LAB                 S
0                              CONS  CHAR                S
1                              CONS  INT                 S                      225
                               CONS  INT                 S
                                                         S

   # INITIALIZATION

   # RDRD

        2

   # TDRD

        1

   # DERMOD

        T(0,1)

   # NUMER

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

   # DENOM

        12*R**2

   # SDER

        0

   # IR

        0

   # IT

        1

# SUB

     ( T(0,0) ,

     T(0,1) ,

     T(0,2) ,
```

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.

```
T(Ø,3) ,
T(Ø,4) ,
T(Ø,5) ,
T(Ø,6) ,
T(Ø,7) ,
T(1,Ø) ,
T(1,1) ,
T(1,2) ,
T(1,3) ,
T(1,4) ,
T(1,5) ,
T(1,6) ,
T(1,7) ,
T(2,Ø) ,
T(2,1) ,
T(2,2) ,
T(2,3) ,
T(2,4) ,
T(2,5) ,
T(2,6) ,
T(2,7) ,
T(3,Ø) ,
T(3,1) ,
T(3,2) ,
T(3,3) ,
T(3,4) ,
T(3,5) ,
T(3,6) ,
T(3,7) ,
T(4,Ø) ,
T(4,1) ,
T(4,2) ,
T(4,3) ,
T(4,4) ,
T(4,5) ,
T(4,6) ,
T(4,7) ,
T(5,Ø) ,
T(5,1) ,
T(5,2) ,
T(5,3) ,
T(5,4) ,
T(5,5) ,
T(5,6) ,
T(5,7) ,
T(6,Ø) ,
T(6,1) ,
T(6,2) ,
T(6,3) ,
T(6,4) ,
T(6,5) ,
T(6,6) ,
T(6,7) ,
T(7,Ø) ,
T(7,1) ,
T(7,2) ,
T(7,3) ,
T(7,4) ,
T(7,5) ,
T(7,6) ,
T(7,7) )
# SUB
    ( T(Ø,1) ,
T(Ø,2) ,
T(Ø,3) ,
T(Ø,4) ,
T(Ø,5) ,
T(Ø,6) ,
T(Ø,7) ,
Ø ,
T(1,1) ,
T(1,2) ,
T(1,3) ,
T(1,4) ,
T(1,5) ,
T(1,6) ,
T(1,7) ,
Ø ,
T(2,1) ,
T(2,2) ,
T(2,3) ,
T(2,4) ,
T(2,5) ,
T(2,6) ,
T(2,7) ,
Ø ,
T(3,1) ,
T(3,2) ,
T(3,3) ,
T(3,4) ,
T(3,5) ,
T(3,6) ,
T(3,7) ,
Ø ,
T(4,1) ,
T(4,2) ,
T(4,3) ,
T(4,4) ,
T(4,5) ,
T(4,6) ,
T(4,7) ,
Ø ,
T(5,1) ,
T(5,2) ,
T(5,3) ,
T(5,4) ,
T(5,5) ,
T(5,6) ,
```

```
        T(5,7) ,
        Ø ,
        T(6,1) ,
        T(6,2) ,
        T(6,3) ,
        T(6,4) ,
        T(6,5) ,
        T(6,6) ,
        T(6,7) ,
        Ø ,
        T(7,1) ,
        T(7,2) ,
        T(7,3) ,
        T(7,4) ,
        T(7,5) ,
        T(7,6) ,
        T(7,7) ,
        Ø )
# DERIV
        ( T(Ø,1) ,
        T(Ø,2) ,
        T(Ø,3) ,
        T(Ø,4) ,
        T(Ø,5) ,
        T(Ø,6) ,
        T(Ø,7) ,
        Ø ,
        T(1,1) ,
        T(1,2) ,
        T(1,3) ,
        T(1,4) ,
        T(1,5) ,
        T(1,6) ,
        T(1,7) ,
        Ø ,
        T(2,1) ,
        T(2,2) ,
        T(2,3) ,
        T(2,4) ,
        T(2,5) ,
        T(2,6) ,
        T(2,7) ,
        Ø ,
        T(3,1) ,
        T(3,2) ,
        T(3,3) ,
        T(3,4) ,
        T(3,5) ,
        T(3,6) ,
        T(3,7) ,
        Ø ,
        T(4,1) ,
        T(4,2) ,
        T(4,3) ,
        T(4,4) ,
        T(4,5) ,
        T(4,6) ,
        T(4,7) ,
        Ø ,
        T(5,1) ,
        T(5,2) ,
        T(5,3) ,
        T(5,4) ,
        T(5,5) ,
        T(5,6) ,
        T(5,7) ,
        Ø ,
        T(6,1) ,
        T(6,2) ,
        T(6,3) ,
        T(6,4) ,
        T(6,5) ,
        T(6,6) ,
        T(6,7) ,
        Ø ,
        T(7,1) ,
        T(7,2) ,
        T(7,3) ,
        T(7,4) ,
        T(7,5) ,
```

```
        T(7,6) ,

        T(7,7) ,

        Ø )

# ISR

        7

# IST

        6

# DELTA

        1.5029262875

# ETIME

        1.5029262875

# MAXIMUM ORDER OF DERIVATIVE TO BE COMPUTED

# ISR

        Ø

# IST

        1

# ISRM

        ( Ø , Ø , Ø , Ø , Ø , Ø , Ø , .NULL. )

# DELTA

        4.1949022499999E-1

# ETIME

        1.9224165125


# DERIV(Ø,Ø)

       - ( 6*DT*R**2*T(0,2) - DR**2*R**2*DIF*T(4,Ø) - 4*DR**2*R*DIF*T(3,0) - 3*DR**2*DIF*T(2,0) - 12*R**2*DIF*T(2,0) - 24*R*DIF*
       T(1,Ø) ) / ( 12*R**2 )

# SPACE DERIVATIVES

# PURE TIME DERIVATIVE
```

```
# IT
     1
# NUMER
     - ( 6*DT*R**2*T(0,2) - DR**2*R**2*DIF*T(4,0) - 4*DR**2*R*DIF*T(3,0) - 3*DR**2*DIF*T(2,0) - 12*R**2*DIF*T(2,0) - 24*R*DIF*
     T(1,0) )
# DENOM
     12*R**2
# DERIV(0,1)
     - ( 6*DT*R**2*T(0,3) - DR**2*R**2*DIF*T(4,1) - 4*DR**2*R*DIF*T(3,1) - 3*DR**2*DIF*T(2,1) - 12*R**2*DIF*T(2,1) - 24*R*DIF*
     T(1,1) ) / ( 12*R**2 )
# SPACE DERIVATIVES
# DERIV
     ( - (399; 6*DT*R**2*T(0,2) - DR**2*R**2*DIF*T(4,0) - 4*DR**2*R*DIF*T(3,0) - 3*DR**2*DIF*T(2,0) - 12*R**2*DIF*T(2,0) - 24*R*DIF*
     T(1,0) ) / ( 12*R**2 ) ,
     - ( 6*DT*R**2*T(0,3) - DR**2*R**2*DIF*T(4,1) - 4*DR**2*R*DIF*T(3,1) - 3*DR**2*DIF*T(2,1) - 12*R**2*DIF*T(2,1) - 24*R*DIF*
     T(1,1) ) / ( 12*R**2 ) ,
     T(0,3) ,
     T(0,4) ,
     T(0,5) ,
     T(0,6) ,
     T(0,7) ,
     0 ,
     T(1,1) ,
     T(1,2) ,
     T(1,3) ,
     T(1,4) ,
     T(1,5) ,
     T(1,6) ,
     T(1,7) ,
     0 ,
     T(2,1) ,
```

```
T(2,2) ,
T(2,3) ,
T(2,4) ,
T(2,5) ,
T(2,6) ,
T(2,7) ,
Ø ,
T(3,1) ,
T(3,2) ,
T(3,3) ,
T(3,4) ,
T(3,5) ,
T(3,6) ,
T(3,7) ,
Ø ,
T(4,1) ,
T(4,2) ,
T(4,3) ,
T(4,4) ,
T(4,5) ,
T(4,6) ,
T(4,7) ,
Ø ,
T(5,1) ,
T(5,2) ,
T(5,3) ,
T(5,4) ,
T(5,5) ,
T(5,6) ,
T(5,7) ,
Ø ,
T(6,1) ,
T(6,2) ,
T(6,3) ,
T(6,4) ,
T(6,5) ,
T(6,6) ,
T(6,7) ,
Ø ,
T(7,1) ,
T(7,2) ,
T(7,3) ,
T(7,4) ,
T(7,5) ,
T(7,6) ,
T(7,7) ,
Ø )
# DELTA
1.5728329540F1
# ETIME
1.7650746Ø525E1
# RORD
2
```

```
 # TORD

       1

 # DERMOD

       T(Ø,1)

 # NUMER

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

 # DENOM

       12*R**2

 # DELTA

       1.85115975

 # FTIME

       1.95Ø19Ø58Ø25E1

 # DFLTA

       2.94938Ø49999E-2

 # ETIME

       1.953139960Ø75E1


*** NORMAL RETURN FROM MAIN PROCEDURE

*** RUN STATISTICS

     19.7ØØ SECONDS ELAPSED
     13107Ø WORDS IN WORKSPACE
         14 DIGITS IN SHORT INTEGERS
         28 DIGITS IN LONG  INTEGERS
          Ø GARBAGE COLLECTIONS
     125165 WORDS OF WORKSPACE NEVER USED

$EJ

# INITIALIZATION

# RORD

       2
```

```
≠ TORD
        1
≠ DERMOD
        T(0.1)
≠ NUMER
      - DIF * ( 6*DT*R**2*T(2,1) + 12*DT*R*T(1,1) - DR**2*R**2*T(4,0) - 4*DR**2*R*T(3,0) - 3*DR**2*T(2,0) - 12*R**2*T(2,0) -
      24*R*T(1,0) )
≠ DENOM
        12*R**2
≠ SDER
        0
≠ IR
        0
≠ IT
        1
≠ SUB
        ( T(0,0) +
        T(0,1) +
        T(0,2) +
        T(0,3) +
        T(0,4) +
        T(0,5) +
        T(0,6) +
        T(0,7) +
        T(1,0) +
        T(1,1) +
        T(1,2) +
        T(1,3) +
```

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.

```
T(1,4) ,
T(1,5) ,
T(1,6) ,
T(1,7) ,
T(2,0) ,
T(2,1) ,
T(2,2) ,
T(2,3) ,
T(2,4) ,
T(2,5) ,
T(2,6) ,
T(2,7) ,
T(3,0) ,
T(3,1) ,
T(3,2) ,
T(3,3) ,
T(3,4) ,
T(3,5) ,
T(3,6) ,
T(3,7) ,
T(4,0) ,
T(4,1) ,
T(4,2) ,
T(4,3) ,
T(4,4) ,
T(4,5) ,
T(4,6) ,

T(4,7) ,
T(5,0) ,
T(5,1) ,
T(5,2) ,
T(5,3) ,
T(5,4) ,
T(5,5) ,
T(5,6) ,
T(5,7) ,
T(6,0) ,
T(6,1) ,
T(6,2) ,
T(6,3) ,
T(6,4) ,
T(6,5) ,
T(6,6) ,
T(6,7) ,
T(7,0) ,
T(7,1) ,
T(7,2) ,
T(7,3) ,
T(7,4) ,
T(7,5) ,
T(7,6) ,
T(7,7) )
≠ SUB
( T(0,1) ,

T(0,2) ,
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) ,
^ ,
T(2,1) ,
T(2,2) ,
T(2,3) ,
T(2,4) ,
T(2,5) ,
T(2,6) ,
T(2,7) ,
^ ,
T(3,1) ,
T(3,2) ,
T(3,3) ,
T(3,4) ,
T(3,5) ,

T(3,6) ,
T(3,7) ,
n ,
T(4,1) ,
T(4,2) ,
T(4,3) ,
T(4,4) ,
T(4,5) ,
T(4,6) ,
T(4,7) ,
n ,
T(5,1) ,
T(5,2) ,
T(5,3) ,
T(5,4) ,
T(5,5) ,
T(5,6) ,
T(5,7) ,
^ ,
T(6,1) ,
T(6,2) ,
T(6,3) ,
T(6,4) ,
T(6,5) ,
T(6,6) ,
T(6,7) ,
n ,
```

```
    T(7,1) ,
    T(7,2) ,
    T(7,3) ,
    T(7,4) ,
    T(7,5) ,
    T(7,6) ,
    T(7,7) ,
    0 )
≠ DERIV
    ( T(0,1) ,
    T(0,2) ,
    T(0,3) ,
    T(0,4) ,
    T(0,5) ,
    T(0,6) ,
    T(0,7) ,
    0 ,
    T(1,1) ,
    T(1,2) ,
    T(1,3) ,
    T(1,4) ,
    T(1,5) ,
    T(1,6) ,
    T(1,7) ,
    0 ,
    T(2,1) ,
    T(2,2) ,
    T(2,3) ,
    T(2,4) ,
    T(2,5) ,
    T(2,6) ,
    T(2,7) ,
    0 ,
    T(3,1) ,
    T(3,2) ,
    T(3,3) ,
    T(3,4) ,
    T(3,5) ,
    T(3,6) ,
    T(3,7) ,
    0 ,
    T(4,1) ,
    T(4,2) ,
    T(4,3) ,
    T(4,4) ,
    T(4,5) ,
    T(4,6) ,
    T(4,7) ,
    0 ,
    T(5,1) ,
    T(5,2) ,
    T(5,3) ,
    T(5,4) ,
    T(5,5) ,
    T(5,6) ,
    T(5,7) ,
    0 ,
    T(6,1) ,
    T(6,2) ,
    T(6,3) ,
    T(6,4) ,
    T(6,5) ,
    T(6,6) ,
    T(6,7) ,
    0 ,
    T(7,1) ,
    T(7,2) ,
    T(7,3) ,
    T(7,4) ,
    T(7,5) ,
    T(7,6) ,
    T(7,7) ,
    0 )
≠ ISR
    7
≠ IST
    6
≠ DELTA
    1.51039504
≠ ETIME
    1.51039504
```

```
# MAXIMUM ORDER OF DERIVATIVE TO BE COMPUTED
# ISR
        2
# IST
        0
# ISRM
        ( 2 , 0 , 0 , 0 , 0 , 0 , 0 , .NULL. )
# DELTA
        4.1290988749 98E-1
# ETIME
        1.9233049275
# DERIV(0,0)
      - DIF * ( 6*DT*R**2*T(2,1) + 12*DT*R*T(1,1) - DR**2*R**2*T(4,0) - 4*DR**2*R*T(3,0) - 3*DR**2*T(2,0) - 12*R**2*T(2,0) -
      24*R*T(1,0) ) / ( 12*R**2 )
# SPACE DERIVATIVES
# IR
      1
# NUMER
      - DIF * ( 6*DT*R**2*T(2,1) + 12*DT*R*T(1,1) - DR**2*R**2*T(4,0) - 4*DR**2*R*T(3,0) - 3*DR**2*T(2,0) - 12*R**2*T(2,0) -
      24*R*T(1,0) )
# DENOM
      12*R**2
# DERIV(1,0)
      - DIF * ( 6*DT*R**3*T(3,1) + 12*DT*R**2*T(2,1) - 12*DT*R*T(1,1) - DR**2*R**3*T(5,0) - 4*DR**2*R**2*T(4,0) + DR**2*R*T(3,0) +
      6*DR**2*T(2,0) - 12*R**3*T(3,0) - 24*R**2*T(2,0) + 24*R*T(1,0) ) / ( 12*R**3 )
# IR
      2
# NUMER
      - DIF * ( 6*DT*R**3*T(3,1) + 12*DT*R**2*T(2,1) - 12*DT*R*T(1,1) - DR**2*R**3*T(5,0) - 4*DR**2*R**2*T(4,0) + DR**2*R*T(3,0) +
      6*DR**2*T(2,0) - 12*R**3*T(3,0) - 24*R**2*T(2,0) + 24*R*T(1,0) )
# DENOM
      12*R**3
```

```
* DERIV(2,0)
      - DIF * ( 6*DT*R**4*T(4,1) + 12*DT*R**3*T(3,1) - 24*DT*R**2*T(2,1) + 24*DT*R*T(1,1) - DR**2*R**4*T(6,0) - 4*DR**2*R**3*T(5,0) +
      5*DR**2*R**2*T(4,0) + 4*DR**2*R*T(3,0) - 18*DR**2*T(2,0) - 12*R**4*T(4,0) - 24*R**3*T(3,0) + 48*R**2*T(2,0) - 48*R*T(1,0) ) /
      ( 12*R**4 )
* DERIV
      ( - DIF * ( 6*DT*R**2*T(2,1) + 12*DT*R*T(1,1) - DR**2*R**2*T(4,0) - 4*DR**2*R*T(3,0) - 3*DR**2*T(2,0) - 12*R**2*T(2,0) -
      24*R*T(1,0) ) / ( 12*R**2 ) ,
      T(0,2) ,
      T(0,3) ,
      T(0,4) ,
      T(0,5) ,
      T(0,6) ,
      T(0,7) ,
      0 ,
      - DIF * (3911 6*DT*R**3*T(3,1) + 12*DT*R**2*T(2,1) - 12*DT*R*T(1,1) - DR**2*R**3*T(5,0) - 4*DR**2*R**2*T(4,0) +
      DR**2*R*T(3,0) + 6*DR**2*T(2,0) - 12*R**3*T(3,0) - 24*R**2*T(2,0) + 24*R*T(1,0) ) / ( 12*R**3 ) ,
      T(1,2) ,
      T(1,3) ,
      T(1,4) ,
      T(1,5) ,
      T(1,6) ,
      T(1,7) ,
      0 ,
      - DIF * ( 6*DT*R**4*T(4,1) + 12*DT*R**3*T(3,1) - 24*DT*R**2*T(2,1) + 24*DT*R*T(1,1) - DR**2*R**4*T(6,0) - 4*DR**2*R**3*T(5,0) +
      5*DR**2*R**2*T(4,0) + 4*DR**2*R*T(3,0) - 18*DR**2*T(2,0) - 12*R**4*T(4,0) - 24*R**3*T(3,0) + 48*R**2*T(2,0) - 48*R*T(1,0) ) /
      T(2,2) ,
      T(2,3) ,
      T(2,4) ,
      T(2,5) ,
      T(2,6) ,
      T(2,7) ,
      0 ,
```

```
T(3,1) ,
T(3,2) ,
T(3,3) ,
T(3,4) ,
T(3,5) ,
T(3,6) ,
T(3,7) ,
0 ,
T(4,1) ,
T(4,2) ,
T(4,3) ,
T(4,4) ,
T(4,5) ,
T(4,6) ,
T(4,7) ,
0 ,
T(5,1) ,
T(5,2) ,
T(5,3) ,
T(5,4) ,
T(5,5) ,
T(5,6) ,
T(5,7) ,
0 ,
T(6,1) ,
T(6,2) ,
T(6,3) ,
T(6,4) ,
T(6,5) ,
T(6,6) ,
T(6,7) ,
0 ,

T(7,1) ,
T(7,2) ,
T(7,3) ,
T(7,4) ,
T(7,5) ,
T(7,6) ,
T(7,7) ,
0 )

# DELTA
        3.63010688425E1
# ETIME
        3.822437377E1
# RORD
        2
# TORD
        1
# DERMOD
        T(0,1)
# NUMER
        - DIF * ( 6*DT*R**2*DIF*T(4,0) + 24*DT*R*DIF*T(3,0) - DR**2*R**2*T(4,0) - 4*DR**2*R*T(3,0) - 3*DR**2*T(2,0) - 12*R**2*T(2,0) -
        24*R*T(1,0) )

# DENOM
        12*R**2
# DELTA
        3.1373692625
# ETIME
        4.13617430325E1
# DELTA
        2.861710500019E-[illegible]
# ETIME
        4.13903601375E1


*** NORMAL RETURN FROM MAIN PROCEDURE
*** RUN STATISTICS
    41.641 SECONDS ELAPSED
    131070 WORDS IN WORKSPACE
        14 DIGITS IN SHORT INTEGERS
        28 DIGITS IN LONG  INTEGERS
         0 GARBAGE COLLECTIONS
    124228 WORDS OF WORKSPACE NEVER USED


  COMPLETE T3LCCAL3RJ
```

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 orderin δt or δr. 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:

```
SECØND = 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

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

after line 129.

The second modification increase 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 DERMØD. After line 10 insert the following:

```
INTEGER NPASS = 1
```

After line 30 insert the following:

```
AG: CONTINUE
```

After line 141 insert the following:

```
REWIND(25)
```

Replace lines 145 through 147 with the following:

```
IF(SDER.EQ.0)GØ TØ BB
```

Replace line 165 with the following:

```
BB: CONTINUE
    WRITE "END OF PASS", NPASS
    IF(NPASS .GT. 10) GØ TØ ST
    NPASS = NPASS + 1
    GØ TØ AG
```

After line 167 insert the following:

```
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:

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

After line 139 insert the following:

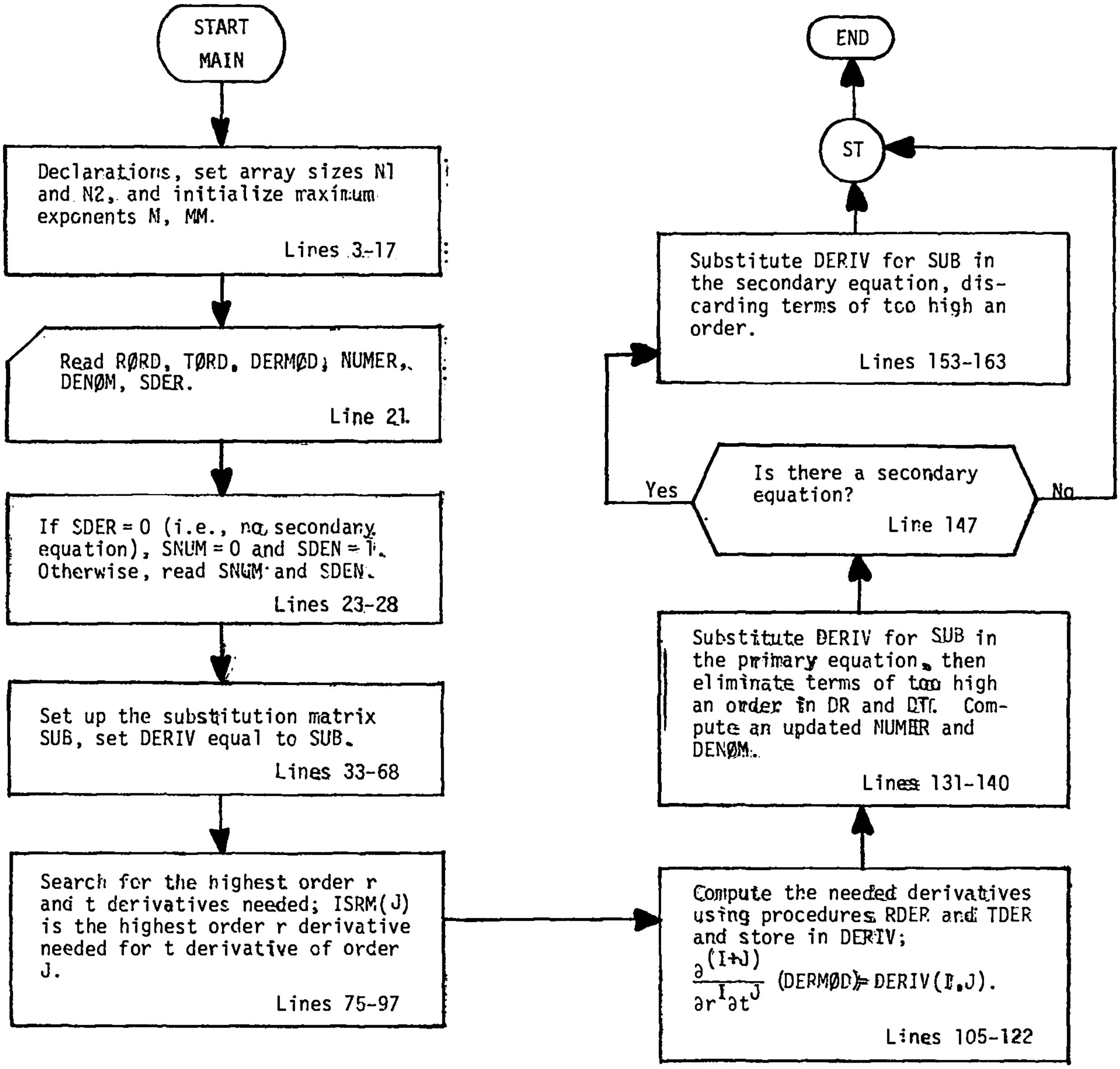
```
DERIV(0,0) = DERIV(0,0)(SUB(0,0) = DERIV(0,0))
```

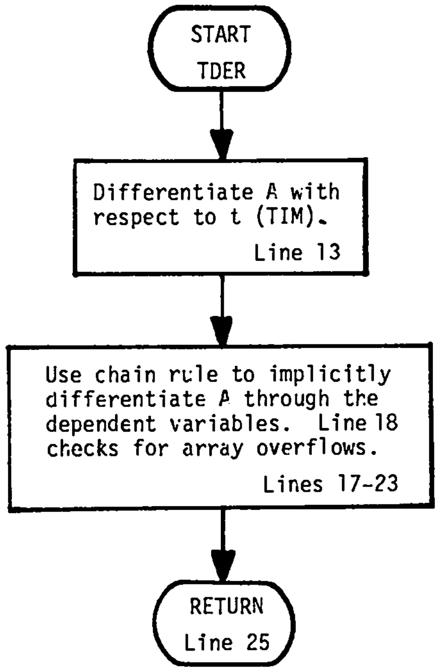
After line 156 insert the following:

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

APPENDIX D

FLOW CHARTS FOR THE TIME DERIVATIVE ELIMINATION PROGRAM





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

☆ U.S. GOVERNMENT PRINTING OFFICE: 1977 – 777-018/95