2

X-641-73-202

PREPRINT

70438

THE NUMERICAL SOLUTION OF ORDINARY DIFFERENTIAL EQUATIONS BY THE TAYLOR SERIES METHOD

ALLAN SILVER
EDWARD SULLIVAN

JULY 1973

GSFC — GODDARD SPACE FLIGHT CENTER —
GREENBELT, MARYLAND

76

The Numerical Solution of Ordinary Differential Equations by the Taylor Series Method

Allan Silver
and
Edward Sullivan

Laboratory for Space Physics
NASA-Goddard Space Flight Center
Greenbelt, Maryland 20771

I. Introduction

The Taylor series method [1] has long been regarded as an efficient procedure for solving systems of ordinary differential equations. Frequently, it is necessary to algebraically manipulate the differential system into an equivalent system. The Taylor coefficients for this modified system may be simply written. However, the required modification is a tedious and error prone task for all but the simplest systems. For this reason, the Taylor series method has often been excluded by numerical analysts from consideration as a general purpose integrator.

In Moore [2], the procedures for recasting a system, which is reducible to a rational form, have been described in detail. Barton, Willers, and Zahar [3] describe techniques for automatic step length prediction, local error estimation, and for choosing the proper number of terms in the series. The authors also include a comparison of the Taylor series method with the fourth order Runge-Kutta method [4] and the Bulirsch-Stoer rational extrapolation method [5]. For a wide range of accuracy, it was found that the Bulirsch-Stoer method generally required five times the amount of computing time, and the factor for the Runge-Kutta method varied from five to one hundred.

A method for the automatic reduction of arbitrary differential systems is described in Barton, Willers, and Zahar [6]. Also presented is a procedure to generate the computer routine which evaluates the Taylor series coefficients of the reduced system. The system reduction and program generation are analogous to the output from a compiler, and the differential equations and initial values are treated as simple language statements that are input to the "compiler". The particular implementation in [6] is an interactive program written for the Atlas 2 computer in Cambridge, England. The target language is Atlas machine language code.

In this paper, an implementation allowing wider usage is presented. The "compiler" is written in PL/I, and the target language is Fortran IV. In Section II, the reduction of a differential system to rational form is described along with the procedures required for automatic numerical integration. In Section III, the Taylor series method is compared with the Bulirsch-Stoer method and with the Nordsieck version of the Adams predictor-corrector method [7] for a number of differential equations.

In Section IV, algorithms using the Taylor series method to find the zeroes of a given differential equation and to evaluate partial derivatives are presented.

Section V discusses the PL/I implementation of the

Barton et al. algorithm. Appendix A contains an annotated listing of the PL/I program which performs the reduction and code generation. Included in Appendix B are listings of the Fortran routines used by the Taylor series method. Finally, Appendix C has a compilation of all the recurrence formulas used to generate the Taylor coefficients for non-rational functions which may appear in the defining system of equations.

II. The Taylor Series Method

Consider the following differential system

$$\frac{d\bar{y}}{dt} = \bar{f}(t,\bar{y}),\ \bar{y}(a)=\bar{\alpha},\ a\leq t\leq b \tag{2.1}$$

where the f_i are rational functions. To apply the Taylor series method to this system, the Taylor coefficients for the expansion about the point $t_o=a$ are computed. The dependent variables y_i are then evaluated at $t=t_1$, with

$$y_i(t_1)=\sum_{j=o}^{\infty} \frac{d^j y_i(t_o)}{dt^j} \frac{(t_1-t_o)^j}{j!} \tag{2.2}$$

The value t_o is now replaced by t_1 and the process repeated until the y_i at the value $t_1=b$ are evaluated

Initially, it may appear that the applicability of the method only to differential systems involving rational functions is a severe limitation on the usefulness of the method. However, functions such as sin, cos, exp, etc., are solutions of rational differential systems. Consequently, a large class of solutions of non-rational differential systems have equivalent representations as solutions of rational differential systems.

To illustrate this point, the function y satisfying the differential equation

$$\frac{dy}{dt} = e^{\sin(y)}+e^{\cos(y)},\ y(0)=0,\ 0 \leq t \leq \frac{\pi}{2} \tag{2.3}$$

may be written as the function u_1 in the system

$$\begin{aligned}
\frac{du_1}{dt} &= u_4+u_5 \\
\frac{du_2}{dt} &= u_3(u_4+u_5) \\
\frac{du_3}{dt} &= -u_2(u_4+u_5) \\
\frac{du_4}{dt} &= u_4u_3(u_4+u_5) \\
\frac{du_5}{dt} &= -u_5u_2(u_4+u_5) \\
u^T(0) &= [0,0,1,1,e] \quad , \qquad 0 \leq t \leq \frac{\pi}{2}
\end{aligned} \tag{2.4}$$

where $u_2=\sin(u_1)$, $u_3=\cos(u_1)$, $u_4=e^{u_2}$, $u_5=e^{u_3}$.

To obtain the canonical system equivalent to (2.4), auxiliary variables are introduced so that each equation in the canonical system represents a single operation of either addition, subtraction, multiplication, or division. Once the canonical system has been generated and the order of evaluation determined, it is a simple task for the computer to produce the formulas for the coefficients.

In the implementation of the method it is important to determine how to best evaluate expressions of the form

$$y_i(t_1) = \sum_{j=o}^{j_{max}} y_i^{(j)}(t_o)(t_1-t_o)^j \tag{2.5}$$

where $y_i^{(j)}(t_o) = \frac{1}{j!} \frac{d^jy_i}{dt^j}(t_o)$.

Also, it is necessary to decide whether j_{max} should be a constant value over the interval of integration or whether j_{max} should be changed at each integration step. Other questions involve the procedure for varying step length and the method of estimating local truncation error.

It was found for a number of test differential equations, including those in Section III, that Horner's method [8] for evaluating (2.5) proved to be the most efficient. Horner's method applied to (2.5) is given by

$$\begin{aligned} &\text{a)}\quad y_i(t_1)=y_i^{(j_{max})}(t_o)\ (t_1-t_o) \\ &\text{b)}\quad y_i(t_1)=y_i^{(j)}(t_o)+(t_1-t_o)y_i(t_1) \\ &\qquad\qquad \text{for } j=j_{max}-1,\ldots,0 \end{aligned} \qquad (2.6)$$

Relative error in the Taylor series solution is controlled by methods analogous to those commonly used for other discrete integrators. The interval length is varied from step to step in order to yield a local relative truncation error less than some preset error bound. The error term resulting from the truncation to j_{max} terms of the Taylor series for $y_i(t_1)$ expanded about t_o is

$$y_i^{(j_{max}+1)}(\xi)(t_1-t_o)^{j_{max}+1} \qquad t_o\leq\xi\leq t_1$$

Thus, a local relative error bound of E requires that the step length $h=t_1-t_o$ satisfy

$$h^{j_{max}+1} \leq E \min_i \left[\left|\frac{N_i}{y_i^{(j_{max}+1)}(t_o)}\right| : N_i = \left\{ \begin{matrix} y_i(t_o) & \text{for} & y_i(t_o) = 0 \\ 1 & \text{for} & y_i(t_o) = 0 \end{matrix} \right\} \right] \qquad (2.7)$$

where i varies over the set of indices for which $y_i^{(j_{max}+1)}(t_o)\neq 0$. If $y_i^{(j_{max}+1)}(t_o)=0$ for all i, then h is set to step to the end of the range.

For the differential equations considered in Section III, the fixed j_{max} which proved to be most efficient was equal to the number of significant decimal digits carried by the computer. This was also found to be true for the equations tested in [6]. For many problems where large functional changes occur over the integration interval, and computation time is critical, a variable j_{max} may produce a very efficient procedure. For a further discussion of numerical integration methods which are optimized by changing the order at each step, see [9] and [10].

III. Comparison

An age old problem confronting numerical analysts is the generation of effective procedures for the comparison of computational methods. It is virtually impossible to include such characteristics as simplicity of method, implementation effort, reliability, and efficiency in a conclusive evaluation. Almost all comparisons of numerical integrators are made solely on the basis of efficiency - usually measured by the number of integration steps or the computer time required to obtain solutions of equal accuracy.

With third generation machines, the concurrent execution of programs, and optimizing compilers, the computer time required for solution is subject to wide fluctuations. These fluctuations are often of the same order of magnitude as the computation times being measured. Also, during the computation, there is an overhead charge incurred when index registers are saved, arguments are passed, and loops are generated.

Many implementations of a numerical algorithm will reduce the overhead at the expense of generality. It is unfair to compare on the basis of computer time, routines which differ in their implementation philosophy, because for the moderate sized problems generally used as test cases the overhead is often a significant portion of the computation time. Consequently, a less general, low overhead method may perform competitively with a less efficiently programmed and more general method.

In comparing the Taylor series method with other methods, significant factors such as the extra storage needed, the difficulty in learning to use the "compiler", and the effort in debugging the Taylor coefficient routine if the "compiler" malfunctions, are difficult to include. Further difficulties result because the Taylor series method integrates a different system of equations than do the usual methods.

To eliminate implementation dependence from the estimate of a method's efficiency, each test problem was integrated to determine the number of derivative evaluations required for solution. This number should be approximately constant for a given method regardless of implementation. For each derivative evaluation routine, the number of machine (360/91) cycles required for one pass through this routine was determined. Table III-A shows the number of cycles required for some typical operations.

TABLE III-A

OPERATION	NUMBER OF CYCLES*
D.P. Load and Store	0
D.P. Add and Subtract	2
D.P. Multiply	3
D.P. Divide	12
F.P. Add, Subtract, Load and Store	1
F.P. Multiply	11
F.P. Divide	37
D.P. Sin and Cos	217
D.P. Exponential	217
D.P. Square Root	133
D.P. Power	400

F.P. = Integer Arithmetic
D.P. = Double Precision Floating Point Arithmetic

*Not including overlap or simultaneous operations.

The number of cycles required to pass arguments from the calling routine was not counted and the overlap or simultaneous execution of operations was not considered.

The methods selected for comparison were the Bulirsch-Stoer rational extrapolation and the Nordsieck version of the Adams predictor-corrector. Both of these methods require a number of functional evaluations to obtain a starting step size which satisfies the accuracy condition. If the initial estimate for the step size is far off, the number of evaluations used in starting could be quite large. For the problems considered here, the number of evaluations required to start were not counted. The test problems are five representative non-trivial differential equations encountered in a computation laboratory:

Problem 1. Bessel Function

$$Y'' = Y(2/t^2 - 1)$$
$$Y(0) = 0$$
$$Y'(0) = 0$$
$$Y''(0) = 2/3$$
$$0 \leq t \leq 25\pi/4$$

Solution: $Y(t) = \sin(t)/t - \cos(t) = tj_o(t)$

Problem 2. Coulomb Function [11]

$Y'' = (-1 + 1/t)Y$

$Y(0) = 0$

$Y'(0) = (\Pi/(e^{\pi}-1))^{1/2}$

$0 \leq t \leq 20$

Solution: $Y(t) = F_o(1/2,t)$

Problem 3. Restricted 3-body problem [5]

$X'' = X + 2Y' - a'(X + g) - a(X - g')$

$Y'' = Y - 2X' - a'Y - aY$

$X(0) = 1.2$

$X'(0) = 0$

$Y(0) = 0$

$Y'(0) = -1.04935750983$

$g = 1/82.45,\ g' = 1 - g$

$a = g/((X - g')^2 + Y^2)^{3/2},\ a' = g'/((X + g)^2 + Y^2)^{3/2}$

$0 \leq t \leq 6.192169331396$

Solution: The given range for t is one period.

Problem 4.

$Y' = -Y + (1 + t)\cos(te^t)$

$Y(0) = 0$

$0 \leq t \leq 5$

Solution: $Y(t) = e^{-t}\sin(te^t)$

Problem 5. A stiff equation [12]

$$X' = -2000X + 1000Y + 1000$$
$$Y' = X - Y$$
$$X(0) = 0$$
$$Y(0) = 0$$
$$1 \leq t \leq 4$$

Solution:

$$X(t)=1+A_1e^{-\lambda_1 t}+A_2e^{-\lambda_2 t}$$
$$Y(t)=1+B_1e^{-\lambda_1 t}+B_2e^{-\lambda_2 t}$$
$$\lambda_1=+2000.5001 \ldots$$
$$\lambda_2=+.49987500 \ldots$$
$$A_1=-.49975000 \ldots$$
$$A_2=-.50024999 \ldots$$
$$B_1=+.00024993746 \ldots$$
$$B_2=-1.0002499 \ldots$$

Table III-B lists some of the results of testing four of the five problems. The column labeled "error" refers to the relative error of the computed solution at the end of the interval. "Cycles" is the number of machine cycles required for each evaluation. "DE" refers to the number of evaluations required to integrate over the given interval. Finally, the column labeled "R" contains the ratio

$$\frac{\text{DE}\left(\begin{matrix}\text{Comparison}\\\text{Method}\end{matrix}\right) \times \text{Cycles}\left(\begin{matrix}\text{Comparison}\\\text{Method}\end{matrix}\right)}{\text{DE}\left(\begin{matrix}\text{Taylor Series}\\\text{Method}\end{matrix}\right) \times \text{Cycles}\left(\begin{matrix}\text{Taylor Series}\\\text{Method}\end{matrix}\right)}$$

TABLE III-B

TAYLOR SERIES METHOD

PROBLEM	ERROR	CYCLES	DE
1	1.2×10^{-10}	789	15
2	4.1×10^{-9}	736	10
3	2.7×10^{-9}	23769	103
4	4.7×10^{-9}	2524	557

NORDSIECK METHOD

PROBLEM	ERROR	CYCLES	DE	R
1	7.9×10^{-10}	22	661	1.2
2	5.5×10^{-9}	19	741	1.9
3	1.0×10^{-9}	349	2340	0.1
4	2.4×10^{-9}	445	18374	5.8

BULIRSCH-STOER RATIONAL EXTRAPOLATION METHOD

PROBLEM	ERROR	CYCLES	DE	R
1	1.7×10^{-10}	22	1288	2.4
2	1.6×10^{-9}	19	790	2.0
3	1.0×10^{-9}	349	5769	0.8
4	1.6×10^{-9}	445	6612	2.1

where the comparison method is either Nordsieck or Bulirsch-Stoer.

The Taylor series method is superior to Nordsieck and Bulirsch-Stoer on Problems 1, 2, and 4 and inferior on Problem 3. Other results, not presented here, show that the Nordsieck and Bulirsch-Stoer methods are very inefficient for Problem 5, while the Taylor series method handles this problem well.

Once the user masters the fairly simple art of setting up the input for program generation, he has an easy means for applying the Taylor series method. If greater efficiency is required, the program may be optimized by the user who has some knowledge of Fortran. On the other hand, if an error occurs, the program may be difficult to debug. Finally, it should be noted that there exists an important class of problems where no Taylor series method program can at present be generated. In general, however, the method is a valuable tool for solving many problems and is certainly worth trying.

IV. Finding Zeros; Partial Differentiation

In this section two algorithms are presented which may be used in conjunction with the Taylor series method. The first algorithm finds the zeros of a function and the second algorithm is used to find the partial derivatives of a function of several variables.

The method used to solve for the zeros of a function is that of series inversion. The relevant theorem is quoted here without proof [13].

Given the power series

$$f=f_o+\sum_{k=1}^{\infty} a_k(t-t_o)^k \tag{4.1}$$

with positive radius of convergence and $a_1 \neq 0$, then there exists a unique power series

$$t=t_o+\sum_{k=1}^{\infty} b_k(f-f_o)^k \tag{4.2}$$

with positive radius of convergence and such that the two series are inverses in sufficiently small neighborhoods of t_o and f_o and $b_1 = 1/a_1$.

To develop a recursion formula for the coefficients b_k in (4.2), solve for $(f-f_o)$ in (4.1) and substitute into (4.2), resulting in

$$t-t_o=\sum_{k=1}^{\infty} b_k \left[\sum_{j=1}^{\infty} a_j(t-t_o)^j\right]^k . \tag{4.3}$$

Letting

$$\sum_{j=k}^{\infty} c_{jk}(t-t_o)^j = \left[\sum_{j=1}^{\infty} a_j (t-t_o)^j\right]^k \quad \text{for } k\geq 1 \tag{4.4}$$

and interchanging the order of summation in (4.3) leads to

$$t-t_o=\sum_{j=1}^{\infty} (t-t_o)^j \sum_{k=1}^{j} c_{jk}b_k. \tag{4.5}$$

Equating powers of $(t-t_o)$ in (4.5) yields

$$\begin{aligned} &b_1=1/c_{11} \\ &b_j=\Big(\sum_{k=1}^{j-1} c_{jk}b_k\Big)/c_{jj} \qquad \text{for } j\geq 2 \ . \end{aligned} \tag{4.6}$$

Rewriting (4.4) in terms of previously computed coefficients, we find

$$\begin{aligned} \sum_{j=k}^{\infty} c_{jk}(t-t_o)^j &= \sum_{j=k-1}^{\infty} c_{j,k-1}(t-t_o)^j \sum_{j=1}^{\infty} a_j (t-t_o)^j \\ &=\sum_{r=1}^{\infty} \sum_{s=k-1}^{\infty} c_{s,k-1}a_r (t-t_o)^{r+s} \\ &\qquad\qquad \text{for } k\geq 2 \ . \end{aligned} \tag{4.7}$$

Substituting j=r+s and interchanging the order of summation yields

$$\sum_{j=k}^{\infty} c_{jk}(t-t_o)^j=\sum_{j=k}^{\infty} (t-t_o)^j \sum_{r=1}^{j-k+1} c_{j-r,k-1}a_r \qquad \text{for } k\geq 2 \ . \tag{4.8}$$

Finally, equating powers of $(t-t_o)$ yields

$$c_{jk}=\sum_{r=1}^{j-k+1} c_{j-r,k-1}\, a_r \quad \text{for } k\geq 2, \ j\geq k \ . \tag{4.9}$$

Also, note that

$$c_{j1}=a_j \qquad \text{for } j\geq 1 \quad . \tag{4.10}$$

The following summarizes the algorithm to find t' such that f(t')=0 when the a_j are known, t_o is given sufficiently close to t', and $f_o=f(t_o)$.

1) $c_{11}=a_1$, $b_1=1/c_{11}$

2) $c_{j1}=a_j$

3) $c_{jk} = \sum\limits_{r=1}^{j-k+1} c_{j-r,k-1}\, a_r$ for $2\leq k\leq j$ (4.11)

4) $b_j= \sum\limits_{k=1}^{j-1} c_{jk}b_k /c_{jj}$

Repeat 2) thru 4) for j=2,3...

5) $t'=t_o+\sum_{k=1}^{\infty} b_k(-f_o)^k$

To illustrate the application of this method, the differential equation for the ninth degree Legendre polynomial was integrated and the zeros of the function computed by series inversion. The results were accurate to the requested precision.

For the computation of the partial derivative of a function of several variables $f(y_1, y_2,...,y_n)$ with respect to y_i, the Taylor series coefficients for the differential system

$$\frac{dy_j}{dt} = \delta_{ij} \qquad j=1,...,n \tag{4.12}$$

are evaluated along with the coefficients for the function f(t). The derivative of f with respect to t may be written as

$$\frac{df}{dt} = \sum_{s=1}^{n} \frac{\partial f}{\partial y_s} \frac{dy_s}{dt} \quad . \tag{4.13}$$

Substituting (4.12) intc(4.13),it is clear that the desired partial derivative is the first Taylor coefficient of f. This procedure may be applied to any number of functions and was used to evaluate the Jacobian of the system given in Problem 3. The results of this computation were as accurate as the input data.

V. PL/I Implementation

The program to generate a Fortran subroutine which evaluates recursive Taylor series coefficients for a system of differential equations has been written in PL/I. The PL/I language was chosen, instead of a string processing language like SNOBOL, because PL/I contains an adequate set of string manipulating functions and because of the similarity between PL/I and Fortran statements. Since the PL/I statements are Fortran-like, changes may be incorporated into the processing program to suit individual needs, with greater facility than might otherwise be the case.

In the implementation of [3], the defining system may contain derivatives of arbitrary order and the differentiation operator may appear on the right hand side of the equations. Without a serious loss in generality, the current implementation is restricted to systems of first order differential equations and the differentiation operator may not appear on the right hand side of the equations.

The program reads in the defining system of equations from the PL/I SYSIN data set, and the equations are checked for balance with respect to parentheses, but no determination is yet made as to whether they represent valid expressions. The program then attempts to generate the Fortran subroutine to evaluate the Taylor series coefficients.

It is not a difficult task to add information about the interval of integration, the accuracy required per integration step, etc. to the input definition of the system of equations. This information may then be edited into appropriate driver routines. The next job steps are compilation and execution of the Fortran program. However, these additions are dependent upon the computer installation and upon individual requirements. The PL/I program is written and annotated so that modifications, similar to the ones just mentioned, are fairly straightforward.

The first input card to the PL/I program is a control card containing the words DIFFERENTIAL EQUATIONS, which may appear anywhere in columns 1 to 72. Sequence numbers are permitted in columns 73-80. The word EQUATIONS may be optionally followed by the letters SP or DP, which is a request to generate a single or double precision routine. The default value is single precision. The nomenclature for the $i^{\underline{th}}$ equation in the differential system is Y(1,I)=f(T,Y), where the first subscript of Y denotes differentiation with respect to the independent variable T. f(T,Y) represents a valid Fortran expression. If it is more convenient to specify the system with a different independent and dependent variable, say R and V, then it is necessary to include (R,V) on the first control card. The differential equations are specified next with a free form format in columns 1-72. Each differential equation is ended with a semi-colon, except for the last equation, which is terminated with a colon. Any equations which are useful in defining the differential system may be included and are order independent. The next card following the differential equations is a control card containing the words INITIAL VALUES. The initial values are then specified in the same manner as the differential equations.

To illustrate a sample input to the processing program, consider Problem 3 above written as four first order equations. The input data required to specify the construction of a double precision routine may have the following form

```
DIFFERENTIAL EQUATIONS DP(T,U)
U(1,1) = U(3); U(1,2) = U(4);
U(1,3) = U(1) + 2.D0*U(4) - AP*(U(1) + G) - A*(U(1) - GP);
U(1,4) = U(2) - 2.D0*U(3) - U(2)*(AP + A);
G = 1.D0/82.45D0; GP = 1.D0 - G;
A = G/DSQRT((U(1)- GP)**2 + U(2)**2)**3;
AP = GP/DSQRT((U(1)+ G)**2 + U(2)**2)**3:
INITIAL VALUES
U(1) = 1.2D0; U(2) = 0.D0;U(3) = 0.D0;U(4) = -1.04935750983:
```

The generated Fortran routine will have the structure

```
SUBROUTINE COEFF (U, ITSMAX)
IMPLICIT REAL*8(A-H,O-Z)
DIMENSION  U(ITSMAX, 1)
Fortran statements necessary to compute the
1 to ITSMAX Taylor coefficients for the equiva-
lent canonical system given U(I), I=1,4.
RETURN
```

```
ENTRY INITAL (T,U,ITSMAX)
┌
│ Initialization of all Taylor coefficients
│ to zero followed by the assignment of the
│ initial values specified as input data.
└
RETURN
END
```

With the above routine and the two Fortran routines listed in Appendix B, edited in the appropriately indicated places, a complete Fortran program for the numerical integration of the sample problem may be developed.

Standard output from the processing program contains listings of the defining differential equations and the generated Fortran routine. Since it is necessary to have some measure of the computer time required per pass through the COEFF routine in order to properly assess the effectiveness of the Taylor series method compared to other popular methods for solving differential equations, an operations count in terms of additions and multiplications is also printed.

The process by which the Fortran routine is generated is very similar to the way a compiler generates assembler language routines. For a complete description of the algorithm see [3]. The differential system is reduced to canonical form, which is the representation of the system in terms of the elementary operations of + - * /. The decomposition is accomplished by the method of bounded context translation [14]. The next step consists of an elimination of redundant operations from the canonical system. After the system has been optimized, a tree search is performed to determine the computational order. For some equations, it may be desirable to examine a number of the intermediate quantities in this process. Coding DEBUG in the PARM field of the processing programs EXEC statement will produce this listing. For the Riccati equation, $y' = y^2 + 3t^2$, the DEBUG listing has the form given in Appendix D.

RMAT is the procedure which performs the decomposition of the differential system. LEVEL denotes the current level of recursive calls to the procedure. The integer K denotes the element in the equation being scanned. TYPE is an integer representing the $K^{\underline{th}}$ element. Table V-A is a listing of the correspondence between the integers and the elements. The E in (C,O,V|E) denotes the print mode that lists the input equation to the procedure. The equation is enclosed by the delimiters #$. C,O,V represents the print mode that lists the $K^{\underline{th}}$ element which is either a constant, operator, or variable. If the $K^{\underline{th}}$ element is a constant, it is replaced by $\#\ell$. The integer ℓ designates the position of this constant in a tabulation of all constants that appear in the differential system. The constant table is

TABLE V-A

TYPE	ELEMENT
-1	constant
0	variable
1	+
2	-
3	*
4	/
5	=
6	(
7	)
8	# (left delimiter)
9	$ (right delimiter)
10	% (function specification)
11	**

listed after the optimized canonical form.

The heading on the right indicates entries into the recurrence matrix, where the canonical system is eventually stored. R represents the row of the matrix, OP the operation, and A(1), A(2) the two possible arguments. A(3) is the name associated with this operation. If there is no external name associated with this operation, the name is generally represented as ?r, where r indicates the row in the matrix storing the result of the operation. The symbol $ in the recurrence matrix is used for the composite operation = . The first differential equation processed is the one for the independent variable, which makes the system autonomous. After the last equation has been processed, the complete recurrence matrix is listed both before and after it has been optimized. As mentioned earlier, the constant table is listed at this point.

The next step involves searching the recurrence matrix to initialize the matrix D described in [3]. The D matrix is used to determine the computational order of evaluation of the coefficients. The dimension of the matrix is the number of rows in the recurrence matrix. Briefly, starting with the result of the operation for a given row in the recurrence matrix, the arguments of the operation are traced backwards thru the recurrence matrix to ascertain

TABLE V-B

%	Function
1	exp
2	$\log_{10}$
3	$\log_{e}$
4	sin
5	cos
6	tan
7	sinh
8	cosh
9	tanh
10	sqrt

their dependence upon other operations. The DMAT ENTRY statement lists the row currently being initialized, and finally the entire D matrix is listed. To aid in an interpretation of the recurrence matrix, a table is constructed showing the correspondence between this matrix and the set of dependent variables in the canonical system. An integer pair, ij, in this table, indicates that the result of the operation in the ith row of the recurrence matrix is the jth dependent variable.

In the reduction to canonical form, special functions which appear may cause their defining differential equations to be appended to the differential system. In this implementation, the special functions are left in the reduced system and the corresponding coefficients for these functions are hard coded in the program generating routine. The symbol %j, where j represents an integer constant, is used to represent functions in the recurrence matrix. Table V-B shows the correspondence between the integers j and the functions they represent.

This completes the description of the intermediate quantities required in the Fortran COEFF routine construction. The listings should be useful in debugging any malfunctioning of the processing program for a given differential system.

REFERENCES

1. Collatz, L. The Numerical Treatment of Differential Equations, Springer-Verlag, Berlin, 1960.

2. Moore, R.E. Interval Analysis, Prentice-Hall, Englewood Cliffs, N.J., 1966, pp. 107-118.

3. Barton, D., Willers, I.M., and Zahar, R.V.M. "The Automatic Solution of Systems of Ordinary Differential Equations by the Method of Taylor Series", The Computer Journal, V. 14, 1971, pp. 243-248.

4. System/360 Scientific Subroutine Package Programmers Manual, IBM, subroutine RKGS.

5. Bulirsch, R., and Stoer, J. "Numerical Treatment of Ordinary Differential Equations by Extrapolation Methods", Numerische Mathematik, V. 8, 1966, pp. 1-13.

6. Barton, D., Willers, I.M., and Zahar, R.V.M. "Taylor Series Methods for Ordinary Differential Equations - An Evaluation", Proc. Math. Software Symposium, Purdue Univ., 1970, pp. 369-390.

7. Eiserike, H., and Silver, A. "A Study of Nordsieck-Type Predictor-Corrector Methods", NASA GSFC X-641-70-199, Revised 1971.

8. Henrici, P. Elements of Numerical Analysis, Wiley, New York, 1964, pp. 51-52.

9. Gear, C.W. "The Automatic Integration of Ordinary Differential Equations", Comm. ACM, V. 14, 1971, pp. 176-190.

10. Estes, R.H., and Lancaster, E.R. "Optimized Computation with Recursive Power Series Integration", NASA GSFC X-643-68-80, 1968.

11. Abramowitz, M. "Coulomb Wave Functions" in Handbook of Mathematical Functions, National Bureau of Standards, 1965, pp. 537-554.

12. Liniger, W., and Odeh, F. "A-Stable, Accurate Averaging of Multistep Methods for Stiff Differential Equations", IBM J. Res. Develop. V. 16, 1972, pp. 335-348.

13. Knopp, K. Infinite Sequences and Series, Dover, New York, 1956, pp. 119-124.

14. Graham, R. "Bounded Context Translation", AFIPS-SJCC, V. 25, 1964, pp. 17-29.

APPENDIX A

```
TAYLOR: PROC(PARM) OPTIONS(MAIN);
/**********************************************************************
 * DRIVER PROCEDURE USED TO CONSTRUCT A FORTRAN SUBROUTINE WHICH      *
 * EVALUATES THE RECURSIVE TAYLOR COEFFICIENTS DERIVED FROM A         *
 * SYSTEM OF ORDINARY DIFFERENTIAL EQUATIONS. FOR A DESCRIPTION OF    *
 * THE ALGORITHM SEE 'THE AUTCMATIC SOLUTION OF SYSTEMS OF            *
 * ORDINARY DIFFERENTIAL EQUATIONS BY THE TAYLOR SERIES METHOD'.      *
 * BY D. BARTON ET AL. CCMPUTER JOURNAL V 14 (1971) PP. 243-248       *
 *********************************************************************/
     DCL
      BFDC ENTRY(BIN FIXED) RETURNS(CHAR(15) VAR).
      BREAKF ENTRY(CHAR(*) VAR.CHAR(*) VAR.CHAR(*) VAR).
      CODE ENTRY(CHAR(*) VAR.BIT(1).BIT(1)).
      COUNT ENTRY(CHAR(*) VAR.CHAR(*) VAR) RETURNS(BIN FIXED).
      SPAN ENTRY(CHAR(*) VAR.CHAR(*) VAR.CHAR(*) VAR.CHAR(*) VAR);
     DCL
      (CS.WS) CHAR(400) VAR EXT.DVRBL CHAR(4) VAR EXT.
      IVRBL CHAR(4) VAR EXT.R( 500.4) CHAR(15) VAR.
      RMAX BIN FIXED INIT(0).D(*.*) BIN FIXED CTL.KFMAX EXT INIT(0).
      ERROR BIN FIXED.CC BIN FIXED INIT(1).IED EXT.
      DEBUG BIT(1) EXT.NEQ EXT INIT(0).KO BIT(1).NSGMA EXT INIT(0).
      FL FILE OUTPUT.SN CHAR(4) VAR.CB CHAR(15) VAR;
     DCL
      (NMUL.NADD.NWTS.NMTL.NATS.NATL) INIT(0) EXT.PARM CHAR(100) VAR.
      LBLA(3) LABEL INIT(LW.LW.LD).LBLB(3) LABEL INIT(LX.LD.LD).
      CST(100) CHAR(25) VAR EXT.IC EXT INIT(1).NEQTNS EXT;
/*                                                                    */
     CALL STINT;
     IF INDEX(PARM.'DEBUG')¬=0 THEN DEBUG='1'B;
/* READ IN THE SYSTEM OF EQUATIONS */
     CST(1)='0.5';
     CALL INPUT(ERROR.CC);
     IF IED=0 THEN CST(1)='0.5'; ELSE CST(1)='0.5D0';
     GO TO LBLA(ERROR);
LW:  NEQTNS=COUNT(CS.'Y(1.')+1;
/* INSERT DIFFERENTIAL EQUATION FOR THE INDEPENDENT VARIABLE TO MAKE
   THE SYSTEM AUTONOMOUS */
     CS='Y(1.'||BFDC(NEQTNS)||')=1.0'||';'||CS||'#';
/* READ IN THE INITIAL VALUES */
     CC=2;
     CALL INPUT(ERROR.CC);
     CS=CS||DVRBL||'('||BFDC(NEQTNS)||')='||IVRBL||';';
     GO TO LBLB(ERROR);
LX:  IPASS=1;
/* FACTOR EACH DIFFERENTIAL EQUATION INTO ELEMENTARY OPERATIONS */
LY:  DO WHILE(SUBSTR(CS.1.1)¬='#');
        NEQ=NEQ+1;
        CALL BREAKF(CS.';'.WS);
        CALL RMAT('#'||WS||'$'.R.RMAX.KO);
```

```
         IF NEQ=1 THEN R(RMAX.4)=IVRBL;
      END;
      IF ¬DEBUG THEN GO TO LZ;
      PUT EDIT('RECURRENCE MATRIX') (SKIP(1).X(60).A);
      DO I=1 TO RMAX;
         PUT EDIT(I.R(I.1).R(I.2).R(I.3).R(I.4)) (SKIP,X(60).
          F(2).X(1).A(2).X(5).3 A(15));
      END;
/* ELIMINATE REDUNDANT OPERATIONS FROM THE RECURRENCE MATRIX */
LZ:   CALL OPTMZE(R.RMAX);
      IF ¬DEBUG THEN GO TO LB;
      PUT EDIT('OPTIMIZED RECURRENCE MATRIX') (SKIP(1).X(60).A);
LA:   DO I=1 TO RMAX;
         PUT EDIT(I.R(I.1).R(I.2).R(I.3).R(I.4)) (SKIP.X(60).F(2).
          X(1).A(2).X(5).3 A(15));
      END;
LB:   IF ¬DEBUG THEN GO TO LBC;
      PUT EDIT('CONSTANT TABLE') (SKIP(2).A);
      DO I=1 TO IC;
         PUT EDIT('#'.I.'='.CST(I)) (COLUMN(MOD(I-1.4)*29+1).
          A.F(2).A.A);
      END;
LBC: ALLOCATE D(RMAX.RMAX);
/* GENERATE THE MATRIX D WHICH IS USED TO DETERMINE THE ORDER IN WHICH
   THE TAYLOR COEFFICIENTS ARE COMPUTED */
      CALL DMAT(R.RMAX.D);
      IF ¬DEBUG THEN GO TO LC;
      PUT EDIT('D MATRIX') (SKIP(2).A);
      PUT EDIT(((I.'.'.J.D(I.J) DO J=1 TO RMAX) DO I=1 TO RMAX))
       (SKIP.8 (F(2).A.F(2).F(4).X(5)));
/* GENERATE THE FORTRAN ROUTINE TO COMPUTE THE TAYLOR COEFFICIENTS */
      CALL COGE(R.RMAX.D.KO);
      IF ¬KO THEN GO TO LD;
LC: FREE D;
      IPASS=IPASS+1;
      IF IPASS>2 THEN GO TO LD;
      CALL CODE('C'.'0'B.'1'B);
      CALL CODE('       ENTRY INITAL('||IVRBL||'.'||DVRBL||'.ITSMAX)'.
       '0'B.'1'B);
      CALL CODE('       DO 2001 ITS=1.ITSMAX'.'0'B.'1'B);
      CALL CODE('       DO 2001 IXV=1.'||BFDC(RMAX).'0'B.'1'B);
      CALL CODE('2001   Y(ITS.IXV=0.0'.'0'B.'1'B);
      DO WHILE(CS¬='');
         CALL BREAKF(CS.';'.WS);
         CALL SPAN(WS.'('.')'.SN);
         CALL BREAKF(WS.')'.CB);
         CALL CODE(DVRBL||'(1.'||SN||')'||WS.'1'B.'0'B);
      END;
      CALL CODE(DVRBL||'(2.'||BFDC(NEQTNS)||')=1.0'.'0'B.'0'B);
```

```
    CALL CODE('        RETURN','0'B,'1'B);
    CALL CODE('        END','0'B,'1'B);
    PUT EDIT('OPERATION CCUNT (OC) FOR ONE PASS THRU THE CUEFF '
     ||'ROUTINE') (SKIP(3),A);
    PUT EDIT('ITSMAX - THE NUMBER OF COEFFICIENTS COMPUTED')
     (SKIP(2),A);
    PUT EDIT('AS      - AN ADDITION OR SUBTRACTION') (SKIP,A);
    PUT EDIT('MD      - A MULTIPLICATION OR DIVISION') (SKIP,A);
    WS='OC = ('||BFDC(NADD)||' + ('||BFDC(NATL)||' + '||BFDC(NATS)||
     '*ITSMAX)*ITSMAX/2)*AS + ('||BFDC(NMUL)||' + ('||BFDC(NMTL)||
     ' + '||BFDC(NMTS)||'*ITSMAX)*ITSMAX/2*MD';
    PUT EDIT(WS) (SKIP(2),A);
LD: END TAYLOR;
```

```
* PROCESS;
 TAYLOR: PROC(PARM) OPTIONS(MAIN);
 /*********************************************************************
  * DRIVER PROCEDURE USED TO CONSTRUCT A FORTRAN SUBROUTINE WHICH      *
  * EVALUATES THE RECURSIVE TAYLOR COEFFICIENTS DERIVED FROM A         *
  * SYSTEM OF ORDINARY DIFFERENTIAL EQUATIONS. FOR A DESCRIPTION OF    *
  * THE ALGORITHM SEE 'THE AUTCMATIC SOLUTION OF SYSTEMS OF            *
  * ORDINARY DIFFERENTIAL EQUATIONS BY THE TAYLOR SERIES METHOD'.      *
  * BY D. BARTON ET AL. CCMPUTER JOURNAL V 14 (1971) PP. 243-248       *
  *********************************************************************/
      CCL
       B=DC ENTRY(BIN FIXED) RETURNS(CHAR(15) VAR).
       BREAKF ENTRY(CHAR(*) VAR.CHAR(*) VAR.CHAR(*) VAR).
       CODE ENTRY(CHAR(*) VAR.BIT(1).BIT(1)).
       COUNT ENTRY(CHAR(*) VAR.CHAR(*) VAR) RETURNS(BIN FIXED).
       SPAN ENTRY(CHAR(*) VAR.CHAR(*) VAR.CHAR(*) VAR.CHAR(*) VAR);
      DCL
       (:S.WS) CHAR(400) VAR EXT.DVRBL CHAR(4) VAR EXT.
       IVRBL CHAR(4) VAR EXT.R( 500.4) CHAR(15) VAR.
       RMAX BIN FIXED INIT(0).D(*.*) BIN FIXED CTL.KFMAX EXT INIT(0).
       ERROR BIN FIXED.CC BIN FIXED INIT(1).IED EXT.
       DEBUG BIT(1) EXT.NEQ EXT INIT(0).KD BIT(1).NSGMA EXT INIT(0).
       FL =ILE OUTPUT.SN CHAR(4) VAR.CB CHAR(15) VAR;
      DCL
       (NMUL.NADD.NMTS.NMTL.NATS.NATL) INIT(0) EXT.PARM CHAR(100) VAR.
       LBLA(3) LABEL INIT(LW.LW.LD).LBLB(3) LABEL INIT(LX.LD.LD).
       CST(100) CHAR(25) VAR EXT.IC EXT INIT(1).NEQTNS EXT;
/*                                                                   */
      CALL STINT;
      IF INDEX(PARM.'DEBUG')¬=0 THEN DEBUG='1'B;
/* READ IN THE SYSTEM OF EQUATIONS */
      CST(1)='0.5';
      CALL INPUT(ERROR.CC);
      IF IED=0 THEN CST(1)='0.5'; ELSE CST(1)='0.5D0';
      GO TO LBLA(ERROR);
LW:   NEQTNS=COUNT(CS.'Y(1.')+1;
/* INSERT DIFFERENTIAL EQUATION FOR THE INDEPENDENT VARIABLE TO MAKE
   THE SYSTEM AUTCNCMOUS */
      CS='Y(1.'||BFDC(NEQTNS)||')=1.0'||';'||CS||'#';
/* READ IN THE INITIAL VALUES */
      CC=2;
      CALL INPUT(ERROR.CC);
      CS=CS||DVRBL||'('||BFDC(NEQTNS)||')='||IVRBL||';';
      GO TO LBLB(ERROR);
LX:   IPASS=1;
/* FACTOR EACH DIFFERENTIAL EQUATION INTO ELEMENTARY OPERATIONS */
LY:   DO WHILE(SUBSTR(CS.1.1)¬='#');
         NEQ=NEC+1;
         CALL BREAKF(CS.';'.WS);
```

```
        CALL RMAT('#'||WS||'$',R,RMAX,KO);
        IF NEQ=1 THEN R(RMAX,4)=IVRBL;
     END;
     IF ¬DEBUG THEN GO TO LZ;
     PUT EDIT('RECURRENCE MATRIX') (SKIP(1),X(60),A);
     DO I=1 TO RMAX;
        PUT EDIT(I,R(I,1),R(I,2),R(I,3),R(I,4)) (SKIP,X(60),
         F(2),X(1),A(2),X(5),3 A(15));
     END;
/* ELIMINATE REDUNDANT OPERATIONS FROM THE RECURRENCE MATRIX */
LZ:  CALL OPTMZE(R,RMAX);
     IF ¬DEBUG THEN GO TO LB;
     PUT EDIT('OPTIMIZED RECURRENCE MATRIX') (SKIP(1),X(60),A);
LA:  DO I=1 TO RMAX;
        PUT EDIT(I,R(I,1),R(I,2),R(I,3),R(I,4)) (SKIP,X(60),F(2),
         X(1),A(2),X(5),3 A(15));
     END;
LB:  IF ¬DEBUG THEN GO TO LBC;
     PUT EDIT('CONSTANT TABLE') (SKIP(2),A);
     DO I=1 TO IC;
        PUT EDIT('#',I,'=',CST(I)) (COLUMN(MOD(I-1,4)*29+1),
         A,F(2),A,A);
     END;
LBC: ALLOCATE D(RMAX,RMAX);
/* GENERATE THE MATRIX D WHICH IS USED TO DETERMINE THE ORDER IN WHICH
   THE TAYLOR COEFFICIENTS ARE COMPUTED */
     CALL DMAT(R,RMAX,D);
     IF ¬DEBUG THEN GO TO LC:
     PUT EDIT('D MATRIX') (SKIP(2),A);
     PUT EDIT(((I,',',J,D(I,J) DO J=1 TO RMAX) DO I=1 TO RMAX))
      (SKIP,8 (F(2),A,F(2),F(4),X(5)));
/* GENERATE THE FORTRAN ROUTINE TO COMPUTE THE TAYLOR COEFFICIENTS */
     CALL CCGE(R,RMAX,D,KO);
     IF ¬KO THEN GC TO LD;
LC: FREE D;
     IPASS=IPASS+1;
     IF IPASS>2 THEN GO TO LD;
     CALL CODE('C','0'B,'1'B);
     CALL CODE('      ENTRY INITAL('||IVRBL||','||DVRBL||',ITSMAX)',
      '0'B,'1'B);
     CALL CODE('      DO 2001 ITS=1,ITSMAX','0'B,'1'B);
     CALL CODE('      DO 2001 IXV=1,'||BFDC(RMAX),'0'B,'1'B);
     CALL CODE('2001  Y(ITS,IXV=0.0','0'B,'1'B);
     DO WHILE(CS¬='');
        CALL BREAKF(CS,':',WS);
        CALL SPAN(WS,'(',')',SN);
        CALL BREAKF(WS,')',CB);
        CALL CODE(DVRBL||'(I,'||SN||')'||WS,'1'B,'0'B);
     END;
     CALL CODE(DVRBL||'(2,'||BFDC(NEQTNS)||')=1.0','0'B,'0'B);
     CALL CODE('        RETURN','0'B,'1'B);
     CALL CODE('        END','0'B,'1'B);
     PUT EDIT('OPERATION COUNT (OC) FOR ONE PASS THRU THE COEFF '
      ||'ROUTINE') (SKIP(3),A);
     PUT EDIT('ITSMAX - THE NUMBER OF COEFFICIENTS COMPUTED')
      (SKIP(2),A);
     PUT EDIT('AS       - AN ADDITION OR SUBTRACTION') (SKIP,A);
     PUT EDIT('MD       - A MULTIPLICATION OR DIVISION') (SKIP,A);
     WS='OC = ('||BFDC(NADD)||' + ('||BFDC(NATL)||' + '||BFDC(NATS)||
      '*ITSMAX)*ITSMAX/2)*AS + ('||BFDC(NMUL)||' + ('||BFDC(NMTL)||
      ' + '||BFDC(NMTS)||'*ITSMAX)*ITSMAX/2*MD';
     PUT EDIT(WS) (SKIP(2),A);
LD: END TAYLOR;
```

```
* PROCESS:
 CODE: PROC(STRING.SUM.CMMNT) RECURSIVE;
 /* PROCEDURE TRANSFORMS THE INPUT 'STRING' INTO FORTRAN CARD IMAGES */
      DCL
       EXTRACT ENTRY(CHAR(*) VAR.CHAR(*) VAR.CHAR(*) VAR).
       LIBF ENTRY(CHAR(*) VAR.BIN FIXED).
       SIGMA ENTRY(CHAR(*) VAR);
      DCL ST CHAR(400) VAR.(SUM.CMMNT) BIT(1).STRING CHAR(*) VAR.
       SN CHAR(7) VAR.FL FILE OUTPUT EXT.IED EXT;
      DCL NSGMA STATIC BIN FIXED EXT.SYSA BIT(1) EXT.IC INIT(1).
        SQN BIN FIXED STATIC INIT(10000). EIS(3) CHAR(8) VAR EXT;
 /*                                                                    */
      IF ¬SYSA|LENGTH(STRING)<11 THEN GO TO LA;
      SN=SUBSTR(STRING.7.5);
      IF SN='SUBRO' | SN='BLOCK' THEN PUT PAGE;
 LA:  IF ¬SUM THEN GO TO LC;
 LB:  CALL EXTRACT(STRING.EIS(3).ST);
      IF ST='' THEN GO TO LC;
      CALL SIGMA(ST);
      GO TO LB;
 LC:  IF ¬CMMNT THEN GO TO LD;
      SQN=SQN+100;
      PUT FILE(FL) EDIT(STRING.'000'.SQN)(SKIP.A.COLUMN(73).A.F(5));
      IF SYSA THEN PUT EDIT(STRING.'000'.SQN)(SKIP.A.COLUMN(73).A.F(5));
      RETURN;
 LD:  SN='       ';
      IF ¬CMMNT THEN CALL LIBF(STRING.IED);
      DO I=1 TO 6 WHILE(VERIFY(SUBSTR(STRING.I.1).'0123456789 ')=0);END;
      IF I¬=1 THEN SN=SUBSTR(SUBSTR(STRING.1.I-1) || SN.1.6);
      STRING=SUBSTR(STRING.I);
      LS=LENGTH(STRING);
      DO I=1 TO LS BY 65;
         ST=SN || SUBSTR(STRING.IC.MIN(65.LS-IC+1));
         SQN=SQN+100;
         PUT FILE(FL) EDIT (ST.'000',SQN) (SKIP.A.COLUMN(73).A.F(5));
         IF SYSA THEN PUT EDIT(ST.'000'.SQN) (SKIP.A.COLUMN(73).A.F(5));
         IC=IC+65;
         IF I=1 THEN SN='     X ';
      END;
 END CODE;
```

```
* PROCESS;
 COGE: PROC(R,RMAX,D,KO);
 /* PROCEDURE GENERATES THE FORTRAN TAYLOR COEFFICIENT ROUTINE */
      DCL
       BFDC ENTRY(BIN FIXED) RETURNS(CHAR(15) VAR),
       BFTC ENTRY(BIN FLOAT(53),BIN FIXED)  RETURNS(CHAR(50) VAR),
       BREAKF ENTRY(CHAR(*) VAR,CHAR(1),CHAR(*) VAR),
       CODE ENTRY(CHAR(*) VAR,BIT(1),BIT(1)),
       COL4 ENTRY RETURNS(BIN FIXED),
       FUDGE ENTRY RETURNS(BIT(1)),
       OP ENTRY(CHAR(1)) RETURNS(BIN FIXED),
       REPLACE ENTRY(CHAR(*) VAR,CHAR (*) VAR,CHAR(* ) VAR,BIT(1)),
       SPAN ENTRY(CHAR(*) VAR,CHAR(*) VAR,CHAR(*) VAR,CHAR(* ) VAR);
      DCL
       R( *,*) CHAR( *) VAR,D(*,*) BIN FIXED,DR(*) BIT(1) CTL,KO BIT(1),
       CTBL(*) CHAR(15) VAR CTL,DVRBL CHAR(4) VAR EXT,(DMAX,RMAX)
       BIN FIXED,(M(*),C(*)) BIN FIXED CTL,FTBL(100,2) BIN FIXED EXT,
       LBL(11) LABEL INIT(PMS,PMS,MPY,DVD,EQL,ERR,
       ERR,ERR,INT,FN,EXP),CST(100) CHAR(25) VAR EXT,CO CHAR(1),IED EXT,
       L(4) LABEL,OC(10) CHAR(1) EXT,WS CHAR( 400) VAR EXT,NEQTNS EXT;
      DCL
       LEPMAX(3) INIT(3,4,2),LEP INIT(0),CH CHAR(1),
       IVFN CHAR(10) VAR,IVRBL CHAR(4) VAR EXT,
       DEBUG BIT(1) EXT,DT(*,*) BIN FIXED CTL,
       C48 CHAR(1) EXT,(NMUL,NADD,NMTS,NMTL,NATS,NATL) EXT,
       IQ BIT(1) INIT('1'B),ADS CHAR(50) VAR,EPLG CHAR(200) VAR;
      DCL
       (ARG(2) CHAR(25) VAR,(CA3,ZERO) BIT(1),(LHS,LHSARG) CHAR(15) VAR,
       (LP(2),RP(2),UO(2)) CHAR(1) VAR,KCMAX,KFMAX,TSS CHAR(5) VAR
       ) EXT;
 /*                                                                    */
      ZERO='1'B;
      EPLG=
       'SUBROUTINE COEFF('||DVRBL||',ITSMAX);IMPLICIT REAL*'||C48||
       '(A-H,O-Z);'||'DIMENSION '||DVRBL||'(ITSMAX,1);1000 DO 2000 '||
       'ITS=2,ITSMAX;ITSM1=ITS-1;ITSP1=ITS+1;FITSM1=FLOAT(ITSM1);'||
       '2000CONTINUE;RETURN;';
      IVFN=DVRBL||'('||BFDC(NEQTNS)||')';
      DMAX=DIM(D,1);
      ALLOCATE DR(DMAX),CTBL(RMAX),M(DMAX),C(DMAX),DT(RMAX,RMAX);
      DR='0'B;  DT=D;
 /* CCRRESPONDENCE TABLE BETWEEN R(I,4) & Y(J) */
      KC=NEQTNS-1;
      DO I=1 TO RMAX;
         IF SUBSTR(R(I,4),1,MIN(LENGTH(R(I,4)),LENGTH(DVRBL)))=DVRBL
          THEN CALL SPAN(R(I,4),'(',')',CTBL(I));
          ELSE DO; KC=KC+1; CTBL(I)=BFDC(KC); END;
      END;
      IF DEBUG
```

```
     THEN DO; PUT EDIT('CORRESPONDENCE BETWEEN RECURRENCE MATRIX ROWS '
               ||'AND THE Y ARRAY') (SKIP(2),A);
              PUT EDIT((I,CTBL(I) DO I=1 TO RMAX)) (SKIP,12 (F(3),X(1),
               A(3),X(4)));
          END;
/* EVALUATION OF THE SET M */
     KM=0;
     DO J=1 TO DMAX;
        DO I=1 TO DMAX;
           IF DR(I) THEN GO TO LB;
           IF DT(I,J)>2 THEN GO TO LC;
LB:     END;
        GO TO LE;
LC:     KM=KM+1; M(KM)=J;
LE:  END;
     IF KM=0 THEN GO TO LEA;
     PUT EDIT('** PROLOG IS NOT CURRENTLY IMPLEMENTED **') (SKIP(2),A);
     KO='0'B;
     RETURN;
LEA: PUT PAGE EDIT('** LISTING OF THE GENERATED FORTRAN ROUTINE **')
      (A); PUT SKIP;
LF:  LEP=LEP+1;
     DO I=1 TO LEPMAX(LEP);
        CALL BREAKF(EPLG,';',WS);
        CALL CODE(WS,'0'B,'0'B);
     END;
     IF EPLG¬='' THEN GO TO LG;
     FREE DR,CTBL,M,C,DT;
     RETURN;
/* EVALUATION OF THE SET C */
LG:  KC=0;
     IF ZERO THEN TSS='(1,'; ELSE TSS='(ITS,';
     DO I=1 TO DMAX;
        IF DR(I) THEN GO TO LI;
        IF ¬ZERO & IO
        THEN DO; IF R(I,1)='$' THEN DO; KCMAX=I; GO TO LK; END;
                  GO TO LI;
             END;
        DO J=1 TO DMAX;
           IF DT(I,J)>0 THEN GO TO LI;
        END;
        KC=KC+1; KCMAX=I; C(KC)=I;
LI:  END;
     IF IO &¬ZERO THEN DO; IO='0'B; GO TO LG; END;
     IF KC>0 THEN GO TO LK;
     PUT EDIT('** EQUATIONS ARE NOT WELL POSED **') (SKIP(2),A);
     KO='0'B;
     PUT EDIT('D MATRIX') (SKIP(2),A);
     PUT EDIT(((I,',',J,D(I,J) DO J=1 TO RMAX) DO I=1 TO RMAX))
```

```
      (SKIP.8 (F(2).A.F(2).F(4).X(5)));
     IF DEBUG
     THEN PUT EDIT(((I.'.'.J.DT(I.J) DO J=1 TO RMAX) DO I=1 TO RMAX))
            (SKIP.8 (F(2).A.F(2).F(4).X(5)));
     RETURN;
LK:  LHSARG=CTBL(COL4(R(KCMAX.4).R.RMAX));
     LHS=DVRBL||TSS||LHSARG||')=';
     CA3=(SUBSTR(R(KCMAX.3).1.1)='#');
     KA=0; UO.LP.RP='';
     IF R(KCMAX.1)='**'
     THEN DO; NOP=11; CO=' '; END;
     ELSE DO; NOP=OP(R(KCMAX.1)); CO=OC(NOP); END;
     DO I=1 TO 2;
        IF R(KCMAX.1)='X' & I=1 THEN GO TO LL;
        IF SUBSTR(R(KCMAX.I+1).1.1)='#'
         THEN DO; KA=KA+I;
                  IF NOP<3 & ¬ZERO
                  THEN DO; ARG(I)=''; IF NOP=1|I=2 THEN CO=''; END;
                  ELSE ARG(I)=CST(SUBSTR(R(KCMAX.I+1).2));
              END;
        ELSE DO; IF R(KCMAX.I+1)=IVFN
                  THEN ARG(I)=CTBL(COL4(IVRBL.R.RMAX));
                  ELSE ARG(I)=CTBL(COL4(R(KCMAX.I+1).R.RMAX));
                  CH=SUBSTR(R(KCMAX.I+1).1.1);
                  IF CH='+' | CH='-'
                  THEN DO; UO(I)=CH; LP(I)='('; RP(I)=')'; END;
              END;
LL:  END;
     IF KA=0 THEN KA=4;
     IF NOP=9 THEN GO TO INT;
     IF NOP=11 THEN GO TO EXP;
     IF NOP=5
     THEN DO; IF ¬ZERO THEN GO TO LT: LHS='': LP(2).RP(2)='': END:
     GO TO L(KA);
L(1):IF NOP=4
     THEN WS='-SIGMA(IXV=2.ITS:'||DVRBL||'(IXV.'||ARG(2)||')*'||DVRBL||
           'ITSP1-IXV.'||LHSARG||')))/'||DVRBL||'(1.'||ARG(2)||')';
      ELSE WS=ARG(1)||CO||LP(2)||UO(2)||DVRBL||TSS||ARG(2)||')'||RP(2);
     GO TO L34;
L(2): WS=UO(1)||DVRBL||TSS||ARG(1)||')'||CO||LP(2)||ARG(2)||RP(2);
     GO TO L34;
L(3): IF ¬ZERO THEN GO TO LT;
     WS=LHS||UO(1)||ARG(1)||CO||LP(2)||UO(2)||ARG(2)||RP(2);
L34: IF NOP<3 THEN NADD=NADD+1;
     IF NOP<5 & NOP>2 THEN NMUL=NMUL+1;
     GO TO LS;
L(4): GO TO LBL(NOP);
FN:  IF ¬FUDGE(CTBL.DR.FTBL.R.RMAX.DT) THEN RETURN: ELSE GO TO LU:
ERR: PUT EDIT('** ILLEGAL OPERATOR IN COGE **') (SKIP(2).A);
```

```
     KO='0'B;
     RETURN;
/* EQUALITY OF TWO SERIES */
EQL:  WS=UO(2)||DVRBL||TSS||ARG(2)||')';
     GO TO LS;
/* ADDITION OR SUBTRACTION OF TWO SERIES */
PMS: WS=UO(1)||DVRBL||TSS||ARG(1)||')'||CO||LP(2)||UO(2)||DVRBL||TSS||
      ARG(2)||')'||RP(2);
     NADD=NADD+1;
     GO TO LS;
/* MULTIPLICATION OF TWO SERIES */
MPY: IF ¬ZERO
     THEN DO; WS='SIGMA(IXV=1.ITS;'||DVRBL||'(IXV,'||ARG(1)||')*'||
               DVRBL||'(ITSP1-IXV,'||ARG(2)||'))';
              NMTS=NMTS+1; NMTL=NMTL+1;
          END;
     ELSE
MDZ:      DO; WS=UO(1)||DVRBL||TSS||ARG(1)||')'||CO ||LP(2)||UO(2)||
               DVRBL||TSS||ARG(2)||')'||RP(2);
              NMUL=NMUL+1;
          END;
     GO TO LS;
/* DIVISION OF ONE SERIES BY ANOTHER */
DVD: IF ZERO THEN GO TO MDZ;
     WS='('||DVRBL||'(ITS,'||ARG(1)||
      ')-SIGMA(IXV=2.ITS;'||DVRBL||'(IXV,'||ARG(2)||')*'||
      ⊃VRBL||'(ITSP1-IXV,'||LHSARG||')))/'||DVRBL||'(1,'||ARG(2)||')';
     NMTS=NMTS+1; NMTL=NMTL-1;
     GO TO LS;
INT: IF ZERO THEN GO TO LT;
     IF SUBSTR(R(KCMAX.3).1.1)='#' THEN  GO TO LT;
     WS=UO(2)||DVRBL||'(ITSM1,'||ARG(2)||')/FITSM1';
     CALL CODE(LHS||WS,'1'B,'0'B);
     GO TO LT;
/* SERIES RAISED TO A POWER */
EXP:IF KA=3
     THEN DO; IF ¬ZERO THEN GO TO LT;
              WS=LP(1)||UO(1)||ARG(1)||RP(1)||'**'||LP(2)||ARG(2)||
               RP(2);
              NADD=NADD+200;
              GO TC LS;
          END;
     IF ZERO
     THEN DO; WS=LP(1)||UO(1)||DVRBL||TSS||ARG(1)||')'||RP(1)||
               '**'||LP(2)||ARG(2)||RP(2);
              NADD=NADD+200;
          END;
     ELSE DO; IF IED=1 THEN CALL REPLACE(ARG(2),'D','E','0'B);
              ADS=BFTC(ARG(2)+1.0,IED);
```

```
                  IF IED=1 THEN CALL REPLACE(ADS,'E','D','0'B);
                  IF VERIFY(SUBSTR(ARG(2),1,1),'+-')=0
                  THEN ADS='('||ADS||')';
                  WS='SIGMA(IXV=1,ITSM1;('||ARG(2)||'-(IXV-1)*'||ADS||
                   '/FITSM1)*'||DVRBL||'(IXV,'||LHSARG||')*'||DVRBL||
                   '(ITSP1-IXV,'||ARG(1)||'))/'||DVRBL||'(1,'||ARG(1)||')';
                  NATS=NATS+3; NATL=NATL-3; NMTS=NMTS+3; NMTL=NMTL-3;
              END;
LS:   CALL CODE(LHS||WS,'1'B,'0'B);
LT:   DR(KCMAX)='1'B;
      DO I=1 TO DMAX;
         IF ¬DR(I) & DT(I,KCMAX)>=0 THEN DT(I,KCMAX)=DT(I,KCMAX)-1;
      END;
LU:   DO I=1 TO DMAX;
         IF ¬DR(I) THEN GO TO LG;
      END;
      ZERO='0'B;
      DR='0'B; DT=0;
      GO TO LF;
END CODE;
```

```
* PROCESS;
 COL4: PROC(CS,RM,RMAX) RETURNS(BIN FIXED);
 /********************************************************************
  * THE PROCEDURE SEARCHES COLUMN 4 OF THE RECURRENCE MATRIX TO FIND *
  * THE ROW NUMBER WHICH CONTAINS CS                                 *
  ********************************************************************/
      DCL CS CHAR(*) VAR,I,RM(*,*) CHAR(*) VAR,RMAX BIN FIXED,
       CT CHAR(15) VAR;
      CT=SUBSTR(CS,2-VERIFY(SUBSTR(CS,1,1),'+-'));
      DO I=1 TO RMAX;
         IF RM(I,4)=CT THEN RETURN(I);
      END;
      PUT EDIT('** ERROR IN INPUT EQUATIONS '''||CS||''' CAN NOT BE '||
       'FOUND IN COLUMN 4 OF THE RECURRENCE MATRIX **') (SKIP(2),A);
      STOP;
 END COL4;
```

```
* PROCESS;
 DMAT: PROC(RM,RMAX,DM);
 /*******************************************************************
  * PROCEDURE CCNSTFUCTS THE #ATRIX D WHICH IS USED TO DETERMINE      *
  * THE ORDER OF TAYLCR COEFFICIENT EVALUATION                        *
  *******************************************************************/
      DCL
       BFDC ENTRY(BIN FIXED) RETURNS(CHAR(15) VAR),
       COL4 ENTRY RETURNS(BIN FIXED),
       S?AN ENTRY(CHAR(*) VAR,CHAR(*) VAR,CHAR(*) VAR,CHAR(*) VAR);
      DCL
       RM(*,*) CHAR( *) VAR,RMAX BIN FIXED,DM(*,*) BIN FIXED,
       R BIN FIXED,L(500,2) BIN FIXED,CH CHAR(1),(IVRBL,DVRBL)
       CHAR(4) VAR EXT,JCH CHAR(2) VAR,VN CHAR(4) VAR,DEBUG BIT(1) EXT;
 /*                                                                 */
      DM=-1;
      DO R=1 TO RMAX;
         IF DEBUG
         THEN DO; IF R=1
                  THEN PUT FOIT('DMAT ENTRY  ',R) (SKIP(2),A,F(2));
                  ELSE PUT EDIT(',',R) (A(1),F(3));
              END;
         N=0; I=1;
         L(1,1)=R; L(1,2)=1+(RM(L(1,1),1)='='|RM(L(1,1),1)='%'
          |RM(L(1,1),1)='$');
 LA:     IF RM(L(I,1),1)='''' THEN N=N+1;
         L1=L(I,1); L2=L(I,2)+1;
         CH=SUBSTR(RM(L1,L2),1,1);
         IF CH='?'
         THEN DO; L(I+1,1)=SUBSTR(RM(L1,L2),2);
                  IC4=COL4(RM(L1,L2),RM,RMAX);
 LC:              IF DM(R,IC4)<N+1 THEN DM(R,IC4)=N+1;
                  L(I+1,2)=1+(RM(L(I+1,1),1)='='|
                   RM(L(I+1,1),1)='X'|RM(L(I+1,1),1)='$');
                  I=I+1;
                  GO TO LA;
              END;
         CALL SPAN(')'||RM(L1,L2),')','(',VN);
         IF VN¬=''
         THEN IF VERIFY(SUBSTR(VN,1,1),'+-')=0 THEN VN=SUBSTR(VN,2);
         IF VN=DVRBL
         THEN DO; IC4=COL4(RM(L1,L2),RM,RMAX);
                  IF DM(R,IC4)<N THEN DM(R,IC4)=N;
                  GO TO LB;
              END;
         IF CH='#' THEN GC TO LB;
         IF RM(L1,L2)=IVREL
         THEN DO; IF DM(R,1)<N THEN DM(R,1)=N;
                  GO TO LB;
```

```
             END;
        ELSE DO; IC4=COL4(RM(L1,L2),RM,RMAX);
                 L(I+1,1)=IC4;
                 GO TO LC;
             END;
LB:     IF L(I,2)=2 | RM(R,1)='=' | RM(R,1)='X'
        THEN DO; I=I-1;
                 IF I=0 THEN GO TO LR; ELSE GO TO LB;
             END;
        L(I,2)=2;
        GO TO LA;
LR:  END;
ND:  END DMAT;
```

```
* PROCESS;
 EXTRACT: PROC(STRING,WORD,EXS);
 /* PROCEDURE EXTRACTS THE SUMMATION OPERATOR FROM THE INPUT STRING   */
      DCL
       BFDC ENTRY(BIN FIXED) RETURNS(CHAR(15) VAR),
       BLNCD ENTRY(CHAR(*) VAR) RETURNS(BIN FIXED),
       REPLACE ENTRY(CHAR(*) VAR,CHAR(*) VAR,CHAR(* ) VAR,BIT(1));
      DCL (STRING,WORD) CHAR(*) VAR,EXS CHAR(*) VAR,
       (NSGMA,NSMTR) STATIC BIN FIXED EXT,EIS(3) CHAR(8) VAR EXT;
 /*                                                                   */
      EXS='';
      CALL EXT(WORD,MRKRA,MRKRB);
      IF MRKRA=0 THEN RETURN;
      DO I=1 TO 3;
         IF EIS(I)=WORD THEN GO TO LA;
         CALL EXT(WORD,MI,LI);
         IF MI<MRKRA THEN RETURN;
 LA:  END;
      EXS=SUBSTR(STRING,MRKRA,MRKRB-MRKRA+1);
      IF WORD=EIS(3)
      THEN CALL REPLACE(STRING,EXS,'SGMA'||BFDC(NSGMA+1),'0'B);
      IF WORD=EIS(2)
      THEN CALL REPLACE(STRING,EXS,'SMTR'||BFDC(NSMTR+1),'0'B);
      IF WORD=EIS(1) THEN CALL REPLACE(STRING,EXS||';','','0'B);
      EXS=SUBSTR(EXS,INDEX(EXS,'(')+1);
      EXS=SUBSTR(EXS,1,LENGTH(EXS)-1)||';';
 /*                                                                   */
 EXT: PROC(W,MR,IL);
      DCL W CHAR(*) VAR,MR,IL;
      MR,IL=0;
 LD:  IF MR+1<LENGTH(STRING)
      THEN IX=INDEX(SUBSTR(STRING,MR+1),W); ELSE IX=0;
      IF IX=0
      THEN
 LDA:       DO; MR=0; RETURN; END;
      ELSE MR=IX+MR;
      IL=MR-1;
 LE:  IX=INDEX(SUBSTR(STRING,IL+1),')');
      IF IX=0 THEN GO TO LDA; ELSE IL=IL+IX;
      IF BLNCD(SUBSTR(STRING,MR,IL-MR+1))¬=0 THEN GO TO LE;
      IF INDEX(SUBSTR(STRING,MR,IL),'=')=0 THEN GO TO LD;
      IF MR=1 | VERIFY(SUBSTR(STRING,MR-1,1),'+-*/)(=')=0 THEN RETURN;
      GO TO LD;
 END EXT;
 END EXTRACT;
```

```
* PROCESS;
 FUDGE: PROC(CTBL.DR.FTBL.R.RMAX.DT) RETURNS(BIT(1));
 /*****************************************************************
  * PROCEDURE CONSTRUCTS THE RECURSIVE TAYLOR COEFFICIENTS FOR      *
  * FUNCTIONS SUCH AS EXP, SIN, COS, TAN ETC WHICH MAY APPEAR       *
  * IN THE DIFFERENTIAL EQUATIONS                                   *
  *****************************************************************/
      DCL
       CODE ENTRY(CHAR(*) VAR.BIT(1).BIT(1)).
       COL4 ENTRY RETURNS(BIN FIXED);
      DCL
       DR(*) BIT(1).CTBL(*) CHAR(*) VAR.R(*.*) CHAR(*) VAR.FTBL(*.*)
       BIN FIXED.FN(13.2) CHAR(6) VAR EXT.RMAX BIN FIXED.
       (ARG(2) CHAR(25) VAR, (CA3.ZERO) BIT(1), (LHS.LHSARG) CHAR(15)
       VAR, (LP(2).RP(2).UO(2)) CHAR(1) VAR, KCMAX, KFMAX, TSS CHAR(5)
       VAR, (IVRBL.DVRBL) CHAR(4) VAR  ) EXT.VNLHS CHAR(15) VAR,
       LHSA(3) CHAR(15) VAR, FNL( 9) LABEL INIT(EXP.L10.LN,
       SCT.SCT.SCT.HSCT.HSCT.HSCT).PW CHAR(1) VAR,
       (NMUL.NADD.NMTS.NMTL.NATS.NATL) EXT.IED EXT.DT(*.*) BIN FIXED.
       ALPHA(2) CHAR(20) VAR INIT('4.342945E-1'.'4.34294481903251 8D-1');
 /*                                                                */
      DO KF=1 TO KFMAX;
         IF KCMAX<=FTBL(KF.2) & KCMAX>=FTBL(KF.1) THEN GO TO LA;
      END;
      PUT EDIT('** FUNCTION NUMBER '.KCMAX.' IS NOT IN TABLE **')
       (SKIP(2).A.F(2).A);
      RETURN('0'B);
 LA:  I=0;
      DO K=FTBL(KF.1) TO FTBL(KF.2);
         DR(K)='1'B;
         DO J=1 TO RMAX;
            IF ¬DR(J) & DT(J.K)>=0 THEN DT(J.K)=DT(J.K)-1;
         END;
         I=I+1;
         LHSA(I)=CTBL(COL4(R(K.4).R.RMAX));
      END;
      NF=R(KCMAX.2);
      IF ¬ZERO THEN GO TO LB;
      I=0;
      DO K=FTBL(KF.1) TO FTBL(KF.2);
         I=I+1;
         KX=R(K.2);
         VNLHS=DVRBL||'(1.'||LHSA(I)||')=';
         IF CA3
         THEN CALL CODE(VNLHS||FN(KX.2)||'('||UO(2)||ARG(2)||'))'.'0'B.
                   '0'B);
         ELSE CALL CODE(VNLHS||FN(KX.2)||'('||UO(2)||DVRBL||'(1.'||
                   ARG(2)||'))'.'0'B.'0'B);
      END;
```

```
     GO TO LZ;
LB:  IF CA3 THEN RETURN('1'B); ELSE GO TO FNL(NF);
/* SERIES COEFFICIENTS FOR EXPONENTIAL FUNCTION */
EXP: CALL CODE(LHS||'SIGMA(IXV=2.ITS;'||'(IXV-1)*'||
      DVRBL||'IXV,'||ARG(2)||')*'||DVRBL||'(ITSP1-IXV,'||
      LHSA(1)||')/FITSM1'. '1'B,'0'B);
     NMUL=NMUL+1; NMTS=NMTS+2; NMTL=NMTL-1;
     GO TO LZ;
/* SERIES COEFFICIENTS FOR LCG BASE 10 FUNCTION */
L10: CALL CODE('IF (ITS.EQ.2) '||LHS||ALPHA(IED+1)||'*'||DVRBL ||'(2,'
      ||ARG(2)||')/'||DVRBL||'(1,'||ARG(2)||')'.'0'B.'0'B);
     CALL CODE('IF (ITS.GT.2) '||LHS||'('||ALPHA(IED+1)||'*'
      ||DVRBL||TSS||ARG(2)||')-SIGMA(IXV=2.ITSM1;(IXV-1)*'||
      DVR3L||'(ITSP1-IXV,'||ARG(2)||')*'||DVRBL||'(IXV,'||
      LHSA(1)||'))/FITSM1)/' ||DVRBL||'(1,'||ARG(2)||')'.'1'B.'0'B);
     NMUL=NMUL+3; NADD=NADD+1; NMTS=NMTS+2; NMTL=NMTL-2;
     GO TO LZ;
/* SERIES COEFFICIENTS FOR LCG BASE E FUNCTION */
LN:  CALL CODE('IF (ITS.EQ.2) '||LHS||DVRBL||TSS||ARG(2)||')/'
      ||DVRBL||'(1,'||ARG(2)||')','0'B.'0'B);
     CALL CODE('IF (ITS.GT.2) '||LHS||'('||DVRBL||TSS||ARG(2)
      ||')-'||'SIGMA(IXV=2.ITSM1;(IXV-1)*'||DVRBL||'(ITSP1-IXV,'||
      ARG(2)||')*'||DVRBL||'(IXV,'||LHSARG||'))/FITSM1/'||DVRBL||'(1,'
      ||ARG(2)||')/'||DVRBL||'(1,'||ARG(2)||')'.'1'B.'0'B);
     NMUL=NMUL+2; NADD=NADD+1; NMTS=NMTS+2; NMTL=NMTL-2;
     GO TO LZ;
/* SERIES COEFFICIENTS FOR THE SIN. COS. TAN FUNCTIONS */
SCT: PM='-';
     GO TO LC;
/* SERIES COEFFICIENTS FOR THE HYPERBOLIC SINH. CJSH. TANH FUNCTIONS */
HSCT: PM='';
LC: CALL CODE(DVRBL||TSS||LHSA(1)||')=SIGMA(IXV=2.ITS;(IXV-1)*'||
      DVR3L||'(IXV,'||ARG(2)||')*'||DVRBL||'(ITSP1-IXV,'||LHSA(2)
      ||'))/FITSM1' .'1'B.'0'B);
     CALL CODE(DVRBL||TSS||LHSA(2)||')='||PM||'SIGMA(IXV=2.ITS;'||
      '(IXV-1)*'||DVRBL||'IXV,'||ARG(2)||')*'||DVRBL||'(ITSP1-IXV,'
      ||LHSA(1)||'))/FITSM1' .'1'B.'0'B);
     CALL CODE(DVRBL||TSS||LHSA(3)||')=('||DVRBL||TSS||LHSA(1)||
      ')-SIGMA(IXV=2.ITS;'||DVRBL||'(IXV,'||LHSA(2)||')*'||DVRBL||
      '(ITSP1-IXV,'||LHSA(3)||')))/'||DVRBL||'(1,'||LHSA(2)||')'.
      '1'B.'0'B);
     NMUL=NMUL+3; NADD=NADD+1; NMTS=NMTS+6; NMTL=NMTL-6;
LZ: RETURN('1'B);
END FUDGE;
```

```
* PROCESS;
 INPUT: PROC(ERROR,CC);
 /*********************************************************************
  * PROCEDURE READS THE DEFINING SYSTEM OF DIFFERENTIAL EQUATIONS     *
  * FROM THE SYSIN DATA SET, AND CHECKS TO SEE THAT THE EQUATIONS     *
  * ARE BALANCED WITH RESPECT TO PARENTHESES                          *
  ********************************************************************/
      DCL
       BLNCD ENTRY(CHAR(*) VAR) RETURNS(BIN FIXED),
       COUNT ENTRY(CHAR(*) VAR,CHAR(1)) RETURNS(BIN FIXED),
       DELETE ENTRY(CHAR(*) VAR,CHAR(1)),
       SPAN ENTRY(CHAR(*) VAR,CHAR(*) VAR,CHAR(*) VAR,CHAR(*) VAR);
      DCL
       (ERROR,CC) BIN FIXED, LINE CHAR(80) VAR,
       CW(2) CHAR(25) VAR INIT('DIFFERENTIALEQUATIONS',
       'INITIALVALUES'),  CS CHAR(400) VAR EXT, WS CHAR(400) VAR EXT,
       IED EXT, C48 CHAR(1) EXT,(IVRBL,DVRBL) CHAR(4) VAR EXT;
 /*                                                                  */
      ON ENDFILE(SYSIN)
      BEGIN; PUT EDIT('** EOF READING SYSIN **') (SKIP(2),A);
             GO TO LPA;
      END;
      IF CC=1
      THEN DO; MRKR=0;
               PUT EDIT('** TAYLOR SERIES PROGRAM JAN. 1973'||
                 ' VERSION - LISTING OF INPUT EQUATIONS **') (A);
               ERROR=1;
           END;
      ELSE MRKR=INDEX(CS,'#');
      PUT SKIP;
 LP:  GET EDIT(LINE) (A(80));
      PUT EDIT(LINE) (SKIP,COLUMN(4),A(80));
      LINE=SUBSTR(LINE,1,72);
      CALL DELETE(LINE,' ');
      IF CC¬=0
      THEN DO; IF INDEX(LINE,CW(CC))¬=0 THEN GO TO LQ;
               PUT EDIT('** THE FOLLOWING CONTROL CARD IS INVALID **',
                LINE) (SKIP(2),A,SKIP,A);
 LPA:          ERROR=3;
               RETURN;
 LQ:           IF CC=1
               THEN DO; IF INDEX(LINE,'DP')¬=0
                        THEN DO; IED=1; C48='8'; END;
                        ELSE DO; IED=0; C48='4'; END;
                        IF INDEX(LINE,'(')¬=0
                        THEN DO; CALL SPAN(LINE,'(',',',IVRBL);
                                 CALL SPAN(LINE,',',')',DVRBL);
                             END;
                    END;
```

```
              CC=0; PUT SKIP; GO TO LP;
          END;
     CS=CS||LINE;
     IF INDEX(LINE,':')=0  THEN GO TO LP;
     PUT SKIP;
     CS=SUBSTR(CS,1,LENGTH(CS)-1)||';';
     DO WHILE(MRKR<LENGTH(CS));
        MC=INDEX(SUBSTR(CS,MRKR+1),';');
        WS=SUBSTR(CS,MRKR+1,MC-1);
        MRKR=MRKR+MC;
        IF BLNCD(WS)¬=0
        THEN DO; PUT EDIT('** THE FOLLOWING EXPRESSION HAS AN '||
                  'INCORRECT PAIRING OF PARENTHESES **',WS)
                  (SKIP(2),A,SKIP,A);
                 ERROR=2;
             END;
        IF COUNT(WS,'=')¬=1
        THEN DO; PUT EDIT('** SYNTAX ERROR IN THE FOLLOWING EXPRESSION '
                  ||' **',WS) (SKIP(2),A,SKIP,A);
                 ERROR=2;
             END;
     END;
ND: END INPUT;
```

```
* PROCESS;
 OP: PROC(CH) RETURNS(BIN FIXED);
 /*********************************************************************
  * PROCEDURE COMPARES THE INPUT CHARACTER CH WITH THE CHARACTERS     *
  * +, -, *, /, =, (, ), #, $, %                                      *
  *********************************************************************/
      DCL CH CHAR(1),I,OC(10) CHAR(1) EXT;
      DO I=1 TO 10;
         IF CH=OC(I) THEN RETURN(I);
      END;
      RETURN(0);
 END CP;
```

```
* PROCESS;
 OPTMZE: PROC(M,RMAX);
 /* PROCEDURE ELIMINATES REDUNDANT OPERATIONS FROM THE R MATRIX */
       DCL
        BFDC ENTRY(BIN FIXED) RETURNS(CHAR(15) VAR),
        (I,J,K,R,RMAX) BIN FIXED,IMP BIT(1),M(*,*) CHAR(15) VAR,
        IX(2,2) INIT(2,3,3,2);
 LA:   R=0; IMP='0'B;
 LAB: R=R+1; I=R+1;
 LB:   DO WHILE(I<=RMAX);
          IF M(R,1)¬=M(I,1) THEN GO TO LD;
          IF M(R,1)='+'|M(R,1)='-'|M(R,1)='*'|M(R,1)='/'
          THEN JMAX=2; ELSE JMAX=1;
          DO J=1 TO JMAX;
             DO K=1 TO 2;
                IF M(R,K+1)¬=M(I,IX(K,J)) THEN GO TO LC;
             END;
             IMP='1'B;
             CALL RAD(I,R);
             GO TO LE;
 LC:      END;
 LD:      I=I+1;
 LE:   END;
       IF R<RMAX-1 THEN GO TO LAB;
       IF IMP THEN GO TO LA;
 /*                                                                       */
 RAD: PROC(I,J);
 /**********************************************************************
  * PROCEDURE DELETES ROW I, AND REPLACES REFERENCES TO ROW I WITH     *
  * ROW J IN THE RECURRENCE MATRIX                                     *
  *********************************************************************/
       DCL I,J,K,L,ROW BIN FIXED;
       K=I+1;
       DO WHILE(K<=RMAX);
          M(K-1,1)=M(K,1);
          DO L=2 TO 4;
             IF SUBSTR(M(K,L),1,1)¬='?'
             THEN DO; M(K-1,L)=M(K,L); GO TO LF; END;
             ROW=SUBSTR(M(K,L),2);
             IF ROW=I THEN ROW=J; ELSE IF ROW>I THEN ROW=ROW-1;
             M(K-1,L)='?' || BFDC(ROW);
 LF:      END;
          K=K+1;
       END;
       RMAX=RMAX-1;
 END RAD;
 END OPTMZE;
```

```
* PROCESS;
 RMAT: PROC(CS,R,RMAX,KO) RECURSIVE;
 /*******************************************************************
  * FACTORIZATION OF THE DIFFERENTIAL SYSTEM INTO CANONICAL FORM     *
  * USING THE ALGORITHM DESCRIBED IN 'BOUNDED CONTEXT TRANSLATION',  *
  * BY R. GRAHAM, AFIPS-SJCC V 25 (1964), P. 21                      *
  *******************************************************************/
      DCL
       OP ENTRY(CHAR(1)) RETURNS(BIN FIXED),
       BFDC ENTRY(BIN FIXED) RETURNS(CHAR(15) VAR),
       REPLACE ENTRY(CHAR(*) VAR,CHAR(*) VAR,CHAR(*) VAR,BIT(1)),
       SFNL ENTRY(CHAR(*) VAR) RETURNS(BIN FIXED),
       SPAN ENTRY(CHAR(*) VAR,CHAR(*) VAR,CHAR(*) VAR,CHAR(*) VAR);
      DCL P(11) BIN FIXED STATIC          INIT(6,6,7,7,5,3,4,1,2,8,8),
       S( 500) CHAR(15) VAR INIT(( 500) ''), L( 250) CHAR(15) VAR,
       TYPE BIN FIXED,R(*,*) CHAR(15) VAR,CS CHAR(*) VAR,
       WT CHAR(400) VAR,FN CHAR(4) VAR,FTBL(100,2) BIN FIXED EXT,
       WS CHAR(400) VAR,RMAX BIN FIXED,DVRBL CHAR(4) VAR EXT;
      DCL CST(100) CHAR(25) VAR EXT,IC EXT,KO BIT(1),IED EXT,
       NEST BIN FIXED STATIC INIT(0),DEBUG BIT(1) EXT,NEQ EXT,
       PMD CHAR(36) VAR EXT;
 /*                                                                 */
      KO='1'B;
      NEST=NEST+1;
      IF NEST=1 & NEQ=1 & DEBUG
      THEN DO;
               PUT PAGE;
               PUT EDIT('RMAT ENTRY',' LEVEL K TYPE (C, O, V | E)',
                ' R OP','A(1)','A(2)','A(3)') (SKIP(2),A,SKIP(1),A
                X(33),A,X(5),A,3 (X(11),A));
      END;
      DO I=1 TO LENGTH(CS)/45+1;
         PUT EDIT(SUBSTR(CS,1+(I-1)*45,MIN(45,LENGTH(CS)-(I-1)*45)))
          (SKIP,X(15),A);
      END;
      J=1;
      K=2;
      MRKR=2;
      S(1)=SUBSTR(CS,1,1);
      TYPE=8;
      L(1)=S(1);
 L1:  IF S(K)='' THEN CALL CHECK;
      IF DEBUG THEN PUT EDIT(NEST,K,TYPE,S(K)) (SKIP,3 F(4),X(3),A);
      IF TYPE<1 THEN GO TO L2;
      IF TYPE¬=6 THEN GO TO LA;
 L2:  J=J+1;
      L(J)=S(K);
 L3:  K=K+1;
      IF K>DIM(S,1)
```

```
     THEN DO; PUT EDIT('** OVERFLOW IN S TABLE **') (SKIP(2),A);
             KO='0'B; RETURN;
         END;
     GO TO L1;
LA:  IF P(OP(L(J-1)))<P(TYPE) THEN GO TO LB;
     RMAX=RMAX+1;
     IF L(J-1)¬='=' THEN GO TO LAB;
     IF LENGTH(L(J-2))<LENGTH(DVRBL) | SUBSTR(L(J-2),1,LENGTH(DVRBL))
      ¬=DVRBL THEN GO TO LAA;
     CALL SPAN(L(J-2),'.',')',FN);
     R(RMAX,1)='$';
     R(RMAX,2)=DVRBL||'('||FN||')';
     R(RMAX,3)=L(J);
     R(RMAX,4)=R(RMAX,2);
     GO TO LAC;
LAA: IF SUBSTR(L(J),1,1)¬='?' | SUBSTR(L(J),2)¬=BFDC(RMAX-1)
      THEN GO TO LAB;
     RMAX=RMAX-1;
     R(RMAX,4)=L(J-2);
     GO TO LAC;
LAB: IF L(J-1)='**' & INDEX(L(J),'#')¬=0
     THEN DO; IF CST(SUBSTR(L(J),2))¬='2' THEN GO TO LAZ;
             R(RMAX,1)='*';
             R(RMAX,2)=L(J-2);
             R(RMAX,3)=L(J-2);
         END;
     ELSE
LAZ:     DO; R(RMAX,1)=L(J-1);
             R(RMAX,2)=L(J-2);
             R(RMAX,3)=L(J);
         END;
     IF L(J-1)='='
     THEN R(RMAX,4)=R(RMAX,2); ELSE R(RMAX,4)='?'||BFDC(RMAX);
LAC: IF DEBUG THEN PUT EDIT(RMAX,(R(RMAX,M) DO M=1 TO 4))
      (SKIP,X(60),F(2),X(1),A(2),X(5),4 A(15));
     J=J-2;
     L(J)='?' || BFDC(RMAX);
     GO TO LA;
LB:  IF TYPE=7 THEN DO; L(J-1)=L(J); J=J-1; GO TO L3; END;
     IF TYPE¬=9 THEN GO TO L2;
     IF K=2 THEN CS=S(2); ELSE CS='?'||BFDC(RMAX);
     NEST=NEST-1;
/*                                                                   */
CHECK: PROC RECURSIVE;
/*******************************************************************
 * PROCEDURE SCANS A SEQUENCE OF CHARACTERS BEGINNING WITH MRKR TO  *
 * DETERMINE WHETHER THEY SPECIFY A CONSTANT, VARIABLE, OR OPERATOR *
 *******************************************************************/
     DCL
```

```
      BLNCD ENTRY(CHAR(*) VAR) RETURNS(BIN FIXED),
      FLIP BIT(1), CH CHAR(1),STRING CHAR(15) VAR INIT(''),
      BREAKB ENTRY(CHAR(*) VAR,CHAR(*) VAR,CHAR(*) VAR),
      BREAKF ENTRY(CHAR(*) VAR,CHAR(*) VAR,CHAR(*) VAR);
/*                                                                      */
     CH=SUBSTR(CS,MRKR,1);
     S(K)=CH;
     MRKR=MRKR+1;
     IO=OP(CH);
     IF IO=0 THEN GO TO LG;
     IF (TYPE=5|TYPE=6)&IO<3
     THEN IF VERIFY(SUBSTR(CS,MRKR,1),'.0123456789')¬=0
          THEN GO TO LGA; ELSE GO TO LJ;
     IF TYPE>0&TYPE¬=6&TYPE¬=7&IO¬=6&IO¬=7
     THEN GO TC LJ;
     IF IO=9 THEN DO; TYPE=IO; GO TO ND; END;
     CH=SUBSTR(CS,MRKR,1);
     NOP=OP(CH);
     IF NOP=3 & IO=3 THEN DO; S(K)=S(K) || CH; MRKR=MRKR+1; END;
     IF LENGTH(S(K))=2 THEN TYPE   =11; ELSE TYPE   =IO;
     GO TO ND;
LG:  IF VERIFY(S(K),'ABCDEFGHIJKLMNOPQRSTUVWXYZ?')¬=0 THEN GO TO LJ;
/* PROCESS A VARIABLE */
LGA: FLIP='0'B; TYPE=0; WS=S(K);
LH:  CH=SUBSTR(CS,MRKR,1);
     NOP=OP(CH);
     IF NOP=0 THEN GO TO LI;
     IF NOP¬=6 & NOP¬=7 & ¬FLIP THEN GO TO LIA;
     IF BLNCD(WS||CH)<0 THEN GO TO LIA;
     FLIP='1'B;
LI:  WS=WS||CH;
     MRKR=MRKR+1;
     IF ¬FLIP | BLNCD(WS)¬=0 THEN GO TO LH;
     CALL SPAN(')'||WS,')','(',WT);
     IF WT=DVRBL THEN
LIA:  DO; S(K)=WS; GO TO ND; END;
     LFN=SFNL(WS);
     IF LFN=0 THEN GO TO LIA;
     CALL BREAKF(WS,'(',WT);
     CALL BREAKB(WS,')',WT);
     IF VERIFY(WS,PMD) =0 THEN GO TO LIB;
     WS='#TEMPN='||WS||'$';
     CALL RMAT(WS,R,RMAX,KO);
     IF¬KO THEN RETURN;
     R(RMAX,4)='?'||BFDC(RMAX);
LIB: CALL SFNC(LFN,WS,R,RMAX,S(K),FTBL,KO);
     IF ¬KO THEN RETURN;
     GO TO ND;
/* PROCESS A CONSTANT */
```

```
LJ:    TYPE=-1;
     IF IC=100
     THEN DO; PUT EDIT('** OVERFLOW IN CONSTANT TABLE **') (SKIP(2),A);
              KO='0'B; RETURN;
          END;
     IC=IC+1;
     CST(IC)=S(K);
LJA: CH=SUBSTR(CS.MRKR,1);
     IF VERIFY(CH,'.0123456789ED+-')¬=0 THEN GO TO LK;
     IF (CH='+'|CH='-')&VERIFY(SUBSTR(CST(IC).LENGTH(CST(IC)),1),
      'ED')¬=0 THEN GO TO LK;
     IF (CH='E' | CH='D') & VERIFY(STRING,'.1234567890')¬=0
      THEN GO TO LK;
     CST(IC)=CST(IC)||CH;
     STRING=STRING||CH;
     MRKR=MRKR+1;
     GO TO LJA;
LK:  IF IC¬=1
     THEN DO; IF IED=1 & INDEX(CST(IC).'.')¬=0
              THEN DO; CALL REPLACE(CST(IC).'E','D','0'B);
                       IF INDEX(CST(IC).'D')=0
                       THEN CST(IC)=CST(IC)||'D0';
                   END;
              DO I=1 TO IC-1;
                 IF CST(I)¬=CST(IC) THEN GO TO LJB;
                 S(K)='#'||BFDC(I);
                 IC=IC-1;
                 RETURN;
LJB:          END;
          END;
     S(K)='#' || BFDC(IC);
     RETURN;
ND: END CHECK;
END RMAT;
```

```
* PROCESS;
 SFNC: PROC(LFN.WS.R,RMAX.S.FTBL.KO);
 /********************************************************************
  * PROCEDURE INSERTS THE PROPER ENTRIES IN THE R MATRIX FOR AN       *
  * ALLOWABLE FUNCTION REFERENCE                                      *
  *******************************************************************/
      DCL BFDC ENTRY(BIN FIXED) RETURNS(CHAR(15) VAR).RMAX BIN FIXED.
       R(*,*) CHAR( *) VAR,S CHAR(*) VAR,WS CHAR(*) VAR,
       CLFN CHAR(15) VAR.FTBL(*,*) BIN FIXED.KFMAX EXT.DEBUG BIT(1) EXT.
       KFL(10) INIT(1.2.3.4,4.4,7.7.7.10).
       KFU(10) INIT(1.2.3.6.6.6,9.9.9.10);
 /*                                                                 */
      CLFN=BFDC(LFN);
      DO I=1 TO RMAX;
         IF  R(I.1)¬='X' |  R(I.2)¬=CLFN |  R(I.3)¬=WS
          THEN GO TO LA;
          ELSE DO; S='?' || BFDC(I); RETURN; END;
 LA:  END;
      DO K=KFL(LFN) TO KFU(LFN);
         RMAX=RMAX+1;
         IF K=10
         THEN DO; R(RMAX.1)='**';
                  R(RMAX.2)=WS;
                  R(RMAX.3)='#1';
                  GO TO LB;
              END;
         R(RMAX.1)='X';
         R(RMAX.2)=BFDC(K);
         R(RMAX.3)=WS;
 LB:     R(RMAX.4)='?' || BFDC(RMAX);
         IF DEBUG
         THEN PUT EDIT(RMAX.(R(RMAX.M) DO N=1 TO 4)) (SKIP.X(60).F(2).
                X(1).A(2),X(5).4 A(15));
         IF K=LFN THEN S='?'||BFDC(RMAX);
      END;
      KFMAX=KFMAX+1;
      IF KFMAX>DIM(FTBL.1)
      THEN DO; PUT EDIT('** OVERFLOW IN FUNCTION TABLE **')
                 (SKIP(2).A);
               KO='0'B; RETURN;
           END;
      FTBL(KFMAX.2)=RMAX;
      FTBL(KFMAX.1)=RMAX-KFU(LFN)+KFL(LFN);
 END SFNC;
```

```
* PROCESS;                                                                  00000100
 SFNL: PROC(WS) RETURNS(BIN FIXED);                                         00000200
 /* PROCEDURE DETERMINES WHETHER WS IS AN ALLOWABLE FUNCTION */             00000300
      DCL                                                                   00000400
       WS CHAR(*) VAR,NF EXT,F CHAR(6) VAR,FN(10,2) CHAR(6) VAR EXT;        00000500
      F=SUBSTR(WS,1,INDEX(WS,'(')-1);                                       00000600
      DO I=1 TO NF; DO J=1 TO 2;                                            00000700
         IF F=FN(I,J) THEN RETURN(I);                                       00000800
      END; END;                                                             00000900
      RETURN(0);                                                            00001000
 END SFNL;                                                                  00001100
```

```
* PROCESS;
 SIGMA: PROC(STRING) RECURSIVE;
 /* PROCEDURE CONSTRUCTS A FORTRAN DO LOOP FOR A SUMMATION OPERATOR  */
      DCL
       BFDC ENTRY(BIN FIXED) RETURNS(CHAR(15) VAR),
       BREAKF ENTRY(CHAR(*) VAR,CHAR(1),CHAR(*) VAR),
       CODE ENTRY(CHAR(*) VAR,BIT(1),BIT(1)),
       EXTRACT ENTRY(CHAR(*) VAR,CHAR(*) VAR,CHAR(*) VAR);
      DCL EXA CHAR(400) VAR,NSGMA BIN FIXED STATIC EXT,
       (STN,SMN) CHAR(5) VAR,EIS(3) CHAR(8) VAR EXT,STRING CHAR(*) VAR;
 /*                                                                   */
      NSGMA=NSGMA+1;
      SMN=BFDC(NSGMA);
      STN=BFDC(1000+NSGMA);
      CALL BREAKF(STRING,';',EXA);
      CALL CODE('SGMA'||SMN||'=0.0','0'B,'0'B);
      CALL CODE('DO '||STN||' '||EXA,'0'B,'0'B);
      CALL CODE(STN||'SGMA'||SMN||'=SGMA'||SMN||'+('||SUBSTR(STRING,1,
       LENGTH(STRING)-1)||')','1'B,'0'B);
 END SIGMA;
```

```
* PROCESS;
 STINT: PROC;
 /* PROCEDURE INITIALIZES ALL EXTERNAL BIT AND CHARACTER STRING VBL'S */
      DCL(
       NF INIT(10),(IVRBL,DVRBL) CHAR(4) VAR,EIS(3) CHAR(8) VAR,
       OC(10) CHAR(1),FN(13,2) CHAR(6) VAR,SYSA BIT(1),PMD CHAR(36) VAR
       ) EXT;
 /*                                                                    */
      SYSA='1'B;
      PMD='ABCDEFGHIJKLMNOPQRSTUVWXYZ1234567890';
      IVRBL='T'; DVRBL='Y';
      EIS(1)='EQUATION'; EIS(2)='INTEGRAL'; EIS(3)='SIGMA';
      OC( 1)='+'; OC( 2)='-'; OC( 3)='*'; OC( 4)='/'; OC( 5)='=';
      OC( 6)='('; OC( 7)=')'; OC( 8)='#'; OC( 9)='$'; OC(10)='X';
      FN( 1,1)=    'EXP'; FN( 1,2)=   'DEXP';
      FN( 2,1)='ALOG10'; FN( 2,2)='DLOG10';
      FN( 3,1)=  'ALOG'; FN( 3,2)=   'DLOG';
      FN( 4,1)=    'SIN'; FN( 4,2)=   'DSIN';
      FN( 5,1)=    'COS'; FN( 5,2)=   'DCOS';
      FN( 6,1)=    'TAN'; FN( 6,2)=   'DTAN';
      FN( 7,1)=  'SINH'; FN( 7,2)= 'DSINH';
      FN( 8,1)=  'CCSH'; FN( 8,2)= 'DCOSH';
      FN( 9,1)=  'TANH'; FN( 9,2)= 'DTANH';
      FN(10,1)=  'SQRT'; FN(10,2)= 'DSQRT';
 END STINT;
```

```
* PROCESS;
 BFDC: PROC(N) RETURNS(CHAR(15) VAR);
 /*********************************************************************
  * BIN FIXED NUMBER N IS CONVERTED TO CHARACTER STRING WITH ALL      *
  * BLANKS RESULTING FROM THE CONVERSION DELETED                      *
  *********************************************************************/
      DCL DELETE ENTRY(CHAR(*) VAR,CHAR(1)),CS CHAR(15) VAR;
      CS=CHAR(N);
      CALL DELETE(CS,' ');
      RETURN(CS);
 END BFDC;
```

```
* PROCESS;
 BFTC: PROC(BF,IED) RETURNS(CHAR(50) VAR);
 /*********************************************************************
  * BIN FLOAT(53) NUMBER BF IS CONVERTED TO CHARACTER STRING WITH     *
  * ALL BLANKS RESULTING FROM THE CONVERSION DELETED                  *
  *********************************************************************/
      DCL
       DELETE ENTRY(CHAR(*) VAR,CHAR(1)),
       REPLACE ENTRY(CHAR(*) VAR,CHAR(*) VAR,CHAR(*) VAR,BIT(1)),
       BF BIN FLOAT(53),CH CHAR(50) VAR,SF BIN FLOAT;
      IF IED=0 THEN DO; SF=BF; CH=SF; END; ELSE CH=BF;
      CALL DELETE(CH,' ');
      IF IED=1 THEN CALL REPLACE(CH,'E','D','0'B);
      RETURN(CH);
 END BFTC;
```

```
* PROCESS;
 BLNCD: PROC(CS) RETURNS(BIN FIXED);
 /*********************************************************************
  * THE DIFFERENCE BETWEEN THE NUMBER OF RIGHT AND LEFT PARENTHESES   *
  * IN THE CHARACTER STRING CS IS RETURNED                            *
  *********************************************************************/
      DCL CS CHAR(*) VAR, C CHAR(1) VAR, (IL,IR) INIT(0), I;
      DO I=1 TO LENGTH(CS);
         C=SUBSTR(CS,I,1);
         IF C='(' THEN IL=IL+1;
         IF C=')' THEN IR=IR+1;
      END;
      RETURN (IL-IR);
 END BLNCD;
```

```
* PROCESS;
 BREAKB: PROC(CS,C,BS);
 /*****************************************************************
  * THE CHARACTER STRING TO THE RIGHT OF THE CHARACTER VARIABLE C IN *
  * THE CHARACTER STRING CS IS PLACED IN THE CHARACTER STRING BS     *
  * C || BS IS DELETED FROM CS                                       *
  * EXAMPLE:  CS = 'ABCDE'                                           *
  *           CALL BREAKB(CS,'C',BS)                                 *
  * GENERATES CS = 'AB', BS = 'DE'                                   *
  *******************************************************************/
      DCL CS CHAR(*) VAR, C CHAR(1), BS CHAR(*) VAR,IX;
      BS='';
      DO I=LENGTH(CS) TO 1 BY -1;
         IF SUBSTR(CS,I,1)¬=C THEN GO TO LA;
         IF I+1<LENGTH(CS) THEN BS=SUBSTR(CS,I+1);
         IF I>1 THEN CS=SUBSTR(CS,1,I-1); ELSE CS='';
         GO TO LB;
 LA:  END;
 LB:  END BREAKB;
```

```
* PROCESS;
 BREAKF: PROC(CS,C,BS);
 /*****************************************************************
  * THE CHARACTER STRING TO THE LEFT OF THE CHARACTER VARIABLE C IN  *
  * THE CHARACTER STRING CS IS PLACED IN THE CHARACTER STRING BS     *
  *  BS || C IS DELETED FROM CS                                      *
  * EXAMPLE:  CS = 'ABCDE'                                           *
  *           CALL BREAKF(CS,'C',BS)                                 *
  * GENERATES CS = 'DE', BS = 'AB'                                   *
  *******************************************************************/
      DCL CS CHAR(*) VAR, C CHAR(1), BS CHAR(*) VAR,IX;
      BS='';
      IX=INDEX(CS,C)-1;
      IF IX=-1 THEN RETURN;
      IF IX<1 THEN GO TO LA;
      BS=SUBSTR(CS,1,IX);
 LA:  IF IX+2<=LENGTH(CS) THEN CS=SUBSTR(CS,IX+2); ELSE CS='';
 LB:  END BREAKF;
```

```
* PROCESS;
 COUNT: PROC(CS,C) RETURNS(BIN FIXED);
 /*******************************************************************
  * THE NUMBER OF TIMES THE CHARACTER STRING C APPEARS IN THE        *
  * CHARACTER STRING CS IS RETURNED                                  *
  *******************************************************************/
      DCL CS CHAR(*) VAR, C CHAR(*) VAR,(MRKR,IC) INIT(0);
 LA:  IXC= INDEX(SUBSTR(CS,MRKR+1),C);
      MRKR=MRKR+IXC;
      IF  IXC¬=0 THEN IC=IC+1;
      IF  IXC=0 | MRKR=LENGTH(CS) THEN RETURN(IC);
      GO TO LA;
 END COUNT;
```

```
* PROCESS;
 DELETE: PROC(CS,C);
 /*******************************************************************
  * EXAMPLE:  CS = 'ABCDE'                 CS = 'A''BCD''E'          *
  *           CALL DELETE(CS,'C')          CALL DELETE(CS,'''')      *
  * GENERATES CS = 'ABDE'                  CS = 'AE'                 *
  *******************************************************************/
     .DCL C CHAR(1), CS CHAR(*) VAR, I INIT(2),
          FLIP BIN FIXED INIT(1),SYM CHAR(1) INIT('''');
      CS=' ' || CS || ' ';
      IF C=SYM THEN GO TO LH;
      DO WHILE(I<LENGTH(CS));
         IF  SUBSTR(CS,I,1)=C
          THEN CS=SUBSTR(CS,1,I-1) || SUBSTR(CS,I+1);
          ELSE I=I+1;
      END;
      GO TO LIA;
 LH:  DO WHILE(I<LENGTH(CS));
         IF SUBSTR(CS,I,1)=SYM
          THEN DO; FLIP=-FLIP; GO TO LHA; END;
         IF FLIP<0 THEN
 LHA:     DO; CS=SUBSTR(CS,1,I-1) || SUBSTR(CS,I+1); GO TO LI; END;
          ELSE I=I+1;
 LI:  END;
 LIA: CS=SUBSTR(CS,2,LENGTH(CS)-2);
 END DELETE;
```

```
* PROCESS;
 LIBF: PROC(CS,IED);
 /********************************************************************
  * IF IED = 0 ALL DOUBLE PRECISION FORTRAN LIBRARY FUNCTIONS IN THE *
  * CHARACTER STRING CS ARE REPLACED WITH SINGLE PRECISION FUNCTIONS *
  * AND VICE VERSA IF IED = 1                                      *
  ********************************************************************/
      DCL IED,     CS CHAR(*) VAR,NF INIT(25),
       REPLACE ENTRY(CHAR(*) VAR,CHAR(*) VAR,CHAR(*) VAR,BIT(1));
      DCL FN(25,2) CHAR(6) VAR INIT(
          'EXP',   'DEXP', 'ALOG10', 'DLOG10',  'ARSIN', 'DARSIN',
        'ARCOS', 'DARCOS',   'ATAN',  'DATAN',  'ATAN2', 'DATAN2',
          'SIN',   'DSIN',    'COS',   'DCOS',    'TAN',   'DTAN',
        'COTAN', 'DCOTAN',   'SQRT',  'DSQRT',   'TANH',  'DTANH',
         'SINH',  'DSINH',   'COSH',  'DCOSH',    'ERF',   'DERF',
         'ERFC',  'DERFC',  'GAMMA', 'DGAMMA', 'ALGAMA', 'DLGAMA',
         'AMOD',   'DMOD',    'ABS',   'DABS',  'AMAX1',  'DMAX1',
        'AMIN1',  'DMIN1',  'FLOAT', 'DFLOAT',   'SIGN',  'DSIGN',
         'ALOG',   'DLOG'  );
      DO I=1 TO NF;
         CALL REPLACE(CS,FN(I,2-IED),FN(I,1+IED),'1'B);
      END;
 END LIBF;
```

```
* PROCESS;
 REPLACE: PROC(CS,SA,SB,OP);
 /*****************************************************************
  * ALL APPEARANCES OF THE STRING SA IN THE CHARACTER STRING CS ARE  *
  * REPLACED WITH THE STRING SB IF THE BIT(1) VARIABLE OP = '0'B     *
  * IF OP = '1'B ONLY THOSE OCCURENCES OF SA BOUNDED ON THE RIGHT BY *
  * THE NULL CHARACTER OR +-/*)( ARE REPLACED WITH SB                *
  * EXAMPLE:     CS = 'A+BA'                                         *
  *              CALL REPLACE(CS,'A','Z','0'B)                       *
  * GENERATES    CS = 'Z+BZ'                                         *
  * WHILE        CALL REPLACE(CS,'A','Z','1'B)                       *
  * GENERATES    CS = 'Z+BA'                                         *
  * RESTRICTION: THE CHARACTER % MAY NOT APPEAR IN CS, SA, OR SB     *
  ******************************************************************/
      DCL
       CHECK ENTRY(BIN FIXED) RETURNS(BIT(1)),
       VERIFY ENTRY(CHAR(*) VAR,CHAR(*) VAR) RETURNS(BIN FIXED),
       (CS,SA,SB) CHAR(*) VAR,OP BIT(1),OR BIT(1) INIT('0'B);
 /*                                                                */
      LSA=LENGTH(SA);
 LA:  MRKSA=INDEX(CS,SA);
      IF MRKSA=0 THEN GO TO LB;
      IF ¬QR THEN DO; CS=' '||CS||' '; MRKSA=MRKSA+1; END;
      QR='1'B;
      CS=SUBSTR(CS,1,MRKSA-1) || '%' || SUBSTR(CS,MRKSA+LSA);
      GO TO LA;
 LB:  IF ¬QR THEN RETURN;
      MRKSA=INDEX(CS,'%');
      IF MRKSA=0 THEN GO TO LC;
      IF ¬OP | CHECK(MRKSA)
       THEN CS=SUBSTR(CS,1,MRKSA-1) || SB || SUBSTR(CS,MRKSA+1);
       ELSE CS=SUBSTR(CS,1,MRKSA-1) || SA || SUBSTR(CS,MRKSA+1);
      GO TO LB;
 LC:  CS=SUBSTR(CS,2,LENGTH(CS)-2);
 /*                                                                */
 CHECK: PROC(IX) RETURNS(BIT(1));
      DCL IX BIN FIXED,(IL,IR) BIN FIXED INT INIT(0);
      IF IX-1>0 THEN IL=VERIFY(SUBSTR(CS,IX-1,1),'=(+-*/)');
      IF IX+1<=LENGTH(CS)
       THEN IR=VERIFY(SUBSTR(CS,IX+1,1),'=(+-*/)');
      RETURN(IL+IR=0);
 END CHECK;
 END REPLACE;
```

```
* PROCESS;
 SPAN: PROC(CS,SA,SB,SC);
 /*********************************************************************
  * THE CHARACTER STRING IN CS SPANNED BY SA AND SB IS RETURNED IN SC*
  * EXAMPLE:   CS = 'ABCDEF'                                          *
  *            CALL SPAN(CS,'AB','F',SC)                              *
  * GENERATES  SC = 'CDE'                                             *
  *********************************************************************/
      DCL (CS,SA,SB,SC) CHAR(*) VAR;
      SC='';
      ISTART=INDEX(CS,SA)+LENGTH(SA);
      ISTOP=INDEX(SUBSTR(CS,ISTART),SB);
      IF ISTART>0&ISTOP>1 THEN SC=SUBSTR(CS,ISTART,ISTOP-1);
 END SPAN;
```

```
* PROCESS;
 TRIM: PROC(STRING);
 /* ALL TRAILING BLANKS ARE DELETED FROM THE STRING CS */
          DCL STRING CHAR(*) VAR;
          DO I=LENGTH(STRING) TO 1 BY -1 WHILE (SUBSTR(STRING,I,1)=' ');
          END;
          IF I=0 THEN STRING=''; ELSE STRING=SUBSTR(STRING,1,I);
 END TRIM;
```

APPENDIX B

```
      SUBROUTINE TAYLOR (TI,YI,TF,YF,NCF,NEQ,INIT,EPS,RANGE)
      IMPLICIT REAL*8 (A-H,O-Z)
      COMMON /COUNT/ KEV,KIS,IER
      COMMON /PAR/ HMIN
      REAL*8 YI(NCF,1),YF(NCF,1)
      LOGICAL*4 INIT
      IF (.NOT.INIT)  GO TO 11
C**** INITIALIZATION
      IER = 0
      KEV = 0
      KIS = 0
      NCL = NCF - 1
      EX = 1.D0/NCL
      NEQ1 = NEQ + 1
      CR = DSIGN(1.D0,RANGE)
      CALL INITAL(TI,YF,NCF)
      CALL COEFF (YF,NCF)
      DO 18  K = 1,NEQ
      DO 18  J = 1,NCF
   18 YI(J,K) = YF(J,K)
C**** COMPUTE R(H)
   11 R = 0.D0
      DO 10  K = 1,NEQ
      RJ = YI(NCF,K)
      IF (YI(1,K).NE.0.D0)  RJ = RJ/YI(1,K)
   10 R = DMAX1(R,DABS(RJ))
C**** COMPUTE H
      IF (R.NE.0.D0)  GO TO 23
      H = RANGE
      GO TO 24
   23 H = CR*(EPS/R)**EX
   24 IF (DABS(H).GT.HMIN)  GO TO 17
C**** ERROR: H TOO SMALL
      IER = - 2
      RETURN
```

```
C**** TAKE A STEP
   17 DO 12  K = 1,NEQ
   12 YF(1,K) = YI(NCL,K)
      DO 13  JJ = 2,NCL
      J = NCF - JJ
      DO 13  K = 1,N EQ
   13 YF(1,K) = YI(J,K) + H*YF(1,K)
      TF = TI + H
      YF(1,NEQ1) = TF
      CALL COEFF(YF,NCF)
      KEV = KEV + 1
      RANGE = RANGE - H
      KIS = KIS + 1
      RETURN
      END
```

```
      SUBROUTINE INTERP (TI,YI,NCF,NEQ,TW,W)
      IMPLICIT REAL*8 (A-H,O-Z)
      REAL*8 YI(NCF,1),W(1)
      H = TW - TI
      NCL = NCF - 1
      DO 21  K = 1,NEQ
   21 W(K) = YI(NCL,K)
      DO 22  JJ = 2,NCL
      J = NCF - JJ
      DO 22  K = 1,NEQ
   22 W(K) = YI(J,K) + H*W(K)
      RETURN
      END
      SUBROUTINE ZERO(T,F,A,TZ,B,C,N)
      IMPLICIT REAL*8(A-H,O-Z)
      DIMENSION A(N),B(N),C(N,N)
      C(1,1)=A(1)
      YP=-F
      B(1)=1.D0/C(1,1)
      TZ=T+B(1)*YP
      DO 500 K=2,N
      C(K,1)=A(K)
      DO 300 J=2,K
      C(K,J)=0.D0
      IMAX=K-J+1
      DO 300 I=1,IMAX
  300 C(K,J)=C(K,J)+C(K-I,J-1)*A(I)
      B(K)=0.0D0
      IMAX=K-1
      DO 400 I=1,IMAX
  400 B(K)=B(K)+C(K,I)*B(I)
      B(K)=-B(K)/C(K,K)
      YP=YP*(-F)
      TZ=TZ+B(K)*YP
  500 CONTINUE
      RETURN
      END
```

APPENDIX C

In the following compilation of the recurrence coefficients for commonly used non-rational functions, it is assumed that the $k^{\underline{th}}$ Taylor coefficients for the function A(t) are known and the $k^{\underline{th}}$ coefficient ($k \geq 1$) for $B=f(A)$ is sought.

$$A(t) = \sum_{j=o}^{\infty} a_j (t-t_o)^j$$

$$B(t) = \sum_{j=o}^{\infty} b_j (t-t_o)^j$$

For each function, the functional relationship is listed first, followed by the defining differential equation and finally by the Taylor coefficients. Many functions such as sin, cos, tan are derived from a coupled differential system and consequently are listed together. The symbol ' denotes differentiation with respect to the independent variable t.

$$B = \exp(A)$$

$$B' = BA'$$

$$b_k = (\sum_{j=1}^{k} ja_j b_{k-j})/k$$

$$B = \log_e(A), \ a_o > 0$$

$$B' = A'/A$$

$$b_1 = a_1/a_o$$

$$b_k = (a_k - \sum_{j=1}^{k-1} ja_{k-j} b_j \ /k)/a_o \ , \ k \geq 2$$

$$B = \log_{10}(A) \ , \ a_o > 0$$

$$B' = \alpha A'/A \ , \ \alpha = \log(e)$$

$$b_1 = \alpha a_1/a_o$$

$$b_k = (\alpha a_k - \sum_{j=1}^{k-1} ja_{k-j} b_j \ /k)/a_o \ , \ k \geq 2$$

$$B = A^{\alpha} \ , \ \alpha \text{ real}, \ a_o > 0$$

$$B' = \alpha BA'/A$$

$$b_k = \sum_{j=0}^{k-1} (\alpha - j(\alpha + 1)/k) b_j a_{k-j})/a_o$$

$$B = \sin(A), \ C = \cos(A), \ D = \tan(A), \ c_o \neq 0$$

$$B' = CA', \ C' = -BA', \ D = B/C$$

$$b_k = (\sum_{j=1}^{k} ja_j c_{k-j})/k$$

$$c_k = -(\sum_{j=1}^{k} ja_j b_{k-j})/k$$

$$d_k = (b_k - \sum_{j=1}^{k} c_j d_{k-j})/c_o$$

$$B = \sinh(A),\ C = \cosh(A),\ D = \tanh(A),\ c_o \neq 0$$

$$B' = CA',\ C' = BA',\ D = B/C$$

$$b_k = \left(\sum_{j=1}^{k} j a_j c_{k-j}\right)/k$$

$$c_k = \left(\sum_{j=1}^{k} j a_j b_{k-j}\right)/k$$

$$d_k = \left(b_k - \sum_{j=1}^{k} c_j d_{k-j}\right)/c_o$$

The Taylor coefficients for the operations +, -, *, / are also included.

$$C = A + B$$

$$c_k = a_k + b_k$$

$$C = A - B$$

$$c_k = a_k - b_k$$

$$C = A*B$$

$$c_k = \sum_{j=o}^{k} a_{k-j} b_j$$

$$C = A/B$$

$$c_k = \left(a_k - \sum_{j=1}^{k} b_j c_{k-j}\right)/b_o$$

APPENDIX D

```
RMAT ENTRY
 LEVEL K TYPE (C, Q , V | E)                     R OP      A(1)          A(2)      A(3)
                #Y(1,2)=1.0$
   1    2   0   Y(1,2)
   1    3   5   =
   1    4  -1   #2
   1    5   9   $
                #Y(1,1)=Y(1)**2+3.D0*T**2$       1 $       Y(2)          #2        Y(2)
   1    2   0   Y(1,1)
   1    3   5   =
   1    4   0   Y(1)
   1    5  10   **
   1    6  -1   #3
   1    7   1   +
                                                 2 *       Y(1)          Y(1)      ?2
   1    8  -1   #4
   1    9   3   *
   1   10   0   T
   1   11  10   **
   1   12  -1   #3
   1   13   9   $                                3 *       T             T         ?3
                                                 4 *       #4            ?3        ?4
                                                 5 +       ?2            ?4        ?5
                                                 6 $       Y(1)          ?5        Y(1)
                                                 RECURRENCE MATRIX
                                                 1 $       Y(2)          #2        T
                                                 2 *       Y(1)          Y(1)      ?2
                                                 3 *       T             T         ?3
                                                 4 *       #4            ?3        ?4
                                                 5 +       ?2            ?4        ?5
                                                 6 $       Y(1)          ?5        Y(1)
                                                 OPTIMIZED RECURRENCE MATRIX
                                                 1 $       Y(2)          #2        T
                                                 2 *       Y(1)          Y(1)      ?2
                                                 3 *       T             T         ?3
                                                 4 *       #4            ?3        ?4
                                                 5 +       ?2            ?4        ?5
                                                 6 $       Y(1)          ?5        Y(1)
```

```
CONSTANT TABLE

# 1=0.5D0                  # 2=1.0D0            # 3=2                # 4=3.D0

DMAT ENTRY  1, 2, 3, 4, 5, 6

D MATRIX

 1, 1 -1    1, 2 -1    1, 3 -1    1, 4 -1    1, 5 -1    1, 6 -1    2, 1 -1    2, 2 -1

 2, 3 -1    2, 4 -1    2, 5 -1    2, 6  0    3, 1 -1    3, 2 -1    3, 3 -1    3, 4 -1

 3, 5 -1    3, 6 -1    4, 1 -1    4, 2 -1    4, 3  1    4, 4 -1    4, 5 -1    4, 6 -1

 5, 1 -1    5, 2  1    5, 3  1    5, 4  1    5, 5 -1    5, 6  0    6, 1 -1    6, 2  1

 6, 3  1    6, 4  1    6, 5  1    6, 6  0

CORRESPONDENCE BETWEEN RECURRENCE MATRIX ROWS AND THE Y ARRAY

 1 2          2 3          3 4          4 5          5 6          6 1
```