10-20

~~DO NOT COPY~~
~~COPIES AVAILABLE~~
~~CODE 2627, B43, R14~~

~~2620PE~~

UNCLASSIFIED

NRL Report 8611

A FORTRAN-Based Program for Computerized Algebraic Manipulation

R. R. DASENBROCK

Systems Research Branch
Aerospace Systems Division

September 21, 1982

PLEASE RETURN THIS COPY TO:

NAVAL RESEARCH LABORATORY
WASHINGTON, D.C. 20375
ATTN: CODE 2628

Because of our limited supply you are requested to return this copy as soon as it has served your purposes so that it may be made available to others for reference use. Your cooperation will be appreciated.

NDW-NRL-5070/2616 (1-84)

NAVAL RESEARCH LABORATORY
Washington, D.C.

Approved for public release; distribution unlimited.

UNCLASSIFIED

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER NRL Report 8611	2. GOVT ACCESSION NO.
3. RECIPIENT'S CATALOG NUMBER	
4. TITLE (and Subtitle) A FORTRAN-BASED PROGRAM FOR COMPUTERIZED ALGEBRAIC MANIPULATION	5. TYPE OF REPORT & PERIOD COVERED Final report on one phase of the problem.
	6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) R. R. Dasenbrock	8. CONTRACT OR GRANT NUMBER(s)
9. PERFORMING ORGANIZATION NAME AND ADDRESS Naval Research Laboratory Washington, DC 20375	10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS 61153N-14 RR014-02-41-6152 79-1474-0-2
11. CONTROLLING OFFICE NAME AND ADDRESS Office of Naval Research Arlington, VA 22217	12. REPORT DATE September 21, 1982
	13. NUMBER OF PAGES 73
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office)	15. SECURITY CLASS. (of this report) UNCLASSIFIED
	15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Machine algebra
Computer algebra
Algebraic manipulation
Symbolic manipulation

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

This report describes a program which can perform literal algebraic manipulations on a high-speed electronic computer. Emphasis is concentrated on the manipulation of the harmonic series occurring in the classical theory of lightly perturbed dynamical systems. The program is written in FORTRAN and can be made operational on any computer possessing a FORTRAN compiler and at least a 32-bit word length.

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE
S/N 0102-014-6601

UNCLASSIFIED

CONTENTS

INTRODUCTION 1

PROGRAM OBJECTIVES 2

INTERNAL REPRESENTATION 3

STORAGE MANAGEMENT 3

DATA CHAIN MANIPULATIONS 5

ALGEBRAIC MANIPULATIONS 8

THE EXTENDED TWO-DIMENSIONAL REPRESENTATION 14

SUMMARY OF THE STORAGE-ALLOCATION METHODS 19

PROBLEM EXAMPLES 19

- Multiplication 19
- A Binomial Expansion 22
- Transformation into a Polynomial Form 24
- Computation of the Legrendre Polynomials 26
- A Series Solution to a Differential Equation 28
- A Transcendental Equation 30
- An Example of Polynomial Denominators 31

CONCLUDING REMARKS 32

ACKNOWLEDGMENTS 32

REFERENCES 33

APPENDIX A — Program Listing 34

UNCLASSIFIED

A FORTRAN-BASED PROGRAM FOR COMPUTERIZED ALGEBRAIC MANIPULATION

INTRODUCTION

In 1973 the author described a computer-automated algebraic manipulation program [1] to process and manipulate a Poisson series. The purpose of the present report is to describe a much revised and improved version of this original work. The original employed a one-dimensional chained format to represent each series. It proved quite easy to implement and required a minimal amount of overhead within the machine. However for certain applications, such as planetary theory, a literal polynomial denominator is required in order to complete the theory. A two-dimensional representation proved more advantageous to represent the divisors. In the latter part of this report this extented representation is described in detail along with an improved version of the one-dimensional chained-format representation.

The earliest efforts in programming literal algebraic expansions on a computer were by Herget and Musen [2] in 1958. However, due to the limitations of the IBM-650 in use at the time, little progress was made until the introduction of the high-speed computers in the early 1960s. FORMAC [3] was introduced in 1964 and was followed by MATHLAB [4], REDUCE [5], and MACSYMA [6] a few years later. These languages were quite general and required little programming skill but unfortunately could not efficiently satisfy the widely varied demands of each individual user. This was especially true for the more serious researchers attempting to solve the massive algebraic problems encountered in lightly perturbed dynamical systems. Many minutes of central-processor (CPU) time were required when solving seemingly simple problems, and users were almost always plagued with failures due to storage overflow when attempting to solve sizable problems.

To satisfy their own specialized requirements, some researchers began to construct their own algebraic manipulation programs to satisfy their own specialized needs. For example, Deprit et al. [7] have introduced a program called E.S.P. (Echeloned Series Processor) to compute an analytical lunar ephemeris. Van Flandern and Pulkkinen [8], Jefferys [9], and Broucke [10] have constructed similar programs to manipulate the Poisson series occurring in classical perturbation theory. Unfortunately these programs are for the most part neither documented or in an easily exportable form. An excellent survey of many of the applications of algebraic manipulation programs in physics may be found in Barton and Fitch [11].

The program described here grew mainly out of the author's unsuccessful attempts at implementing REDUCE on the NRL CDC 3800 in 1971. Since that time, due to the laboratory's acquisition of a PDP-10, REDUCE and MATHLAB have both become available through DECUS, and both have been implemented on-site at NRL. MACSYMA has also become available through the ARPANET. MACSYMA represents one of the most extensive developments to date in this field. It has been most helpful in solving small-scale problems of general interest. However, storage limitations [12] at the host computer site preclude the execution of very large scale problems. For this discussion a large-scale problem involves expressions of 50,000 terms or more.

Manuscript submitted March 16, 1982.

For a program to be effective, one should be able to reproduce a computation of the size of, say, Delaunay's original lunar theory [13] in a reasonably small amount of CPU time (several minutes or less) on any present-day, large-size electronic computer. Although this requirement may seem unrealistic, it is called for because a fundamental requirement of the program being considered here is the ability to significantly expand upon such a theory in a reasonable amount of CPU time. Here a reasonable increment of CPU time is defined as the maximum that can be assigned to a single run at most computer centers (usually several hours or less). A larger amount of time is considered unreasonable in relation to efficient machine usage.

For the machine to reproduce in several minutes what Delaunay accomplished in 20 years by hand, a machine-to-man speed factor of over several hundred thousand to one is desired (if Delaunay spent at least several hours per day on this task). From the author's experience, it appears that most of the general algebraic (symbol) manipulators in use today have attained a speed factor of not more than 5000 to 1. Thus, these systems appear to be of limited use when performing massive algebraic calculations, and more specialized programs or machines are required.

The overall modern high-speed electronic computer can outpace its human counterpart by over 10,000,000 to 1 when performing numerical computations. However, when manipulating algebraic expressions, the same computer becomes over 1000 times less efficient. It seems that much improvement in software (and hardware) development is needed over what is apparently present technology. This situation may be changed soon with the construction of the LISP machine [14]. This is a new computer system which is hard-wired for the LISP programming language to provide a high performance and economical implementation of this language. Since MACSYMA is written in LISP, its performance should increase by an order of magnitude. An unfortunate disadvantage is that portability is sacrificed completely for such systems, except between machines of identical design.

All computerized algebraic manipulation programs have two things in common. It cannot be accurately determined in advance how long the program will run, and it cannot be predicted in advance how much storage will be required for intermediate and final results. When the size and length of the program approach the limit of a particular machine, storage overflow may result. Sometimes a simple rearrangement of the calculations will bring success. In more extreme cases a total reformulation of the problem is required. Automated algebraic manipulations may be approached in two ways: as a problem of symbolically matching and substituting strings of characters through an applicative language such as FORMAC or REDUCE or as a problem of applying algebraic laws of operation on fixed data structures [15,16]. The latter approach is simpler and easier to implement and thus is the approach that is followed.

PROGRAM OBJECTIVES

The construction of a general-purpose algebraic manipulator suitable to the needs of a diverse multidisciplinary research community appeared to be not feasible for the reasons presented in the preceding section. Therefore it was decided to restrict the range of applicability of the manipulator to a specific area of research and at the same time produce a result which is fast and efficient within its domain. FORTRAN compatibility of the program and a high efficiency factor were specific goals of the developmental program. Portability of the program between machines of different manufacture was of high priority. Standard FORTRAN-IV programming was to be adhered to whenever possible. Exotic methods of coding were discouraged, even though advantages could be realized on machines of a particular design.

A program was therefore devised to manipulate the sometime lengthy harmonic series occurring in the study of lightly perturbed dynamical systems such as found in astrodynamics. Desired manipulator capabilities include the addition, subtraction, multiplication, simplification, differentiation, and

UNCLASSIFIED

integration of one or more series. With the successful implementation of these algebraic operations, it would be possible to produce high-order series solutions to a set of first-order differential equations that characterize the mathematical foundation of certain astrodynamical and other problems.

INTERNAL REPRESENTATION

Many problems in applied mathematics can be solved by series expansions. One such series is the so-called Poisson series [15], which can be written in the form

$$\Sigma \begin{Bmatrix}\text{rational}\\ \text{fraction}\end{Bmatrix} \cdot \begin{Bmatrix}\text{polynomial of}\\ \text{several variables}\end{Bmatrix} \cdot \begin{Bmatrix}\sin\\ \cos\end{Bmatrix} \cdot \begin{Bmatrix}\text{argument involving a}\\ \text{linear combination of}\\ \text{several angles}\end{Bmatrix}.$$

This form is the underlying mathematical form used by this program and in classical perturbation theory. In addition, this series is significant in algebraic manipulation because it contains several important properties:

- The sum, difference, and product of two Poisson series is a Poisson series.
- The substitution of one Poisson series into another yields a Poisson series.
- The symbolic integration and differentiation does not change the form of the series.

Internally to the machine, each term of the Poisson series is represented by 12 32-bit words. These words, which are in common to all subroutines, are

NEXT(k) word giving the location (index) of the next term in the list

N(k) word giving the integer numerator

M(k) word giving the integer denominator

KTERM(j,k) nine words giving the polynomial index containing the packed integer exponents whose values are restricted from −128 to +127, a sine-cosine bit (−1 or +1 for sine or cosine respectively), and a trigonometric-argument index containing the packed integer coefficients whose values are restricted from −128 to +127.

Each word contains four packed coefficients which can represent the polynomial exponents, the trigonometric argument coefficients, or the sine-cosine bit. The leading integer polynomial and trigonometric coefficient is restricted to the range of −64 to +63, whereas the remaining coefficients are restricted to −127 to 128 inclusive. Each algebraic term, denoted by k, requires 12 words of storage, which is the space required by **NEXT, N, M,** and **KTERM**.

STORAGE MANAGEMENT

The success of any algebraic manipulator depends on the effectiveness of the storage management procedures that have been devised. Elementary list-processing techniques are used to handle efficiently the sometimes large series that are generated as intermediate and final results. The chain of elements comprising a series is denoted as a list. Distinction is made between an active list containing one or more elements and an inactive list denoting the null series (Fig. 1). The arrays **NEXT, N, M,** and

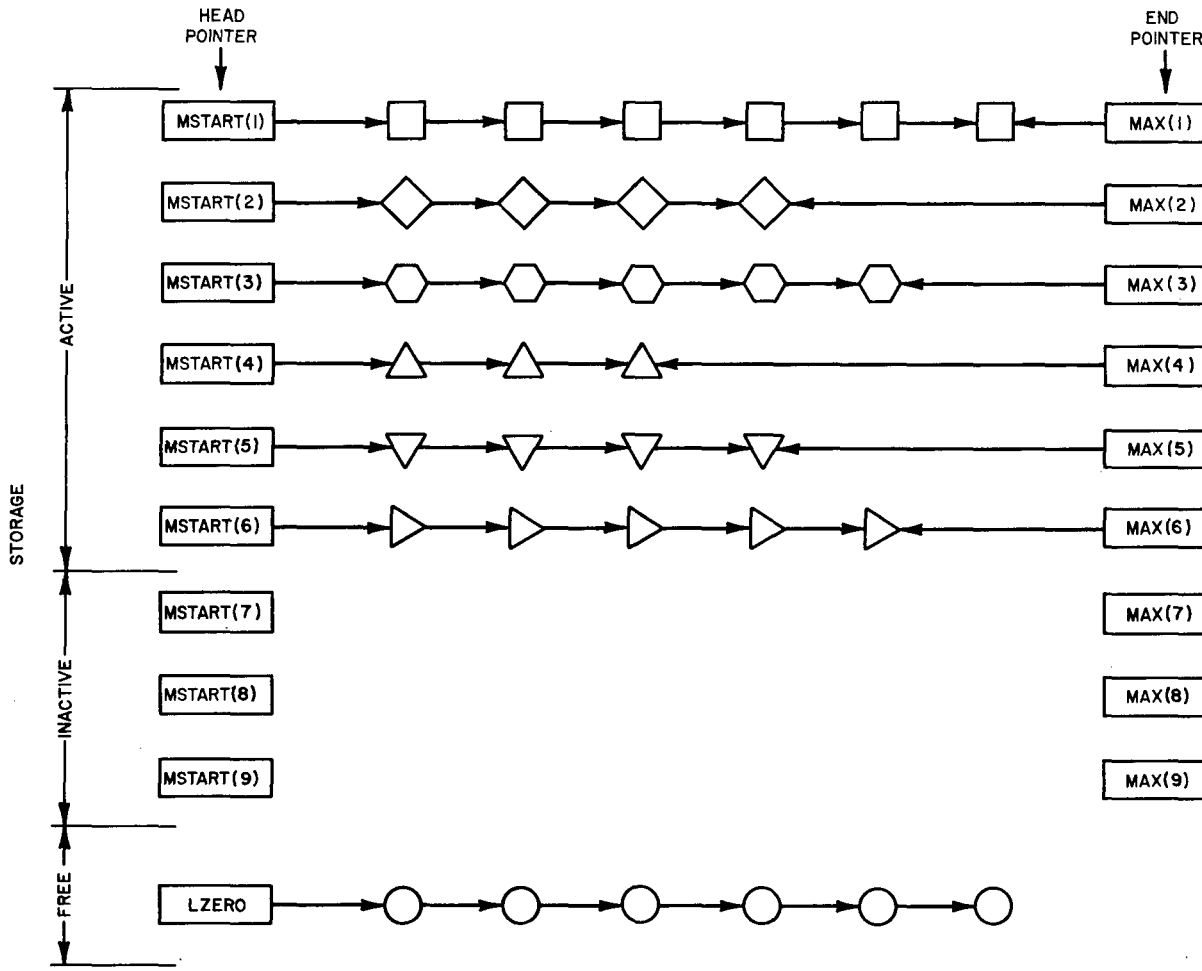

Fig. 1 — Internal list structure (symbolic). Round symbols denote free storage throughout this report

KTERM are dimensioned in the driver program to sufficient size to accommodate the problem at hand. From the author's experience, a dimension of 10,000 will suffice for most of the moderately sized problems, whereas a dimension of over 50,000 may be required for the larger scale problems. Since 12 words are required for each term, 600,000 words will be required to store 50,000 terms within memory. This is approaching the limit of the storage available on many present-day computers. A reformulation of the problem may be necessary to reduce storage requirements.

The number of different series is limited to 1000. A few series each having many terms or many series each having a few terms cannot be exceeded. In other words, space is allocated to the nonvanishing terms only. The remaining space is denoted as free storage. Also, adjacent elements of a particular list may be interlaced with the terms of separate lists and the free storage, as in Fig. 2.

The location of the lead term of a particular list I is given by the head pointer **MSTART(I)**, and its terminal location is given by the end pointer **MAX(I)**. The elements of each list are chained together by pointers; that is, the location of the (j + 1)th term is given by **NEXT(j)**. The location of

UNCLASSIFIED

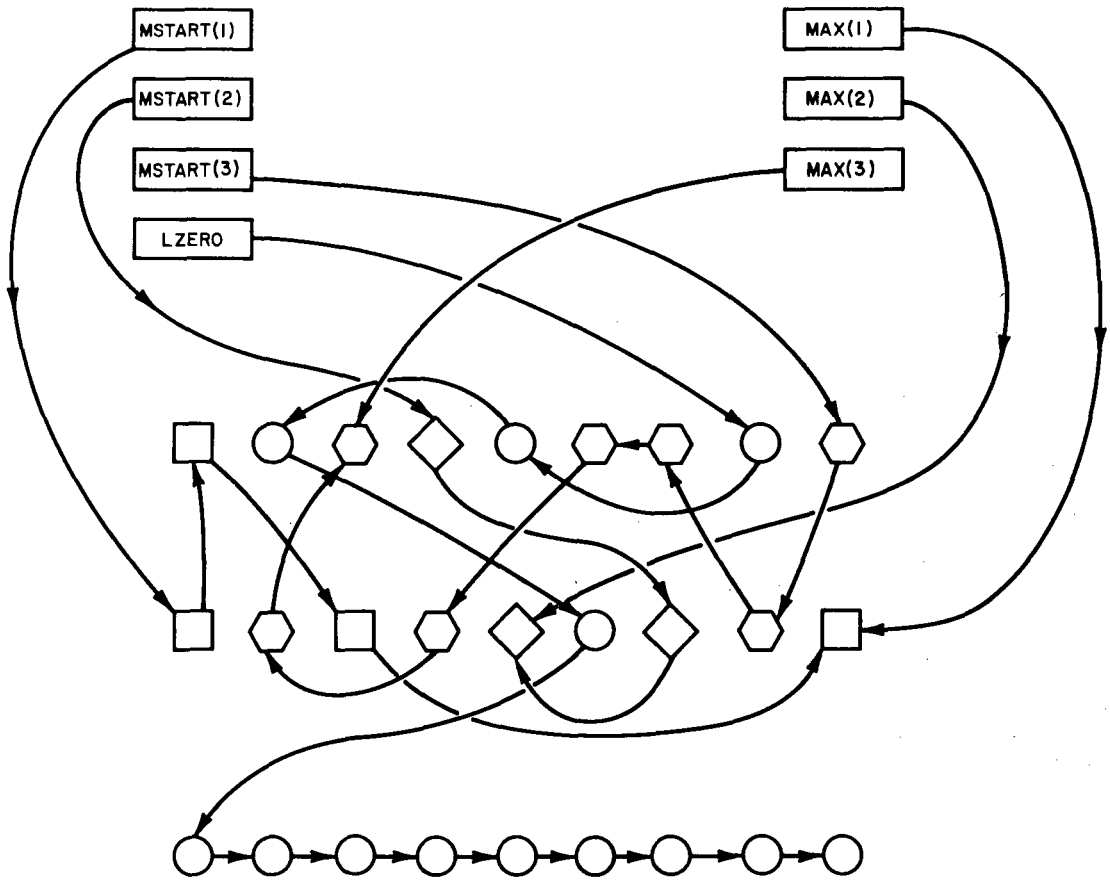

Fig. 2 — Example of an actual internal list structure

the preceding element is not given; a chain cannot be followed in its reverse direction. The storage not allocated to active storage is denoted as free storage, which is also kept in chained format, with the lead-cell location being given by **LZERO**. The space required for each multiplication, addition, etc. is taken as needed from free storage, whereas the unneeded space created as a result of simplification is returned to the free storage, where it is again available for use as required. "Garbage collection" (the collection of unused space into a single area of contiguous storage available for future use) is carried out continuously in this manner. The storage management is automatically carried out by the data chain routines (to be described next) and thus is of little concern to the user.

DATA CHAIN MANIPULATIONS

A number of subroutines will now be described which manipulate the series in various ways. A hierarchy of operations is constructed starting from the simplest and growing in complexity to the more complicated operations. Each successive procedure depends on those before it; thus the coding is kept relatively simple, most routines being kept to less than one page of FORTRAN statements. Most of these subroutines are required by the mathematical and algebraic routines to be described later and are generally not required explicitly in using the program.

Subroutine START

START is called initially at the start of the program. It performs a number of initial housekeeping duties, of which the most pertinent are as follows:

- The head pointers **MSTART(I)** and **MAX(I)** are initialized.
- The total memory to be used is set up as free storage in chained format. **LZERO** denotes the leading free storage cell in chained format. All the memory is in free storage initially.

Subroutine SWITCH(I,J)

SWITCH(I,J) interchanges the correspondence of the series I with that of J; that is, $I \rightleftarrows J$.

Subroutine ZERO(I)

ZERO(I) negates the series I and releases its occupied space to free storage. To illustrate how this subroutine operates, consider the list arrangement of Fig. 3a. A call to **ZERO(I)** rearranges the pointers, and the resulting list arrangement is shown in Fig. 3b. **MSTART(I)** is set to zero, and I becomes a null series.

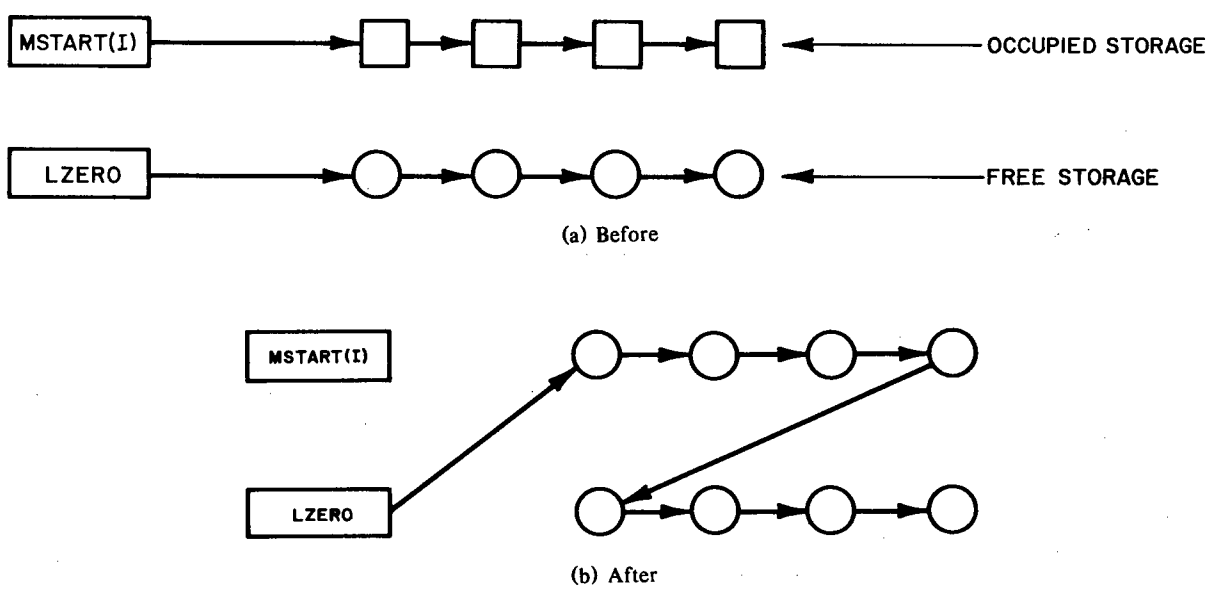

Fig. 3 — Typical list arrangement before and after CALL ZERO(I)

Subroutine TRANSF(I,J)

TRANSF(I,J) negates the series J (if active) and returns the space to free storage. A new list J is then created equivalent to but separate from I. Consider the particular list structure of Fig. 4a. After a call to **TRANSF(I,J)** the arrangement becomes that of Fig. 4b. The series I is left unchanged.

UNCLASSIFIED

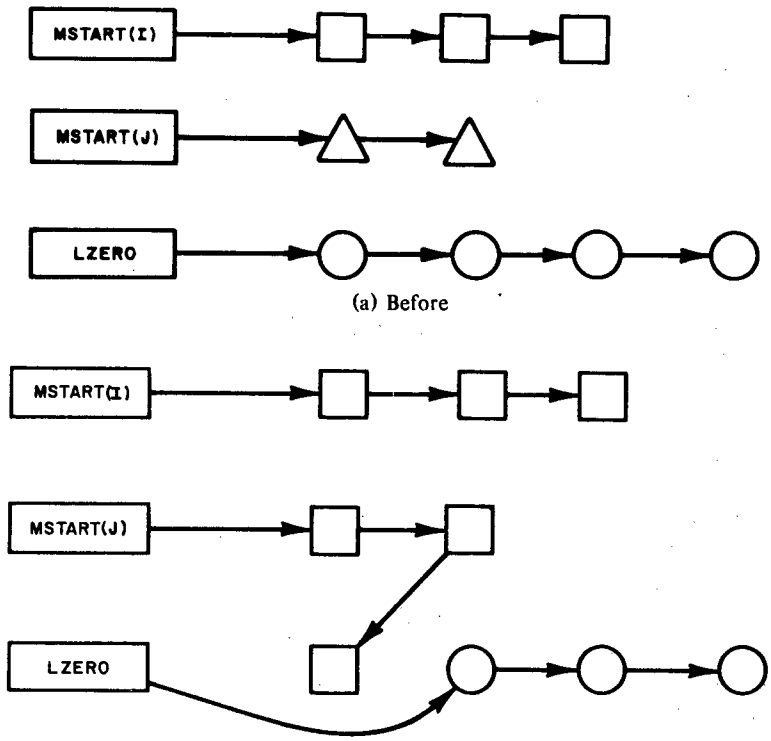

Fig. 4 — Typical list arrangement before and after CALL TRANSF(I,J)

Subroutine LINK(I)

LINK(I) returns all entries of the series I which are zero to free storage, leaving only the nonzero entries chained together. Figure 5 describes a call to the subroutine **LINK** which then applies it to a sample chain. As shown in Fig. 5b, the null terms are returned to the free-cell list.

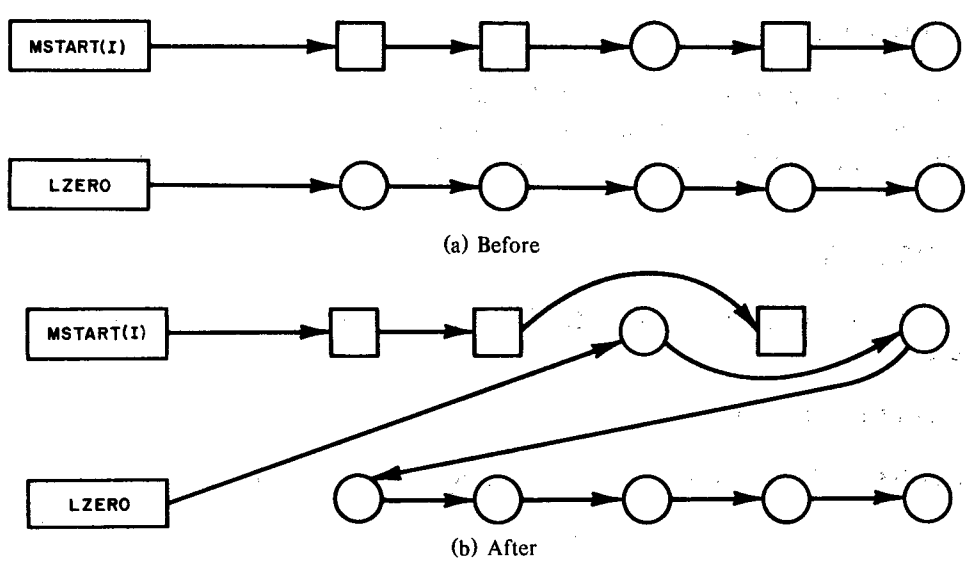

Fig. 5 — List arrangement before and after CALL LINK(I)

Subroutine SIMP(I)

SIMP(I) collects like entries in I. The terms are ordered according to the argument index, the sine-cosine bit, and the polynomial index respectively during this operation. The routine will sort in place; no additional intermediate storage will be required. The resulting unneeded space is released to free storage.

ALGEBRAIC MANIPULATIONS

Another set of subroutines has been devised to carry out the various mathematical and algebraic operations. These subroutines and the speed with which they are carried out depend on the data-chain procedures just mentioned.

Whenever possible in these subroutines, the coefficients are computed as exact integer fractions. However, it is impossible to avoid the unwieldly fractions which appear when higher order expansions of certain functions are computed. When this is the case, a nonexact representation is used. Here the computational speed is increased, since the integer fractional coefficients need not be searched for common factors. However, a new problem appears. Due to roundoff errors, terms with small coefficients sometimes appear when theoretically equal terms are subtracted. Provision for deleting these terms are made because badly needed storage is wasted if these small coefficients are retained. A threshold **EPSLON** is set to delete these unwanted coefficients. The use of nonexact coefficients is usually necessary when extensive high-order calculations are performed and is also convenient when literal expressions are evaluated numerically. The logical variable **NEXACT** is set to **.TRUE.** if the rational coefficients cannot be preserved. These coefficients, when encountered, are printed in a floating-point format.

Subroutine FACTOR(N,M)

A call to the subroutine **FACTOR(N,M)** automatically removes the common factors from the numerator and denominator (N,M). When working with inexact coefficients, this subroutine negates any term whose coefficient is less then **EPSLON**, a lower limit on the size of the coefficient. This limit is set according to the physics of the problem at hand, that is, several orders of magnitude less than the smallest coefficient physically meaningful. The Euclidean factorization algorithm [17] is used.

Subroutine ADD(I,J,K)

If $I \neq J \neq K$, the subroutine **ADD(I,J,K)** adds the series J to I and places the simplified result in K. If $K = I$ the routine acts as an accumulator: $I + J \rightarrow I$. This routine is extremely fast—almost instantaneous as compared to some of the other algebraic routines (such as **MULT**) to be described later.

Subroutine SUB(I,J,K)

SUB(I,J,K) which is similar to **ADD(I,J,K)**, subtracts the series J from I and places the simplified result in K.

Subroutine MULCON(I,n,m)

MULCON(I,n,m) multiplies the series I term by term by the rational fraction n/m.

UNCLASSIFIED

Subroutine MULT(I,J,K)

MULT(I,J,K) multiplies each term of the series I with each term of J to form a result which is stored in K. Provision is made to truncate any term whose order exceeds a specified value. During multiplication, the following trigonometric identities are introduced implicitly:

$$2 \cos x \cos y = \cos(x + y) + \cos(x - y),$$

$$2 \cos x \sin y = \sin(x + y) - \sin(x - y),$$

$$2 \sin x \cos y = \sin(x + y) + \sin(x - y),$$

$$2 \sin x \sin y = \cos(x + y) + \cos(x - y),$$

$$\cos(-x) = + \cos x,$$

$$\sin(-x) = - \sin x,$$

$$\cos(0) = 1,$$

$$\sin(0) = 0.$$

For these trigonometric identities, two terms are produced (labeled A and B in Fig. 6) for each product $I_m J_n$. The required space for both A and B is taken from free storage. Next the partial result K must be searched for like entries. If such a term is found, the new term A or B is combined with it. If no such entries are found, as is shown in the example of Fig. 7, the new terms are inserted (in order) in the partial list K. The actual terms are not moved in memory; only the pointers are recomputed.

The searching operation just described can be very time consuming if not carried out efficiently. Since the partial result K is always ordered, it is not necessary to search its entire length each time a new term is produced. A binary search scheme has been devised using this fact to sort efficiently. First the new term is compared to the middle term of the list. This comparison determines whether the lower or upper half of the list need next be searched. The remaining portion is again halved, and the new term is again compared with its midpoint. This procedure is carried out repetitively until the

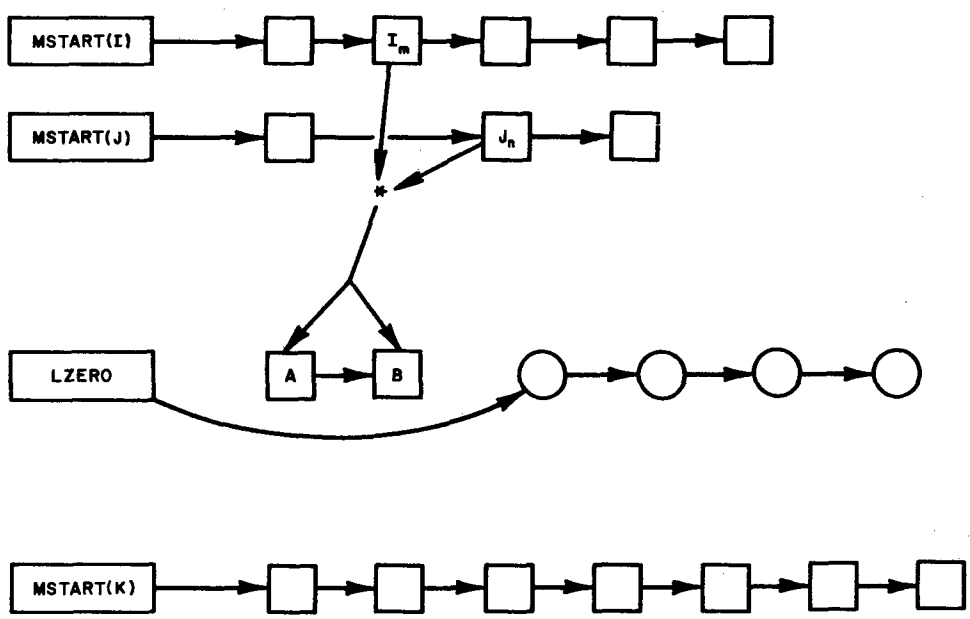

Fig. 6 — List arrangement during multiplication of subproduct $I_m J_n$

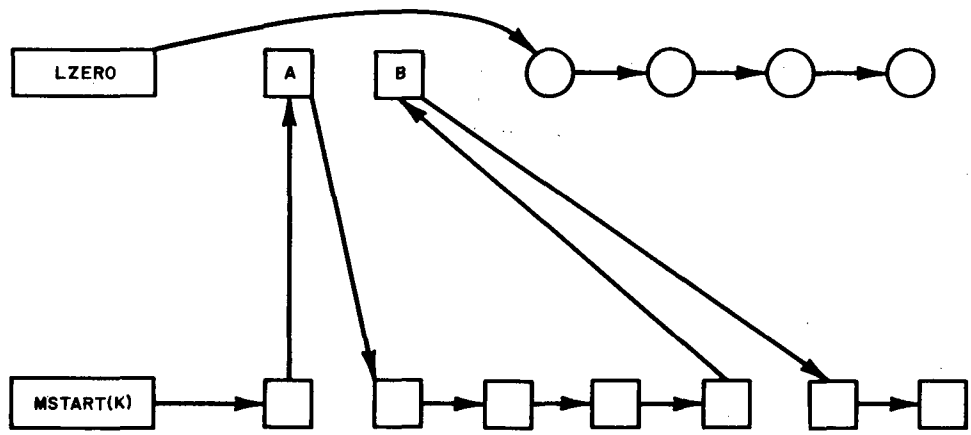

Fig. 7 — List arrangement after the new terms (labeled A and B) are inserted in K

correct location for the insertion at the new term has been determined. For a list of 4000 terms only about 12 such comparisons need be made. Actually, because the partial list K is always kept in chained format, the midpoints of each successive half-interval cannot be determined by formula directly. To circumvent this problem, the pointers (which give the location of each successive term) are mapped onto a linear array. It is this array that is searched, and the locations of the required midpoints are obtained directly. As this mapping is in itself time consuming, it need only be carried out a dozen or so times during the course of a long multiplication. The effect of this procedure is to make the program running time solely dependent on the total number of multiplications performed and eliminate its dependence on the length of the final result. For example, a multiplication which produced a result of over 4000 terms required over 1 hour of CPU time without the optimal search. After the implementation of this technique, the time was reduced to less than 2 minutes.

During multiplication truncation is carried out by the limits specified by **KCHOP**, **MORDER**, and **KFACT**. For example if the ith variable is to be truncated at order j + 1, then set **KCHOP(i)** = j. If more than one polynomial variable is a high-order small quantity, then **MORDER** and **KFACT** should be used. Suppose the ith and jth polynomial variables are first-order and third-order small quantities respectively. Further suppose the result of a multiplication is to be truncated at order m. Then set **MORDER** = m, **KFACT(i)** = 1, and **KFACT(j)** = 3. The truncated terms are not stored, and only the desired portion of the result is produced.

Subroutine POWER(I,n,J)

POWER(I,n,J) multiplies the series I by itself n − 1 times and stores the result in location J; that is, I**n → J.

Subroutine EXPAND(I,k,m,n,J)

EXPAND(I,k,m,n,J) calculates the binomial expansion of the quantity $(1 + I)^{m/n}$, where I << 1. The pair (m,n) are integers. The expansion is terminated at order k in I. The result is stored in location J.

UNCLASSIFIED

Subroutine DERIV(I,j,K)

DERIV(I,j,K) differentiates the series stored in location I term by term with respect to the jth packed variable. The result is stored as the Kth chain. I remains unaltered unless I = K. If j is positive, the derivative is taken with respect to the jth packed polynomial variable. If j is negative, the |j|th trigometric variable is assumed.

Subroutine NDERIV(I,j,n,K)

NDERIV(I,j,n,K) differentiates the Ith chain with respect to the jth packed variable n times. The result is stored in location K.

Subroutine BRACKE(I,J,m,n,K)

BRACKE(I,J,m,n,K) calculates the Poisson bracket $(I,J)_{m,n}$ with respect to the variables m,n. The result is stored in location K.

Subroutine INTEG(I,j,K)

INTEG(I,j,K) integrates each term of I with respect to the jth packed variable. The result is stored in location K.

Subroutine DEnm(I,NUM,DEMON,J1,...,Jn,K,L1,...,Lm)

DEnm(I,NUM,DEMON,J1,...,Jn,K,L1,...,Lm) is used to input the various terms of a series to the machine. The term (NUM/DEMON) X1**J1 ... Xn**Jn (sin/cos) (L1*Y1 + ... + Lm*Ym) is added to the contents of the series I. J1, ..., Jn and L1, ..., Lm represent the first n and m packed polynomial and trigonometric exponent and coefficient values respectively. This routine and others similar are used to input series to the machine. At present only the routines **DE44**, **DE124**, **DE168**, and **DE204** are implemented. These represent the values for (n,m) of (4,4), (12,4), (16,8), and (20,4) respectively. In principle only one routine, namely DE 248, need be implemented. However as two lines of FORTRAN code are required for each use, it is too lengthy for most applications. Location I may assume any number from 1 through 1000; that is, 1000 different series may be represented.

Subroutine SUBAB(I,J,m,n,ν)

SUBAB(I,J,m,n,ν) substitutes $m^2 = 1 - n^2$ in the Ith chain, where m and n represent the mth and nth packed variable. That is, when sin i and cos i are stored as polynomial variables, the substitution becomes $\cos^2 i \rightarrow 1 - \sin^2 i$. This substitution will in effect search for the trigonometric identity $\sin^2 i + \cos^2 i = 1$. The operation is carried out ν times, and the result is stored in location J. In the expression $315/512 \cos^4 i - 135/256 \cos^2 i + 27/512$, a call to **SUBAB(I,J,m,n,2)** performs the substitution $1 - \sin^2 i$ twice. The result is $315/512 \sin^4 i - 45/64 \sin^2 i + 9/64$. This substitution sometimes greatly reduces the size of certain intermediate and final expressions. The result is stored in location J.

Subroutine CUT(I,J)

CUT(I,J) deletes all the nonperiodic terms in the series I and stores the result in location J. The routine requires no additional space if I = J.

Subroutine SECULA(I,J)

SECULA(I,J) is similar to **CUT(I,J)**, except that the periodic terms are eliminated. Only the nontrigonometric terms remain.

Subroutine TRUN(I,J,j,n)

TRUN(I,J,j,n) negates all terms above order n in the jth packed polynomial variable. In general only the terms below order **MORDER** are retained during a multiplication. The rest are negated before they are stored and simplified. The result is stored in location J if $I \neq J$. If the preservation of I is not required, I may be set to J, so that no additional space is required.

Subroutine SELECT(I,J,j,n)

SELECT(I,J,j,n) retains all terms of order n in the jth packed polynomial variable of the series I and removes all other members. The result is placed in location J. If $I = J$, the selection is performed in place, and no additional space is required.

Subroutine CHOOSE(I,J,j,n)

CHOOSE(I,J,j,n) is similar to **SELECT**, except that all terms of periodicity n in the jth packed trigonometric argument are retained. The result is stored in location J.

Subroutine TAYLOR(I,J,j,m,K)

TAYLOR(I,J,j,m,K) performs a Taylor-series expansion on the series I with respect to x, which denotes the jth packed variable:

$$f(x + \Delta X) = \sum_{k=0}^{m} \frac{1}{k!} \frac{\partial_k}{\partial x^K} [f(x)] (\Delta X)^k. \tag{1}$$

The f(x) and x terms are stored as I and J respectively. The expansion is terminated at order m in Δx. The result is stored in location K.

Subroutine EVAL(I,J,j,A)

EVAL(I,J,j,A) substitutes the floating-point number A for the jth packed polynomial variable. The result is stored as J. Rational coefficients are not preserved, and floating-point coefficients are computed.

Subroutine OUTLP(I)

OUTLP(I) prints or punches the series I in a literal form. The punched form is FORTRAN compatible and can be inserted as part of another FORTRAN program with no additional modifications. The output may be given as described in two forms: either with the coefficients of each term presented as exact integer fractions whenever possible and the remaining coefficients presented as floating-point numbers or with all coefficients presented as floating-point numbers.

Subroutine SHORT(I)

SHORT(I) is similar to **OUTLP**, except that only the first and last ten terms are printed.

UNCLASSIFIED

Subroutine SPLIT(I,J,iv)

SPLIT(I,J,iv) splits the series I into two components, I and J. I contains only those terms that do not depend on the ivth packed polynomial variable. J contains all the remaining terms. No additional storage is required. The initial contents of J, if any, are released to free storage.

Subroutine TAEVAL(I,J,iv,k)

TAEVAL(I,J,iv,k) evaluates the series I for the ivth trigometric packed variable in k multiples of $\pi/2$. Rational coefficients are preserved. The result is placed in J. Its initial contents, if any, are released to free storage.

Subroutine KEVAL(I,J,iv,KNUM,KDEMON)

KEVAL(I,J,iv,KNUM,KDEMON) substitutes the rational fraction KNUM/KDEMON for the ivth packed variable. The resulting series is evaluated, simplified, and placed in location J. Rational coefficients are preserved.

Subroutine MULONE(I)

MULONE(I) truncates the series I according to the criteria specified by **KFACT**, **KCHOP**, and **MORDER**.

Subroutine MULVAR(I,iv,num)

MULVAR(I,iv,num) multiplies the series I by the ivth packed variable to power num.

Subroutine DEFONE(I)

DEFONE(I) sets the series I to unity.

Subroutine ERASE(I,iv)

ERASE(I,iv) erases the dependence of the series I on the ivth packed variable; that is, it sets the ivth variable to unity and evaluates and simplifies the resulting series.

Subroutine IN(I,L)

IN(I,L) reads the series stored on the Lth logical unit into central memory as the series I.

Subroutine OUT(I,L)

OUT(I,L) reads out the series I onto the Lth logical unit. This subroutine is used when no storage is available in central memory. The resulting released space then becomes available for additional computations.

Subroutine TFORM(I,J,iv,IS,IC)

TFORM(I,J,iv,IS,IC) performs a transformation on the series I which places a trigometric variable of position iv into a polynomial in powers of sin(iv) and cos(iv). These are loaded into positions IS and IC as polynomial variables. For example cos 3A will be transformed into $4\cos^3 A - 3\cos A$. The result is stored in location J.

Subroutine SUBST(I,J,iv,L)

SUBST(I,J,iv,L) substitutes the series J for the ivth packed variable in the series I. The result is stored in location L. The routine is valid for only positive exponents of the series J.

Subroutine ACCUM(I,J)

ACCUM(I,J) adds the series J to I, and J becomes the null series. No additional storage is required for this computation, because I and J are merely combined. If I = J, then I → 2I.

THE EXTENDED TWO-DIMENSIONAL REPRESENTATION

A one-dimensional chained format has been employed up to this time to represent the algebraic expressions internal to the machine. This scheme has proven easy to implement. However the limitations of this method become rather apparent when processing very large expressions: those of 50,000 terms or more. This is due in part to the following reasons:

- The polynomial and trigonometric nodes must be recopied for each term regardless of their equivalence to other nodes. Most periodic series contain many like nodes and therefore must be recopied for each term.

- The one-dimensional search, although efficient, can be time consuming. The efficiency of the program is directly dependent on the time required to insert the terms as they are produced during multiplication. The time required for sorting the many thousands of terms can be reduced by use of the two-dimensional procedure in spite of the increased overhead.

- The method does not adapt well to the more generalized Poisson series, that is, those series containing literal polynomial divisors.

Figure 8 describes an alternate two-dimensional method [16]. The polynomial divisors may be linked to either or both the trigonometric and coefficient node for generality. In some problems encountered in astrodynamics a polynomial divisor will appear during the integration of a series with several time-dependent trigonometric arguments. If they are all slowly varying, they cannot be expanded and must be carried as a divisor of each trigonometric node. They need not be recopied for each coefficient node.

Each polynomial and trigonometric node is listed only once. All of the numerical coefficients associated with each trigonometric node are linked left and right through each coefficient node. Likewise all the numerical coefficients associated with each polynomial node are linked up and down through each coefficient node. Each chain of trigonometric and polynomial nodes are also linked forward and backward. Two separate free-cell lists are maintained: one for the coefficient and the other for the polynomial and coefficient nodes which have identical structure.

For example the following expression can be represented in both the one-dimensional or new two-dimensional extended representation:

$$1/2\, X^2Y \cos(A + B) + 2/3\, X^2Y \cos(2A - B) + 3/8\, X^3Y^2 \cos(2A - B) + 2/3\, X^3Y^4 \cos(3A + B) + 9/7\, X^2Y \cos(4A - B) + 5/8\, X^3Y^4 \cos(4A - B). \quad (2)$$

Figure 9 describes the internal representation, and Fig. 10 shows the structure in the extended two-dimensional representation.

UNCLASSIFIED

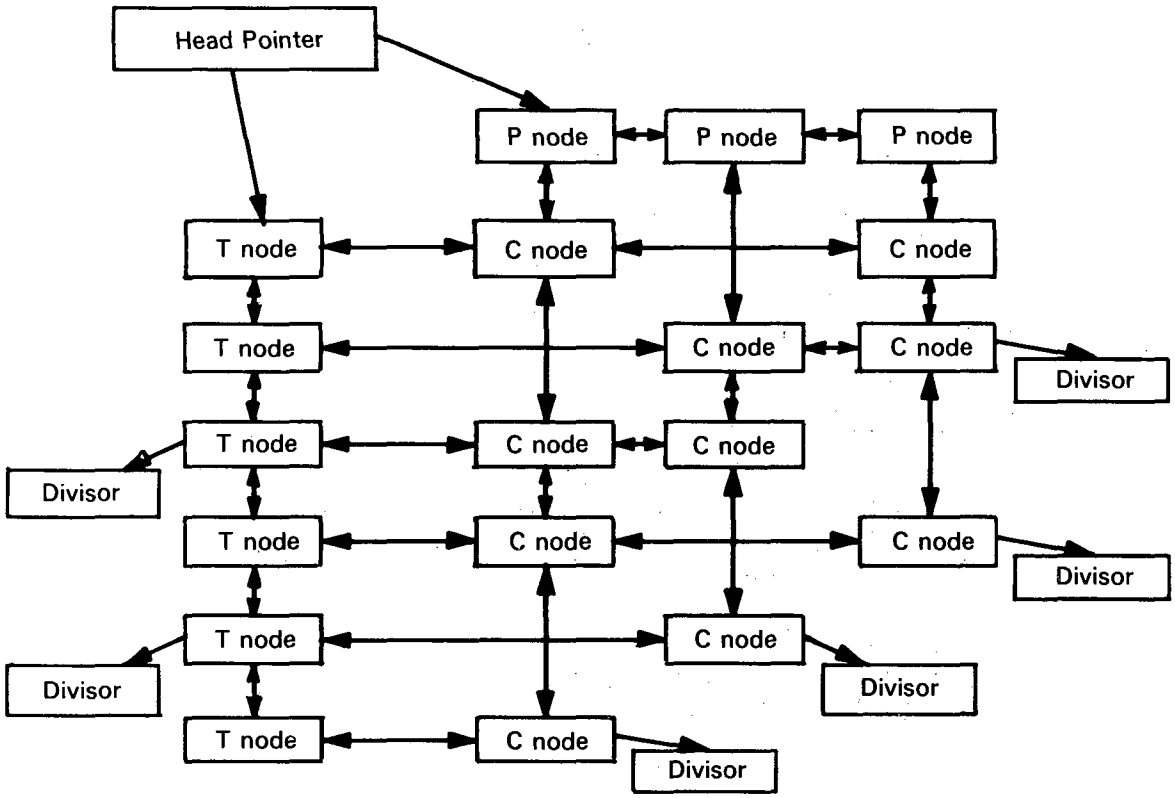

Fig. 8 — Two-dimensional extended representation (P = polynomial, T = trigonometric, and C = coefficient)

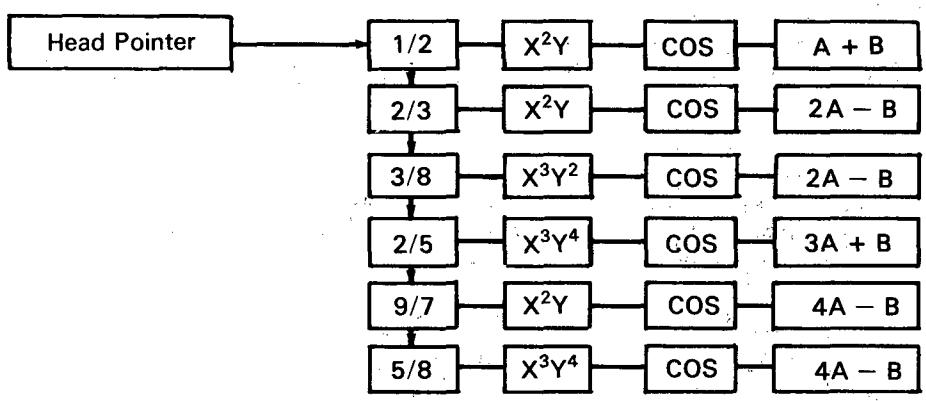

Fig. 9 — One-dimensional representation of Eq. (2)

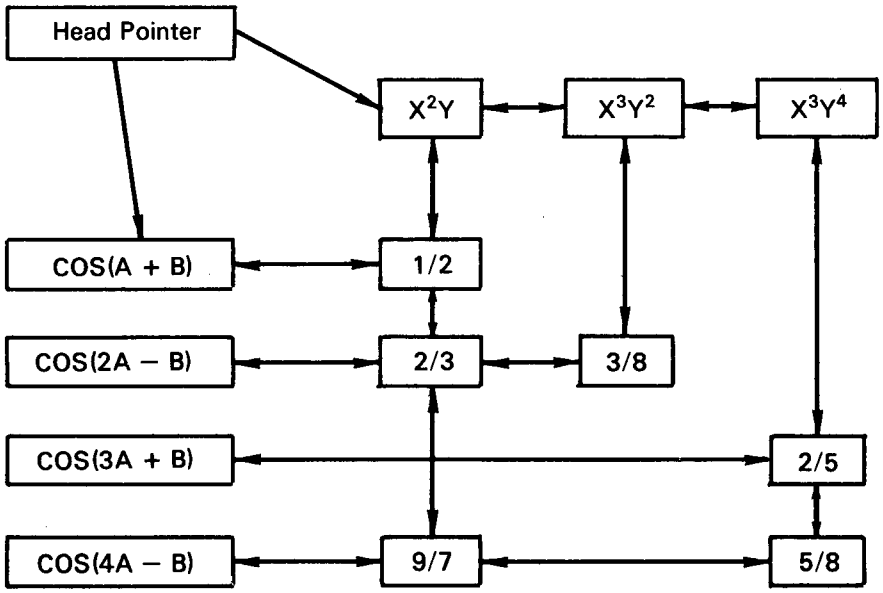

Fig. 10 — Extended two-dimensional representation of Eq. (2)

The preceding example did not include polynomial divisors. Figure 11 describes how the following expression containing polynomial divisors is represented in the machine:

$$\frac{1/2X^{**}2^{*}Y^{*}\cos(A+B)}{(X+2Y)} + \frac{1/4Z^{**}2^{*}\cos(2A-B)}{(Z-3X)^{*}(X+2Y^{**}2)}. \tag{3}$$

Figure 12 describes the head pointer **MSTART** for the two-dimensional method. It contains the positions of the leading polynomial and trigonometric nodes. Figure 13 describes the structure of each polynomial node. It contains three pointers. Two are used to denote the next and preceding polynomial node respectively. The third points to the leading coefficient node.

Figure 14 describes the trigonometric node. Its structure is identical to the polynomial node with the addition of one pointer to reference the polynomial divisor (if any). The coefficient node is shown

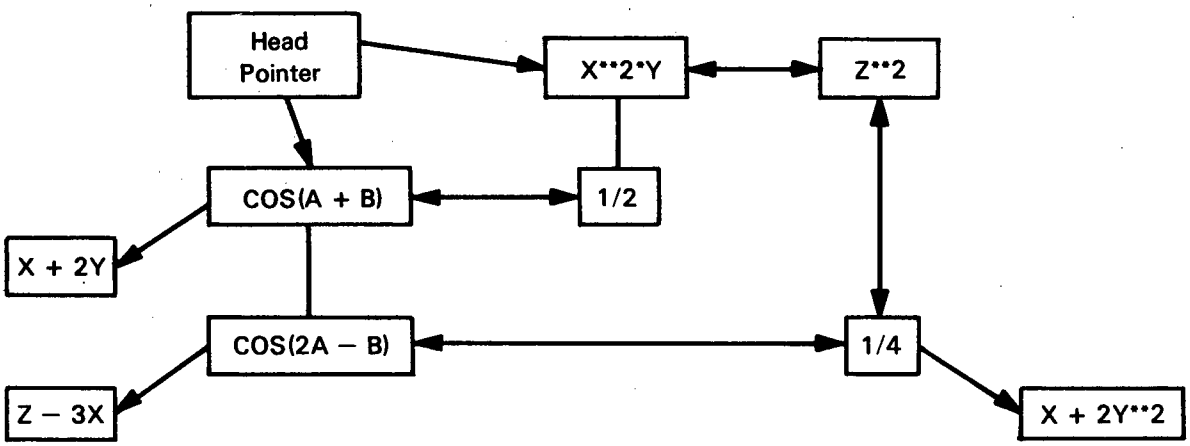

Fig. 11 — The representation of polynomial divisors

UNCLASSIFIED

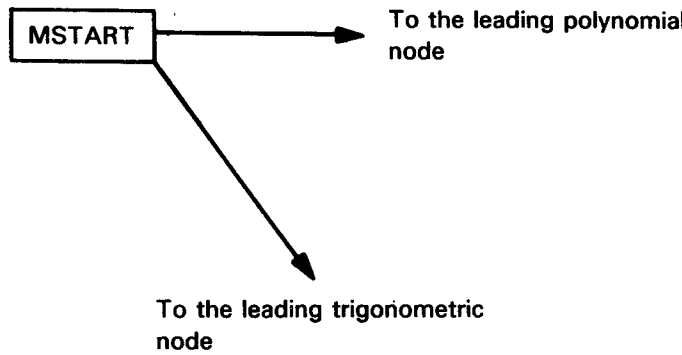

Fig. 12 — A head pointer for the two-dimensional method

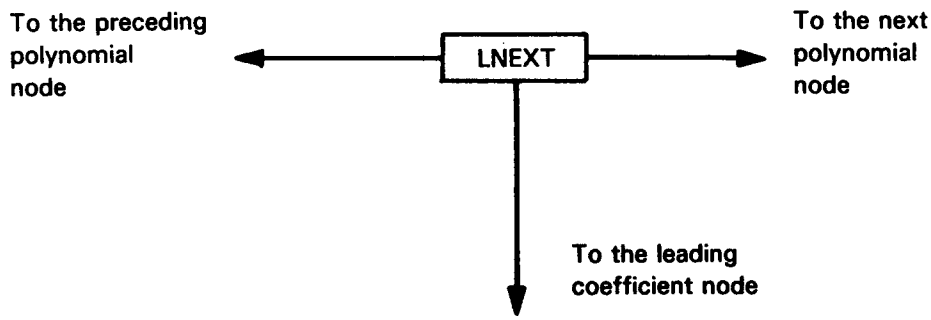

Fig. 13 — Polynomial node

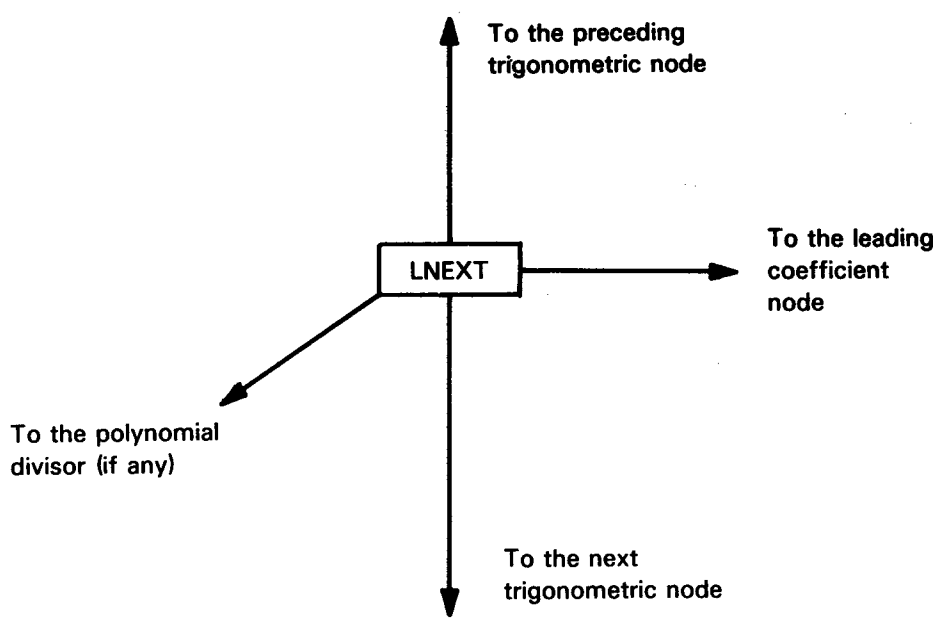

Fig. 14 — Trigonometric node

in Fig. 15. It is the most complicated as it contains seven pointers. Four pointers link the preceding and successive nodes both vertically and horizontally. Two pointers link the polynomial and trigonometric nodes at the head of the chain. These are not absolutely necessary, because these nodes could be recovered by following the chain linking each coefficient back to its origin. However, this can be time consuming, and the storage saved is not worth the loss in efficiency. The seventh pointer locates the polynomial divisor (if any) associated with that node.

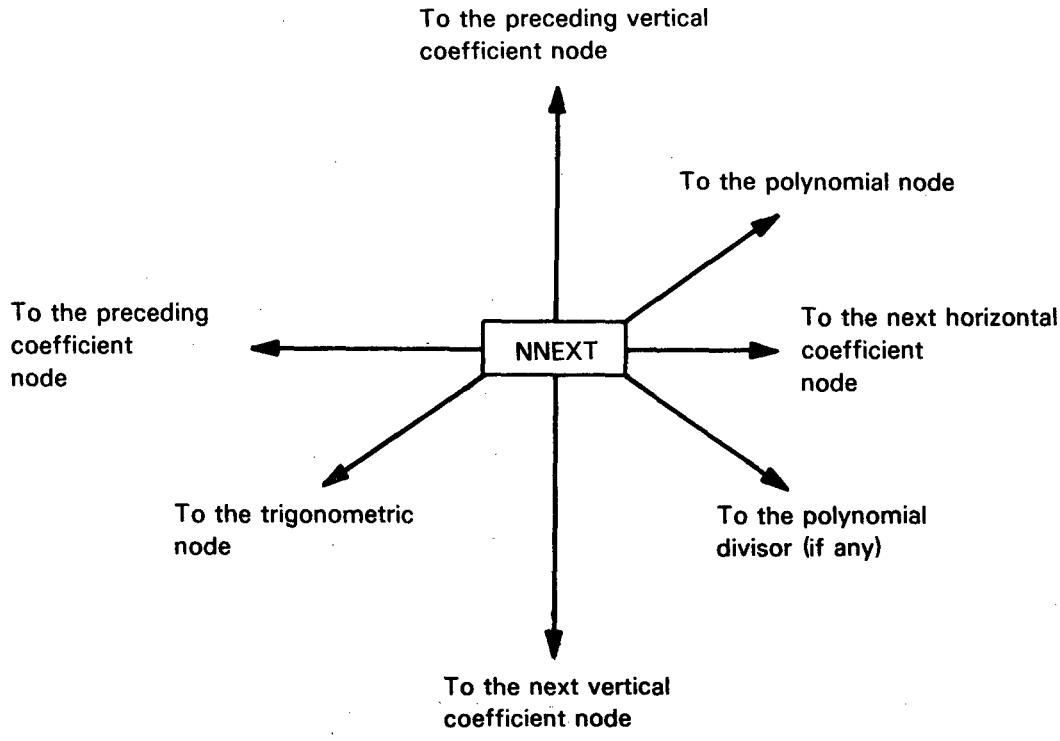

Fig. 15 — Coefficient node

Figure 16 shows the free-cell head pointer **LZERO**. Its two entries contain the location of the head pointers of the unused (available) coefficient nodes **NNEXT** and trigonometric and polynomial nodes **LNEXT**. The polynomial divisors when needed are drawn from **LNEXT**.

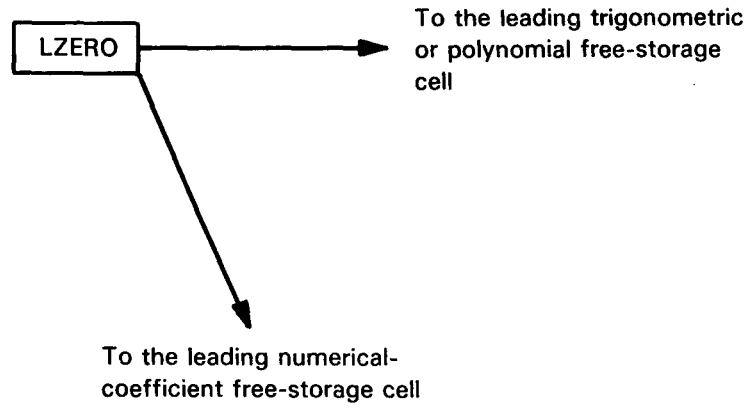

Fig. 16 — Free-cell-storage head pointer

UNCLASSIFIED

The two-dimensional version of the program has been coded; however, it is still in the experimental stage and not ready for export. The author will make a listing available to any interested person. It is not included in the listing at the end of this report.

SUMMARY OF THE STORAGE-ALLOCATION METHODS

Thus in the overall development of the program, three storage-allocation schemes have been pursued:

- Unchained sequential. This method was used in an earlier undocumented version of the program. It is probably the most efficient method of storing the terms, because no pointers (and the space they occupy) are required. Adjacent terms in each series are stored in adjacent locations in memory. However, it is probably the least efficient in terms of running time, because only the less-efficient sorting algorithms can be applied. This is especially true for the sort-in-place operations required for massive calculations. Peripheral disk storage could be used to take advantage of the more efficient routines, but again valuable CPU time is required for the necessary data-transfer operations. This method proved simplest to implement and was fastest for problems involving expressions of no more than a hundred terms.

- One-dimensional chained format. This is the method described in the first part of this report. It appears to be a good compromise between storage and computational efficiency. It is fairly easy to implement and proved superior for problems involving expressions of 10,000 terms or less. The required overhead is minimal, because only one pointer is required per term. An efficient sort-in-place algorithm is used to promote efficiency during the multiplication and simplification of massive expressions.

- Two-dimensional chained format. This method, just described, proved most difficult to implement, because seven pointers are required for each coefficient node and four for each trigonometric and polynomial node. During multiplication and simplification these pointers must be constantly manipulated, which is time consuming to implement. In theory this method should be the most efficient, because it should strike a best compromise between storage and computational efficiency, especially if a fairly dense coefficient matrix is encountered. This has not been realized in practice, since the coefficient array tends to be sparse: several terms or less per polynomial or trigonometric node. This method has the potential of being the most efficient computationally, especially during the multiplication of two large expressions. In practice this also has not been realized. However, this format should permit the implementation of a highly efficient sort-in-place algorithm. It is anticipated that when this algorithm is implemented, some improvement in computational speed will be realized.

PROBLEM EXAMPLES

Multiplication

The first example is a simple multiplication. Suppose the following two expressions are to be multiplied together, which are stored in the arbitrary locations #100 and #200 respectively:

$$\#100 = -\,4/7\ UY^2\sin(A + 3B - 5D) + 1/2\ V^3XY^6\cos(6B + 2C - D) + 2/3\ X^2Y \tag{4}$$

and

$$\#200 = -\,U^7V^8 - 9/5\ UV^2\cos(C - 7D) + V^3XY^8\sin(9A - 3B). \tag{5}$$

The following driver program is required for this example:

```
      PROGRAM AMULT
C     EXAMPLE OF A MULTIPLICATION
      COMMON/LPOLY/LPOLY(24)/LARG/LARG(8)
      CALL START

C     ASSIGN NAMES TO VARIABLES
      LPOLY(1)=1HU
      LPOLY(2)=1HV
      LPOLY(3)=1HX
      LPOLY(4)=1HY
      LARG (1)=1HA
      LARG (2)=1HB
      LARG (3)=1HC
      LARG (4)=1HD

C     DEFINE FIRST SERIES AND PRINT
      CALL DE44(100,+1,2,0,3,1,6,+1,0,6,2,-1)
      CALL DE44(100,-4,7,1,0,0,2 -1,1,3,0,-5)
      CALL DE44(100,+2,3,0,0,2,1,+1,0,0,0,0)
      CALL OUTLP(100)

C     DEFINE SECOND SERIES AND PRINT
      CALL DE44(200,+1,1,0,3,1,8 -1,9,-3,0,0)
      CALL DE44(200,-9,5,1,2,0,0,+1,0,0,1,-7)
      CALL DE44(200 -1,1,7,8,0,0,+1,0,0,0,0)
      CALL OUTLP(200)

C     MULTIPLY TOGETHER - PLACE RESULT IN #300
      CALL MULT(100,200,300)
      CALL OUTLP(300)
      STOP
      END
```

The necessary subroutines to support this driver program are listed in Appendix A.

UNCLASSIFIED

The program output is listed below, with the input series #100 and #200 being shown along with the output, #300:

```
START OF EXECUTION




WRITE SERIES 100
LENGTH =     3

    -4/7*U*Y**2*SIN(A+3B-5D)
    +1/2*V**3*X*Y**6*COS(6B+2C-D)
    +2/3*X**2*Y




WRITE SERIES 200
LENGTH =     3

    -U**7*V**8
    -9/5*U*V**2*COS(C-7D)
    +V**3*X*Y**8*SIN(9A-3B)




WRITE SERIES 300
LENGTH =    13

    +4/7*U**8*V**8*Y**2*SIN(A+3B-5D)
    -1/2*U**7*V**11*X*Y**6*COS(6B+2C-D)
    -2/3*U**7*V**8*X**2*Y
    +18/35*U**2*V**2*Y**2*SIN(A+3B+C-12D)
    +18/35*U**2*V**2*Y**2*SIN(A+3B-C+2D)
    -9/20*U*V**5*X*Y**6*COS(6B+3C-8D)
    -9/20*U*V**5*X*Y**6*COS(6B+C+6D)
    +2/7*U*V**3*X*Y**10*COS(10A-5D)
    -2/7*U*V**3*X*Y**10*COS(8A-6B+5D)
    -6/5*U*V**2*X**2*Y*COS(C-7D)
    +1/4*V**6*X**2*Y**14*SIN(9A+3B+2C-D)
    +1/4*V**6*X**2*Y**14*SIN(9A-9B-2C+D)
    +2/3*V**3*X**3*Y**9*SIN(9A-3B)
```

A Binomial Expansion

As an example demonstrating how a series expansion may be computed, $1/(1 + E \sin M)^{1/2}$ is to be expanded to eighth order in E. To truncate the expansion at order eight, **KFACT(4)** is set to 1, and **MORDER** is set to 8. E is the 4th packed polynomial variable and M is the 4th trigonometric packed variable. The following sample driver program is required to compute this series:

```
      PROGRAM AXPAND
C     EXAMPLE OF AN EXPANSION
      COMMON/II/KFACT(24)/JJ/MORDER
      COMMON/LPOLY/LPOLY(24)/LARG/LARG(8)
      LOGICAL NPRINT

      CALL START

C     DEFINE VARIABLES
      LPOLY(4)=1HE
      LARG (4)=1HM

C     SET TRUNCATION LIMITS
      MORDER=8
      KFACT(4)=1

C     DEFINE E*SIN(M)
      CALL DE44(100,1,1,0,0,0,1,-1,0,0,0,1)

C     COMPUTE (1 + E*SIN(M))**(-1/2) TO 8TH ORDER IN "E"
      CALL EXPAND(100,8,-1,2,200)
      CALL OUTLP(100)
      CALL OUTLP(200)
      STOP
      END
```

UNCLASSIFIED

The program output is listed below:

```
START OF EXECUTION



EXPAND (1 +100)**(-1/ 2)
 TO ORDER  8 IN100
 RESULT STORED IN200
 TOTAL TIME =   0.15 SEC


WRITE SERIES 100
LENGTH =    1

    +E*SIN(M)



WRITE SERIES 200
LENGTH =   25

    +6435/4194304*E**8*COS(8M)
    -6435/524288*E**8*COS(6M)
    +45045/1048576*E**8*COS(4M)
    -45045/524288*E**8*COS(2M)
    +225225/4194304*E**8
    +429/131072*E**7*SIN(7M)
    -3003/131072*E**7*SIN(5M)
    +9009/131072*E**7*SIN(3M)
    -15015/131072*E**7*SIN(M)
    -231/32768*E**6*COS(6M)
    +693/16384*E**6*COS(4M)
    -3465/32768*E**6*COS(2M)
    +1155/16384*E**6
    -63/4096*E**5*SIN(5M)
    +315/4096*E**5*SIN(3M)
    -315/2048*E**5*SIN(M)
    +35/1024*E**4*COS(4M)
    -35/256*E**4*COS(2M)
    +105/1024*E**4
    +5/64*E**3*SIN(3M)
    -15/64*E**3*SIN(M)
    -3/16*E**2*COS(2M)
    +3/16*E**2
    -1/2*E*SIN(M)
    +1
```

Transformation into a Polynomial Form

Certain problems that appear in astrodynamics require that an expression be transformed into a polynomial form. As an example to show how this may be accomplished, suppose that X cos(19A) + Y sin(16A) is to be transformed into a polynomial in first powers of sin(A) and cos(A). The driver program for this example is as follows:

```
      PROGRAM ATRANS
C     EXAMPLE OF A POLYNOMIAL TRANSFORMATION
      COMMON/LPOLY/LPOLY(24)/LARG/LARG(8)
      CALL START

C     DEFINE "SA"=SIN(A) AND "CA"=COS(A)
      LPOLY(1)=2HSA
      LPOLY(2)=2HCA
      LPOLY(3)=1HX
      LPOLY(4)=1HY
      LARG (1)=1HA

C     DEFINE X*COS(19A)+Y*SIN(16A)
      CALL DE44(100,1,1,0,0,1,0,+1,19,0,0,0)
      CALL DE44(100,1,1,0,0,0,1,-1,16,0,0,0)
      CALL OUTLP(100)

C     TRANSFORM THE SERIES IN LOCATION #100 INTO
C     A PURELY POLYNOMIAL FORM
      CALL TFORM(100,200,-1,1,2)
      CALL OUTLP(200)

C     NOW DO THE REVERSE PROCEDURE - I.E., LINEARIZE THE SERIES
      CALL DE44(10,1,1,0,0,0,0,-1,1,0,0,0)
      CALL DE44(12,1,1,0,0,0,0,+1,1,0,0,0)
      CALL OUTLP(10)
      CALL OUTLP(12)
      CALL SUBST(200,10,1,250)
      CALL SUBST(250,12,2,300)

C     PRINT RESULT AND COMPARE WITH ORIGINAL SERIES IN #100
      CALL OUTLP(300)
      STOP
      END
```

UNCLASSIFIED

The program output is shown below:

```
START OF EXECUTION

WRITE SERIES 100
LENGTH =    2

    +X*COS(19A)
    +Y*SIN(16A)

TRANSFORM  100 200  -1   1   2
SUBSTITUTE 200 200   2   1   9
 TOTAL TIME =   0.46 SEC
 TOTAL TIME =   1.33 SEC

WRITE SERIES 200
LENGTH =   18

    -262144*SA**18*CA*X
    +1114112*SA**16*CA*X
    -32768*SA**15*CA*Y
    -1966080*SA**14*CA*X
    +114688*SA**13*CA*Y
    +1863680*SA**12*CA*X
    -159744*SA**11*CA*Y
    -1025024*SA**10*CA*X
    +112640*SA**9*CA*Y
    +329472*SA**8*CA*X
    -42240*SA**7*CA*Y
    -59136*SA**6*CA*X
    +8064*SA**5*CA*Y
    +5280*SA**4*CA*X
    -672*SA**3*CA*Y
    -180*SA**2*CA*X
    +16*SA*CA*Y
    +CA*X
```

```
WRITE SERIES  10
LENGTH =    1

    +SIN(A)

WRITE SERIES  12
LENGTH =    1

    +COS(A)

SUBSTITUTE 200  10   1 250
SUBSTITUTE 250  12   2 300

WRITE SERIES 300
LENGTH =    2

    +X*COS(19A)
    +Y*SIN(16A)
```

The result is shown in series #200. To check the result, sin(A) and cos(A) are both substituted into the expression using the routine **SUBST**. The result is in location #300 and agrees with the original expression in #100, as it should.

Computation of the Legrendre Polynomials

The Legrendre polynomials can be computed from the expansion

$$(1 - 2hX + h^2)^{-1/2}. \tag{6}$$

The coefficient of h^k will be the kth Legrendre polynomial. The code to perform this expansion and selection is listed below:

```
      PROGRAM LEGRENDRE
      COMMON/II/KFACT(24)/JJ/MORDER
      COMMON/LPOLY/LPOLY(24)/LARG/LARG(8)

      CALL START
      LPOLY(1)=1HX
      LPOLY(2)=1HH
      KFACT(1)=0
      KFACT(2)=1
      MORDER=6

C     DEFINE  H**2-2*H*X
      CALL DE44(60,1,1,0,2,0,0,+1,0,0,0,0)
      CALL DE44(60,-2,1,1,1,0,0,1,0,0,0,0)
      CALL OUTLP(60)

C     EXPAND TO INVERSE SQUARE ROOT
C     PLACE RESULT IN LOCATION #70
      CALL EXPAND(60,MORDER -1,2,70)

C     SELECT POWERS OF "H" - THEIR COEFFICIENTS
C     WILL BE THE LEGRENDRE POLYNOMIALS
      DO 10 K=1,MORDER
      CALL SELECT(70,100+K,2,K)
      CALL ERASE(100+K,2)
   10 CALL OUTLP(100+K)
      END
```

UNCLASSIFIED

The program output is shown below:

```
START OF EXECUTION


WRITE SERIES  60
LENGTH =    2

    -2*X*H
    +H**2


EXPAND (1 + 60)**(-1/ 2)
 TO ORDER  6 IN 60
 RESULT STORED IN 70
 TOTAL TIME =   0.12 SEC

WRITE SERIES 101
LENGTH =    1

    +X


WRITE SERIES 102
LENGTH =    2

    +3/2*X**2
    -1/2


WRITE SERIES 103
LENGTH =    2

    +5/2*X**3
    -3/2*X


WRITE SERIES 104
LENGTH =    3

    +35/8*X**4
    -15/4*X**2
    +3/8


WRITE SERIES 105
LENGTH =    3

    +63/8*X**5
    -35/4*X**3
    +15/8*X


WRITE SERIES 106
LENGTH =    4

    +231/16*X**6
    -315/16*X**4
    +105/16*X**2
    -5/16
```

A Series Solution to a Differential Equation

An excellent example is a series solution to Kepler's equation relating the orbital mean anomaly to the true anomaly. This example is chosen because the result may be compared with the published solution in the literature [18]. The differential equation to be considered is

$$\frac{df}{dM} = (1 - e^2)^{-3/2}\,(1 + e\cos f)^2 \text{ (with } f = M \text{ when } M = 0). \tag{7}$$

The problem is to compute $f - M$ to a high order in e and to assume $e << 1$. To first order in e:

$$\frac{df}{dM} = 1 + 2e\cos f$$

$$\rightarrow f - M = 2e\sin M. \tag{8}$$

To second order in e, the first order-solution is used on the right-hand side:

$$\frac{df}{dM} = (1 + 3/2e^2 + \ldots)\,(1 + e\cos(M + 2e\sin M))^2$$

$$\rightarrow f - M = 2e\sin M + 5/4e^2\sin 2M + \ldots. \tag{9}$$

This procedure could in principle be carried to any order in eccentricity. The driver program required to solve Kepler's equation to eighth order in eccentricity is listed below:

```
      PROGRAM KEPLER
      COMMON/II/KFACT(24)/JJ/MORDER
      COMMON/LPOLY/LPOLY(24)/LARG/LARG(8)
      COMMON/NN/NPRINT
      LOGICAL NPRINT
      CALL START
      NPRINT=.TRUE.

C     SET TRUNCATION LIMITS
      KFACT(4)=1
      JOR=8
      MORDER=JOR

C     DEFINE VARIABLE NAMES
      LPOLY(4)=1HE
      LARG (4)=1HM

C     CARRY OUT EXPANSION
      CALL DEFONE(101)
      CALL DE44(102,-1,1,0,0,0,2,+1,0,0,0,0)
      CALL DE44(103,+1,1,0,0,0,0,+1,0,0,0,0)
      CALL DE44(103,+1,1,0,0,0,1,+1,0,0,0,1)
      CALL EXPAND(102,JOR/2,-3,2,105)
      CALL MULT(103,103,106)
      CALL MULT(105,106,107)
      CALL SUB(107,101,110)
      DO 30 MORDER=1,JOR
      CALL TAYLOR(110,500,-4,MORDER,112)
      CALL CUT(112,112)
   30 CALL INTEG(112,-4,500)
      CALL OUTLP(500)
      END
```

UNCLASSIFIED

The program output, that is, the solution of Kepler's equation to eighth order in E, is listed below:

```
START OF EXECUTION


EXPAND (1 +102)**(-3/ 2)
 TO ORDER  4 IN102
 RESULT STORED IN105
 TOTAL TIME =   0.04 SEC
TAYLOR SERIES EXPANSION  110  500   -4    1  112
 TIME FOR TAYLOR SERIES EXPANSION =   0.02 SEC
TAYLOR SERIES EXPANSION  110  500   -4    2  112
 TIME FOR TAYLOR SERIES EXPANSION =   0.09 SEC
TAYLOR SERIES EXPANSION  110  500   -4    3  112
 TIME FOR TAYLOR SERIES EXPANSION =   0.18 SEC
TAYLOR SERIES EXPANSION  110  500   -4    4  112
 TIME FOR TAYLOR SERIES EXPANSION =   0.34 SEC
TAYLOR SERIES EXPANSION  110  500   -4    5  112
 TIME FOR TAYLOR SERIES EXPANSION =   0.62 SEC
TAYLOR SERIES EXPANSION  110  500   -4    6  112
 TIME FOR TAYLOR SERIES EXPANSION =   1.13 SEC
TAYLOR SERIES EXPANSION  110  500   -4    7  112
 TIME FOR TAYLOR SERIES EXPANSION =   1.97 SEC
TAYLOR SERIES EXPANSION  110  500   -4    8  112
 TIME FOR TAYLOR SERIES EXPANSION =   3.35 SEC


WRITE SERIES 500
LENGTH =   20

    +556403/322560*E**8*SIN(8M)
    -7913/4480*E**8*SIN(6M)
    +4123/11520*E**8*SIN(4M)
    +43/5760*E**8*SIN(2M)
    +47273/32256*E**7*SIN(7M)
    -5957/4608*E**7*SIN(5M)
    +95/512*E**7*SIN(3M)
    +107/4608*E**7*SIN(M)
    +1223/960*E**6*SIN(6M)
    -451/480*E**6*SIN(4M)
    +17/192*E**6*SIN(2M)
    +1097/960*E**5*SIN(5M)
    -43/64*E**5*SIN(3M)
    +5/96*E**5*SIN(M)
    +103/96*E**4*SIN(4M)
    -11/24*E**4*SIN(2M)
    +13/12*E**3*SIN(3M)
    -1/4*E**3*SIN(M)
    +5/4*E**2*SIN(2M)
    +2*E*SIN(M)
```

A Transcendental Equation

Kepler's equation relating the eccentric anomaly E to the mean anomaly M is M = E − ecc sin E. The program and solution of this equation for E as a function of M to sixth order in ecc are as follows.

```
      PROGRAM INVERT
C     IN "M=E-ECC*SIN(E)"  SOLVE FOR "E" AS A FUNCTION OF "M"
      COMMON/LPOLY/LPOLY(24)/LARG/LARG(8)
      COMMON/II/KFACT(24)/JJ/MORDER
      CALL START
      LPOLY(4)=3HECC
      LARG (1)=1HM
      CALL DE44(101,+1,1,0,0,0,1,-1,1,0,0,0)
      CALL DE44(102,+1,1,0,0,0,1,-1,1,0,0,0)
      KFACT(4)=1
      DO 700 MORDER=1,6
      CALL TAYLOR(102,101,-1,MORDER,103)
  700 CALL SWITCH(103,101)
      CALL ROUTLP(101)
      END
```

```
WRITE SERIES 101
LENGTH =    12

      +27/80*ECC**6*SIN(6M)
      +125/384*ECC**5*SIN(5M)
      -4/15*ECC**6*SIN(4M)
      +1/3*ECC**4*SIN(4M)
      -27/128*ECC**5*SIN(3M)
      +3/8*ECC**3*SIN(3M)
      +1/48*ECC**6*SIN(2M)
      -1/6*ECC**4*SIN(2M)
      +1/2*ECC**2*SIN(2M)
      +1/192*ECC**5*SIN(M)
      -1/8*ECC**3*SIN(M)
      +ECC*SIN(M)
```

UNCLASSIFIED

An Example of Polynomial Denominators

The following series shows how the output might appear if literal polynomial divisors are present:

```
WRITE SERIES  20
LENGTH =    10

    +1/8*A**2*B**4*C**6*D**8*SIN(12A+10B+8C+6D)
    (12AD+10BD+8CD+6DD)(12AZ+10BZ+8CZ+6ZZ)

    -9/8*A**2*B**2*C**12*D**4*COS(10A+5B+4C+5D)
    (10AD+5BD+4CD+5DD)(10AZ+5BZ+4CZ+5ZZ)

    -81/32*A**2*C**18*SIN(8A+4D)
    (8AD+4DD)(8AZ+4ZZ)

    -1/4*A**2*B**3*C**12*D**6*COS(7A+8B+3C+5D)
    (7AD+8BD+3CD+5DD)(7AZ+8BZ+3CZ+5ZZ)

    -2/3*A**5*B**5*C**5*D**5*COS(7A+7B+7C+7D)
    (7AD+7BD+7CD+7DD)(7AZ+7BZ+7CZ+7ZZ)

    +7/4*A**2*B**6*C**5*D**7*SIN(6A+5B+4C+6D)
    (6AD+5BD+4CD+6DD)(6AZ+5BZ+4CZ+6ZZ)

    +7/4*A**2*B**6*C**5*D**7*SIN(6A+5B+4C)
    (6AD+5BD+4CD)(6AZ+5BZ+4CZ)

    +2/3*A**5*B**5*C**5*D**5*COS(5A+3B+C-D)
    (5AD+3BD+CD-DD)(5AZ+3BZ+CZ-ZZ)

    -9/8*A**2*B*C**18*D**2*SIN(5A+3B-C+4D)
    (5AD+3BD-CD+4DD)(5AZ+3BZ-CZ+4ZZ)

    +1/4*A**2*B**3*C**12*D**6*COS(5A+2B+5C+D)
    (5AD+2BD+5CD+DD)(5AZ+2BZ+5CZ+ZZ)
```

In this example, one polynomial is referenced from the trigonometric node, and the other is referenced from the coefficient node. The terms in this example are examples of the most general terms that the system can represent thus far.

CONCLUDING REMARKS

The following concluding remarks are in order:

- It appears that algebraic manipulations of interest in classical perturbation theory may be easily performed on a high-speed computer using the FORTRAN language.

- The use of elementary list-processing techniques appears to be quite advantageous when manipulating large series internally to the machine. This method of storage management was tried along with a packed format representation (adjacent terms in a series stored in adjacent locations in memory). Whereas the latter method proved simpler in implementation, it proved to be nearly useless when larger scale calculations were performed which required the most efficient use of core storage.

- From experience it appears that for problems of small to intermediate size the one-dimensional representation appears to be superior to the more complicated two-dimensional scheme. However for large-scale problems and/or those requiring polynomial denominators, the latter method appears to be slightly advantageous.

- Extremely lengthy algebraic computations requiring 10 min or more of CPU time may require additional storage (disk or drum) for storage of intermediate results.

- The program can be run on any computer possessing a FORTRAN compiler and a 32-bit (or greater) word length with few modifications. It may be adapted to any machine with a shorter word length but with major modifications.

- The program will perform about 100 multiplications per second with no truncation on a PDP 11/780. For example, the product of a 50-term series with itself will require about 25 s assuming no truncation occurs. If substantial truncation takes place, the required time is greatly reduced. About 1 man-year's work of algebra can be reproduced in 1 minute of CPU time on a PDP 11/780.

ACKNOWLEDGMENTS

The program has been made available to several other researchers. Some have carried out major revisions in the code to suit their own individual needs and requirements. The author has in turn borrowed heavily from their suggested improvements and modifications. To this end the author is indebted to A. Deprit of the National Bureau of Standards, D. Richardson of the University of Cincinnati, V. A. Brumberg of the Institute for Theoretical Astronomy at Leningrad, USSR, Victor Slablinksi of Comsat, and K. Alfriend, S. Coffey, B. Kaufman, H. Pickard, and W. Harr of the Naval Research Laboratory.

UNCLASSIFIED

REFERENCES

1. R. Dasenbrock, "Algebraic Manipulation by Computer," NRL Report 7564, June 1973.

2. P. Herget and P. Musen, "The Calculation of Literal Expressions," Astron. J. **64**, 11-20 (Feb. 1959).

3. J.E. Sammet and E.R. Bond, "Introduction to FORMAC," IEEE Trans. Electron. Comput. **EC-13**, 386-394 (Aug. 1964).

4. M. Manove, S. Bloom, and G. Engelman, "Rational Functions in MATHLAB," Mitre Corporation, Bedford, Mass., MPT-35, Aug. 1966.

5. A.C. Hearn, "REDUCE User's Manual," Stanford Computational Center, Stanford, Calif., 1967.

6. "Proceedings of the 1977 MACSYMA User's Conference," held at Berkley Calif., July 27-29, 1977, NASA CP-2012.

7. A. Deprit, J. Henrad, and A. Rom, "Analytical Lunar Ephemeris, the Mean Motions," Boeing Scientific Research Laboratories, DI-82-0990, Aug. 1970.

8. T.C. Von Flandern and K.F. Pulkkinen, "Low-Precision Formulae for Planetary Positions," The Astrophysical Journal Supplement Series **41**, Nov. 1979.

9. W.H. Jefferys, "A FORTRAN-Based List Processor for Poisson Series," Celestial Mech. **2**(4), 474-480 (Nov. 1970).

10. R. Broucke, "A FORTRAN-IV System for the Manipulation of Symbolic Poisson Series with Applications to Celestial Mechanics," Univ. of Texas, IASOM TR 80-3, Jan. 1980.

11. D.R. Barton and J.P. Fitch, "Applications of Algebraic Manipulation Programs in Physics," Rep. Prog. Physics **35**, 235-314 (1972).

12. R.J. Fateman, "Is a LISP machine different from a FORTRAN machine?" SIGSAM **12** (3), Aug. 1978.

13. Mem. de L'Acad. des Sc., Vol. XXVIII.

14. A. Bawden et al., "LISP Machine Progress Report," MIT Artificial Intelligence Laboratory, Memo 444, Aug. 1977.

15. A. Deprit, "Never Say NO to a Computer," J. of Guidance and Control, **4**(6), 577, Nov. 1981.

16. A. Deprit, private communication.

17. B.A. Brumberg, "Algorithms of Celestial Mechanics," Vol. 1, Mathematical Software for Digital Computers, Institute of Theoretical Astronomy, Academy of Sciences, USSR, 1974.

18. D. Brouwer and G.M. Clemence, *Methods of Celestial Mechanics*, Academic Press, New York, 1961, p. 77.

Appendix A
PROGRAM LISTING

Listed below are subroutine START and, in alphabetical order, the most important routines comprised by the system. In the interest of conserving space, some of the more specialized subroutines are not included in the listing. Also, the two-dimensional version is not included. The complete package is available from the author.

```
      SUBROUTINE START
      IMPLICIT REAL*8 (A-H,O-Z)
   70 FORMAT (///)
   80 FORMAT(19H START OF EXECUTION)
      COMMON/AA/NEXT(1000)/BB/N(1000)/CC/M(1000)/DD/KTERM(9,1000)
      COMMON/HH/MSTART(1050)/HI/MAX(1050)/NC/NOCHOP
      COMMON/GH/NUTIL(30)/LL/INDEX/KORE/KORE
      COMMON/II/KFACT(24)/JJ/MORDER/KK/LZERO/UU/LMAP(1024)
      COMMON/MM/NOFACT/NN/NPRINT/NM/LPRINT/OO/NEXACT
      COMMON/RR/EPSLON/SS/NOFIX/IJ/KCHOP(24)/NT/NOTRUN
      COMMON/LPOLY/LPOLY(24)/LARG/LARG(8)
      LOGICAL NOFACT,NPRINT,LPRINT,NOFIX,NEXACT,NOTRUN,NOCHOP
      DATA NPY/6/,NTG/2/,NVAR/8/,NTIP/9/,NTGP/7/
      DATA INDEX/Z40808080/,KORE/1000/
      DATA LPOLY/1HA,1HB,1HC,1HD,1HE,1HF,1HG,1HH,
     *           1HI,1HJ,1HK,1HL,1HM,1HN,1HO,1HP,
     *           1HQ,1HR,1HS,1HT,1HU,1HV,1HW,1HX/
      DATA LARG /1HA,1HB,1HC,1HD,1HE,1HF,1HG,1HH/
      PRINT 80
      PRINT 70
      NOFACT=.FALSE.
      LPRINT=.TRUE.
      NPRINT=.TRUE
      NOFIX=.TRUE.
      NEXACT=.FALSE.
      NOTRUN=.FALSE.
      NOCHOP=.FALSE.
      EPSLON=1.0D-9
      MORDER=63
      LZERO=1
      DO 10 K=1,24
      KCHOP(K)=63
   10 KFACT(K)=0
      DO 30 K=1,30,1
   30 NUTIL(K)=0
      DO 40 K=1,1050,1
      MSTART(K)=0
   40 MAX(K)=0
      DO 50 K=1,KORE,1
```

UNCLASSIFIED

```
      NEXT(K)=K+1
      N(K)=0
      M(K)=+1
      KTERM(NTIP,K)=+1
      DO 50 LJ=1,NVAR
   50 KTERM(LJ,K)=INDEX
      NEXT(KORE)=0
      RETURN
      END


      SUBROUTINE ACCUM(ISA,ISB)
      IMPLICIT REAL*8 (A-H,O-Z)
  150 FORMAT(11H ACCUMULATE,I4,3H TO,I4)
  160 FORMAT(21H TIME TO ACCUMULATE =,F7.2,4H SEC)
      COMMON/AA/NEXT(1)/BB/N(1)/CC/M(1)/DD/KTERM(9,1)
      COMMON/HH/MSTART(1)/HI/MAX(1)/KK/LZERO
      COMMON/LL/INDEX/NN/NPRINT
      LOGICAL NPRINT
      DATA NPY/6/,NTG/2/,NVAR/8/,NTIP/9/,NTGP/7/
      IF (ISA.EQ.ISB) CALL MULCON(ISA,2,1)
      IF (ISA.EQ.ISB) RETURN
      LT=0
      KA=MSTART(ISA)
      KB=MSTART(ISB)
      IF (KB.LE.0) RETURN
      IF (KA.LE.0) CALL SWITCH(ISA,ISB)
      IF (KA.LE.0) RETURN
      XNB=TMLEFT(MS)
      ISC=ISA
   10 DO 20 LX=1,NTIP
      IF (KTERM(LX,KB)-KTERM(LX,KA)) 40,20,30
   20 CONTINUE
      IF (LT.LE.0) MSTART(ISC)=KA
      IF (LT.GT.0) NEXT(LT)=KA
      LT=KA
      NA=NEXT(KA)
      NB=NEXT(KB)
      CALL REFACT(N(KA),M(KA),N(KB),M(KB),1,1,N(KA),M(KA),0)
      N(KB)=0
      M(KB)=1
      DO 25 LX=1,NVAR
   25 KTERM(LX,KB)=INDEX
      KTERM(NTIP,KB)=+1
      NEXT(KB)=LZERO
      LZERO=KB
      KA=NA
      KB=NB
      IF (KA.LE.0) GO TO 60
      IF (KB.LE.0) GO TO 50
      GO TO 10
```

```
 30 IF (LT.LE.0) MSTART(ISC)=KB
    IF (LT.GT.0) NEXT(LT)=KB
    LT=KB
    KB=NEXT(KB)
    IF (KB.GT.0) GO TO 10
    GO TO 50
 40 IF (LT.LE.0) MSTART(ISC)=KA
    IF (LT.GT.0) NEXT(LT)=KA
    LT=KA
    KA=NEXT(KA)
    IF (KA.GT.0) GO TO 10
    GO TO 60
 50 NEXT(LT)=KA
    MAX(ISC)=MAX(ISA)
    GO TO 70
 60 NEXT(LT)=KB
    MAX(ISC)=MAX(ISB)
 70 CALL LINK(ISC)
    MSTART(ISB)=0
    MAX(ISB)=0
    IF (NPRINT) RETURN
    XNA=TMLEFT(MS)
    XNT=XNB-XNA
    PRINT 150,ISB,ISA
    PRINT 160,XNT
    RETURN
    END


    SUBROUTINE ADD(ISA,ISB,ISC)
    IMPLICIT REAL*8 (A-H,O-Z)
150 FORMAT(4H ADD,I4,3H TO,I4,10H RESULT IN,I4)
160 FORMAT(14H TIME TO ADD =,F7.2,4H SEC)
170 FORMAT(24H STORAGE OVERFLOW IN ADD)
    COMMON/AA/NEXT(1)/BB/N(1)/CC/M(1)/DD/KTERM(9,1)
    COMMON/HH/MSTART(1)/HI/MAX(1)/KK/LZERO
    COMMON/LL/INDEX/NN/NPRINT
    LOGICAL NPRINT
    DATA NPY/6/,NTG/2/,NVAR/8/,NTIP/9/,NTGP/7/
    IF (ISA.EQ.ISB) CALL TRANSF(ISA,ISC)
    IF (ISA.EQ.ISB) CALL MULCON(ISC,2,1)
    IF (ISA.EQ.ISB) RETURN
    IF (ISA.EQ.ISC) CALL ADDACC(ISC,ISB)
    IF (ISB.EQ.ISC) CALL ADDACC(ISC,ISA)
    IF (ISA.EQ.ISC.OR.ISB.EQ.ISC) RETURN
    LT=0
    KA=MSTART(ISA)
    KB=MSTART(ISB)
    IF (KA.GT.0.AND.KB.LE.0) CALL TRANSF(ISA,ISC)
    IF (KA.LE.0.AND.KB.GT.0) CALL TRANSF(ISB,ISC)
    IF (KA.GT.0.AND.KB.LE.0) RETURN
```

UNCLASSIFIED

```
      IF (KA.LE.0.AND.KB.GT.0) RETURN
      CALL ZERO(ISC)
      IF (KA.LE.0.AND.KB.LE.0) RETURN
      XNB=TMLEFT(MS)
   10 DO 20 LX=1,NTIP
      IF (KTERM(LX,KB)-KTERM(LX,KA)) 40,20,30
   20 CONTINUE
      NT=LZERO
      IF (NT.LE.0) GO TO 200
      LZERO=NEXT(NT)
      NEXT(NT)=0
      CALL REFACT(N(KA),M(KA),N(KB),M(KB),1,1,N(NT),M(NT),0)
      DO 110 LX=1,NTIP
  110 KTERM(LX,NT)=KTERM(LX,KA)
      KA=NEXT(KA)
      KB=NEXT(KB)
      IF (N(NT).EQ.0) GO TO 120
      IF (LT.LE.0) MSTART(ISC)=NT
      IF (LT.GT.0) NEXT(LT)=NT
      LT=NT
      GO TO 140
  120 M(NT)=1
      KTERM(NTIP,NT)=+1
      DO 130 LX=1,NVAR
  130 KTERM(LX,NT)=INDEX
      NEXT(NT)=LZERO
      LZERO=NT
      GO TO 140
   30 LC=KB
      KB=NEXT(KB)
      GO TO 100
   40 LC=KA
      KA=NEXT(KA)
      GO TO 100
  100 NT=LZERO
      IF (NT.LT.0) GO TO 200
      LZERO=NEXT(NT)
      NEXT(NT)=0
      IF (LT.LE.0) MSTART(ISC)=NT
      IF (LT.GT.0) NEXT(LT)=NT
      N(NT)=N(LC)
      M(NT)=M(LC)
      DO 111 LX=1,NTIP
  111 KTERM(LX,NT)=KTERM(LX,LC)
      LT=NT
  140 IF (KA.GT.0.AND.KB.GT.0) GO TO 10
      IF (KA.LE.0.AND.KB.GT.0) GO TO 30
      IF (KA.GT.0.AND.KB.LE.0) GO TO 40
      MAX(ISC)=NT
      CALL LINK(ISC)
      IF (NPRINT) RETURN
```

```
      XNA=TMLEFT(MS)
      XNT=XNB-XNA
      PRINT 150,ISA,ISB,ISC
      PRINT 160,XNT
      RETURN
  200 PRINT 170
      CALL QUIT
      STOP
      END


      SUBROUTINE ADDACC(ISA,ISB)
      IMPLICIT REAL*8 (A-H,O-Z)
  150 FORMAT(11H ACCUMULATE,I4,3H TO,I4)
  160 FORMAT(14H TIME TO ADD =,F7.2,4H SEC)
  170 FORMAT(27H STORAGE OVERFLOW IN ADDACC)
      COMMON/AA/NEXT(1)/BB/N(1)/CC/M(1)/DD/KTERM(9,1)
      COMMON/HH/MSTART(1)/HI/MAX(1)/KK/LZERO
      COMMON/LL/INDEX/NN/NPRINT
      LOGICAL NPRINT
      DATA NPY/6/,NTG/2/,NVAR/8/,NTIP/9/,NTGP/7/
      IF (ISA.EQ.ISB) CALL MULCON(ISA,2,1)
      IF (ISA.EQ.ISB) RETURN
      KA=MSTART(ISA)
      KB=MSTART(ISB)
      IF (KA.LE.0.AND.KB.GT.0) CALL TRANSF(ISB,ISA)
      IF (KA.LE.0.AND.KB.GT.0) RETURN
      IF (KA.GT.0.AND.KB.LE.0) RETURN
      IF (KA.LE.0.AND.KB.LE.0) RETURN
      XNB=TMLEFT(MS)
      MSTART(ISA)=0
      MAX(ISA)=0
      LT=0

   10 DO 20 LX=1,NTIP
      IF (KTERM(LX,KB)-KTERM(LX,KA)) 40,20,30
   20 CONTINUE
      NT=KA
      CALL REFACT(N(KA),M(KA),N(KB),M(KB),1,1,N(NT),M(NT),0)
      KA=NEXT(KA)
      KB=NEXT(KB)
      NEXT(NT)=0
      IF (N(NT).EQ.0) GO TO 24
      IF (LT.LE.0) MSTART(ISA)=NT
      IF (LT.GT.0) NEXT(LT)=NT
      LT=NT
      GO TO 100
   24 M(NT)=1
      KTERM(NTIP,NT)=+1
      DO 25 LX=1,NVAR
   25 KTERM(LX,NT)=INDEX
```

UNCLASSIFIED

```
      NEXT(NT)=LZERO
      LZERO=NT
      GO TO 100
   30 NT=LZERO
      IF (NT.LE.0) GO TO 200
      LZERO=NEXT(NT)
      NEXT(NT)=0
      IF (LT.LE.0) MSTART(ISA)=NT
      IF (LT.GT.0) NEXT(LT)=NT
      N(NT)=N(KB)
      M(NT)=M(KB)
      DO 35 LX=1,NTIP
   35 KTERM(LX,NT)=KTERM(LX,KB)
      KB=NEXT(KB)
      LT=NT
      GO TO 100
   40 NT=KA
      IF (LT.LE.0) MSTART(ISA)=NT
      IF (LT.GT.0) NEXT(LT)=NT
      KA=NEXT(KA)
      NEXT(NT)=0
      LT=NT
      GO TO 100
  100 IF (KA.GT.0.AND.KB.GT.0) GO TO 10
      IF (KA.LE.0.AND.KB.GT.0) GO TO 30
      IF (KA.GT.0.AND.KB.LE.0) GO TO 40
      MAX(ISA)=LT
      CALL LINK(ISA)
      IF (NPRINT) RETURN
      XNA=TMLEFT(MS)
      XNT=XNB-XNA
      PRINT 150,ISB,ISA
      PRINT 160,XNT
      RETURN
  200 PRINT 170
      CALL QUIT
      STOP
      END


      SUBROUTINE QUIT
      RETURN
      END


      SUBROUTINE CHOOSE(ISA,ISB,IVARIB,NUM)
      IVVV=-IABS(IVARIB)
      CALL COMBIN(ISA,ISB,IVVV,NUM,5)
      RETURN
      END
```

```
      SUBROUTINE COLECT(ISA,KEYC)
   20 FORMAT (33H **** CALL GARBAGE COLLECTOR ****,5X,2I8)
   30 FORMAT (39H **** STORAGE EXCEEDED IN MULTIPLY ****)
      COMMON/VV/NUMPAR,KOUNT,KOLECT,KSCAN/NN/NPRINT
      COMMON/TT/NTOTAL/NUMBER/NUMBER
      LOGICAL NPRINT,KSCAN
      IF (KEYC.EQ.0) GO TO 10
      NUMBER=NUMBER+1
      NTA=KOUNT      IF (NUMBER.LT.10) GO TO 10
      PRINT 30
      CALL QUIT
      STOP
   10 CALL LINK (ISA)
      IF (.NOT.NPRINT) PRINT 20,NTA,NTOTAL
      RETURN
      END


      SUBROUTINE COMBIN(ISA,ISB,IVARIB,NUM,KFUN)
  101 FORMAT(29H RETAIN ONLY PERODIC TERMS IN,I3)
  102 FORMAT(30H RETAIN ONLY CONSTANT TERMS IN,I3)
  103 FORMAT(16H RETAIN TERMS IN,I3,9H TO ORDER,I3,4H WRT,I3)
  104 FORMAT(16H RETAIN TERMS IN,I3,9H OF POWER,I3,4H WRT,I3)
  105 FORMAT(16H RETAIN TERMS IN,I3,15H OF PERIODICITY,I3,4H WRT,I3)
  109 FORMAT(16H RETAIN TERMS IN,I4,19H WITH DEPENDENCE ON,I4)
  106 FORMAT(17H  STORE RESULT IN,I4)
  107 FORMAT(27H STORAGE OVERFLOW IN COMBIN)
      COMMON/AA/NEXT(1)/BB/N(1)/CC/M(1)/DD/KTERM(9,1)
      COMMON/HH/MSTART(1)/HI/MAX(1)
      COMMON/KK/LZERO/LL/INDEX/NN/NPRINT
      LOGICAL NPRINT
      DIMENSION IA(4)
      DATA NPY/6/,NTG/2/,NVAR/8/,NTIP/9/,NTGP/7/
      IF (ISA.NE.ISB) CALL ZERO(ISB)
      IF (NPRINT) GO TO 10
      IF (KFUN.EQ.1) PRINT 101,ISA
      IF (KFUN.EQ.2) PRINT 102,ISA
      IF (KFUN.EQ.3) PRINT 103,ISA,NUM,IVARIB
      IF (KFUN.EQ.4) PRINT 104,ISA,NUM,IVARIB
      IF (KFUN.EQ.5) PRINT 105,ISA,NUM,IVARIB
      IF (KFUN.EQ.6) PRINT 109,ISA,IVARIB
      PRINT 106,ISB
   10 K=MSTART(ISA)
      IF (K.LE.0) RETURN
      IF (ISA.EQ.ISB) GO TO 60
      J=LZERO
      MSTART(ISB)=LZERO
   20 JN=J
      GO TO (21,21,23,23,23,23),KFUN
   21 DO 26 LJ=NTGP,NVAR
      IF (KFUN.EQ.1.AND.KTERM(LJ,K).NE.INDEX) GO TO 36
```

UNCLASSIFIED

```
      IF (KFUN.EQ.2.AND.KTERM(LJ,K).NE.INDEX) GO TO 38
   26 CONTINUE
      IF (KFUN.EQ.1) GO TO 38
      IF (KFUN.EQ.2) GO TO 36
   23 JW=(IABS(IVARIB)+3)/4
      IF (IVARIB.LT.0) JW=JW+NPY
      CALL UNPACK(KTERM(JW,K),IA)
      JV=MOD(IABS(IVARIB),4)
      IF (JV.EQ.0) JV=4
      IF (KFUN.EQ.3.AND.IA(JV).LE.NUM) GO TO 36
      IF (KFUN.EQ.4.AND.IA(JV).EQ.NUM) GO TO 36
      IF (KFUN.EQ.5.AND.IA(JV).EQ.NUM) GO TO 36
      IF (KFUN.EQ.6.AND.IA(JV).NE.0) GO TO 36
      GO TO 38
   36 N(J)=N(K)
      M(J)=M(K)
      DO 37 LJ=1,NTIP
   37 KTERM(LJ,J)=KTERM(LJ,K)
      JN=NEXT(J)
   38 KN=NEXT(K)
      IF (JN.GT.0) GO TO 40
      PRINT 107
      CALL QUIT
      STOP
   40 IF (KN.EQ.0) GO TO 50
      J=JN
      K=KN
      GO TO 20
   50 LZERO=NEXT(J)
      NEXT(J)=0
      MAX(ISB)=J
      CALL LINK(ISB)
      RETURN
   60 GO TO (61,61,63,63,63,63),KFUN
   61 DO 66 LJ=NTGP,NVAR
      IF (KFUN.EQ.1.AND.KTERM(LJ,K).NE.INDEX) GO TO 90
      IF (KFUN.EQ.2.AND.KTERM(LJ,K).NE.INDEX) GO TO 80
   66 CONTINUE
      IF (KFUN.EQ.1) GO TO 80
      IF (KFUN.EQ.2) GO TO 90
   63 JW=(IABS(IVARIB)+3)/4
      IF (IVARIB.LT.0) JW=JW+NPY
      CALL UNPACK(KTERM(JW,K),IA)
      JV=MOD(IABS(IVARIB),4)
      IF (JV.EQ.0) JV=4
      IF (KFUN.EQ.3.AND.IA(JV).LE.NUM) GO TO 90
      IF (KFUN.EQ.4.AND.IA(JV).EQ.NUM) GO TO 90
      IF (KFUN.EQ.5.AND.IA(JV).EQ.NUM) GO TO 90
      IF (KFUN.EQ.6.AND.IA(JV).NE.0) GO TO 90
   80 N(K)=0
   90 K=NEXT(K)
```

```
      IF (K.GT.0) GO TO 60
      CALL LINK(ISA)
      RETURN
      END


      SUBROUTINE CUT(ISA,ISB)
      IVARIB=0
      NUM=0
      CALL COMBIN(ISA,ISB,IVARIB,NUM,1)
      RETURN
      END


      SUBROUTINE DEFONE(ISA)
      COMMON/AA/NEXT(1)/BB/N(1)/CC/M(1)/DD/KTERM(9,1)
      COMMON/HH/MSTART(1)/HI/MAX(1)/KK/LZERO/LL/INDEX
      DATA NPY/6/,NTG/2/,NVAR/8/,NTIP/9/,NTGP/7/
      CALL ZERO(ISA)
      MSTART(ISA)=LZERO
      MAX(ISA)=LZERO
      K=LZERO
      N(K)=1
      M(K)=1
      DO 10 LJ=1,NVAR
   10 KTERM(LJ,K)=INDEX
      KTERM(NTIP,K)=1
      LZERO=NEXT(K)
      NEXT(K)=0
      RETURN
      END


      SUBROUTINE DERINT(ISA,IV,ISB,KFUN)
   40 FORMAT (14H DIFFERENTIATE,I3,7H W.R.T.,I3,10H RESULT IN,I3)
   45 FORMAT (27H STORAGE OVERFLOW IN DERINT)
   50 FORMAT (10H INTEGRATE,I3,7H W.R.T.,I3,10H RESULT IN,I3)
      COMMON/AA/NEXT(1)/BB/N(1)/CC/M(1)/DD/KTERM(9,1)
      COMMON/GH/NUTIL(1)/HH/MSTART(1)/HI/MAX(1)/NN/NPRINT
      COMMON/KK/LZERO
      LOGICAL NPRINT
      DIMENSION IA(4),IB(4)
      DATA NPY/6/,NTG/2/,NVAR/8/,NTIP/9/,NTGP/7/
      IF (KFUN.EQ.1.AND..NOT.NPRINT) PRINT 40,ISA,IV,ISB
      IF (KFUN.EQ.2.AND..NOT.NPRINT) PRINT 50,ISA,IV,ISB
      IF (KFUN.EQ.1.AND.ISA.NE.ISB)  GO TO 100
      CALL TRANSF(ISA,ISB)
      K=MSTART(ISB)
      IF (K.LE.0) RETURN
      IF (IV.LT.0) GO TO 30
      JW=(IABS(IV)+3)/4
```

UNCLASSIFIED

```
      JV=MOD(IABS(IV),4)
      IF (JV.EQ.0) JV=4
   10 CALL UNPACK(KTERM(JW,K),IA)
      IF (KFUN.EQ.1) GO TO 20
      IF (KFUN.EQ.2) GO TO 25
   20 KCON=IA(JV)
      IA(JV)=IA(JV)-1
      CALL REFACT (N(K),M(K),1,1,KCON,1,N(K),M(K),1)
      NUTIL(12)=NUTIL(12)+1
      GO TO 28
   25 IA(JV)=IA(JV)+1
      KCON=IA(JV)
      CALL REFACT (N(K),M(K),1,1,1,KCON,N(K),M(K),1)
      NUTIL(13)=NUTIL(13)+1
   28 CALL REPACK(IA,KTERM(JW,K))
      K=NEXT(K)
      IF (K.GT.0) GO TO 10
      CALL LINK(ISB)
      RETURN
   30 JW=(IABS(IV)+3)/4+NPY
      JV=MOD(IABS(IV),4)
      IF (JV.EQ.0) JV=4
   31 CALL UNPACK(KTERM(JW,K),IB)
      IF (KFUN.EQ.1) GO TO 32
      IF (KFUN.EQ.2) GO TO 34
   32 KCON=-IB(JV)*KTERM(NTIP,K)
      CALL REFACT (N(K),M(K),1,1,KCON,1,N(K),M(K),1)
      NUTIL(12)=NUTIL(12)+1
      GO TO 36
   34 KCON=+IB(JV)*KTERM(NTIP,K)
      CALL REFACT (N(K),M(K),1,1,1,KCON,N(K),M(K),1)
      NUTIL(13)=NUTIL(13)+1
   36 KTERM(NTIP,K)=-KTERM(NTIP,K)
      K=NEXT(K)
      IF (K.GT.0) GO TO 31
      CALL LINK (ISB)
      RETURN
  100 CALL ZERO(ISB)
      K=MSTART(ISA)
      IF (K.LE.0) RETURN
      MSTART(ISB)=LZERO
      J=MSTART(ISB)
      IF (IV.LT.0) GO TO 80
      JW=(IABS(IV)+3)/4
      JV=MOD(IABS(IV),4)
      IF (JV.EQ.0) JV=4
   60 N(J)=N(K)
      M(J)=M(K)
      DO 62 LJ=1,NTIP
   62 KTERM(LJ,J)=KTERM(LJ,K)
      CALL UNPACK(KTERM(JW,J),IA)
```

```
      KCON=IA(JV)
      IA(JV)=IA(JV)-1
      CALL REFACT (N(J),M(J),1,1,KCON,1,N(J),M(J),1)
      NUTIL(12)=NUTIL(12)+1
      CALL REPACK(IA,KTERM(JW,J))
      JN=NEXT(J)
      KN=NEXT(K)
      IF (KN.LE.0) GO TO 90
      IF (JN.LE.0) GO TO 92
      IF (N(J).NE.0) J=JN
      K=KN
      GO TO 60
   80 JW=(IABS(IV)+3)/4+NPY
      JV=MOD(IABS(IV),4)
      IF (JV.EQ.0) JV=4
   81 N(J)=N(K)
      M(J)=M(K)
      DO 83 LJ=1,NTIP
   83 KTERM(LJ,J)=KTERM(LJ,K)
      CALL UNPACK(KTERM(JW,J),IB)
      KCON=-IB(JV)*KTERM(NTIP,J)
      CALL REFACT (N(J),M(J),1,1,KCON,1,N(J),M(J),1)
      NUTIL(12)=NUTIL(12)+1
      KTERM(NTIP,J)=-KTERM(NTIP,J)
      JN=NEXT(J)
      KN=NEXT(K)
      IF (KN.LE.0) GO TO 90
      IF (JN.LE.0) GO TO 92
      IF (N(J).NE.0) J=JN
      K=KN
      GO TO 81
   90 MAX(ISB)=J
      LZERO=NEXT(J)
      NEXT(J)=0
      CALL LINK(ISB)
      RETURN
   92 PRINT 45
      CALL QUIT
      STOP
      END


      SUBROUTINE DERIV(ISA,IV,ISB)
      CALL DERINT(ISA,IV,ISB,1)
      RETURN
      END
```

UNCLASSIFIED

```
      SUBROUTINE DE44(ISA,NUM,NDEMON,I1,I2,I3,I4,
     *                KTRIG,J1,J2,J3,J4)
      COMMON/AA/NEXT(1)/BB/N(1)/CC/M(1)/DD/KTERM(9,1)
      COMMON/HH/MSTART(1)/HI/MAX(1)/KK/LZERO/LL/INDEX
      DIMENSION IP(4),IA(4)
      DATA NPY/6/,NTG/2/,NVAR/8/,NTIP/9/,NTGP/7/
      IP(1)=I1
      IP(2)=I2
      IP(3)=I3
      IP(4)=I4
      IA(1)=J1
      IA(2)=J2
      IA(3)=J3
      IA(4)=J4
      K=LZERO
      LZERO=NEXT(K)
      NEXT(K)=0
      N(K)=NUM
      M(K)=NDEMON
      KTERM(NTIP,K)=KTRIG
      DO 10 LJ=1,NVAR
   10 KTERM(LJ,K)=INDEX
      CALL REPACK(IP,KTERM(1,K))
      CALL REPACK(IA,KTERM(7,K))
      CALL SETUP(0,0)
      CALL INSERT(ISA,K)
      RETURN
      END


      SUBROUTINE ERASE (ISA,IV)
   20 FORMAT (6H ERASE,I3,4H WRT,I3)
      COMMON/AA/NEXT(1)/BB/N(1)/CC/M(1)/DD/KTERM(9,1)
      COMMON/HH/MSTART(1)/HI/MAX(1)/NN/NPRINT
      LOGICAL NPRINT
      DIMENSION IAA(4)
      IF (.NOT.NPRINT) PRINT 20, ISA,IV
      K=MSTART(ISA)
      IF (K.EQ.0) RETURN
      JW=(IV+3)/4
      JV=MOD(IABS(IV),4)
      IF (JV.EQ.0) JV=4
   10 CALL UNPACK(KTERM(JW,K),IAA)
      IAA(JV)=0
      CALL REPACK (IAA,KTERM(JW,K))
      K=NEXT(K)
      IF (K.GT.0) GO TO 10
      CALL SIMP(ISA)
      RETURN
      END
```

```
      SUBROUTINE EXPAND (ISA,IORDER,IN,IM,ISB)
   30 FORMAT (12H EXPAND (1 +,I3,4H)**(,I2,1H/,I2,1H))
   40 FORMAT (10H  TO ORDER,I3,3H IN,I3)
   50 FORMAT (18H  RESULT STORED IN,I3)
   60 FORMAT (14H  TOTAL TIME =,F7.2,4H SEC)
      COMMON/NN/NPRINT/NM/LPRINT/LL/INDEX
      LOGICAL NPRINT,LPRINT,PNPRNT
      XNB=TMLEFT(MS)
      PNPRNT=NPRINT
      IF (LPRINT) NPRINT=.TRUE.
      PRINT 30, ISA,IN,IM
      PRINT 40, IORDER,ISA
      PRINT 50, ISB
      CALL ZERO (ISB)
      ISC=1003
      ISD=1004
      CALL ZERO (ISC)
      CALL ZERO (ISD)
      CALL DEFONE(ISB)
      CALL DEFONE(ISC)
      IF (IN.EQ.0) GO TO 20
      IF (IORDER.EQ.0) GO TO 20
      JN=IN
      JM=IM
      DO 10 K=1,IORDER,1
      JN=IN-(K-1)*IM
      JM=K*IM
      CALL FACTOR (JN,JM)
      CALL MULT (ISA,ISC,ISD)
      CALL ZERO (ISC)
      CALL MULCON (ISD,JN,JM)
      IF (K.LT.IORDER) CALL ADD (ISB,ISD,ISB)
      IF (K.EQ.IORDER) CALL ACCUM (ISB,ISD)
   10 CALL SWITCH (ISD,ISC)
   20 CALL ZERO (ISC)
      CALL ZERO (ISD)
      IF (LPRINT) NPRINT=PNPRNT
      XNA=TMLEFT(MS)
      XNT=XNB-XNA
      PRINT 60, XNT
      RETURN
      END
```

UNCLASSIFIED

```
      SUBROUTINE FACTOR (NN,MM)
      IF (NN.EQ.0) MM=1
      KA=NN
      KB=MM
   10 KC=MOD(KA,KB)
      IF (KC.EQ.0) GO TO 20
      KA=KB
      KB=KC
      GO TO 10
   20 KF=IABS(KB)
      NN=NN/KF
      MM=MM/KF
      RETURN
      END


      SUBROUTINE INSERT (ISA,KTA)
      IMPLICIT REAL*8 (A-H,O-Z)
  260 FORMAT (34H POINTER NEGATIVE DURING PARTITION)
  270 FORMAT (27H POINTER NEGATIVE IN INSERT)
      COMMON/AA/NEXT(1)/BB/N(1)/CC/M(1)/DD/KTERM(9,1)
      COMMON/GH/NUTIL(1)/HH/MSTART(1)/HI/MAX(1)/KK/LZERO/LL/INDEX
      COMMON/TT/NTOTAL/UU/LMAP(1)/VV/NUMPAR,KOUNT,KOLECT,KSCAN
      LOGICAL KSCAN
      DATA NPY/6/,NTG/2/,NVAR/8/,NTIP/9/,NTGP/7/
      NUTIL(17)=NUTIL(17)+1
      CALL SWIFNC(KTA)
      IF (MSTART(ISA).GT.0) GO TO 10
      MSTART(ISA)=KTA
      MAX(ISA)=KTA
      NEXT(KTA)=0
      KOUNT=1
      RETURN
   10 IF (N(KTA).EQ.0) GO TO 190
      IF (KOUNT.LT.KOLECT) GO TO 12
      CALL COLECT(ISA,0)
      CALL SETUP(NTOTAL,1)
      KOLECT=KOLECT+300
   12 IF (NUMPAR.GE.1024) GO TO 40
      IPAR=16*NUMPAR
      IF (KOUNT.LE.IPAR) GO TO 40
      KSCAN=.FALSE.
      NUMPAR=2*NUMPAR
      XSUM=0.0D0
      YSUM=0.0D0
      XKOUNT=KOUNT
      DXK=XKOUNT/NUMPAR
      KS=MSTART(ISA)
      DO 30 MX=1,NUMPAR
      LMAP(MX)=KS
      XSUM=XSUM+DXK
```

```
      IF (MX.EQ.NUMPAR) GO TO 30
      DO 20 MY=1,500
      IF (YSUM.GE.XSUM) GO TO 30
      YSUM=YSUM+1.0D0
      IF (KS.LE.0) GO TO 230
   20 KS=NEXT(KS)
   30 CONTINUE
   40 MSB=MSTART(ISA)
      L=MSTART(ISA)
      LAST=0
      IF (KSCAN) GO TO 130
      MX=NUMPAR/2
      LX=LMAP(MX)
      MDX=NUMPAR/4
   50 DO 60 LJ=1,NTIP
      IF (KTERM(LJ,LX)-KTERM(LJ,KTA)) 80,60,100
   60 CONTINUE
      GO TO 90
   80 IF (MDX.EQ.0) MDX=1
      MX=MX-MDX
      MDX=MDX/2
      IF (MX.LE.0) GO TO 110
      LX=LMAP(MX)
      GO TO 50
   90 L=LX
      GO TO 180
  100 IF (MDX.EQ.0) GO TO 120
      MX=MX+MDX
      LX=LMAP(MX)
      MDX=MDX/2
      GO TO 50
  110 LX=MSB
  120 L=LX
  130 DO 140 LJ=1,NTIP
      IF (KTERM(LJ,L)-KTERM(LJ,KTA)) 160,140,200
  140 CONTINUE
      GO TO 180
  160 IF (LAST.NE.0) GO TO 170
      MSTART(ISA)=KTA
      NEXT(KTA)=L
      KOUNT=KOUNT+1
      RETURN
  170 NEXT(LAST)=KTA
      NEXT(KTA)=L
      KOUNT=KOUNT+1
      RETURN
  180 CALL REFACT (N(KTA),M(KTA),N(L),M(L),1,1,N(L),M(L),0)
  190 N(KTA)=0
      M(KTA)=1
      KTERM(NTIP,KTA)=1
      DO 195 LJ=1,NVAR
```

UNCLASSIFIED

```
195 KTERM(LJ,KTA)=INDEX
    NEXT(KTA)=LZERO
    LZERO=KTA
    NUTIL(5)=NUTIL(5)+1
    RETURN
200 LN=NEXT(L)
    IF (LN) 240,210,220
210 MAX(ISA)=KTA
    NEXT(L)=KTA
    NEXT(KTA)=0
    KOUNT=KOUNT+1
    RETURN
220 LAST=L
    L=LN
    NUTIL(15)=NUTIL(15)+1
    GO TO 130
230 PRINT 260
    CALL QUIT
    STOP
240 PRINT 270
    CALL QUIT
    STOP
    END


    SUBROUTINE INTEG(ISA,IV,ISB)
    CALL DERINT(ISA,IV,ISB,2)
    RETURN
    END


    FUNCTION IRGSGN(K)
    COMMON/DD/KTERM(9,1)/LL/INDEX
    DATA NPY/6/,NTG/2/,NVAR/8/,NTIP/9/,NTGP/7/
    DO 100 LJ=NTGP,NVAR
    IF (KTERM(LJ,K)-INDEX) 200,100,300
100 CONTINUE
    IRGSGN=0
    RETURN
200 IRGSGN=-1
    RETURN
300 IRGSGN=+1
    RETURN
    END
```

```
      FUNCTION KEVEN(L)
      J=L/2
      M=J*2
      IF (M.EQ.L) KEVEN=1
      IF (M.NE.L) KEVEN=-1
      RETURN
      END


      SUBROUTINE LINK (ISA)
      COMMON/AA/NEXT(1)/BB/N(1)/CC/M(1)/DD/KTERM(9,1)
      COMMON/GH/NUTIL(1)/HH/MSTART(1)/HI/MAX(1)
      COMMON/KK/LZERO/LL/INDEX/TT/NTOTAL
      DATA NPY/6/,NTG/2/,NVAR/8/,NTIP/9/,NTGP/7/
      K=MSTART(ISA)
      MSA=MSTART(ISA)
      NTOTAL=0
      LNZP=0
      IF (MSA.GT.0) GO TO 10
      MAX(ISA)=0
      RETURN
   10 MSA=0
   20 NADDR=NEXT(K)
      CALL SWIFNC(K)
      IF (N(K).EQ.0) GO TO 30
      NTOTAL=NTOTAL+1
      LNZP=K
      IF (MSA.EQ.0) MSA=K
      IF (NADDR.EQ.0) GO TO 40
      K=NADDR
      GO TO 20
   30 IF (LNZP.GT.0) NEXT(LNZP)=NEXT(K)
      KN=NEXT(K)
      NEXT(K)=LZERO
      LZERO=K
      M(K)=1
      KTERM(NTIP,K)=1
      DO 34 LJ=1,NVAR
   34 KTERM(LJ,K)=INDEX
      NUTIL(4)=NUTIL(4)+1
      IF (KN.EQ.0) GO TO 40
      K=KN
      GO TO 20
   40 MSTART(ISA)=MSA
      MAX(ISA)=LNZP
      IF (LNZP.GT.0) NEXT(LNZP)=0
      RETURN
      END
```

UNCLASSIFIED

```
      SUBROUTINE MULCON(ISA,ICONST,JCONST)
   20 FORMAT (9H MULTIPLY,I3,3H BY,I6,1H/,I6)
      COMMON/AA/NEXT(1)/BB/N(1)/CC/M(1)/DD/KTERM(9,1)
      COMMON/HH/MSTART(1)/HI/MAX(1)/NN/NPRINT/NM/LPRINT
      LOGICAL NPRINT,LPRINT
      ICONS=ICONST
      JCONS=JCONST
      CALL FACTOR (ICONS,JCONS)
      IF (.NOT.NPRINT) PRINT 20, ISA,ICONS,JCONS
      IF (ICONS.EQ.0) CALL ZERO (ISA)
      K=MSTART(ISA)
      IF (K.LE.0) RETURN
   10 CALL REFACT (N(K),M(K),1,1,ICONS,JCONS,N(K),M(K),1)
      K=NEXT(K)
      IF (K.GT.0) GO TO 10
      RETURN
      END


      SUBROUTINE MULONE(ISA)
      COMMON/AA/NEXT(1)/BB/N(1)/CC/M(1)/DD/KTERM(9,1)
      COMMON/HH/MSTART(1)/HI/MAX(1)
      COMMON/NT/NOTRUN/IJ/KCHOP(1)
      COMMON/II/KFACT(1)/JJ/MORDER
      DIMENSION IA(24),IB(24)
      LOGICAL NOTRUN
      DATA NPY/6/,NTG/2/,NVAR/8/,NTIP/9/,NTGP/7/
      NPY4=4*NPY
      K=MSTART(ISA)
      IF (K.LE.0) RETURN
   10 DO 20 LJ=1,NPY
   20 CALL UNPACK(KTERM(LJ,K),IA(4*LJ-3))
      KH=0
      DO 30 LJ=1,NPY4
   30 KH=KH+IA(LJ)*KFACT(LJ)
      IF (KH.GT.MORDER) N(K)=0
      DO 40 KV=1,NPY4
      IB(KV)=IA(KV)-KCHOP(KV)
      IF (IB(KV).GT.0) N(K)=0
   40 CONTINUE
      K=NEXT(K)
      IF (K.GT.0) GO TO 10
      CALL LINK(ISA)
      RETURN
      END
```

```
      SUBROUTINE MULTX(ISA,ISB,ISC)
  350 FORMAT (9H MULTIPLY,3I4,9H TO ORDER,I3)
  370 FORMAT (10H  LENGTH =,I5,9H  TIME  =,F7.2)
      COMMON/AA/NEXT(1)/BB/N(1)/CC/M(1)/DD/KTERM(9,1)
      COMMON/GH/NUTIL(1)/HH/MSTART(1)/HI/MAX(1)/TT/NTOTAL
      COMMON/II/KFACT(1)/JJ/MORDER/KK/LZERO/LL/INDEX/NT/NOTRUN
      COMMON/NN/NPRINT/NUMBER/NUMBER/IJ/KCHOP(1)/NC/NOCHOP
      LOGICAL NPRINT,PNPRNT,NOTRUN,NOCHOP
      DIMENSION IA(24),IB(24),IC(24)
      DATA NPY/6/,NTG/2/,NVAR/8/,NTIP/9/,NTGP/7/
      IF (.NOT.NPRINT) PRINT 350, ISA,ISB,ISC,MORDER
      XNB=TMLEFT(MS)
      NPY4=4*NPY
      CALL LINK(ISA)
      CALL LINK(ISB)
      MSA=MSTART(ISA)
      MSB=MSTART(ISB)
      CALL ZERO (ISC)
      IF (MSA.LE.0.OR.MSB.LE.0) RETURN
      PNPRNT=NPRINT
      NPRINT=.TRUE.
      K=MSA
      J=MSB
      CALL SETUP(0,0)

   20 IF (NOTRUN) GO TO 50
      DO 21 LJ=1,NPY
   21 CALL UNPACK(KTERM(LJ,K),IA(4*LJ-3))
      KH=0
      DO 22 LJ=1,NPY4
   22 KH=KH+IA(LJ)*KFACT(LJ)
   30 IF (NOTRUN) GO TO 50
      DO 31 LJ=1,NPY
   31 CALL UNPACK(KTERM(LJ,J),IB(4*LJ-3))
      JH=KH
      DO 32 LJ=1,NPY4
   32 JH=JH+IB(LJ)*KFACT(LJ)
      IF (JH.GT.MORDER) GO TO 320
      DO 40 KV=1,NPY4
      IC(KV)=IA(KV)+IB(KV)-KCHOP(KV)
      IF (IC(KV)) 40,40,320
   40 CONTINUE

   50 IF (LZERO.GT.0) GO TO 60
      CALL COLECT(ISC,1)
      CALL SETUP(NTOTAL,1)
      GO TO 50
   60 KTA=LZERO
      LZERO=NEXT(KTA)
      NEXT(KTA)=0
   70 IF (LZERO.GT.0) GO TO 80
```

UNCLASSIFIED

```
      CALL COLECT(ISC,1)
      CALL SETUP(NTOTAL,1)
      GO TO 70
   80 KTB=LZERO
      LZERO=NEXT(KTB)
      NEXT(KTB)=0
      DO 84 LJ=1,NPY
      IVA=KTERM(LJ,K)-INDEX
      KTERM(LJ,KTA)=KTERM(LJ,J)+IVA
   84 KTERM(LJ,KTB)=KTERM(LJ,KTA)
      DO 86 LJ=NTGP,NVAR
      IVB=KTERM(LJ,K)-INDEX
      KTERM(LJ,KTA)=KTERM(LJ,J)+IVB
      IVC=KTERM(LJ,K)-KTERM(LJ,J)
   86 KTERM(LJ,KTB)=IVC+INDEX

      IF (KTERM(NTIP,K)) 90,100,100
   90 IF (KTERM(NTIP,J)) 110,150,150
  100 IF (KTERM(NTIP,J)) 210,270,270

C     SIN*SIN TO -COS+COS
  110 CALL REFACT (N(J),M(J),N(K),M(K),-1,+2,N(KTA),M(KTA),1)
      NUTIL(8)=NUTIL(8)+1
      N(KTB)=-N(KTA)
      M(KTB)=+M(KTA)
      KTERM(NTIP,KTB)=+1
      KTERM(NTIP,KTA)=+1
      GO TO 310

C     SIN*COS TO SIN +SIN
  150 CALL REFACT (N(J),M(J),N(K),M(K),+1,+2,N(KTA),M(KTA),1)
      NUTIL(9)=NUTIL(9)+1
      N(KTB)=+N(KTA)
      M(KTB)=+M(KTA)
      KTERM(NTIP,KTB)=-1
      KTERM(NTIP,KTA)=-1
      GO TO 310

C     COS*SIN TO -SIN+SIN
  210 CALL REFACT (N(J),M(J),N(K),M(K),+1,+2,N(KTA),M(KTA),1)
      NUTIL(10)=NUTIL(10)+1
      N(KTB)=-N(KTA)
      M(KTB)=+M(KTA)
      KTERM(NTIP,KTB)=-1
      KTERM(NTIP,KTA)=-1
      GO TO 310

C     COS*COS TO COS+COS
  270 CALL REFACT (N(J),M(J),N(K),M(K),+1,+2,N(KTA),M(KTA),1)
      NUTIL(11)=NUTIL(11)+1
      N(KTB)=+N(KTA)
```

```
      M(KTB)=+M(KTA)
      KTERM(NTIP,KTA)=+1
      KTERM(NTIP,KTB)=+1
      DO 272 LJ=NTGP,NVAR
      IF (KTERM(LJ,KTA).NE.INDEX) GO TO 310
      IF (KTERM(LJ,KTB).NE.INDEX) GO TO 310
  272 CONTINUE
      CALL REFACT(N(KTA),M(KTA),1,1,2,1,N(KTA),M(KTA),1)
      N(KTB)=0
      GO TO 310

  310 CALL INSERT (ISC,KTA)
      CALL INSERT (ISC,KTB)
  320 NJ=NEXT(J)
      NUTIL(16)=NUTIL(16)+1
      IF (NJ.EQ.0) GO TO 330
      J=NJ
      GO TO 30
  330 NK=NEXT(K)
      IF (NK.EQ.0) GO TO 340
      K=NK
      J=MSB
      GO TO 20
  340 CALL LINK (ISC)
      NPRINT=PNPRNT
      IF (NPRINT) RETURN
      XNA=TMLEFT(MS)
      XNT=XNB-XNA
      PRINT 370,NTOTAL,XNT
      CONTINUE
      RETURN
      END


      SUBROUTINE MULVAR (ISA,IV,NUM)
   20 FORMAT (9H MULTIPLY,I3,11H BY VARIBLE,I3,9H TO POWER,I3)
      COMMON/AA/NEXT(1)/BB/N(1)/CC/M(1)/DD/KTERM(9,1)
      COMMON/HH/MSTART(1)/HI/MAX(1)/NN/NPRINT/IJ/KCHOP(1)
      LOGICAL NPRINT
      DIMENSION IAA(4)
      IF (.NOT.NPRINT) PRINT 20, ISA,IV,NUM
      K=MSTART(ISA)
      IF (K.EQ.0) RETURN
      JW=(IV+3)/4
      JV=MOD(IABS(IV),4)
      IF (JV.EQ.0) JV=4
   10 CALL UNPACK(KTERM(JW,K),IAA)
      IAA(JV)=IAA(JV)+NUM
      CALL REPACK (IAA,KTERM(JW,K))
      K=NEXT(K)
      IF (K.GT.0) GO TO 10
```

UNCLASSIFIED

```
      CALL MULONE(ISA)
      RETURN
      END


      SUBROUTINE NDERIV (ISA,IV,NUM,ISB)
   20 FORMAT (10H PDERIV OF,I3,4H WRT,I3,I8,6H TIMES)
   30 FORMAT (18H  RESULT STORED IN,I3)
      COMMON/NN/NPRINT/NM/LPRINT
      LOGICAL NPRINT,LPRINT,PNPRNT
      PNPRNT=NPRINT
      IF (LPRINT) NPRINT=.TRUE.
      PRINT 20, ISA,IV,NUM
      PRINT 30, ISB
      CALL TRANSF (ISA,ISB)
      DO 10 K=1,NUM,1
   10 CALL DERIV (ISB,IV,ISB)
      IF (LPRINT) NPRINT=PNPRNT
      RETURN
      END


      SUBROUTINE OTERM(K)
      IMPLICIT REAL*8 (A-H,O-Z)
  310 FORMAT (200A1,185A1)
  320 FORMAT (5X,130A1)
  321 FORMAT (5X,130A1/(15X,120A1))
  350 FORMAT (1X,I11,A1,I11,24(A1,A3,A2,A2,I4),2A4,8(I4,A4),A1)
  360 FORMAT (        F24.7,24(A1,A3,A2,A2,I4),2A4,8(I4,A4),A1)
      COMMON/AA/NEXT(1)/BB/N(1)/CC/M(1)/DD/KTERM(9,1)
      COMMON/GH/NUTIL(1)/HH/MSTART(1)/HI/MAX(1)
      COMMON/MM/NOFACT/KK/LZERO/LL/INDEX/TT/NTOTAL
      COMMON/LPOLY/LPOLY(1)/LARG/LARG(1)
      COMMON/OTERM/LSTAR,LBLANK,LARROW,LTRIGA,LTRIGB,LPAREN,SOEFF
      DIMENSION KEDIT(97),LEDIT(385),IA(24),IB(24)
      LOGICAL NOFACT,FLOAT,REORDR
      REAL*4 SOEFF
      DATA NPY/6/,NTG/2/,NVAR/8/,NTIP/9/,NTGP/7/
      DATA LBLANK/4H    /
      DATA MPAREN/1H)/
      DATA LCOSA/4H*COS/
      DATA LSINA/4H*SIN/
      DATA LCOSB/4H(   /
      DATA LSINB/4H(   /
      DATA LSLASH/1H//
      DATA KBLANK/1H /
      DATA KPLUSN/1H+/
      DATA MINUSN/1H-/
      DATA LSTAR/1H*/
      DATA LARROW/2H**/
      DATA KZERO/1H0/
```

```
C     SWITCH SIGN OF DEMOMINATOR
      IF (M(K).LT.0) N(K)=-N(K)
      IF (M(K).LT.0) M(K)=-M(K)
      DO 11 LJ=1,NPY
   11 CALL UNPACK(KTERM(LJ,K),IA(4*LJ-3))
      DO 12 LJ=NTGP,NVAR
   12 CALL UNPACK(KTERM(LJ,K),IB(4*(LJ-NPY)-3))

C     DETERMINE IF RATIONAL FRACTION OF NOT
      FLOAT=.FALSE.
      LPAREN=MPAREN
      IF (KTERM(NTIP,K).EQ.-1)  LTRIGA=LSINA
      IF (KTERM(NTIP,K).EQ.-1)  LTRIGB=LSINB
      IF (KTERM(NTIP,K).GE.+0)  LTRIGA=LCOSA
      IF (KTERM(NTIP,K).GE.+0)  LTRIGB=LCOSB
      IF (IRGSGN(K).NE.0) GO TO 14
      LPAREN=LBLANK
      LTRIGA=LBLANK
      LTRIGB=LBLANK

   14 IF (NOFACT) GO TO 50
      SM=M(K)
      SM=DABS(SM)
      IF (SM.LT.1.0D4) GO TO 40
      SM=DLOG10(SM)+0.0001D0
      SM=DMOD(SM,1.0D0)
      IF (SM.LT.0.001D0) FLOAT=.TRUE.
      IF (FLOAT) GO TO 50

C     DETERMINE IF SIGN OR COSINE FOR RATIONAL FRACTION
   40 ENCODE (385,350,KEDIT) N(K),LSLASH,M(K),
     *   (LSTAR,LPOLY(I),LBLANK,LARROW,IA(I),I=1,24),
     *    LTRIGA,LTRIGB,(IB(I),LARG(I),I=1,8),LPAREN
      GO TO 60

C     DETERMINE IF SIGN OR COSINE FOR FLOATING POINT
   50 COEFF=N(K)
      COEFF=COEFF/M(K)
      SOEFF=COEFF
      ENCODE (385,360,KEDIT) SOEFF,
     *   (LSTAR,LPOLY(I),LBLANK,LARROW,IA(I),I=1,24),
     *    LTRIGA,LTRIGB,(IB(I),LARG(I),I=1,8),LPAREN

C     DELETE UNNECESSARY POLYNOMIAL OR TRIG VARIABLES
   60 DO 65 I=1,24
      IF (IA(I).EQ.0) KEDIT(3*I+4) =LBLANK
      IF (IA(I).EQ.0) KEDIT(3*I+5) =LBLANK
   65 IF (IA(I).EQ.0) KEDIT(3*I+6) =LBLANK
      DO 70 I=1,8
      IF (IB(I).EQ.0) KEDIT(2*I+79)=LBLANK
   70 IF (IB(I).EQ.0) KEDIT(2*I+80)=LBLANK
```

UNCLASSIFIED

```
C     DECODE INTO SINGLE CHARACTERS
      DECODE (385,310,KEDIT) (LEDIT(I),I=1,385)

C     DELETE UNITY EXPONENTS AND MULTIPLIERS
      DO 80 I=1,24
      IF (IA(I).NE.+1) GO TO 80
      LEDIT(19+12*I)=KBLANK
      LEDIT(20+12*I)=KBLANK
      LEDIT(24+12*I)=KBLANK
   80 CONTINUE
      DO 90 I=1,8
      IF (IB(I).EQ.+1) LEDIT(316+8*I)=KBLANK
   90 IF (IB(I).EQ.-1) LEDIT(316+8*I)=KBLANK

C     INSERT + SIGNS IN TRIG ARGUMENT LIST
      DO 100 I=1,8
      ISTART=I+1
      IF (IB(I).NE.+0) GO TO 110
  100 CONTINUE
      GO TO 130
  110 IF (ISTART.GT.8) GO TO 130
      DO 120 I=ISTART,8
  120 IF (IB(I).GT.0) LEDIT(313+8*I)=KPLUSN

C     DELETE COEFFICIENTS FOR UNITY VALUES
  130 IF (N(K).GT.0) LEDIT(2)=KPLUSN
      DO 132 LJ=1,NVAR
      IF (KTERM(LJ,K).NE.INDEX) GO TO 134
  132 CONTINUE
      GO TO 160
  134 CONTINUE
      IF (N(K).EQ.-1.AND.M(K).EQ.+1) GO TO 138
      IF (N(K).EQ.+1.AND.M(K).EQ.+1) GO TO 140
      GO TO 160
  138 LEDIT(2)=MINUSN
  140 DO 142 I=3,24
  142 LEDIT(I)=KBLANK
      DO 145 I=1,24
      IF (IA(I).EQ.0) GO TO 145
      LEDIT(13+12*I)=KBLANK
      GO TO 160
  145 CONTINUE
      LEDIT(313)=KBLANK
  160 IF (M(K).EQ.1) LEDIT(13)=KBLANK
      IF (M(K).EQ.1) LEDIT(24)=KBLANK

C     DELETE TRAILING ZEROS IN FLOATING POINT COEFF
      IF (.NOT.NOFACT.AND..NOT.FLOAT) GO TO 190
      MKOUNT=24
      DO 180 I=1,10
      IF (LEDIT(MKOUNT).NE.KZERO) GO TO 190
```

```
      LEDIT(MKOUNT)=KBLANK
  180 MKOUNT=MKOUNT-1

C     COMPRESS INBETWEEN BLANKS
  190 MKOUNT=1
      DO 210 I=1,385
      IF (LEDIT(I).EQ.KBLANK) GO TO 210
      IF (MKOUNT.EQ.I) GO TO 200
      LEDIT(MKOUNT)=LEDIT(I)
      LEDIT(I)=KBLANK
  200 MKOUNT=MKOUNT+1
  210 CONTINUE
      IF (MKOUNT.LE.130) PRINT 320, (LEDIT(I),I=1,MKOUNT)
      IF (MKOUNT.GT.130) PRINT 321, (LEDIT(I),I=1,MKOUNT)
      NUTIL(14)=NUTIL(14)+1
      RETURN
      END


      SUBROUTINE OUTLP (ISA)
      IMPLICIT REAL*8 (A-H,O-Z)
  270 FORMAT (//)
  330 FORMAT (//,13H WRITE SERIES,I4,/,9H LENGTH =,I5,/)
  340 FORMAT (/,2X,17H ** FILE EMPTY **,/)
      COMMON/AA/NEXT(1)/BB/N(1)/CC/M(1)/DD/KTERM(9,1)
      COMMON/TT/NTOTAL/HH/MSTART(1)/HI/MAX(1)
      COMMON/LPOLY/LPOLY(1)/LARG/LARG(1)
      CALL LINK (ISA)
      PRINT 330,ISA,NTOTAL
      K=MSTART(ISA)
      IF (K.GT.0) GO TO 10
      PRINT 340
      RETURN
   10 CALL OTERM(K)
      K=NEXT(K)
      IF (K.GT.0) GO TO 10
      PRINT 270
      RETURN
      END
```

UNCLASSIFIED

```
      SUBROUTINE POWER (ISA,IJKL,ISB)
   70 FORMAT (1X,I3,16H RAISED TO POWER,I3,17H RESULT STORED IN,I3)
   80 FORMAT (44H **** SERIES RAISED TO A NEGATIVE POWER ****)
   90 FORMAT (14H  TOTAL TIME =,F7.2,4H SEC)
      COMMON/NN/NPRINT/NM/LPRINT
      LOGICAL NPRINT,LPRINT,PNPRNT
      XNB=TMLEFT(MS)
      PNPRNT=NPRINT
      IF (LPRINT) NPRINT=.TRUE.
      PRINT 70, ISA,IJKL,ISB
      IF (IJKL.GE.0) GO TO 10
      PRINT 80
      CALL QUIT
      STOP
   10 ISC=1002
      CALL ZERO (ISC)
      IJ=IJKL-1
      IF (IJ) 20,30,40
   20 CALL ZERO (ISB)
      CALL DEFONE(ISB)
      GO TO 60
   30 CALL TRANSF (ISA,ISB)
      GO TO 60
   40 CALL TRANSF (ISA,ISB)
      DO 50 K=1,IJ,1
      CALL MULT (ISA,ISB,ISC)
   50 CALL SWITCH (ISC,ISB)
   60 CALL ZERO (ISC)
      IF (LPRINT) NPRINT=PNPRNT
      XNA=TMLEFT(MS)
      XNT=XNB-XNA
      PRINT 90, XNT
      RETURN
      END


      SUBROUTINE REFACT (NAA,MAA,NBB,MBB,KN,KM,NCC,MCC,KETCH)
      IMPLICIT REAL*8 (A-H,O-Z)
  180 FORMAT (50X,27H LOOSING ACCURACY IN REFACT)
  190 FORMAT (50X,27H DEMONINATOR ZERO IN REFACT)
      COMMON/MM/NOFACT/OO/NEXACT/RR/EPSLON/SS/NOFIX
      DATA EXFACT/1.0D+9/,MLIM/1000/
      DATA ATOL/1.0D-2/,BTOL/1.0D-4/,CTOL/1.0D-6/
      LOGICAL NOFACT,NOFIX,NEXACT
      IF (MAA.NE.0.AND.MBB.NE.0) GO TO 10
      PRINT 190
      STOP
   10 NA=NAA
      MA=MAA
      NB=NBB
      MB=MBB
```

```
      IF (NOFIX) GO TO 50
      IF (KETCH) 20,20,30
   20 CALL FACTOR (MA,MB)
      NCC=NA*MB+MA*NB
      MCC=MA*MBB
      GO TO 40
   30 CALL FACTOR (NA,MB)
      CALL FACTOR (NB,MA)
      NCC=NA*NB
      MCC=MA*MB
   40 NCC=KN*NCC
      MCC=KM*MCC
      GO TO 170
   50 DMAX=2147483647.0D0
      AN=NA
      AM=MA
      BN=NB
      BM=MB
      IF (KETCH) 60,60,70
   60 CN=AN*BM+AM*BN
      CM=AM*BM
      GO TO 80
   70 CN=AN*BN
      CM=AM*BM
   80 CN=KN*CN
      CM=KM*CM
      COEFF=CN/CM
      ABSAB=DABS(COEFF)
      IF (ABSAB.GT.EPSLON) GO TO 90
      NCC=0
      MCC=1
      RETURN
   90 IF (NOFACT) GO TO 130
      IF (IABS(MA).LT.MLIM) GO TO 100
      SA=DABS(AM)
      SA=DLOG10(SA)
      SA=SA+CTOL
      SA=DMOD(SA,1.0D0)
      IF (SA.LT.BTOL) GO TO 130
  100 IF (IABS(MB).LT.MLIM) GO TO 110
      SB=DABS(BM)
      SB=DLOG10(SB)
      SB=SB+CTOL
      SB=DMOD(SB,1.0D0)
      IF (SB.LT.BTOL) GO TO 130
  110 IF (DABS(CN).GT.DMAX) GO TO 120
      IF (DABS(CM).GT.DMAX) GO TO 120
      NCC=IDINT(CN+DSIGN(ATOL,CN))
      MCC=IDINT(CM+DSIGN(ATOL,CM))
      GO TO 170
  120 CALL XACTOR (CN,CM)
```

UNCLASSIFIED

```
      IF (DABS(CN).GT.DMAX) GO TO 130
      IF (DABS(CM).GT.DMAX) GO TO 130
      GO TO 160
130   CN=COEFF*EXFACT
      CM=EXFACT
      NEXACT=.TRUE.
140   IF (DABS(CN).GT.DMAX) GO TO 150
      IF (DABS(CM).GT.DMAX) GO TO 150
      GO TO 160
150   CN=CN/10
      CM=CM/10
      IF (DABS(CN).GE.1.0D3.AND.DABS(CM).GE.1.0D3) GO TO 140
      PRINT 180
      STOP
160   NCC=IDINT(CN+DSIGN(0.5D0,CN))
      MCC=IDINT(CM+DSIGN(0.5D0,CM))
      RETURN
170   CALL FACTOR (NCC,MCC)
      RETURN
      END


      SUBROUTINE REPACK (III,IA)
      DIMENSION III(4)
      DATA KN1/Z00000100/
      DATA KN2/Z00010000/
      DATA KN3/Z01000000/
      IN=(III(4)+128)
      IM=(III(3)+128)*KN1
      IL=(III(2)+128)*KN2
      IK=(III(1)+ 64)*KN3
      IA=IK+IL+IM+IN
      RETURN
      END
```

```
      SUBROUTINE REVERS(ISA)
      COMMON/AA/NEXT(1)/HH/MSTART(1)/HI/MAX(1)
      MSA=MSTART(ISA)
      IF (MSA.EQ.0) RETURN
      MSTART(ISA)=MAX(ISA)
      MAX(ISA)=MSA
      LAST=MSA
      KX=NEXT(MSA)
      NEXT(LAST)=0
      IF (KX.EQ.0) RETURN
      NKX=NEXT(KX)
   10 NEXT(KX)=LAST
      IF (NKX.LE.0) GO TO 20
      LAST=KX
      KX=NKX
      NKX=NEXT(NKX)
      GO TO 10
   20 RETURN
      END


      SUBROUTINE SECULA(ISA,ISB)
      IVARIB=0
      NUM=0
      CALL COMBIN(ISA,ISB,IVARIB,NUM,2)
      RETURN
      END


      SUBROUTINE SELECT(ISA,ISB,IVARIB,NUM)
      CALL COMBIN(ISA,ISB,IVARIB,NUM,4)
      RETURN
      END
```

UNCLASSIFIED

```
      SUBROUTINE SETUP(KA,KEYS)
      IMPLICIT REAL*8 (A-H,O-Z)
      COMMON/VV/NUMPAR,KOUNT,KOLECT,KSCAN
      COMMON/NUMBER/NUMBER/UU/LMAP(1)
      LOGICAL KSCAN
      DO 10 JX=1,1024
   10 LMAP(JX)=0
      IF (KEYS.EQ.0) NUMBER=0
      IF (KEYS.EQ.0) KOLECT=300
      KOUNT=KA
      KSCAN=.TRUE.
      NTEMP=MAX0(KA,1)
      AB=DLOG(DFLOAT(NTEMP))
      AB=AB/DLOG(2.0D0)
      NUMPAR=AB+0.00001D0
      NUMPAR=(2**NUMPAR)/16
      IF (NUMPAR.GT.512) NUMPAR=512
      IF (NUMPAR.LT.2)   NUMPAR=2
      RETURN
      END


      SUBROUTINE SIMP(ISA)
   40 FORMAT (9H SIMPLIFY,I4,10H  LENGTH =,I5,8H  TIME =,F7.2)
      COMMON/AA/NEXT(1)/BB/N(1)/CC/M(1)/DD/KTERM(9,1)
      COMMON/HH/MSTART(1)/HI/MAX(1)/NN/NPRINT/TT/NTOTAL
      LOGICAL NPRINT,KSCAN
      XNB=TMLEFT(MS)
      CALL LINK(ISA)
      CALL REVERS(ISA)
      MSA=MSTART(ISA)
      IF (MSA.EQ.0) RETURN
      ISB=1001
      CALL ZERO(ISB)
      CALL SETUP(0,0)
   20 K=MSTART(ISA)
      IF (K.EQ.0) GO TO 30
      MSTART(ISA)=NEXT(K)
      NEXT(K)=0
      CALL INSERT(ISB,K)
      GO TO 20
   30 CALL SWITCH(ISA,ISB)
      MSTART(ISB)=0
      MAX(ISB)=0
      CALL LINK(ISA)
      IF (NPRINT) RETURN
      XNA=TMLEFT(MS)
      XNT=XNB-XNA
      PRINT 40,ISA,NTOTAL,XNT
      RETURN
      END
```

```
      SUBROUTINE SUBX(ISA,ISB,ISC)
   10 FORMAT (9H SUBTRACT,I3,5H FROM,I3,13H AND STORE IN,I3)
      COMMON/GH/NUTIL(1)/NN/NPRINT/NM/LPRINT
      LOGICAL NPRINT,LPRINT,PNPRNT
      IF (.NOT.NPRINT) PRINT 10, ISB,ISA,ISC
      PNPRNT=NPRINT
      NPRINT=.TRUE.
      N6=NUTIL(6)
      CALL MULCON (ISB,-1,1)
      CALL ADD (ISA,ISB,ISC)
      CALL MULCON (ISB,-1,1)
      ID6=NUTIL(6)-N6
      NUTIL(7)=NUTIL(7)+ID6
      NUTIL(6)=N6
      NPRINT=PNPRNT
      RETURN
      END


      SUBROUTINE SUBAB (ISA,ISB,IVA,IVB,NUM)
   60 FORMAT (11H SUBSTITUTE,5I4)
   70 FORMAT (36H **** STORAGE OVERFLOW IN SUBAB ****)
   80 FORMAT (14H  TOTAL TIME =,F7.2,4H SEC)
      COMMON/AA/NEXT(1)/BB/N(1)/CC/M(1)/DD/KTERM(9,1)
      COMMON/HH/MSTART(1)/HI/MAX(1)
      COMMON/KK/LZERO/NN/NPRINT/NM/LPRINT
      LOGICAL NPRINT,LPRINT,PNPRNT
      DIMENSION IA(4),IB(4)
      DATA NPY/6/,NTG/2/,NVAR/8/,NTGP/7/,NTIP/9/
      PNPRNT=NPRINT
      XNB=TMLEFT(MS)
      IF (LPRINT) NPRINT=.TRUE.
      PRINT 60, ISA,ISB,IVA,IVB,NUM
      CALL TRANSF (ISA,ISB)
      DO 50 J=1,NUM,1
      ISC=1005
      CALL ZERO (ISC)
      K=MSTART(ISB)
      IF (K.LE.0) RETURN
      L=LZERO
      MSTART(ISC)=L
      IF (K.GT.0) GO TO 5
      IF (LPRINT) NPRINT=PNPRNT
      RETURN
    5 JWA=(IVA+3)/4
      JVA=MOD(IVA,4)
      IF (JVA.EQ.0) JVA=4
      JWB=(IVB+3)/4
      JVB=MOD(IVB,4)
      IF (JVB.EQ.0) JVB=4
   10 CALL UNPACK(KTERM(JWA,K),IA)
```

UNCLASSIFIED

```
      IF (IA(JVA).LT.2) GO TO 30
      IA(JVA)=IA(JVA)-2
      CALL REPACK (IA,KTERM(JWA,K))
      N(L)=-N(K)
      M(L)=+M(K)
      DO 15 LJ=1,NTIP
   15 KTERM(LJ,L)=KTERM(LJ,K)
      CALL UNPACK(KTERM(JWB,L),IB)
      IB(JVB)=IB(JVB)+2
      CALL REPACK (IB,KTERM(JWB,L))
      LN=NEXT(L)
      IF (LN.GT.0) GO TO 20
      PRINT 70
      CALL QUIT
      STOP
   20 L=LN
   30 KN=NEXT(K)
      IF (KN.EQ.0) GO TO 40
      K=KN
      GO TO 10
   40 MAX(ISC)=L
      LZERO=NEXT(L)
      NEXT(L)=0
      CALL SIMP(ISB)
   50 CALL ACCUM (ISB,ISC)
      IF (LPRINT) NPRINT=PNPRNT
      XNA=TMLEFT(MS)
      XNT=XNB-XNA
      PRINT 80, XNT
      RETURN
      END
```

```
      SUBROUTINE SUBST (ISA,ISB,IV,ISC)
   30 FORMAT (10X,32H **** NEG EXPONENT IN SUBST ****)
   40 FORMAT (1X,11H SUBSTITUTE,4I4)
      COMMON/AA/NEXT(1)/BB/N(1)/CC/M(1)/DD/KTERM(9,1)
      COMMON/HH/MSTART(1)/HI/MAX(1)
      DIMENSION IA(4)
      PRINT 40, ISA,ISB,IV,ISC
      ISD=1009
      ISE=1010
      ISF=1011
      CALL ZERO (ISC)
      CALL ZERO (ISD)
      CALL ZERO (ISE)
      CALL ZERO (ISF)
      LMIN=+100
      LMAX=-100
      K=MSTART(ISA)
      IF (K.EQ.0) RETURN
      JW=(IABS(IV)+3)/4
      JV=MOD(IABS(IV),4)
      IF (JV.EQ.0) JV=4
   10 CALL UNPACK(KTERM(JW,K),IA)
      NUM=IA(JV)
      IF (NUM.GT.LMAX) LMAX=NUM
      IF (NUM.LT.LMIN) LMIN=NUM
      K=NEXT(K)
      IF (K.GT.0) GO TO 10
      IF (LMIN.LT.0) PRINT 30
      IF (LMIN.LT.0) STOP
      CALL DEFONE(ISD)
      CALL SELECT (ISA,ISC,IV,0)
      DO 20 L=1,LMAX
      CALL MULT (ISD,ISB,ISE)
      CALL SWITCH (ISD,ISE)
      CALL ZERO (ISE)
      IF (L.LT.LMIN) GO TO 20
      CALL SELECT (ISA,ISE,IV,L)
      CALL ERASE (ISE,IV)
      CALL MULT (ISE,ISD,ISF)
      CALL ACCUM (ISC,ISF)
   20 CONTINUE
      CALL ZERO (ISD)
      CALL ZERO (ISE)
      CALL ZERO (ISF)
      RETURN
      END
```

UNCLASSIFIED

```
      SUBROUTINE SWIFNC(K)
      COMMON/BB/N(1)/DD/KTERM(9,1)/LL/INDEX
      DATA NPY/6/,NTG/2/,NVAR/8/,NTIP/9/,NTGP/7/
      IF (N(K).EQ.0) RETURN
      IF (IRGSGN(K)) 10,30,40
   10 DO 20 LJ=NTGP,NVAR
      KTERM(LJ,K)=KTERM(LJ,K)-INDEX
   20 KTERM(LJ,K)=INDEX-KTERM(LJ,K)
      IF (KTERM(NTIP,K).LT.0) N(K)=-N(K)
      RETURN
   30 IF (KTERM(NTIP,K).LT.0) N(K)=0
   40 RETURN
      END


      SUBROUTINE SWITCH(ISA,ISB)
      COMMON/HH/MSTART(1)/HI/MAX(1)
      MSISA=MSTART(ISA)
      MXISA=MAX(ISA)
      MSISB=MSTART(ISB)
      MXISB=MAX(ISB)
      MSTART(ISA)=MSISB
      MAX(ISA)=MXISB
      MSTART(ISB)=MSISA
      MAX(ISB)=MXISA
      RETURN
      END
```

```
      SUBROUTINE TAYLOR (ISA,ISB,IV,NUM,ISC)
   20 FORMAT (24H TAYLOR SERIES EXPANSION,5I5)
   30 FORMAT (36H  TIME FOR TAYLOR SERIES EXPANSION =,F7.2,4H SEC)
      COMMON/NN/NPRINT/NM/LPRINT
      LOGICAL NPRINT,LPRINT,PNPRNT
      XNB=TMLEFT(MS)
      PNPRNT=NPRINT
      PRINT 20, ISA,ISB,IV,NUM,ISC
      ISD=1012
      ISE=1013
      ISF=1014
      CALL ZERO (ISF)
      IF (LPRINT) NPRINT=.TRUE.
      CALL DEFONE(ISD)
      CALL TRANSF (ISA,ISE)
      CALL TRANSF (ISA,ISC)
      DO 10 K=1,NUM,1
      CALL DERIV (ISE,IV,ISE)
      CALL MULCON (ISE,1,K)
      CALL MULT (ISB,ISD,ISF)
      CALL SWITCH (ISF,ISD)
      CALL MULT (ISD,ISE,ISF)
   10 CALL ACCUM (ISC,ISF)
      CALL ZERO (ISD)
      CALL ZERO (ISE)
      CALL ZERO (ISF)
      IF (LPRINT) NPRINT=PNPRNT
      XNA=TMLEFT(MS)
      XNT=XNB-XNA
      PRINT 30, XNT
      RETURN
      END


      FUNCTION TMLEFT(IT)
      TMLEFT=0.0
      RETURN
      END
```

UNCLASSIFIED

```
      SUBROUTINE TRANSF (ISA,ISB)
   50 FORMAT (37H **** STORAGE OVERFLOW IN TRANSF ****)
      COMMON/AA/NEXT(1)/BB/N(1)/CC/M(1)/DD/KTERM(9,1)
      COMMON/GH/NUTIL(1)/HH/MSTART(1)/HI/MAX(1)
      COMMON/KK/LZERO/LL/INDEX
      DATA NPY/6/,NTG/2/,NVAR/8/,NTIP/9/,NTGP/7/
      IF (ISA.EQ.ISB) RETURN
      IF (LZERO.GT.0) GO TO 10
      PRINT 50
      CALL QUIT
      STOP
   10 CALL ZERO (ISB)
      MSTART(ISB)=LZERO
      KF=LZERO
      KI=MSTART(ISA)
      IF (KI.GT.0) GO TO 20
      MSTART(ISB)=0
      MAX(ISA)=0
      MAX(ISB)=0
      RETURN
   20 N(KF)=+N(KI)
      M(KF)=+M(KI)
      DO 24 LJ=1,NTIP
   24 KTERM(LJ,KF)=KTERM(LJ,KI)
      NUTIL(2)=NUTIL(2)+1
      KNI=NEXT(KI)
      IF (KNI.EQ.0) GO TO 40
      KI=KNI
      KNF=NEXT(KF)
      IF (KNF.GT.0) GO TO 30
      PRINT 50
      CALL QUIT
      STOP
   30 KF=KNF
      GO TO 20
   40 LZERO=NEXT(KF)
      NEXT(KF)=0
      MAX(ISA)=KI
      MAX(ISB)=KF
      RETURN
      END


      SUBROUTINE TRUN(ISA,ISB,IVARIB,NUM)
      CALL COMBIN(ISA,ISB,IVARIB,NUM,3)
      RETURN
      END
```

```
      SUBROUTINE UNPACK (IZ,II)
      COMMON/LL/INDEX
      DIMENSION II(4)
      IA=IZ
      IF (IA.EQ.INDEX) GO TO 10
      II(4)=MOD(IA,256)-128
      IA=IA/256
      II(3)=MOD(IA,256)-128
      IA=IA/256
      II(2)=MOD(IA,256)-128
      IA=IA/256
      II(1)=MOD(IA,256)-64
      RETURN
   10 II(1)=0
      II(2)=0
      II(3)=0
      II(4)=0
      RETURN
      END


      SUBROUTINE XACTOR (XN,XM)
      IMPLICIT REAL*8(A-H,O-Z)
      DATA CTOL/1.0D-2/,DTOL/1.0D-5/
      IF (XN.EQ.0.0D0) XM=1.0D0
      XN=XN+DSIGN(CTOL,XN)
      XM=XM+DSIGN(CTOL,XM)
      XN=XN-DMOD(XN,1.0D0)
      XM=XM-DMOD(XM,1.0D0)
      XA=XN+DSIGN(DTOL,XN)
      XB=XM
   10 XC=DMOD(XA,XB)
      IF (DABS(XC).LT.CTOL) GO TO 20
      XA=XB+DSIGN(CTOL,XB)
      XB=XC+DSIGN(CTOL,XC)
      XA=XA-DMOD(XA,1.0D0)
      XB=XB-DMOD(XB,1.0D0)
      XA=XA+DSIGN(DTOL,XA)
      GO TO 10
   20 XN=XN/DABS(XB)
      XM=XM/DABS(XB)
      RETURN
      END
```

UNCLASSIFIED

```
      SUBROUTINE ZERO (ISA)
      COMMON/AA/NEXT(1)/BB/N(1)/CC/M(1)/DD/KTERM(9,1)
      COMMON/GH/NUTIL(1)/HH/MSTART(1)/HI/MAX(1)
      COMMON/KK/LZERO/LL/INDEX
      DATA NPY/6/,NTG/2/,NVAR/8/,NTIP/9/,NTGP/7/
      MSA=MSTART(ISA)
      IF (MSA.EQ.0) GO TO 30
      K=MSA
   10 N(K)=0
      M(K)=+1
      KTERM(NTIP,K)=+1
      DO 15 LJ=1,NVAR
   15 KTERM(LJ,K)=INDEX
      NUTIL(1)=NUTIL(1)+1
      KN=NEXT(K)
      IF (KN.EQ.0) GO TO 20
      K=KN
      GO TO 10
   20 NEXT(K)=LZERO
      LZERO=MSA
   30 MSTART(ISA)=0
      MAX(ISA)=0
      RETURN
      END
```