2627

NRL Report 7564

Algebraic Manipulation by Computer

R. R. DASENBROCK

Operations Research Group
Report 73-1

Mathematics and Information Sciences Division

June 29, 1973

NAVAL RESEARCH LABORATORY
Washington, D.C.

Approved for public release; distribution unlimited. DDC -A

CONTENTS

Abstract 1

INTRODUCTION 1

PROGRAM OBJECTIVES 2

INTERNAL REPRESENTATION 3

STORAGE MANAGEMENT 4

DATA CHAIN MANIPULATIONS 5

ALGEBRAIC MANIPULATIONS 9

PROBLEM EXAMPLES 13

- Exponentiation 13
- Multiplication 14
- A Binomial Expansion 14
- A Polynomial Raised to a Power 15
- The Determination of the Legendre Polynomials 19
- A Series Solution to a Differential Equation 21

CONCLUDING REMARKS 30

REFERENCES 30

APPENDIX A – Program Listing 32

Algebraic Manipulation by Computer

R. R. DASENBROCK

Operations Research Group
Mathematics and Information Sciences Division

Abstract: This report describes a program which has the capability of performing literal algebraic manipulations on a high speed electronic computer. Emphasis is concentrated on the manipulation of the Poisson series occurring in the classical theory of lightly perturbed dynamical systems. The program is written in Fortran and is now operational on a CDC-3800. With little or no modification it could be implemented on any computer possessing a Fortran compiler.

INTRODUCTION

Recent advances in laser satellite tracking techniques provide an order of magnitude improvement in measurement accuracy over earlier techniques. To utilize this new capability, it is apparent that parallel advances be made in the analytical theories describing the orbits of artificial satellites under the influence of the various perturbing forces. To utilize the improved data, the required extension of the theory involves a major effort in developing and manipulating the algebraic series used to describe the equations of motion of the satellite. Such an undertaking is time consuming and prone to errors when using manual methods. Thus it is imperative that computer-automated algebra be developed to manipulate the extensive literal expansions required for an accurate analytical theory.

The earliest efforts in programming literal algebraic expansions on a computer were carried out by Herget (1) in 1958. However, due to the limitations of the IBM-650 in use at that time, little progress was made until the introduction of the high speed computers in the early sixties. Formac (2) was introduced in 1964 and was followed by Mathlab (3), Reduce (4), and I.A.M. (5) a few years later. These languages were quite general and required little programming skill on the part of the programmer 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 involved algebraic problems encountered in celestial mechanics. Many minutes of central processor time were required when solving seemingly simple problems. Users were almost always plagued with system 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 cope with their own specialized needs. For example, Deprit (6) has introduced a program called E.S.P. (Echeloned Series Processor) to compute an analytical lunar ephemeris. In a similar manner, Jefferys (7) and Broucke (8) have constructed similar programs to manipulate the Poisson series occurring in classical perturbation theory.

NRL Problem B01-10, Project RR 003-02-41-6152. This is a final report on one phase of the problem; work is continuing on other phases. Manuscript submitted February 22, 1973.

The program described here grew mainly out of the author's unsuccessful attempts at implementing Reduce at the Naval Research Laboratory (NRL). This language is regarded as one of the best and most versatile algebraic manipulators available at present. However, the program is written in Lisp (9) (a general list-processing language) and requires a Lisp 1.5 compiler for operation. At present Lisp generally has not been operational on any CDC computer with the sole exception of the CDC-6600 at the University of Texas. (It is operational, however, for most IBM machines.) Furthermore, the limited storage capacity of the medium-size NRL CDC-3800 computer is not sufficient to handle Reduce when any reasonable size problem is considered.

Another system, I.A.M. (Interactive Algebraic Manipulation), has been available at NRL on a time-sharing basis. This is a general algebraic manipulator designed for the nonprogrammer. However, it was found to be slow, expensive to use, and incapable of expanding and applying implicit identities to trigonometric series. The program required over 20 s of central processor time on a PDP-10 computer to perform a calculation which required about 3 hr by hand. Although this amounts to a time saving of over 500 to 1, the system is still not effective for extensive calculations.

To be effective, one should be able to reproduce a computation of the size of, say, Delaunay's original lunar theory in a reasonably small amount of central processor (cp) time (several minutes or less) on any present-day, large-size electronic computer. Although this may seem like an unrealistic requirement, it should be recalled that one of the fundamental requirements of the program being considered here is the ability to significantly expand upon such a theory in a reasonable amount of cp time. Here we define a reasonable increment of cp time 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 would be unreasonable in relation to the machine – just as 20 years of manual computation is unreasonable for one man.

As we require that the machine should 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 (this naturally assumes that 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.

It is interesting to note that the overall modern high speed electronic computer is capable of outpacing 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 the present state of the art.

PROGRAM OBJECTIVES

The construction of a general purpose algebraic manipulator suitable to the needs of a diverse multidisciplinary research community appeared to be unfeasible 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. Compatability of the program with the NRL CDC-3800 and an efficiency factor of over 100,000 to 1 were specific goals of the developmental program.

The programming was carried out in Fortran because of its compatability with other machines and the ease of interpretation of this language by other users. However, to make the program user orientated,

it may become necessary to rewrite several key subroutines in assembler language. Currently, the NRL computer center is considering replacing its CDC-3800 with a new and much larger machine. Once this transpires, an attempt to develop an assembler language program may be made.

A program was therefore devised to manipulate the sometimes lengthy Poisson 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 integration of one or more Poisson 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 foundations of certain astrodynamical problems.

INTERNAL REPRESENTATION

Each term of a Poisson series can be written in the form

$$\begin{pmatrix}\text{Rational}\\ \text{fraction}\end{pmatrix} \cdot \begin{pmatrix}\text{Polynomial of}\\ \text{several variables}\end{pmatrix} \cdot \begin{pmatrix}\sin\\ \cos\end{pmatrix} \cdot \begin{pmatrix}\text{Argument involving a linear}\\ \text{combination of several angles}\end{pmatrix}.$$

Many problems in classical perturbation theory can be solved using a series of this type. 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 six 48-bit words. These words, which are in common to all subroutines, are

NEXT(k)	gives the location (index) of the next term in the list
N(k)	integer numerator
M(k)	integer denominator
IEX(k)	polynomial index containing eight packed* integer exponents whose values are restricted from −32 to +31
ITRIG(k)	sine-cosine bit; −1 or +1 for sine and cosine, respectively
IRG(k)	trigonometric argument index containing the (up to eight) packed integer coefficients whose values are restricted from −32 to +31.

Both IEX(k) and IRG(k) are 48-bit octal words of the form

iijjkkllmmooppqq,

where ii, jj, etc. denote the eight packed octal coefficients in the range $0 \leqslant ii \leqslant 77$. The octal constant of 40 is taken as zero, 46 as +6, 35 as −3, etc. Thus the null index is of the form

4040404040404040 (octal).

*"Packed" in this context means the storage of several smaller words into the space of one larger 48-bit word.

Because of a hardware constraint* on the CDC-3800, the null index is restricted to the form

2040404040404040 (octal).

Thus, the leading integer polynomial and trigonometric coefficient is restricted to the range of −16 to +15.

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 which denotes the null series (see Fig. 1). No limit is set as to the total number of different series, provided that the total available memory is not exceeded. In other words, space is allocated to the nonvanishing terms only. Also, adjacent elements of a particular list may be interlaced with the terms of separate lists and the free storage; e.g., see Fig. 2.

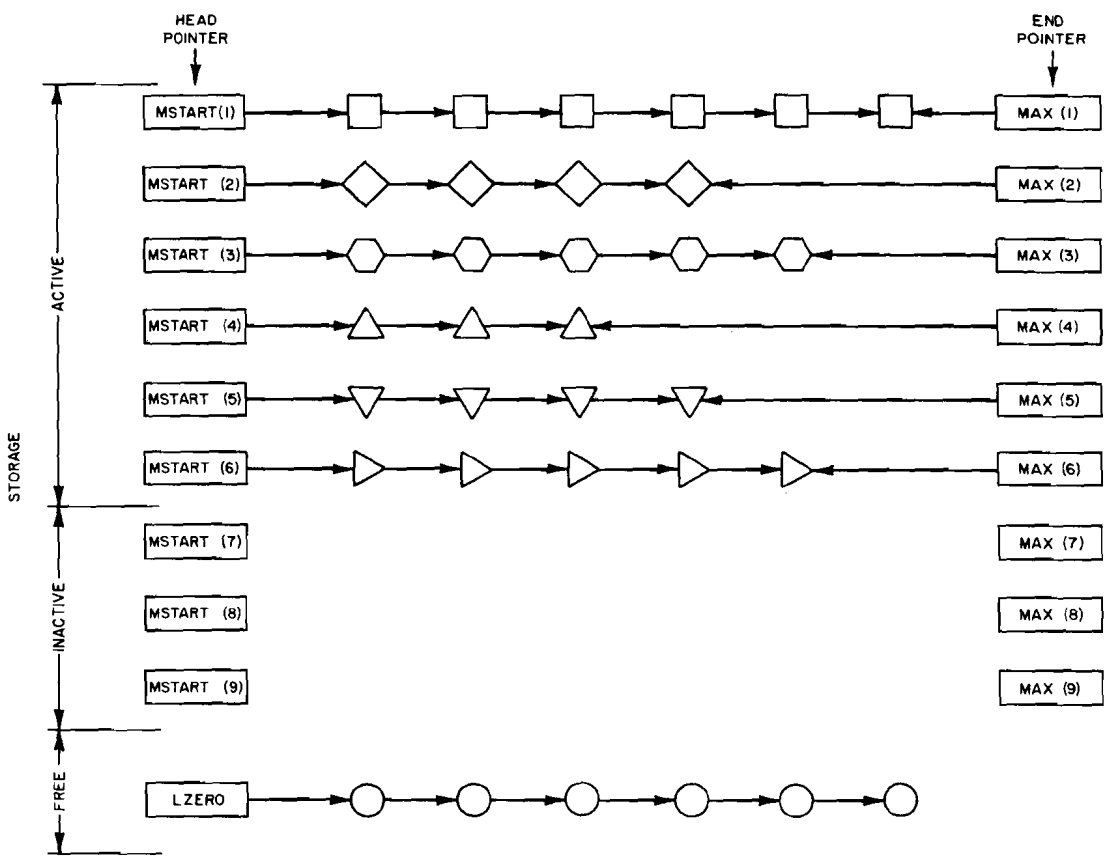

Fig. 1 – Internal list structure (symbolic). Round symbols denote free storage throughout.

*The largest (octal) integer permitted is 3777 7777 7777 7777.

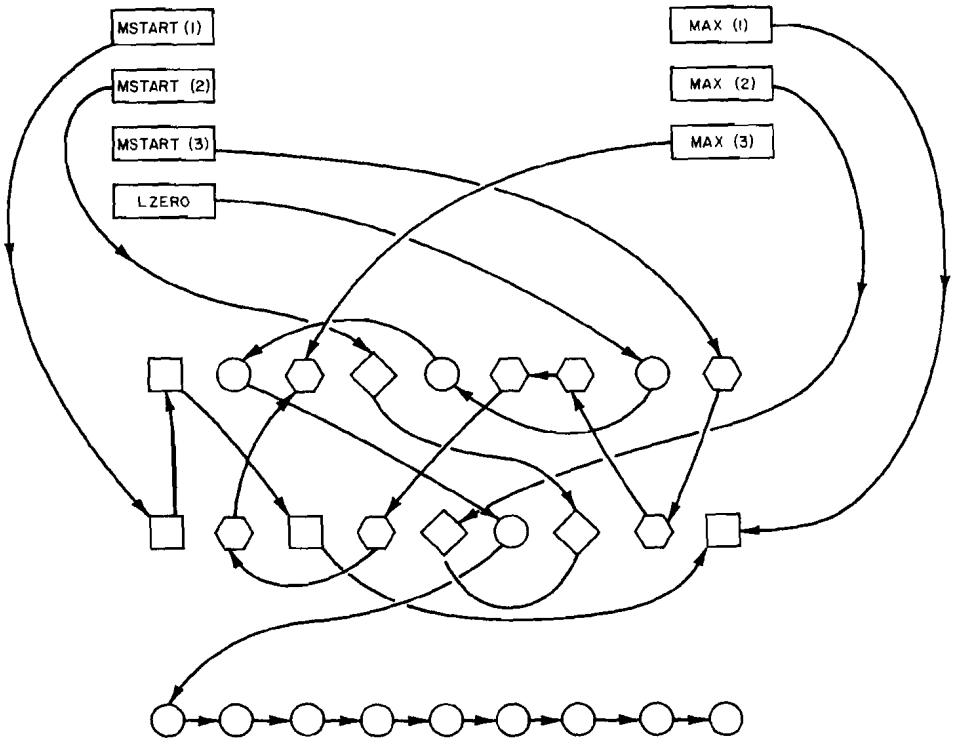

Fig. 2 – Example of actual internal list structure

The location of the lead term of a particular list I is given by the head pointer MSTART(I) and its terminal location by the end pointer MAX(I). The elements of each list are chained together by pointers, i.e., the location of the (j+1)th term is given by NEXT(j). The location of 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 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 cell storage where it is again available for use as required. "Garbage collection" is carried out continuously in this manner. The storage management is automatically carried out by the data chain routines (to be described later) and thus is of no concern to the potential 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 upon those before it and thus the coding is kept relatively simple, most routines being kept to less than one page of Fortran statements. 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

This subroutine 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 is set to denote the first free cell.
- The first 20 prime factors are defined.

Subroutine Switch(I, J)

This routine interchanges the correspondence of the series I with that of J; i.e., I ⇄ J.

Subroutine Zero(I)

This routine negates the series I and releases its occupied space to free storage. To illustrate this subroutine, suppose the list arrangement of Fig. 3a. CALL ZERO(I) rearranges the pointers, and the resulting list arrangement is shown in Fig. 3b. MSTART(I) is set to zero and I becomes the null series.

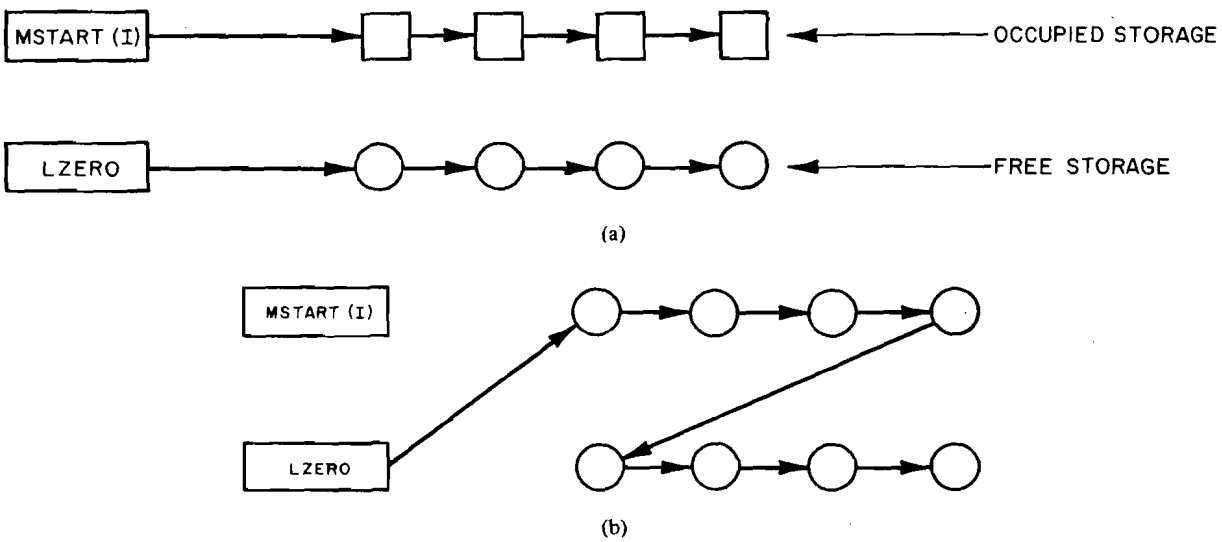

Fig. 3 – List arrangement before and after CALL ZERO(I)

Subroutine TRANSF(I, J)

J is negated (if active) and the space returned to free storage. A new list J is created equivalent to but separate from I. Consider the particular list structure of Fig. 4a. After CALL TRANSF(I,J) the rearrangement of Fig. 4b takes place.

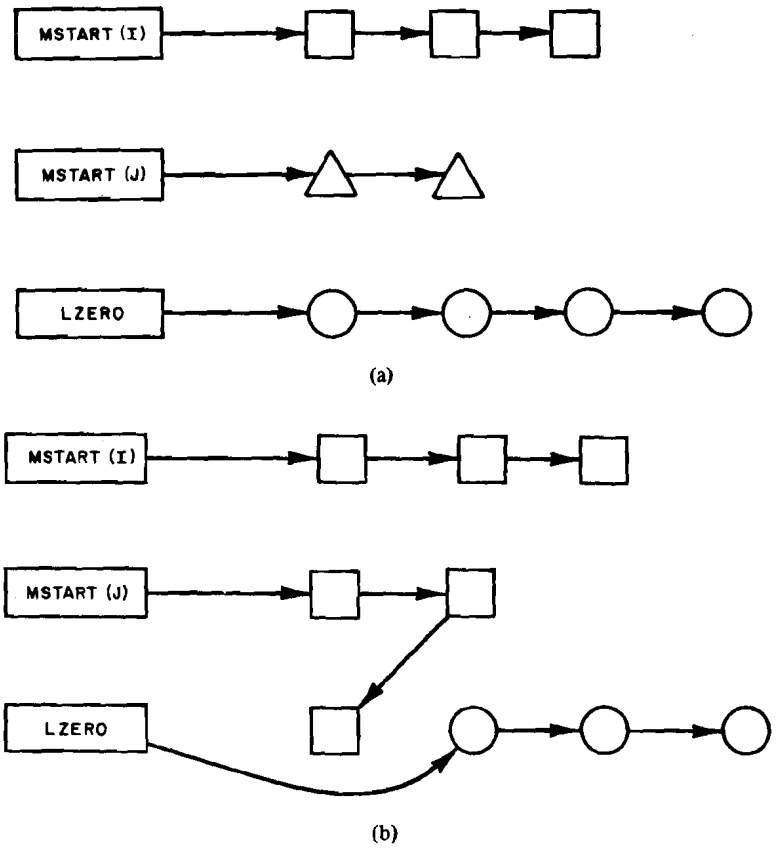

Fig. 4 – List arrangement before and after CALL TRANSF(I,J)

Subroutine PACK(I)

All entries of I which are zero are returned to free storage and only the nonzero entries are chained together: e.g., suppose we have the structure shown in Fig. 5a. After CALL PACK(I) the list is rearranged per Fig. 5b.

Subroutine SIMPLIFY(I)

SIMPLIFY(I) collects like entries in I. The resulting unneeded storage is released to free storage. The terms are ordered according to the argument index, the sine-cosine bit, and the polynomial index, respectively, during this operation.

Subroutine ORDER(I)

The entries of I are rearranged according to the special ordering priorities set up by the user. Usually the terms are ordered in the standard way in the routines SIMPLIFY and MULTIPLY; however, ORDER(I) will rearrange the list to meet the special requirements.

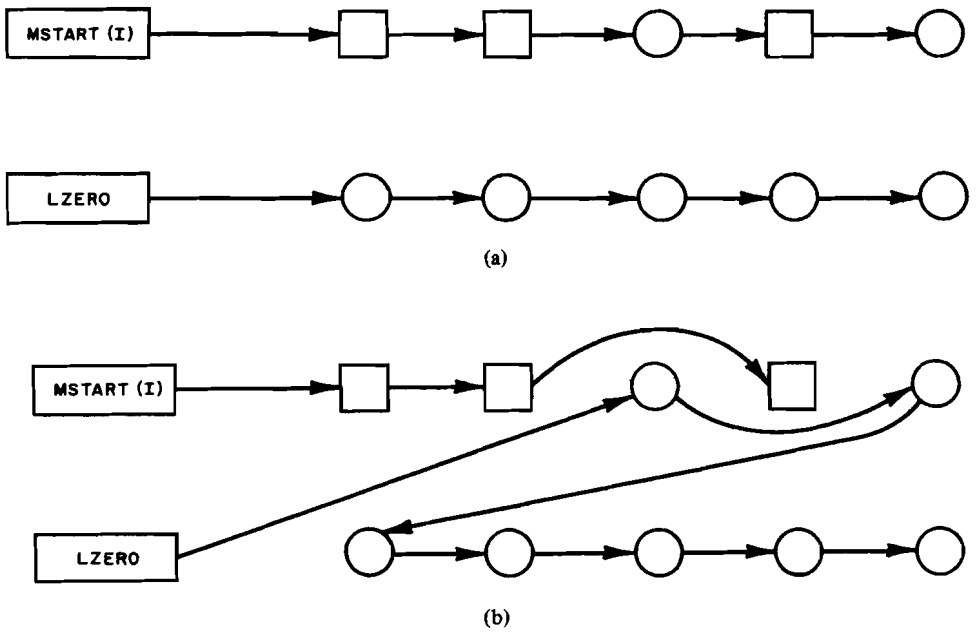

Fig. 5 – List arrangement before and after CALL PACK(I)

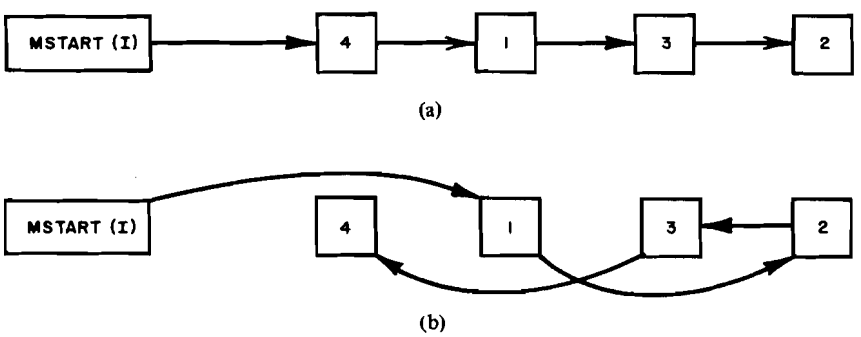

Fig. 6 – List arrangement before and after CALL ORDER(I)

Suppose the unordered list of Fig. 6a exists. CALL ORDER(I) rearranges the pointers as in Fig. 6b. The actual terms are not moved in storage. Only the pointers are recomputed.

Subroutine REVERSE(I)

The order of the series I is reversed. Suppose we have the list arrangement of Fig. 7a. After CALL REVERSE(I), the pointers are rearranged per Fig. 7b. This routine reduces the accumulative addition time significantly. It is most effective when adding a small number of new terms to a much larger, already simplified result.

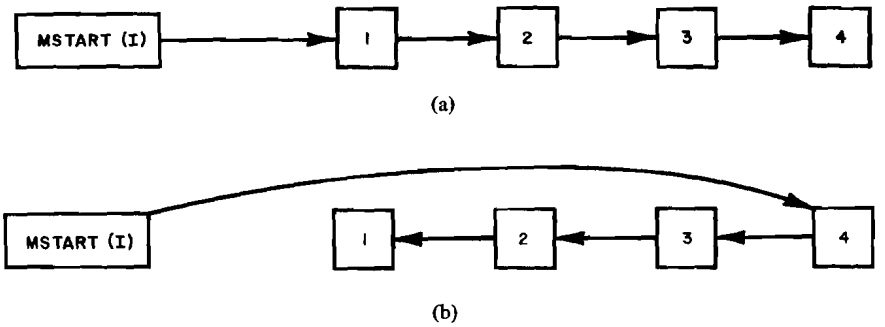

Fig. 7 – List arrangement before and after CALL REVERSE(I)

ALGEBRAIC MANIPULATIONS

Another set of subroutines has been devised to carry out the various mathematical and algebraic operations. These procedures and the speed with which they are carried out depend upon the data chain procedures just described. Before these routines are described, a few words about the computational mode is in order. Whenever possible, 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 floating-point 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 subtracting theoretically equal terms. Provision for negating these terms must be made because badly needed storage is wasted if these small coefficients are retained (see subroutine FACTOR below). The use of floating-point coefficients is almost mandatory when performing extensive high-order calculations and is also convenient when evaluating literal expressions numerically.

Subroutine FACTOR (N, M)

A call to this subroutine automatically removes the first four prime factors (2, 3, 5, 7) from the numerator and denominator (N, M). More or less factorization may be carried out if desired. The first 20 prime factors may be removed if necessary. When working with inexact coefficients, this subroutine negates any term whose coefficient is less than EPSILON, a lower limit on the size of the coefficient. This limit is set according to the physics of the problem at hand; i.e., several orders of magnitude less than the smallest coefficient physically meaningful.

Subroutine ADD (I, J, K)

For $I \neq J \neq K$ the series J is added to I and the simplified result is placed in K. If K = I the routine acts as an accumulator, i.e., $I+J \rightarrow I$.

Subroutine SUB (I, J, K)

This routine is similar to ADD, only J is subtracted from I.

Subroutine MULCONS (I, n, m)

The series I is multiplied term by term by the rational fraction n/m.

Subroutine MULTIPLY (I, J, K)

Each term of the series I is multiplied with that 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:

$$\begin{aligned} 2\cos x\cos y &= \cos(x+y) + \cos(x-y) \\ 2\cos x\sin y &= \sin(x+y) - \sin(x-y) \\ 2\sin x\cos y &= \sin(x+y) + \sin(x-y) \\ 2\sin x\sin y &= -\cos(x+y) + \cos(x-y) \\ \cos(-x) &= +\cos x \\ \sin(-x) &= -\sin x \\ \cos(0) &= 1 \\ \sin(0) &= 0 \end{aligned} \tag{1}$$

Due to the introduction of the above trigonometric identities, two terms are produced (labeled A and B in Fig. 8) for each product $I_m J_n$. The required spaces for A and B are taken from free storage.

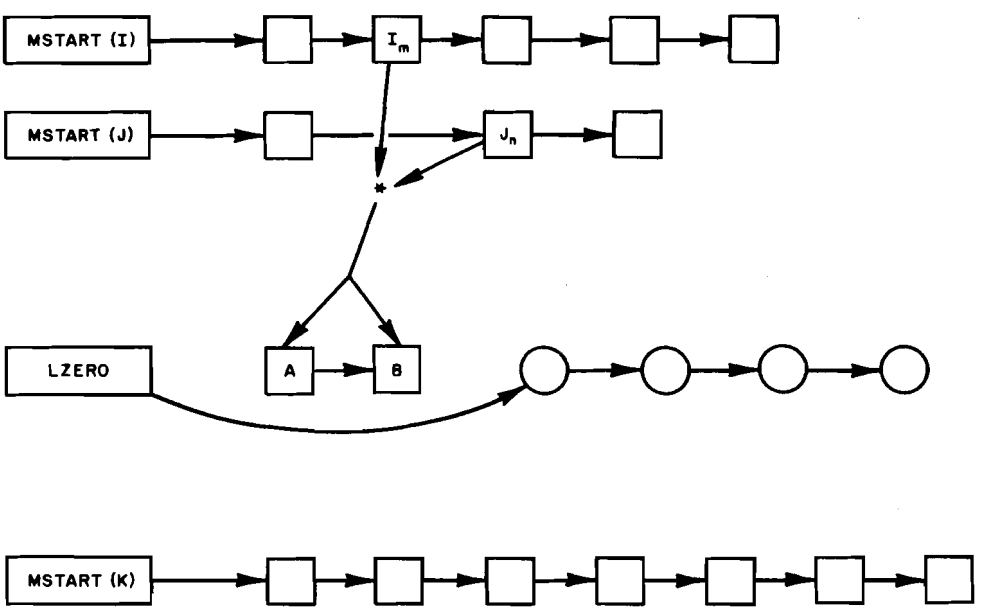

Fig. 8 – List arrangement during multiplication of subproduct $I_m J_n$

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. 9) 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.

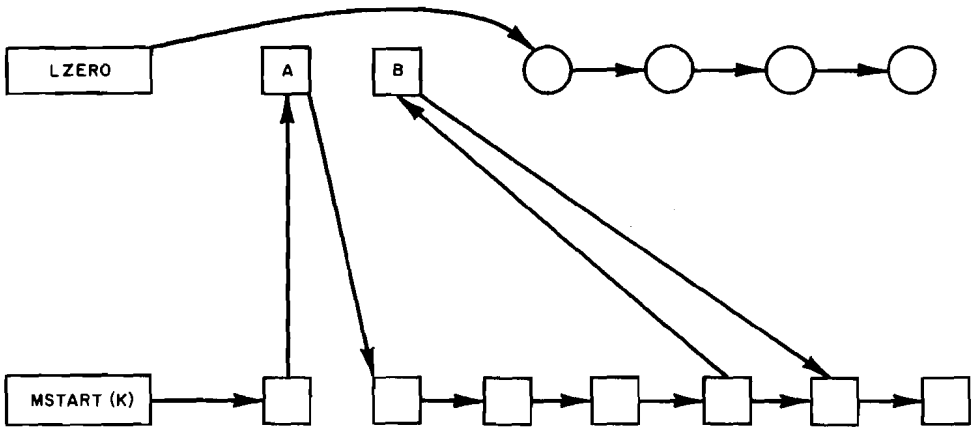

Fig. 9 – List arrangement after A and B terms are inserted in K

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 scheme has been devised using this fact to perform the search in an optimal manner. 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 correct location for the insertion at the new term has been determined. For a list of 4000 terms only 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 the successive term) are mapped onto a linear array. It is this area 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 hr of cp time without the optimal search. After the implementation of this technique, the time was reduced to less than 2 min!

Subroutine POWER (I, n, J)

The series I is multiplied by itself (n−1) times and stored as J; i.e., $I^n \rightarrow J$.

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

The binomial expansion of the quantity $(1+I)^{m/n}$ is computed where $|I| \ll 1$. The pair (m, n) are integers. The expansion is terminated at order k in I. The result is stored as J.

Subroutine DERIV (I, j, K)

The Ith chain is differentiated term by term with respect to the jth packed variable. The result is stored as the Kth chain. I remains unaltered unless I = K.

Subroutine NDERIV (I, j, ℓ, K)

The Ith chain is differentiated with respect to the jth packed variable ℓ times. The result is stored as K.

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

The Poisson bracket $[I, J]_{m,n}$ is computed with respect to the variables m, n. The result is stored as K; i.e.,

$$\left[\frac{\partial I}{\partial m}\frac{\partial J}{\partial n} - \frac{\partial I}{\partial n}\frac{\partial J}{\partial m}\right] \rightarrow K.$$

Subroutine INTEG (I, j, K)

Each term of I is integrated with respect to the jth packed variable. The result is stored as K.

Subroutine SUBAB (I, J, m, n, ℓ)

In the Ith chain, the substitution $m^2 = 1 - n^2$ is made where m, n represent the m and nth packed variable. That is, when sin i, 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 ℓ times and the result is stored as J.

In the expression $315/512 \cos^4 i - 135/256 \cos^2 i + 27/512$, CALL SUBAB (I, J, m, n, 2) performs the substitution, $\cos^2 i = 1 - \sin^2 i$, twice. The result is $315/512 \sin^4 i - 45/64 \sin^2 i + 9/64$. This substitution sometimes results in a substantial reduction in the size of certain intermediate and final expressions.

Subroutine CUT (I, J)

All (a) periodic terms in I are negated. The released memory is returned to free storage. The result is stored in J.

Subroutine SECULAR (I, J)

Similar to CUT(I, J), except that only the periodic terms are negated. Only the constant terms remain.

Subroutine TRUNCATE(I, J, j, ℓ)

All terms above order ℓ in the jth packed polynomial variable are negated. In general only the terms between order LORDER and MORDER are retained during multiplication. The rest are negated before they are stored and simplified. The result is stored as J if $I \neq J$.

Subroutine SELECT (I, J, j, ℓ)

All terms of order ℓ in the jth packed polynomial variable of the series I are retained. All other members are removed.

Subroutine CHOOSE (I, J, j, ℓ)

All terms of periodicity ℓ in the jth packed trigonometric argument are retained.

Subroutine TAYLOR (I, J, ℓ, m, K)

A Taylor series expansion is performed on the series I with respect to the ℓth packed variable;

$$f(x + \Delta X) = \sum_{k=0}^{m} \frac{1}{k!} \frac{\partial^k}{\partial x^k} f(x) (\Delta X)^k. \tag{2}$$

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

Subroutine EVALUATE (I, J, j, A)

The floating-point number A is substituted for the jth packed polynomial variable. The result is stored as J.

Subroutine OUT (I)

The series I is printed or punched 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 different forms.

- The coefficients of each term are given as exact integer fractions whenever possible. The remaining coefficients are presented as floating-point numbers.
- All coefficients are presented as floating-point numbers.

PROBLEM EXAMPLES

Exponentiation

Suppose we wanted to express $(E \sin M)^5$ in the form

$$(E \sin M)^5 = \sum_{i=0}^{5} C_i \sin jM. \tag{3}$$

We store the quantity (E sin M) in location 1. To obtain the final result we write CALL POWER (1, 5, 2). The program computes the desired quantity and places the result in location 2. To obtain the printed result, we write CALL OUT (2). The result is

```
 1*E** 5*SIN( 5M)
16

-5*E** 5*SIN( 3M)
16

 5*E** 5*SIN( 1M)
 8
```

(4)

Multiplication

Let us multiply the series from Eq. (4) by (E sin M). The result as obtained from the subroutine MULTIPLY is

```
 -1*E** 6*COS( 6M)
 32

  3*E** 6*COS( 4M)
 16

-15*E** 6*COS( 2M)
 32

  5*E** 6
 16
```

(5)

The result is essentially $(E \sin M)^5$ (E sin M) or $(E \sin M)^6$.

A Binomial Expansion

Suppose we wished to expand $(1 + E \sin M)^{-1}$ to order 8 in E. We write CALL EXPAND (1, 8, −1, 1, 2) and CALL OUT (2). The result is shown below.

```
  1*E** 8*COS( 8M)
128

  1*E** 7*SIN( 7M)
 64

 -1*E** 8*COS( 6M)
 16

 -7*E** 8*COS( 2M)
 16

-15*E** 6*COS( 2M)
 32

 -1*E** 4*COS( 2M)
  2
```

(Equation 6 Continues)

(Equation 6 Continued)

```
-1*E** 6*COS( 6M)
32

-7*E** 7*SIN( 5M)
64

-1*E** 5*SIN( 5M)
16

 7*E** 8*COS( 4M)
32

 3*E** 6*COS( 4M)
16

 1*E** 4*COS( 4M)
 8

21*E** 7*SIN( 3M)
64

 5*E** 5*SIN( 3M)
16

 1*E** 3*SIN( 3M)
 4

-1*E** 2*COS( 2M)
 2

-35*E** 7*SIN( 1M)
 64

 -5*E** 5*SIN( 1M)
  8

 -3*E** 3*SIN( 1M)
  4

 -1*E** 1*SIN( 1M)

 35*E** 8
128

  5*E** 6
 16

  3*E** 4
  8

  1*E** 2
  2

  1*E** 0
```

(6)

A Polynomial Raised to a Power

An example would be (X+Y+Z)**24. We store the polynomial X+Y+Z in location 1 and write CALL POWER (1,24,2). To obtain the printed result, we write CALL OUT (2). The result is shown below in Eq. (7):

```
     1*X**24
    24*X**23*Y** 1
    24*X**23*Z** 1
   276*X**22*Y** 2
   552*X**22*Y** 1*Z** 1
   276*X**22*Z** 2
  2024*X**21*Y** 3
  6072*X**21*Y** 2*Z** 1
  6072*X**21*Y** 1*Z** 2
  2024*X**21*Z** 3
 10626*X**20*Y** 4
 42504*X**20*Y** 3*Z** 1
 63756*X**20*Y** 2*Z** 2
 42504*X**20*Y** 1*Z** 3
 10626*X**20*Z** 4
 42504*X**19*Y** 5
212520*X**19*Y** 4*Z** 1
425040*X**19*Y** 3*Z** 2
425040*X**19*Y** 2*Z** 3
212520*X**19*Y** 1*Z** 4
 42504*X**19*Z** 5
134596*X**18*Y** 6
807576*X**18*Y** 5*Z** 1
2018940*X**18*Y** 4*Z** 2
```

(Equation 7 Continues)

(Equation 7 Continued)

```
 2691920*X**18*Y** 3*Z** 3
 2018940*X**18*Y** 2*Z** 4
  807576*X**18*Y** 1*Z** 5
  134596*X**18*Z** 6
  346104*X**17*Y** 7
 2422728*X**17*Y** 6*Z** 1
 7268184*X**17*Y** 5*Z** 2
12113640*X**17*Y** 4*Z** 3
12113640*X**17*Y** 3*Z** 4
 7268184*X**17*Y** 2*Z** 5
 2422728*X**17*Y** 1*Z** 6
  346104*X**17*Z** 7
  735471*X**16*Y** 8
 5883768*X**16*Y** 7*Z** 1
20593188*X**16*Y** 6*Z** 2
41186376*X**16*Y** 5*Z** 3
51482970*X**16*Y** 4*Z** 4
41186376*X**16*Y** 3*Z** 5
20593188*X**16*Y** 2*Z** 6
 5883768*X**16*Y** 1*Z** 7
  735471*X**16*Z** 8
 1307504*X**15*Y** 9
11767536*X**15*Y** 8*Z** 1
47070144*X**15*Y** 7*Z** 2
109830336*X**15*Y** 6*Z** 3
164745504*X**15*Y** 5*Z** 4
164745504*X**15*Y** 4*Z** 5
109830336*X**15*Y** 3*Z** 6
 47070144*X**15*Y** 2*Z** 7
 11767536*X**15*Y** 1*Z** 8
  1307504*X**15*Z** 9
  1961256*X**14*Y**10
 19612560*X**14*Y** 9*Z** 1
 88256520*X**14*Y** 8*Z** 2
235350720*X**14*Y** 7*Z** 3
411863760*X**14*Y** 6*Z** 4
494236512*X**14*Y** 5*Z** 5
411863760*X**14*Y** 4*Z** 6
235350720*X**14*Y** 3*Z** 7
 88256520*X**14*Y** 2*Z** 8
 19612560*X**14*Y** 1*Z** 9
  1961256*X**14*Z**10
  2496144*X**13*Y**11
 27457584*X**13*Y**10*Z** 1
137287920*X**13*Y** 9*Z** 2
411863760*X**13*Y** 8*Z** 3
823727520*X**13*Y** 7*Z** 4
1153218528*X**13*Y** 6*Z** 5
1153218528*X**13*Y** 5*Z** 6
823727520*X**13*Y** 4*Z** 7
411863760*X**13*Y** 3*Z** 8
137287920*X**13*Y** 2*Z** 9
 27457584*X**13*Y** 1*Z**10
  2496144*X**13*Z**11
  2704156*X**12*Y**12
 32449872*X**12*Y**11*Z** 1
178474296*X**12*Y**10*Z** 2
594914320*X**12*Y** 9*Z** 3
1338557220*X**12*Y** 8*Z** 4
2141691552*X**12*Y** 7*Z** 5
2498640144*X**12*Y** 6*Z** 6
2141691552*X**12*Y** 5*Z** 7
1338557220*X**12*Y** 4*Z** 8
 594914320*X**12*Y** 3*Z** 9
 178474296*X**12*Y** 2*Z**10
  32449872*X**12*Y** 1*Z**11
  2704156*X**12*Z**12
  2496144*X**11*Y**13
 32449872*X**11*Y**12*Z** 1
194699232*X**11*Y**11*Z** 2
713897184*X**11*Y**10*Z** 3
1784742960*X**11*Y** 9*Z** 4
3212537328*X**11*Y** 8*Z** 5
4283383104*X**11*Y** 7*Z** 6
4283383104*X**11*Y** 6*Z** 7
3212537328*X**11*Y** 5*Z** 8
1784742960*X**11*Y** 4*Z** 9
 713897184*X**11*Y** 3*Z**10
 194699232*X**11*Y** 2*Z**11
  32449872*X**11*Y** 1*Z**12
  2496144*X**11*Z**13
  1961256*X**10*Y**14
 27457584*X**10*Y**13*Z** 1
178474296*X**10*Y**12*Z** 2
713897184*X**10*Y**11*Z** 3
1963217256*X**10*Y**10*Z** 4
3926434512*X**10*Y** 9*Z** 5
5889651768*X**10*Y** 8*Z** 6
6731030592*X**10*Y** 7*Z** 7
5889651768*X**10*Y** 6*Z** 8
3926434512*X**10*Y** 5*Z** 9
1963217256*X**10*Y** 4*Z**10
713897184*X**10*Y** 3*Z**11
178474296*X**10*Y** 2*Z**12
 27457584*X**10*Y** 1*Z**13
  1961256*X**10*Z**14
  1307504*X** 9*Y**15
 19612560*X** 9*Y**14*Z** 1
```

(Equation 7 Continued)

```
 137287920*X**  9*Y**13*Z**  2
 594914320*X**  9*Y**12*Z**  3
1784742960*X**  9*Y**11*Z**  4
3926434512*X**  9*Y**10*Z**  5
6544057520*X**  9*Y**  9*Z**  6
8413788240*X**  9*Y**  8*Z**  7
8413788240*X**  9*Y**  7*Z**  8
6544057520*X**  9*Y**  6*Z**  9
3926434512*X**  9*Y**  5*Z**10
1784742960*X**  9*Y**  4*Z**11
 594914320*X**  9*Y**  3*Z**12
 137287920*X**  9*Y**  2*Z**13
  19612560*X**  9*Y**  1*Z**14
   1307504*X**  9*Z**15
    735471*X**  8*Y**16
  11767536*X**  8*Y**15*Z**  1
  88256520*X**  8*Y**14*Z**  2
 411863760*X**  8*Y**13*Z**  3
1338557220*X**  8*Y**12*Z**  4
3212537328*X**  8*Y**11*Z**  5
5889651768*X**  8*Y**10*Z**  6
8413788240*X**  8*Y**  9*Z**  7
9465511770*X**  8*Y**  8*Z**  8
8413788240*X**  8*Y**  7*Z**  9
5889651768*X**  8*Y**  6*Z**10
3212537328*X**  8*Y**  5*Z**11
1338557220*X**  8*Y**  4*Z**12
 411863760*X**  8*Y**  3*Z**13
  88256520*X**  8*Y**  2*Z**14
  11767536*X**  8*Y**  1*Z**15
    735471*X**  8*Z**16
    346104*X**  7*Y**17
   5883768*X**  7*Y**16*Z**  1
  47070144*X**  7*Y**15*Z**  2
 235350720*X**  7*Y**14*Z**  3
 823727520*X**  7*Y**13*Z**  4
2141691552*X**  7*Y**12*Z**  5
4283383104*X**  7*Y**11*Z**  6
6731030592*X**  7*Y**10*Z**  7
8413788240*X**  7*Y**  9*Z**  8
8413788240*X**  7*Y**  8*Z**  9
6731030592*X**  7*Y**  7*Z**10
4283383104*X**  7*Y**  6*Z**11
2141691552*X**  7*Y**  5*Z**12
 823727520*X**  7*Y**  4*Z**13
 235350720*X**  7*Y**  3*Z**14
  47070144*X**  7*Y**  2*Z**15
   5883768*X**  7*Y**  1*Z**16
    346104*X**  7*Z**17
    134596*X**  6*Y**18
   2422728*X**  6*Y**17*Z**  1
  20593188*X**  6*Y**16*Z**  2
 109830336*X**  6*Y**15*Z**  3
 411863760*X**  6*Y**14*Z**  4
1153218528*X**  6*Y**13*Z**  5
2498640144*X**  6*Y**12*Z**  6
4283383104*X**  6*Y**11*Z**  7
5889651768*X**  6*Y**10*Z**  8
6544057520*X**  6*Y**  9*Z**  9
5889651768*X**  6*Y**  8*Z**10
4283383104*X**  6*Y**  7*Z**11
2498640144*X**  6*Y**  6*Z**12
1153218528*X**  6*Y**  5*Z**13
 411863760*X**  6*Y**  4*Z**14
 109830336*X**  6*Y**  3*Z**15
  20593188*X**  6*Y**  2*Z**16
   2422728*X**  6*Y**  1*Z**17
    134596*X**  6*Z**18
     42504*X**  5*Y**19
    807576*X**  5*Y**18*Z**  1
   7268184*X**  5*Y**17*Z**  2
  41186376*X**  5*Y**16*Z**  3
 164745504*X**  5*Y**15*Z**  4
 494236512*X**  5*Y**14*Z**  5
1153218528*X**  5*Y**13*Z**  6
2141691552*X**  5*Y**12*Z**  7
3212537328*X**  5*Y**11*Z**  8
3926434512*X**  5*Y**10*Z**  9
3926434512*X**  5*Y**  9*Z**10
3212537328*X**  5*Y**  8*Z**11
2141691552*X**  5*Y**  7*Z**12
1153218528*X**  5*Y**  6*Z**13
 494236512*X**  5*Y**  5*Z**14
 164745504*X**  5*Y**  4*Z**15
  41186376*X**  5*Y**  3*Z**16
   7268184*X**  5*Y**  2*Z**17
    807576*X**  5*Y**  1*Z**18
     42504*X**  5*Z**19
     10626*X**  4*Y**20
    212520*X**  4*Y**19*Z**  1
   2018940*X**  4*Y**18*Z**  2
  12113640*X**  4*Y**17*Z**  3
  51482970*X**  4*Y**16*Z**  4
 164745504*X**  4*Y**15*Z**  5
 411863760*X**  4*Y**14*Z**  6
 823727520*X**  4*Y**13*Z**  7
1338557220*X**  4*Y**12*Z**  8
1784742960*X**  4*Y**11*Z**  9
```

(Equation 7 Continued)

```
1963217256*X** 4*Y**10*Z**10
1784742960*X** 4*Y** 9*Z**11
1338557220*X** 4*Y** 8*Z**12
 823727520*X** 4*Y** 7*Z**13
 411863760*X** 4*Y** 6*Z**14
 164745504*X** 4*Y** 5*Z**15
  51482970*X** 4*Y** 4*Z**16
  12113640*X** 4*Y** 3*Z**17
   2018940*X** 4*Y** 2*Z**18
    212520*X** 4*Y** 1*Z**19
     10626*X** 4*Z**20
      2024*X** 3*Y**21
     42504*X** 3*Y**20*Z** 1
    425040*X** 3*Y**19*Z** 2
   2691920*X** 3*Y**18*Z** 3
  12113640*X** 3*Y**17*Z** 4
  41186376*X** 3*Y**16*Z** 5
 109830336*X** 3*Y**15*Z** 6
 235350720*X** 3*Y**14*Z** 7
 411863760*X** 3*Y**13*Z** 8
 594914320*X** 3*Y**12*Z** 9
 713897184*X** 3*Y**11*Z**10
 713897184*X** 3*Y**10*Z**11
 594914320*X** 3*Y** 9*Z**12
 411863760*X** 3*Y** 8*Z**13
 235350720*X** 3*Y** 7*Z**14
 109830336*X** 3*Y** 6*Z**15
  41186376*X** 3*Y** 5*Z**16
  12113640*X** 3*Y** 4*Z**17
   2691920*X** 3*Y** 3*Z**18
    425040*X** 3*Y** 2*Z**19
     42504*X** 3*Y** 1*Z**20
      2024*X** 3*Z**21
       276*X** 2*Y**22
      6072*X** 2*Y**21*Z** 1
     63756*X** 2*Y**20*Z** 2
    425040*X** 2*Y**19*Z** 3
   2018940*X** 2*Y**18*Z** 4
   7268184*X** 2*Y**17*Z** 5
  20593188*X** 2*Y**16*Z** 6
  47070144*X** 2*Y**15*Z** 7
  88256520*X** 2*Y**14*Z** 8
 137287920*X** 2*Y**13*Z** 9
 178474296*X** 2*Y**12*Z**10
 194699232*X** 2*Y**11*Z**11
 178474296*X** 2*Y**10*Z**12
 137287920*X** 2*Y** 9*Z**13
  88256520*X** 2*Y** 8*Z**14
  47070144*X** 2*Y** 7*Z**15
  20593188*X** 2*Y** 6*Z**16
   7268184*X** 2*Y** 5*Z**17
   2018940*X** 2*Y** 4*Z**18
    425040*X** 2*Y** 3*Z**19
     63756*X** 2*Y** 2*Z**20
      6072*X** 2*Y** 1*Z**21
       276*X** 2*Z**22
        24*X** 1*Y**23
       552*X** 1*Y**22*Z** 1
      6072*X** 1*Y**21*Z** 2
     42504*X** 1*Y**20*Z** 3
    212520*X** 1*Y**19*Z** 4
    807576*X** 1*Y**18*Z** 5
   2422728*X** 1*Y**17*Z** 6
   5883768*X** 1*Y**16*Z** 7
  11767536*X** 1*Y**15*Z** 8
  19612560*X** 1*Y**14*Z** 9
  27457584*X** 1*Y**13*Z**10
  32449872*X** 1*Y**12*Z**11
  32449872*X** 1*Y**11*Z**12
  27457584*X** 1*Y**10*Z**13
  19612560*X** 1*Y** 9*Z**14
  11767536*X** 1*Y** 8*Z**15
   5883768*X** 1*Y** 7*Z**16
   2422728*X** 1*Y** 6*Z**17
    807576*X** 1*Y** 5*Z**18
    212520*X** 1*Y** 4*Z**19
     42504*X** 1*Y** 3*Z**20
      6072*X** 1*Y** 2*Z**21
       552*X** 1*Y** 1*Z**22
        24*X** 1*Z**23
         1*Y**24
        24*Y**23*Z** 1
       276*Y**22*Z** 2
      2024*Y**21*Z** 3
     10626*Y**20*Z** 4
     42504*Y**19*Z** 5
    134596*Y**18*Z** 6
    346104*Y**17*Z** 7
    735471*Y**16*Z** 8
   1307504*Y**15*Z** 9
   1961256*Y**14*Z**10
   2496144*Y**13*Z**11
   2704156*Y**12*Z**12
   2496144*Y**11*Z**13
   1961256*Y**10*Z**14
   1307504*Y** 9*Z**15
```

(Equation 7 Continued)

```
735471*Y** 8*Z**16
346104*Y** 7*Z**17
134596*Y** 6*Z**18
 42504*Y** 5*Z**19
 10626*Y** 4*Z**20
  2024*Y** 3*Z**21
   276*Y** 2*Z**22
    24*Y** 1*Z**23
     1*Z**24
```

(7)

The Determination of the Legendre Polynomials

From the definition,

$$(1 - 2\delta E + \delta^2)^{-1/2} = \sum_{i=0}^{\infty} \delta^i \, P_i(E),$$

the quantity $(1 - 2\delta E + \delta^2)^{-1/2}$ is expanded in powers of δ by the routine EXPAND. The coefficients of each δ^i are collected by the subroutine SELECT. The results for $P_i(E)$ $(1 \leqslant i \leqslant 10)$ are given below.

$P_1(E)$:

```
1*E** 1
```

$P_2(E)$:

```
3*E** 2
 2

-1*E** 0
 2
```

$P_3(E)$:

```
5*E** 3
 2

-3*E** 1
 2
```

$P_4(E)$:

```
35*E** 4
 8

-15*E** 2
 4

 3*E** 0
 8
```

$P_5(E)$:

```
63*E** 5
 8

-35*E** 3
 4

 15*E** 1
 8
```

(Equation 8 Continues)

(Equation 8 Continued)

$P_6(E)$:

```
231*E** 6
 16

-315*E** 4
 16

105*E** 2
 16

-5*E** 0
 16
```

$P_7(E)$:

```
429*E** 7
 16

-693*E** 5
 16

315*E** 3
 16

-35*E** 1
 16
```

$P_8(E)$:

```
6435*E** 8
 128

-3003*E** 6
 32

3465*E** 4
 64

-315*E** 2
 32

35*E** 0
 128
```

$P_9(E)$:

```
12155*E** 9
 128

-6435*E** 7
 32

9009*E** 5
 64

-1155*E** 3
 32

315*E** 1
 128
```

(Equation 8 Continued)

```
P10(E):   46189*E**10
            256

        -109395*E** 8
            256

          45045*E** 6
            128

         -15015*E** 4
            128

           3465*E** 2
            256

           -63*E** 0
            256
```

(8)

A Series Solution to a Differential Equation

We choose the solution to Kepler's equation relating the orbital mean anomaly to the true anomaly. This example is chosen, as the result may be compared with the published solution in the literature (10). The differential equation to be considered is

$$\frac{df}{dM} = (1 - e^2)^{-3/2} (1 + e \cos f)^2 \quad (f = M, \text{ when } M = 0). \tag{9}$$

We wish to compute (f – M) to a high order in e. Assume $|e| \ll 1$.

To first order in e: Use

$$\frac{df}{dM} = 1 + 2e \cos f \Rightarrow f - M = 2e \sin M. \tag{10}$$

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

$$\frac{df}{dM} = (1 + 3/2e^2 + \ldots)(1 + e \cos(M + 2e \sin M))^2 \tag{11}$$

$$\Rightarrow f - M = 2e \sin M + 5/4e^2 \sin 2M + \ldots . \tag{12}$$

To order (k+1) in e: The solution to order k is used on the right-hand side as follows:

$$\frac{df}{dM} = (1-e^2)^{-3/2} \left[1 + \sum_{j=0}^{k} \frac{\partial}{\partial M^j} (e \cos M)(f-M)^j \right]^2 . \tag{13}$$

The above procedure was repeated to order 7 in e. The solution involved the multiplication and simplification of over 1500 terms using about 10 s of cp time on a CDC-3800.

The solution for (f − M) to seventh order in e is as follows.

```
                                          -43*E** 5*SIN( 3M)
f-M = 2*E** 1*SIN( 1M)                     64

     -1*E** 3*SIN( 1M)                     95*E** 7*SIN( 3M)
      4                                   512

      5*E** 5*SIN( 1M)                   103*E** 4*SIN( 4M)
     96                                   96

    107*E** 7*SIN( 1M)                  -451*E** 6*SIN( 4M)
   4608                                  480                          (14)

      5*E** 2*SIN( 2M)                  1097*E** 5*SIN( 5M)
      4                                  960

    -11*E** 4*SIN( 2M)                 -5957*E** 7*SIN( 5M)
     24                                 4608

     17*E** 6*SIN( 2M)                  1223*E** 6*SIN( 6M)
    192                                  960

     13*E** 3*SIN( 3M)                 47273*E** 7*SIN( 7M)
     12                                32256
```

We do not propose that Kepler's equation be solved in this way, as more sophisticated methods are available; however, it is obvious that such equations can be solved in this straightforward manner easily and quickly by computer. The method avoids the use of Bessel functions which are required if extensive hand calculations are made.

Another useful relationship is the quantity

$$\frac{r}{a} = \frac{1 - e^2}{1 + e \cos f}. \tag{15}$$

It is desired to compute r/a as a power series in e. The expansion for r/a is computed from Eqs. (14) and (15) as follows:

cos f is expressed in terms of M from the subroutine TAYLOR;

$$\cos f = \sum_{k=0}^{7} \frac{1}{k!} \frac{\partial^k}{\partial M^k} (\cos M)(f - M)^k, \tag{16}$$

where (f − M) has already been computed and is given by Eq. (14). The result for cos f to seventh power in e is listed below.

```
cos f = 1024*E** 7*COS( 8M)
         315

      117649*E** 6*COS( 7M)
       46080

        -162*E** 7*COS( 6M)
          35

          81*E** 5*COS( 6M)
          40

      -30625*E** 6*COS( 5M)
        9216

         625*E** 4*COS( 5M)
         384

          64*E** 7*COS( 4M)
          45

         -12*E** 5*COS( 4M)
           5

           4*E** 3*COS( 4M)
           3

        3969*E** 6*COS( 3M)
        5120

        -225*E** 4*COS( 3M)
         128

           9*E** 2*COS( 3M)
           8

          -2*E** 7*COS( 2M)
          45

           3*E** 5*COS( 2M)
           8

          -4*E** 3*COS( 2M)
           3

           1*E** 1*COS( 2M)

         -49*E** 6*COS( 1M)
        9216                                    (17)

          25*E** 4*COS( 1M)
         192

          -9*E** 2*COS( 1M)
           8

           1*E** 0*COS( 1M)

          -1*E** 1
```

The quantity E cos f is computed from Eq. (17) using the subroutine MULTIPLY, $(1 + e \cos f)^{-1}$ is determined from this result by the routine EXPAND, and $(1 - e^2)(1 + e \cos f)^{-1}$ is obtained from MULTIPLY. The result for r/a is

```
r/a = -16807*E** 7*COS( 7M)
       46080

         -27*E** 6*COS( 6M)
          80

          -1*E** 6*COS( 2M)
          16

           1*E** 4*COS( 2M)
           3
```

(Equation 18 Continues)

(Equation 18 Continued)

```
4375*E** 7*COS( 5M)
9216

-125*E** 5*COS( 5M)
 384

   2*E** 6*COS( 4M)
   5

  -1*E** 4*COS( 4M)
   3

-567*E** 7*COS( 3M)
5120

  45*E** 5*COS( 3M)
 128

  -3*E** 3*COS( 3M)
   8

  -1*E** 2*COS( 2M)
   2

   7*E** 7*COS( 1M)
9216

  -5*E** 5*COS( 1M)
 192

   3*E** 3*COS( 1M)
   8

  -1*E** 1*COS( 1M)

   1*E** 2
   2

   1*E** 0
```

(18)

Multiplying Eq. (17) and Eq. (18), the result for (r/a) cos f to the seventh power in e is as listed.

```
(r/a)cos f = 128*E** 7*COS( 8M)
             315

            16807*E** 6*COS( 7M)
            46080

              -81*E** 7*COS( 6M)
              140

               27*E** 5*COS( 6M)
               80

            -4375*E** 6*COS( 5M)
             9216

              125*E** 4*COS( 5M)
              384

                8*E** 7*COS( 4M)
               45

              -45*E** 4*COS( 3M)
              128

                3*E** 2*COS( 3M)
                8

               -1*E** 7*COS( 2M)
              180

                1*E** 5*COS( 2M)
               16

               -1*E** 3*COS( 2M)
                3

                1*E** 1*COS( 2M)
                2

               -7*E** 6*COS( 1M)
              9216
```

(Equation 19 Continues)

(Equation 19 Continues)

```
 -2*E** 5*COS( 4M)                5*E** 4*COS( 1M)
  5                              192

  1*E** 3*COS( 4M)               -3*E** 2*COS( 1M)
  3                                8

 567*E** 6*COS( 3M)                1*E** 0*COS( 1M)
5120

                                  -3*E** 1
                                   2                          (19)
```

Equations (14), (17), (18), and (19) may be checked with the published solutions in the literature (e.g., Ref. 10, pp. 76, 77). All solutions appear to be correct except for the leading term of Eq. (19). Brouwer apparently dropped this term in his expression.

We need not restrict ourselves to order seven in the solution of Eq. (9). The following result represents the extension of the solution, i.e., Eq. (14), to order 20 in E. The calculation involved 110,800 multiplications, 212,030 simplifications, and required 32 min and 34 s of cp time. Note that not all of the coefficients can be expressed as rational integer fractions.

```
                                                  5*E** 2*SIN( 2M)
                                                  4
f - M = 2*E** 1*SIN( 1M)
                                                -11*E** 4*SIN( 2M)
      -1*E** 3*SIN( 1M)                          24
       4
                                                 17*E** 6*SIN( 2M)
       5*E** 5*SIN( 1M)                         192
      96
                                                 43*E** 8*SIN( 2M)
     107*E** 7*SIN( 1M)                        5760
    4608
                                                677*E**10*SIN( 2M)
    6217*E** 9*SIN( 1M)                       69120
  368640
                                                7237*E**12*SIN( 2M)
  565879*E**11*SIN( 1M)                      967680
 44236800
                                              201221*E**14*SIN( 2M)
 75257101*E**13*SIN( 1M)                    33177600
7431782400
                                            147485557*E**16*SIN( 2M)
2753673031*E**15*SIN( 1M)                 29262643200
32943851520
```

(Equation 20 Continues)

(Equation 20 Continued)

```
0.006918728*E**17*SIN( 1M)

0.005898236*E**19*SIN( 1M)

         13*E** 3*SIN( 3M)
         12

        -43*E** 5*SIN( 3M)
         64

         95*E** 7*SIN( 3M)
        512

       -973*E** 9*SIN( 3M)
      61440

      19503*E**11*SIN( 3M)
    2293760

    1839669*E**13*SIN( 3M)
  367001600

    5343941*E**15*SIN( 3M)
 1258291200

0.003574760*E**17*SIN( 3M)

0.003065427*E**19*SIN( 3M)

        103*E** 4*SIN( 4M)
         96

       -451*E** 6*SIN( 4M)
        480

       4123*E** 8*SIN( 4M)
      11520

      -1619*E**10*SIN( 4M)
      24192

     111929*E**12*SIN( 4M)
    7741440

    1004561*E**14*SIN( 4M)
  348364800

   19476613*E**16*SIN( 4M)
 5971968000

0.004273478*E**18*SIN( 2M)

0.003682707*E**20*SIN( 2M)

        1097*E** 5*SIN( 5M)
         960

       -5957*E** 7*SIN( 5M)
        4608

      164921*E** 9*SIN( 5M)
      258048

     -4305913*E**11*SIN( 5M)
     24772608

    130821245*E**13*SIN( 5M)
   3567255552

   -203628841*E**15*SIN( 5M)
 142690222080

0.002958697*E**17*SIN( 5M)

0.002136669*E**19*SIN( 5M)

        1223*E** 6*SIN( 6M)
         960

       -7913*E** 8*SIN( 6M)
        4480

        7751*E**10*SIN( 6M)
        7168

      -82021*E**12*SIN( 6M)
      215040

      272257*E**14*SIN( 6M)
     2867200

     -4274153*E**16*SIN( 6M)
    315392000

0.004144578*E**18*SIN( 6M)

0.001564250*E**20*SIN( 6M)
```

(Equation 20 Continued)

```
0.002722219*E**18*SIN( 4M)

0.002360083*E**20*SIN( 4M)
      47273*E** 7*SIN( 7M)
      32256

    -1773271*E** 9*SIN( 7M)
      737280

    93521303*E**11*SIN( 7M)
    53084160

  -4855133971*E**13*SIN( 7M)
   6370099200

 255385984003*E**15*SIN( 7M)
1121137459200

 -0.046520105*E**17*SIN( 7M)

  0.009689576*E**19*SIN( 7M)

      556403*E** 8*SIN( 8M)
      322560

    -4745483*E**10*SIN( 8M)
     1451520

    32431949*E**12*SIN( 8M)
    11612160

  -2743090943*E**14*SIN( 8M)
   1916006400

  0.505939442*E**16*SIN( 8M)

 -0.128583404*E**18*SIN( 8M)

  0.027382916*E**20*SIN( 8M)

     10661993*E** 9*SIN( 9M)
      5160960

   -101836961*E**11*SIN( 9M)
     22937600

   8755984901*E**13*SIN( 9M)
   2018508800

  -4531046449*E**15*SIN( 9M)
   1761607680

  1.054917381*E**17*SIN( 9M)

 -0.317514591*E**19*SIN( 9M)

      7281587*E**10*SIN(10M)
      2903040

     -76972457*E**12*SIN(10M)
      12773376

     2032943839*E**14*SIN(10M)
      306561024

  -4.467506512*E**16*SIN(10M)

   2.091520606*E**18*SIN(10M)

  -0.726142570*E**20*SIN(10M)

   62929017101*E**11*SIN(11M)
   20437401600

 -729210780611*E**13*SIN(11M)
   89181388800

  10.011153832*E**15*SIN(11M)

  -7.556348822*E**17*SIN(11M)

   3.982363731*E**19*SIN(11M)

       7218065*E**12*SIN(12M)
       1892352
```

(Equation 20 Continued)

```
-11.093435417*E**14*SIN(12M)

 14.960084231*E**16*SIN(12M)

-12.509955991*E**18*SIN(12M)

  7.336769221*E**20*SIN(12M)

60718862681977*E**13*SIN(13M)
12752938598400

-15.049693699*E**15*SIN(13M)

 22.168114178*E**17*SIN(13M)

-20.348648516*E**19*SIN(13M)

  5.981529320*E**14*SIN(14M)

-20.416704332*E**16*SIN(14M)

 32.617976855*E**18*SIN(14M)

-32.613263099*E**20*SIN(14M)

  7.556537646*E**15*SIN(15M)

-27.698312581*E**17*SIN(15M)

 47.706541979*E**19*SIN(15M)

  9.592446597*E**16*SIN(16M)

-37.578431522*E**18*SIN(16M)

 69.415410321*E**20*SIN(16M)

 12.228525304*E**17*SIN(17M)

-50.985450920*E**19*SIN(17M)

 15.647459425*E**18*SIN(18M)

-69.179694161*E**20*SIN(18M)

 20.089098638*E**19*SIN(19M)

 25.868631638*E**20*SIN(20M)
```

(20)

Equation (20) may be computed and expressed in terms of floating-point coefficients only. The result is Eq. (21).

```
 2.000000000*E** 1*SIN( 1M)
 1.250000000*E** 2*SIN( 2M)
-0.250000000*E** 3*SIN( 1M)
 1.083333333*E** 3*SIN( 3M)
-0.458333333*E** 4*SIN( 2M)
 1.072916667*E** 4*SIN( 4M)
 0.052083333*E** 5*SIN( 1M)
-0.671875000*E** 5*SIN( 3M)
 1.142708333*E** 5*SIN( 5M)
 0.088541667*E** 6*SIN( 2M)
-0.939583333*E** 6*SIN( 4M)
 1.273958333*E** 6*SIN( 6M)
 0.023220486*E** 7*SIN( 1M)
 0.185546875*E** 7*SIN( 3M)
-1.292751736*E** 7*SIN( 5M)
 1.465556796*E** 7*SIN( 7M)
 0.007465278*E** 8*SIN( 2M)
 0.357899306*E** 8*SIN( 4M)
-1.766294643*E** 8*SIN( 6M)
 1.724959697*E** 8*SIN( 8M)
 0.016864692*E** 9*SIN( 1M)
-0.015836589*E** 9*SIN( 3M)
 0.639109778*E** 9*SIN( 5M)
-2.405152723*E** 9*SIN( 7M)
 2.065893361*E** 9*SIN( 9M)
 0.009794560*E**10*SIN( 2M)
-0.066922950*E**10*SIN( 4M)
 1.081333705*E**10*SIN( 6M)
-3.269319748*E**10*SIN( 8M)
 2.508262718*E**10*SIN(10M)
 0.012792042*E**11*SIN( 1M)
 0.008502633*E**11*SIN( 3M)
-0.173817508*E**11*SIN( 5M)
 1.761755352*E**11*SIN( 7M)
-4.439739162*E**11*SIN( 9M)
 3.079110463*E**11*SIN(11M)
 0.007478712*E**12*SIN( 2M)
 0.014458421*E**12*SIN( 4M)
-0.381422061*E**12*SIN( 6M)
 2.792929911*E**12*SIN( 8M)
-6.026007298*E**12*SIN(10M)
 3.814335282*E**12*SIN(12M)
 0.010126387*E**13*SIN( 1M)
 0.005012700*E**13*SIN( 3M)
 0.008270683*E**15*SIN( 1M)
 0.004246983*E**15*SIN( 3M)
-0.001427069*E**15*SIN( 5M)
 0.227791857*E**15*SIN( 7M)
-2.572108705*E**15*SIN( 9M)
10.011153832*E**15*SIN(11M)
-15.049693699*E**15*SIN(13M)
 7.556537646*E**15*SIN(15M)
 0.005040063*E**16*SIN( 2M)
 0.003261339*E**16*SIN( 4M)
-0.013551875*E**16*SIN( 6M)
 0.505939442*E**16*SIN( 8M)
-4.467506512*E**16*SIN(10M)
14.960084231*E**16*SIN(12M)
-20.416704332*E**16*SIN(14M)
 9.592446597*E**16*SIN(16M)
 0.006918728*E**17*SIN( 1M)
 0.003574760*E**17*SIN( 3M)
 0.002958697*E**17*SIN( 5M)
-0.046520105*E**17*SIN( 7M)
 1.054917381*E**17*SIN( 9M)
-7.556348822*E**17*SIN(11M)
22.168114178*E**17*SIN(13M)
-27.698312581*E**17*SIN(15M)
12.228525304*E**17*SIN(17M)
 0.004273478*E**18*SIN( 2M)
 0.002722219*E**18*SIN( 4M)
 0.004144578*E**18*SIN( 6M)
-0.128583404*E**18*SIN( 8M)
 2.091520606*E**18*SIN(10M)
-12.509955991*E**18*SIN(12M)
32.617976855*E**18*SIN(14M)
-37.578431522*E**18*SIN(16M)
15.647459425*E**18*SIN(18M)
 0.005898236*E**19*SIN( 1M)
 0.003065427*E**19*SIN( 3M)
 0.002136669*E**19*SIN( 5M)
 0.009689576*E**19*SIN( 7M)
-0.317514591*E**19*SIN( 9M)
 3.982363731*E**19*SIN(11M)
-20.348648516*E**19*SIN(13M)
47.706541979*E**19*SIN(15M)
-50.985450920*E**19*SIN(17M)
20.089098638*E**19*SIN(19M)
```

(Equation 21 Continues)

(Equation 21 Continued)

```
 0.036672799*E**13*SIN( 5M)
-0.762175567*E**13*SIN( 7M)
 4.337848268*E**13*SIN( 9M)
-8.176714788*E**13*SIN(11M)
 4.761166394*E**13*SIN(13M)
 0.006064966*E**14*SIN( 2M)
 0.002883647*E**14*SIN( 4M)
 0.094955706*E**14*SIN( 6M)
-1.431671075*E**14*SIN( 8M)
 6.631449140*E**14*SIN(10M)
-11.093435417*E**14*SIN(12M)
 5.981529320*E**14*SIN(14M)
 0.003682707*E**20*SIN( 2M)
 0.002360083*E**20*SIN( 4M)
 0.001564250*E**20*SIN( 6M)
 0.027382916*E**20*SIN( 8M)
-0.726142570*E**20*SIN(10M)
 7.336769221*E**20*SIN(12M)
-32.613263099*E**20*SIN(14M)
 69.415410321*E**20*SIN(16M)
-69.179694161*E**20*SIN(18M)
 25.868631638*E**20*SIN(20M)
```

(21)

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.

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

- The program can be run on any computer possessing a Fortran compiler and a 48-bit word length (or greater) with no modifications. It may be adapted to any machine with a shorter word length with minor modifications.

- The program will perform about 100 multiplications per second. This includes the time required for simplification and for the removal of common factors in the numerator and denominator. About one man-years work in algebra can be reproduced in one minute of central processor time on a CDC-3800.

REFERENCES

1. P. Herget and P. Musen, "The Calculation of Literal Expressions," Astron. J. **64**, 11-20 (Feb. 1959)
2. J. E. Sammet and E. R. Bond, "Introduction to FORMAC," IEEE Trans. Electron. Comput. **EC-13**, 386-394 (Aug. 1964)
3. M. Manove, S. Bloom, and G. Engelman, "Rational Functions in Mathlab," MTP-35, Mitre Corporation, Bedford, Mass., Aug. 1966
4. A. C. Hearn, "REDUCE User's Manual," Stanford Computational Center, Stanford, Calif., 1967
5. C. Christensen and M. Karr, "I.A.M., A System for Interactive Algebraic Manipulation," CA-7010-2911, Appl Data Res, Wakefield, Mass. Oct. 1970

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

7. W. H. Jefferys, "A Fortran-Based List Processor for Poisson Series," Celestial Mech. 2 (No. 4), 474-480 Nov. 1970

8. R. Broucke and K. Garthwarte, "A Programming System for Analytical Series Expansions on a Computer," Celestial Mech. (No. 2), 271-284 Oct. 1969

9. J. McCarthy, et al., *Lisp 1.5 Programmer's Manual*, 2nd ed., MIT Press, Cambridge, 1965

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

Appendix A

PROGRAM LISTING

```
      SUBROUTINE START
  100 FORMAT (1H1)
  101 FORMAT (///)
  102 FORMAT (20X,22H START OF EXECUTION AT,I5,6H HOURS,I3,
     * 8H MINUTES,I3,8H SECONDS)
      COMMON/1/NEXT(6500)/2/N(6500)/3/M(6500)
      COMMON/4/IEX(6500)/5/ITRIG(6500)/6/IRG(6500)
      COMMON/7/LFACTL(30),NUTIL(30),MSTART(125),MAX(125)
      COMMON/8/KFACT(10),KMAX,MORDER,LORDER,LZERO,INDEX
      COMMON/9/NOFACT,NORDER,NOSIMP,NPRINT,LPRINT/16/NEXACT
      COMMON/12/LPMAX/13/LPRIME(20)/14/EPSILON/15/NOFIX
      TYPE LOGICAL NOFACT,NORDER,NOSIMP,NPRINT,LPRINT,NOFIX,NEXACT
      DIMENSION MPRIME(20)
      DATA (MPRIME=2,3,5,7,11,13,17,19,23,
     *   29,31,37,41,43,47,53,57,59,61,67)
      CALL TIME(MH,MM,MS)
      PRINT 100
      PRINT 102,MH,MM,MS
      PRINT 101
      NOFACT=.FALSE.
      NORDER=.TRUE.
      NOSIMP=.FALSE.
      NPRINT=.FALSE.
      LPRINT=.TRUE.
      NOFIX=.TRUE.
      NEXACT=.FALSE.
      LPMAX=4B
      EPSILON=1.0E-12
      LORDER=-37B
      MORDER=+37B
      LZERO=1
      INDEX=2040404040404040B
      DO 5 K=1,20,1
    5 LPRIME(K)=MPRIME(K)
      DO 6 K=1,10,1
    6 KFACT(K)=0
      KFACT(8)=1
      KFACT(10)=MORDER
      LFACTL(1)=1
      DO 7 K=2,16,1
      LFACTL(K)=K*LFACTL(K-1)
    7 CONTINUE
      DO 8 K=1,30,1
      NUTIL(K)=0
    8 CONTINUE
      DO 9 K=1,125,1
      MSTART(K)=0
      MAX(K)=0
    9 CONTINUE
      DO 10 K=1,6500,1
      NEXT(K)=K+1
      N(K)=0
      M(K)=ITRIG(K)=+1
      IEX(K)=IRG(K)=INDEX
   10 CONTINUE
      NEXT(6500)=0
      RETURN
      END
```

```
      SUBROUTINE SWITCH(ISA,ISB)
  100 FORMAT (40H **** MSA AND/OR MSB ZERO IN SWITCH ****)
      COMMON/7/LFACTL(30),NUTIL(30),MSTART(125),MAX(125)
      COMMON/8/KFACT(10),KMAX,MORDER,LORDER,LZERO,INDEX
      COMMON/9/NOFACT,NORDER,NOSIMP,NPRINT,LPRINT
      TYPE LOGICAL NOFACT,NORDER,NOSIMP,NPRINT,LPRINT
      MSISA=MSTART(ISA)
      MXISA=MAX(ISA)
      MSISB=MSTART(ISB)
      MXISB=MAX(ISB)
      IF (MSISA.NE.0.AND.MSISB.NE.0) GO TO 10
      IF (NPRINT) GO TO 10
      PRINT 100
   10 CONTINUE
      MSTART(ISB)=MSISA
      MAX(ISB)=MXISA
      MSTART(ISA)=MSISB
      MAX(ISA)=MXISB
      RETURN
      END
```

```
      SUBROUTINE FACTOR(NN,MM)
      COMMON/12/LPMAX/13/LPRIME(20)
      IF (NN.EQ.1.OR.MM.EQ.1) RETURN
      IF (NN.EQ.0.OR.MM.EQ.0) RETURN
      DO 20 K=1,LPMAX,1
      KF=LPRIME(K)
   10 N=NN/KF
      IF (N*KF.NE.NN) GO TO 20
      M=MM/KF
      IF (M*KF.NE.MM) GO TO 20
      NN=N
      MM=M
      GO TO 10
   20 CONTINUE
      RETURN
      END
```

```
SUBROUTINE INDICES(IZ,II)
DIMENSION II(8)
IA=IZ
II(8)=MOD(IA,100B)-40B
IA=IA/100B
II(7)=MOD(IA,100B)-40B
IA=IA/100B
II(6)=MOD(IA,100B)-40B
IA=IA/100B
II(5)=MOD(IA,100B)-40B
IA=IA/100B
II(4)=MOD(IA,100B)-40B
IA=IA/100B
II(3)=MOD(IA,100B)-40B
IA=IA/100B
II(2)=MOD(IA,100B)-40B
IA=IA/100B
II(1)=MOD(IA,100B)-20B
RETURN
END
```

```
SUBROUTINE IMPACT(III,IA)
DIMENSION III(8)
IP=(III(8)+40B)
IO=(III(7)+40B)*100B
IN=(III(6)+40B)*10000B
IM=(III(5)+40B)*1000000B
IL=(III(4)+40B)*100000000B
IK=(III(3)+40B)*10000000000B
IJ=(III(2)+40B)*1000000000000B
II=(III(1)+20B)*100000000000000B
IA=II+IJ+IK+IL+IM+IN+IO+IP
RETURN
END
```

```
      SUBROUTINE REVERSE(ISA)
      COMMON/1/NEXT(6500)/7/IY(30),IZ(30),MSTART(125),MAX(125)
      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)
   50 NEXT(KX)=LAST
      IF (NKX.LE.0) GO TO 70
      LAST=KX
      KX=NKX
      NKX=NEXT(NKX)
      GO TO 50
   70 CONTINUE
      RETURN
      END
```

```
      SUBROUTINE ZERO(ISA)
      COMMON/1/NEXT(6500)/2/N(6500)/3/M(6500)
      COMMON/4/IEX(6500)/5/ITRIG(6500)/6/IRG(6500)
      COMMON/7/LFACTL(30),NUTIL(30),MSTART(125),MAX(125)
      COMMON/8/KFACT(10),KMAX,MORDER,LORDER,LZERO,INDEX
      MSA=MSTART(ISA)
      IF (MSA.EQ.0) RETURN
      K=MSA
   10 CONTINUE
      N(K)=0
      M(K)=ITRIG(K)=+1
      IEX(K)=IRG(K)=INDEX
      NUTIL(1)=NUTIL(1)+1
      KN=NEXT(K)
      IF (KN.EQ.0) GO TO 20
      K=KN
      GO TO 10
   20 CONTINUE
      NEXT(K)=LZERO
      LZERO=MSA
      MSTART(ISA)=0
      MAX(ISA)=0
      RETURN
      END
```

```
      SUBROUTINE TRANSF(INIT,IFINAL)
  100 FORMAT (37H **** STORAGE OVERFLOW IN TRANSF ****)
  101 FORMAT (38H **** MSTART(INIT) ZERO IN TRANSF ****)
      COMMON/1/NEXT(6500)/2/N(6500)/3/M(6500)
      COMMON/4/IEX(6500)/5/ITRIG(6500)/6/IRG(6500)
      COMMON/7/LFACTL(30),NUTIL(30),MSTART(125),MAX(125)
      COMMON/8/KFACT(10),KMAX,MORDER,LORDER,LZERO,INDEX
      COMMON/9/NOFACT,NORDER,NOSIMP,NPRINT,LPRINT
      TYPE LOGICAL NOFACT,NORDER,NOSIMP,NPRINT,LPRINT
      IF (INIT.EQ.IFINAL) RETURN
      IF (LZERO.GT.0) GO TO 5
      PRINT 100
      CALL QUIT
      STOP
    5 CONTINUE
      CALL ZERO(IFINAL)
      MSTART(IFINAL)=LZERO
      KF=LZERO
      KI=MSTART(INIT)
      IF (KI.GT.0) GO TO 10
      IF (NPRINT) GO TO 8
      PRINT 101
    8 CONTINUE
      MSTART(IFINAL)=0
      MAX(INIT)=MAX(IFINAL)=0
      RETURN
   10 CONTINUE
      N(KF)=N(KI)
      M(KF)=M(KI)
      IEX(KF)=IEX(KI)
      ITRIG(KF)=ITRIG(KI)
      IRG(KF)=IRG(KI)
      NUTIL(2)=NUTIL(2)+1
      KNI=NEXT(KI)
      IF (KNI.EQ.0) GO TO 20
      KI=KNI
      KNF=NEXT(KF)
      IF (KNF.GT.0) GO TO 15
      PRINT 100
      CALL QUIT
      STOP
   15 CONTINUE
      KF=KNF
      GO TO 10
   20 CONTINUE
      LZERO=NEXT(KF)
      NEXT(KF)=0
      MAX(INIT)=KI
      MAX(IFINAL)=KF
      RETURN
      END
```

```
      SUBROUTINE PACK(ISA)
  100 FORMAT (27H **** MSA ZERO IN PACK ****)
      COMMON/1/NEXT(6500)/2/N(6500)/3/M(6500)
      COMMON/4/IEX(6500)/5/ITRIG(6500)/6/IRG(6500)
      COMMON/7/LFACTL(30),NUTIL(30),MSTART(125),MAX(125)
      COMMON/8/KFACT(10),KMAX,MORDER,LORDER,LZERO,INDEX
      COMMON/9/NOFACT,NORDER,NOSIMP,NPRINT,LPRINT
      COMMON/10/NTOTAL
      TYPE LOGICAL NOFACT,NORDER,NOSIMP,NPRINT,LPRINT
      K=MSA=MSTART(ISA)
      NTOTAL=LNZP=0
      IF (MSA.GT.0) GO TO 5
      IF (NPRINT) GO TO 2
      PRINT 100
    2 CONTINUE
      MAX(ISA)=0
      RETURN
    5 CONTINUE
      MSA=0
   10 CONTINUE
      NADDR=NEXT(K)
      IF (N(K).EQ.0) GO TO 20
      NTOTAL=NTOTAL+1
      LNZP=K
      IF (MSA.EQ.0) MSA=K
      IF (NADDR.EQ.0) GO TO 30
      K=NADDR
      GO TO 10
   20 CONTINUE
      IF (LNZP.GT.0) NEXT(LNZP)=NEXT(K)
      KN=NEXT(K)
      NEXT(K)=LZERO
      LZERO=K
      M(K)=ITRIG(K)=1
      IEX(K)=IRG(K)=INDEX
      NUTIL(4)=NUTIL(4)+1
      IF (KN.EQ.0) GO TO 30
      K=KN
      GO TO 10
   30 CONTINUE
      MSTART(ISA)=MSA
      MAX(ISA)=LNZP
      IF (LNZP.GT.0) NEXT(LNZP)=0
      RETURN
      END
```

```
      SUBROUTINE SIMPLIFY(ISA)
 1001 FORMAT (9H SIMPLIFY,I4,19H NUMPAR AND KOUNT =,2I5)
 1002 FORMAT (24H TOTAL NUMBER OF TERMS =,I5,8H  TIME =,I5,4H SEC)
 1003 FORMAT (29H POINTER NEGATIVE IN SIMPLIFY)
      COMMON/1/NEXT(6500)/2/N(6500)/3/M(6500)
      COMMON/4/IEX(6500)/5/ITRIG(6500)/6/IRG(6500)
      COMMON/7/LFACTL(30),NUTIL(30),MSTART(125),MAX(125)
      COMMON/8/KFACT(10),KMAX,MORDER,LORDER,LZERO,INDEX
      COMMON/9/NOFACT,NORDER,NOSIMP,NPRINT,LPRINT
      COMMON/10/NTOTAL/11/LMAP(256)
      COMMON/12/LPMAX/13/LPRIME(20)/14/EPSILON/15/NOFIX
      TYPE LOGICAL NOFACT,NORDER,NOSIMP,NPRINT,LPRINT,NOFIX
      TYPE LOGICAL KSCAN,PNPRINT
      IF (NOSIMP) RETURN
      CALL TIME(MH,MM,MS)
      NA=3600*MH+60*MM+MS
      MSA=MSTART(ISA)
      IF (MSA.EQ.0) RETURN
      PNPRINT=NPRINT
      NPRINT=.TRUE.
      CALL PACK(ISA)
      CALL REVERSE(ISA)
      MSA=MSTART(ISA)
      ISB=102 $ CALL ZERO(ISB)
      MSTART(ISB)=MAX(ISB)=MSA
      MSTART(ISA)=NEXT(MSA)
      NEXT(MSA)=0
      KOUNT=1 $ KSCAN=.TRUE.
      NUMPAR=2
      DO 80 JX=1,256,1
   80 LMAP(JX)=0
   90 CONTINUE
      I=MSA=MSTART(ISA)
      IF (MSA.EQ.0) GO TO 300
      MSTART(ISA)=NEXT(MSA)
      NEXT(MSA)=0
      IF (NUMPAR.GE.256) GO TO 100
      IPAR=16*NUMPAR
      IF (KOUNT.LE.IPAR) GO TO 100
      KSCAN=.FALSE.
      NUMPAR=2*NUMPAR
      KS=MSTART(ISB)
      DO 98 MX=1,NUMPAR,1
      LMAP(MX)=KS
      DO 98 MY=1,8,1
      KS=NEXT(KS)
      IF (KS.LE.0) GO TO 250
   98 CONTINUE
  100 L=MSB=MSTART(ISB) $ LAST=0
      IF (KSCAN) GO TO 125
      MX=NUMPAR/2 $ LX=LMAP(MX)
      MDX=NUMPAR/4
  102 IF (IRG(LX)-IRG(I)) 105,103,107
  103 IF (ITRIG(LX)-ITRIG(I)) 105,104,107
  104 IF (IEX(LX)-IEX(I)) 105,106,107
  105 IF (MDX.EQ.0) MDX=1
      MX=MX-MDX $ MDX=MDX/2
      IF (MX.LE.0) GO TO 118
```

```
    LX=LMAP(MX) $ GO TO 102
106 L=LX $ GO TO 160
107 IF (MDX.EQ.0) GO TO 120
    MX=MX+MDX $ LX=LMAP(MX)
    MDX=MDX/2 $ GO TO 102
118 LX=MSB
120 L=LX
125 IF (IRG(L)-IRG(I)) 130,140,170
140 IF (ITRIG(L)-ITRIG(I)) 130,150,170
150 IF (IEX(L)-IEX(I)) 130,160,170
130 IF (LAST.NE.0) GO TO 131
    MSTART(ISB)=I
    NEXT(I)=L
    KOUNT=KOUNT+1
    GO TO 90
131 NEXT(LAST)=I
    NEXT(I)=L
    KOUNT=KOUNT+1
    GO TO 90
160 CALL RFACTOR(N(I),M(I),N(L),M(L),1,1,N(L),M(L),0)
    N(I)=0
    M(I)=ITRIG(I)=1
    IEX(I)=IRG(I)=INDEX
    NEXT(I)=LZERO
    LZERO=I
    NUTIL(5)=NUTIL(5)+1
    GO TO 90
170 LN=NEXT(L)
    IF (LN) 250,171,172
171 NEXT(L)=MAX(ISB)=I
    NEXT(I)=0 $ KOUNT=KOUNT+1
    GO TO 90
172 LAST=L $ L=LN
    NUTIL(15)=NUTIL(15)+1
    GO TO 125
250 PRINT 1003 $ CALL QUIT $ STOP
300 CALL SWITCH(ISA,ISB)
    MSTART(ISB)=MAX(ISB)=0
    CALL PACK(ISA)
    NPRINT=PNPRINT
    IF (NPRINT) RETURN
    CALL TIME(MH,MM,MS)
    NB=3600*MH+60*MM+MS
    NT=NB-NA
    PRINT 1001,ISA,NUMPAR,KOUNT
    PRINT 1002,NTOTAL,NT
    RETURN
    END
```

```
      SUBROUTINE RFACTOR(NAA,MAA,NBB,MBB,KN,KM,N,M,PURPOSE)
 1000 FORMAT (50X,28H LOOSING ACCURACY IN RFACTOR)
      COMMON/12/LPMAX/13/LPRIME(20)/14/EPSILON/15/NOFIX
      COMMON/9/NOFACT,NORDER,NOSIMP,NPRINT,LPRINT/16/NEXACT
      TYPE DOUBLE AN,AM,BN,BM,CN,CM,FN,FM
      TYPE DOUBLE ATOL,BTOL,CTOL,SN,SM
      TYPE DOUBLE DMAX,EXFACT,COEFF
      TYPE LOGICAL NORDER,NOSIMP,NPRINT,LPRINT
      TYPE LOGICAL NOFACT,NOFIX,NEXACT
      TYPE INTEGER PURPOSE
      DATA (MAX=3777777777777777B)
      DATA (EXFACT=1.0D+14)
      DATA (MLIM=1000000)
      DATA (ATOL=0.01D0)
      DATA (BTOL=0.0001D0)
      DATA (CTOL=0.000001D0)
      NA=NAA $ MA=MAA
      NB=NBB $ MB=MBB
      IF  (NOFIX)  GO TO 10
      IF (PURPOSE) 4,4,5
    4 N=NA*MB+MA*NB
      M=MA*MB
      N=KN*N
      M=KM*M
      GO TO 400
    5 N=NA*NB
      M=MA*MB
      N=KN*N
      M=KM*M
      GO TO 400
   10 DMAX=MAX
      AN=NA $ AM=MA
      BN=NB $ BM=MB
      IF (PURPOSE) 15,15,20
   15 CN=AN*BM+AM*BN
      CM=AM*BM
      CN=KN*CN
      CM=KM*CM
      COEFF=CN/CM
      GO TO 30
   20 CN=AN*BN
      CM=AM*BM
      CN=KN*CN
      CM=KM*CM
      COEFF=CN/CM
      GO TO 30
   30 ABSAB=DABS(COEFF)
      IF (ABSAB.GT.EPSILON) GO TO 50
      N=0 $ M=1
      RETURN
   50 IF (NOFACT) GO TO 200
      IF (IABS(MA).LT.MLIM) GO TO 60
      SA=ABS(AM)
      SA=ALOG10(SA)
      SA=SA+CTOL
      SA=AMOD(SA,1.0)
      IF (SA.LT.BTOL) GO TO 200
   60 IF (IABS(MB).LT.MLIM) GO TO 70
```

```
      SB=ABS(BM)
      SB=ALOG10(SB)
      SB=SB+CTOL
      SB=AMOD(SB,1.0)
      IF (SB.LT.BTOL) GO TO 200
   70 IF (DABS(CN).GT.DMAX) GO TO 100
      IF (DABS(CM).GT.DMAX) GO TO 100
      N=IDINT(CN+DSIGN(ATOL,CN))
      M=IDINT(CM+DSIGN(ATOL,CM))
      GO TO 400
  100 DO 120 K=1,LPMAX,1
      KF=LPRIME(K)
  110 FN=CN/KF
      SN=DABS(FN)+BTOL
      SN=DMOD(SN,1.0D0)
      IF (SN.GT.ATOL) GO TO 120
      FM=CM/KF
      SM=DABS(FM)+BTOL
      SM=DMOD(SM,1.0D0)
      IF (SM.GT.ATOL) GO TO 120
      CN=FN
      CM=FM
      GO TO 110
  120 CONTINUE
      IF (DABS(CN).GT.DMAX) GO TO 200
      IF (DABS(CM).GT.DMAX) GO TO 200
      GO TO 300
  200 CN=COEFF*EXFACT
      CM=EXFACT
      NEXACT=.TRUE.
  210 IF (DABS(CN).GT.DMAX) GO TO 220
      IF (DABS(CM).GT.DMAX) GO TO 220
      GO TO 300
  220 CN=CN/10
      CM=CM/10
      IF (DABS(CN).LT.1.0D3.OR.DABS(CM).LT.1.0D3) GO TO 415
      GO TO 210
  300 N=IDINT(CN+DSIGN(0.5D0,CN))
      M=IDINT(CM+DSIGN(0.5D0,CM))
      RETURN
  400 IF (N.EQ.1.OR.M.EQ.1) RETURN
      IF (N.EQ.0.OR.M.EQ.0) RETURN
      DO 420 K=1,LPMAX,1
      KF=LPRIME(K)
  410 NN=N/KF
      IF (NN*KF.NE.N) GO TO 420
      MM=M/KF
      IF (MM*KF.NE.M) GO TO 420
      N=NN
      M=MM
      GO TO 410
  415 PRINT 1000 $ CALL QUIT $ STOP
  420 CONTINUE
      RETURN
      END
```

```
      SUBROUTINE CUT(ISA,ISB)
  100 FORMAT (30H RETAIN ONLY PERIODIC TERMS IN,2I3)
  101 FORMAT (26H **** MSA ZERO IN CUT ****)
  102 FORMAT (34H **** STORAGE OVERFLOW IN CUT ****)
      COMMON/1/NEXT(6500)/2/N(6500)/3/M(6500)
      COMMON/4/IEX(6500)/5/ITRIG(6500)/6/IRG(6500)
      COMMON/7/LFACTL(30),NUTIL(30),MSTART(125),MAX(125)
      COMMON/8/KFACT(10),KMAX,MORDER,LORDER,LZERO,INDEX
      COMMON/9/NOFACT,NORDER,NOSIMP,NPRINT,LPRINT
      TYPE LOGICAL NOFACT,NORDER,NOSIMP,NPRINT,LPRINT
      DIMENSION IA(8)
      IF (NPRINT) GO TO 2
      PRINT 100,ISA,ISB
    2 CONTINUE
      K=MSA=MSTART(ISA)
      IF (MSA.GT.0) GO TO 5
      IF (NPRINT) RETURN
      PRINT 101 $ RETURN
    5 CONTINUE
      IF (ISA.EQ.ISB) GO TO 50
      CALL ZERO(ISB)
      J=MSB=MSTART(ISB)=LZERO
   10 CONTINUE
      JN=J
      IF (ITRIG(K).EQ.0) GO TO 20
      IF (IRG(K).EQ.INDEX) GO TO 20
      N(J)=N(K)
      M(J)=M(K)
      IEX(J)=IEX(K)
      ITRIG(J)=ITRIG(K)
      IRG(J)=IRG(K)
      JN=NEXT(J)
   20 CONTINUE
      KN=NEXT(K)
      IF (JN.GT.0) GO TO 30
      PRINT 102
      CALL QUIT
      STOP
   30 CONTINUE
      IF (KN.EQ.0) GO TO 40
      J=JN $ K=KN
      GO TO 10
   40 CONTINUE
      LZERO=NEXT(J)
      NEXT(J)=0
      MAX(ISB)=J
      CALL PACK(ISB)
      RETURN
   50 CONTINUE
      IF (ITRIG(K).EQ.0) N(K)=0
      IF (IRG(K).EQ.INDEX) N(K)=0
      KN=NEXT(K)
      IF (KN.EQ.0) GO TO 60
      K=KN $ GO TO 50
   60 CONTINUE
      CALL PACK(ISA)
      RETURN
      END
```

```
      SUBROUTINE SECULAR(ISA,ISB)
  100 FORMAT (30H RETAIN ONLY CONSTANT TERMS IN,2I3)
  101 FORMAT (30H **** MSA ZERO IN SECULAR ****)
  102 FORMAT (38H **** STORAGE OVERFLOW IN SECULAR ****)
      COMMON/1/NEXT(6500)/2/N(6500)/3/M(6500)
      COMMON/4/IEX(6500)/5/ITRIG(6500)/6/IRG(6500)
      COMMON/7/LFACTL(30),NUTIL(30),MSTART(125),MAX(125)
      COMMON/8/KFACT(10),KMAX,MORDER,LORDER,LZERO,INDEX
      COMMON/9/NOFACT,NORDER,NOSIMP,NPRINT,LPRINT
      TYPE LOGICAL NOFACT,NORDER,NOSIMP,NPRINT,LPRINT
      DIMENSION IA(8)
      IF (NPRINT) GO TO 2
      PRINT 100,ISA,ISB
    2 CONTINUE
      K=MSA=MSTART(ISA)
      IF (MSA.GT.0) GO TO 5
      IF (NPRINT) RETURN
      PRINT 101 $ RETURN
    5 CONTINUE
      IF (ISA.EQ.ISB) GO TO 50
      CALL ZERO(ISB)
      J=MSB=MSTART(ISB)=LZERO
   10 CONTINUE
      JN=J
      IF (IRG(K).NE.INDEX) GO TO 20
      N(J)=N(K)
      M(J)=M(K)
      IEX(J)=IEX(K)
      ITRIG(J)=ITRIG(K)
      IRG(J)=IRG(K)
      JN=NEXT(J)
   20 CONTINUE
      KN=NEXT(K)
      IF (JN.GT.0) GO TO 30
      PRINT 102
      CALL QUIT
      STOP
   30 CONTINUE
      IF (KN.EQ.0) GO TO 40
      J=JN $ K=KN
      GO TO 10
   40 CONTINUE
      LZERO=NEXT(J)
      NEXT(J)=0
      MAX(ISB)=J
      CALL PACK(ISB)
      RETURN
   50 CONTINUE
      IF (IRG(K).NE.INDEX) N(K)=0
      KN=NEXT(K)
      IF (KN.EQ.0) GO TO 60
      K=KN $ GO TO 50
   60 CONTINUE
      CALL PACK(ISA)
      RETURN
      END
```

```
      SUBROUTINE TRUNCATE(ISA,ISB,IVARIB,NUM)
  100 FORMAT (16H RETAIN TERMS IN,I3,9H TO ORDER,I3,4H WRT,I3)
  101 FORMAT (31H **** MSA ZERO IN TRUNCATE ****)
  102 FORMAT (39H **** STORAGE OVERFLOW IN TRUNCATE ****)
      COMMON/1/NEXT(6500)/2/N(6500)/3/M(6500)
      COMMON/4/IEX(6500)/5/ITRIG(6500)/6/IRG(6500)
      COMMON/7/LFACTL(30),NUTIL(30),MSTART(125),MAX(125)
      COMMON/8/KFACT(10),KMAX,MORDER,LORDER,LZERO,INDEX
      COMMON/9/NOFACT,NORDER,NOSIMP,NPRINT,LPRINT
      TYPE LOGICAL NOFACT,NORDER,NOSIMP,NPRINT,LPRINT
      DIMENSION IA(8)
      IF (NPRINT) GO TO 2
      PRINT 100,ISA,NUM,IVARIB,ISB
    2 CONTINUE
      K=MSA=MSTART(ISA)
      IF (MSA.GT.0) GO TO 5
      IF (NPRINT) RETURN
      PRINT 101 $ RETURN
    5 CONTINUE
      IF (ISA.EQ.ISB) GO TO 50
      CALL ZERO(ISB)
      J=MSB=MSTART(ISB)=LZERO
   10 CONTINUE
      JN=J
      CALL INDICES(IEX(K),IA)
      IF (IA(IVARIB).GT.NUM) GO TO 20
      N(J)=N(K)
      M(J)=M(K)
      IEX(J)=IEX(K)
      ITRIG(J)=ITRIG(K)
      IRG(J)=IRG(K)
      JN=NEXT(J)
   20 CONTINUE
      KN=NEXT(K)
      IF (JN.GT.0) GO TO 30
      PRINT 102
      CALL QUIT
      STOP
   30 CONTINUE
      IF (KN.EQ.0) GO TO 40
      J=JN $ K=KN
      GO TO 10
   40 CONTINUE
      LZERO=NEXT(J)
      NEXT(J)=0
      MAX(ISB)=J
      CALL PACK(ISB)
      RETURN
   50 CONTINUE
      CALL INDICES(IEX(K),IA)
      IF (IA(IVARIB).GT.NUM) N(K)=0
      KN=NEXT(K)
      IF (KN.EQ.0) GO TO 60
      K=KN $ GO TO 50
   60 CONTINUE
      CALL PACK(ISA)
      RETURN
      END
```

```
      SUBROUTINE SELECT(ISA,ISB,IVARIB,NUM)
  100 FORMAT (3H IN,I3,13H SELECT ORDER,I3,4H WRT,I3,9H STORE IN,I3)
  101 FORMAT (29H **** MSA ZERO IN SELECT ****)
  102 FORMAT (37H **** STORAGE OVERFLOW IN SELECT ****)
      COMMON/1/NEXT(6500)/2/N(6500)/3/M(6500)
      COMMON/4/IEX(6500)/5/ITRIG(6500)/6/IRG(6500)
      COMMON/7/LFACTL(30),NUTIL(30),MSTART(125),MAX(125)
      COMMON/8/KFACT(10),KMAX,MORDER,LORDER,LZERO,INDEX
      COMMON/9/NOFACT,NORDER,NOSIMP,NPRINT,LPRINT
      TYPE LOGICAL NOFACT,NORDER,NOSIMP,NPRINT,LPRINT
      DIMENSION IA(8)
      IF (NPRINT) GO TO 2
      PRINT 100,ISA,NUM,IVARIB,ISB
    2 CONTINUE
      K=MSA=MSTART(ISA)
      IF (MSA.GT.0) GO TO 5
      IF (NPRINT) RETURN
      PRINT 101 $ RETURN
    5 CONTINUE
      IF (ISA.EQ.ISB) GO TO 50
      CALL ZERO(ISB)
      J=MSB=MSTART(ISB)=LZERO
   10 CONTINUE
      JN=J
      CALL INDICES(IEX(K),IA)
      IF (IA(IVARIB).NE.NUM) GO TO 20
      N(J)=N(K)
      M(J)=M(K)
      IEX(J)=IEX(K)
      ITRIG(J)=ITRIG(K)
      IRG(J)=IRG(K)
      JN=NEXT(J)
   20 CONTINUE
      KN=NEXT(K)
      IF (JN.GT.0) GO TO 30
      PRINT 102
      CALL QUIT
      STOP
   30 CONTINUE
      IF (KN.EQ.0) GO TO 40
      J=JN $ K=KN
      GO TO 10
   40 CONTINUE
      LZERO=NEXT(J)
      NEXT(J)=0
      MAX(ISB)=J
      CALL PACK(ISB)
      RETURN
   50 CONTINUE
      CALL INDICES(IEX(K),IA)
      IF (IA(IVARIB).NE.NUM) N(K)=0
      KN=NEXT(K)
      IF (KN.EQ.0) GO TO 60
      K=KN $ GO TO 50
   60 CONTINUE
      CALL PACK(ISA)
      RETURN
      END
```

```
      SUBROUTINE CHOOSE(ISA,ISB,IVARIB,NUM)
  100 FORMAT (16H CHOOSE TERMS IN,I3,15H OF PERIODICITY,I3,4H WRT,I3)
  101 FORMAT (29H **** MSA ZERO IN CHOOSE ****)
  102 FORMAT (37H **** STORAGE OVERFLOW IN CHOOSE ****)
  103 FORMAT (17H  STORE RESULT IN,I4)
      COMMON/1/NEXT(6500)/2/N(6500)/3/M(6500)
      COMMON/4/IEX(6500)/5/ITRIG(6500)/6/IRG(6500)
      COMMON/7/LFACTL(30),NUTIL(30),MSTART(125),MAX(125)
      COMMON/8/KFACT(10),KMAX,MORDER,LORDER,LZERO,INDEX
      COMMON/9/NOFACT,NORDER,NOSIMP,NPRINT,LPRINT
      TYPE LOGICAL NOFACT,NORDER,NOSIMP,NPRINT,LPRINT
      DIMENSION IA(8)
      IF (NPRINT) GO TO 2
      PRINT 100,ISA,NUM,IVARIB
      PRINT 103,ISB
    2 CONTINUE
      K=MSA=MSTART(ISA)
      IF (MSA.GT.0) GO TO 5
      IF (NPRINT) RETURN
      PRINT 101 $ RETURN
    5 CONTINUE
      IF (ISA.EQ.ISB) GO TO 50
      CALL ZERO(ISB)
      J=MSB=MSTART(ISB)=LZERO
   10 CONTINUE
      JN=J
      CALL INDICES(IRG(K),IA)
      IF (IA(IVARIB).NE.NUM) GO TO 20
      N(J)=N(K)
      M(J)=M(K)
      IEX(J)=IEX(K)
      ITRIG(J)=ITRIG(K)
      IRG(J)=IRG(K)
      JN=NEXT(J)
   20 CONTINUE
      KN=NEXT(K)
      IF (JN.GT.0) GO TO 30
      PRINT 102
      CALL QUIT
      STOP
   30 CONTINUE
      IF (KN.EQ.0) GO TO 40
      J=JN $ K=KN
      GO TO 10
   40 CONTINUE
      LZERO=NEXT(J)
      NEXT(J)=0
      MAX(ISB)=J
      CALL PACK(ISB)
      RETURN
   50 CONTINUE
      CALL INDICES(IRG(K),IA)
      IF (IA(IVARIB).NE.NUM) N(K)=0
      KN=NEXT(K)
      IF (KN.EQ.0) GO TO 60
      K=KN $ GO TO 50
   60 CONTINUE
      CALL PACK(ISA)
      RETURN
      END
```

```
      SUBROUTINE MULCONS(ISA,ICONST,JCONST)
  100 FORMAT (9H MULTIPLY,I3,3H BY,I6,1H/,I6)
  101 FORMAT (30H **** MSA ZERO IN MULCONS ****)
      COMMON/1/NEXT(6500)/2/N(6500)/3/M(6500)
      COMMON/4/IEX(6500)/5/ITRIG(6500)/6/IRG(6500)
      COMMON/7/LFACTL(30),NUTIL(30),MSTART(125),MAX(125)
      COMMON/8/KFACT(10),KMAX,MORDER,LORDER,LZERO,INDEX
      COMMON/9/NOFACT,NORDER,NOSIMP,NPRINT,LPRINT
      COMMON/12/LPMAX/13/LPRIME(20)/14/EPSILON/15/NOFIX
      TYPE LOGICAL NOFACT,NORDER,NOSIMP,NPRINT,LPRINT,NOFIX
      ICONS=ICONST $ JCONS=JCONST
      CALL FACTOR(ICONS,JCONS)
      IF (NPRINT) GO TO 10
      PRINT 100,ISA,ICONS,JCONS
   10 CONTINUE
      IF (ICONS.EQ.0) CALL ZERO(ISA)
      K=MSA=MSTART(ISA)
      IF (MSA.GT.0) GO TO 50
      IF (NPRINT) RETURN
      PRINT 101
      RETURN
   50 CONTINUE
      CALL RFACTOR(N(K),M(K),1,1,ICONS,JCONS,N(K),M(K),1)
      NK=NEXT(K)
      IF (NK.EQ.0) GO TO 60
      K=NK
      GO TO 50
   60 CONTINUE
      RETURN
      END
```

```
      SUBROUTINE ADD(ISA,ISB,ISC)
  100 FORMAT (4H ADD,I3,3H TO,I3,13H AND STORE IN,I3)
  101 FORMAT (34H **** STORAGE OVERFLOW IN ADD ****)
  102 FORMAT (37H **** MSA AND/OR MSB ZERO IN ADD ****)
  103 FORMAT (12H TIME TO ADD,I4,3H TO,I4,2H =,I4,4H SEC)
      COMMON/1/NEXT(6500)/2/N(6500)/3/M(6500)
      COMMON/4/IEX(6500)/5/ITRIG(6500)/6/IRG(6500)
      COMMON/7/LFACTL(30),NUTIL(30),MSTART(125),MAX(125)
      COMMON/8/KFACT(10),KMAX,MORDER,LORDER,LZERO,INDEX
      COMMON/9/NOFACT,NORDER,NOSIMP,NPRINT,LPRINT
      TYPE LOGICAL NOFACT,NORDER,NOSIMP,NPRINT,LPRINT,PNPRINT
      PNPRINT=NPRINT
      CALL TIME(MH,MM,MS)
      NA=3600*MH+60*MM+MS
      IF (NPRINT) GO TO 2
      PRINT 100,ISA,ISB,ISC
    2 CONTINUE
      CALL TRANSF(ISA,ISC)
      MSB=MSTART(ISB)
      MSC=MSTART(ISC)
      IF (MSB.GT.0.AND.MSC.GT.0) GO TO 5
      IF (NPRINT) GO TO 4
      PRINT 102
    4 CONTINUE
      IF (MSB.EQ.0) RETURN
      CALL TRANSF(ISB,ISC)
      RETURN
    5 IF (LPRINT) NPRINT=.TRUE.
      K=MSC
   10 KN=NEXT(K)
      IF (KN.EQ.0) GO TO 20
      K=KN
      GO TO 10
   20 NEXT(K)=LZERO
      L=MSB
      K=LZERO
   25 N(K)=N(L)
      M(K)=M(L)
      IEX(K)=IEX(L)
      ITRIG(K)=ITRIG(L)
      IRG(K)=IRG(L)
      NUTIL(6)=NUTIL(6)+1
      LN=NEXT(L)
      IF (LN.EQ.0) GO TO 30
      L=LN
      KN=NEXT(K)
      IF (KN.GT.0) GO TO 27
      PRINT 101
      CALL QUIT
      STOP
   27 K=KN
      GO TO 25
   30 LZERO=NEXT(K)
      NEXT(K)=0
      CALL SIMPLIFY(ISC)
      IF (LPRINT) NPRINT=PNPRINT
      IF (NPRINT) RETURN
      CALL TIME(MH,MM,MS)
```

```
      NB=3600*MH+60*MM+MS
      NT=NB-NA
      PRINT 103,ISA,ISB,NT
      RETURN
      END
```

```
      SUBROUTINE SUB(ISA,ISB,ISC)
  100 FORMAT (9H SUBTRACT,I3,5H FROM,I3,13H AND STORE IN,I3)
  101 FORMAT (34H **** STORAGE OVERFLOW IN SUB ****)
  102 FORMAT (37H **** MSA AND/OR MSB ZERO IN SUB ****)
      COMMON/7/LFACTL(30),NUTIL(30),MSTART(125),MAX(125)
      COMMON/9/NOFACT,NORDER,NOSIMP,NPRINT,LPRINT
      TYPE LOGICAL NOFACT,NORDER,NOSIMP,NPRINT,LPRINT,PNPRINT
      IF (NPRINT) GO TO 2
      PRINT 100,ISB,ISA,ISC
    2 CONTINUE
      PNPRINT=NPRINT
      NPRINT=.TRUE.
      N6=NUTIL(6)
      CALL MULCONS(ISB,-1,1)
      CALL ADD(ISA,ISB,ISC)
      CALL MULCONS(ISB,-1,1)
      ID6=NUTIL(6)-N6
      NUTIL(7)=NUTIL(7)+ID6
      NUTIL(6)=N6
      NPRINT=PNPRINT
      RETURN
      END
```

```
      SUBROUTINE MULTIPLY(ISA,ISB,ISC)
 1002 FORMAT (9H MULTIPLY,3I4,11H FROM ORDER,I3,9H TO ORDER,I3)
 1003 FORMAT (39H **** STORAGE EXCEEDED IN MULTIPLY ****)
 1004 FORMAT (27H  NUMPAR,KOUNT,AND NTOTAL =,3I5)
 1005 FORMAT (30H  POINTER NEGATIVE IN MULTIPLY)
 1006 FORMAT (27H  TIME FOR MULTIPLICATION =,I4,4H SEC)
      COMMON/1/NEXT(6500)/2/N(6500)/3/M(6500)
      COMMON/4/IEX(6500)/5/ITRIG(6500)/6/IRG(6500)
      COMMON/7/LFACTL(30),NUTIL(30),MSTART(125),MAX(125)
      COMMON/8/KFACT(10),KMAX,MORDER,LORDER,LZERO,INDEX
      COMMON/9/NOFACT,NORDER,NOSIMP,NPRINT,LPRINT
      COMMON/10/NTOTAL/11/LMAP(256)
      COMMON/12/LPMAX/13/LPRIME(20)/14/EPSILON/15/NOFIX
      TYPE LOGICAL NOFACT,NORDER,NOSIMP,NPRINT,LPRINT,NOFIX
      TYPE LOGICAL KSCAN,PNPRINT
      DIMENSION IA(8),IB(8)
      CALL TIME(MH,MM,MS)
      NA=3600*MH+60*MM+MS
      MSA=MSTART(ISA)
      MSB=MSTART(ISB)
      CALL ZERO(ISC)
      IF (MSA.LE.0.OR.MSB.LE.0) RETURN
      PNPRINT=NPRINT
      NPRINT=.TRUE.
      KSCAN=.TRUE.
      CALL PACK(ISA)
      CALL PACK(ISB)
      MSA=MSTART(ISA)
      MSB=MSTART(ISB)
      NUMPAR=2
      DO 4 JX=1,256,1
    4 LMAP(JX)=0
      K=MSA $ J=MSB
    5 CALL INDICES(IEX(K),IA)
      KH=   KFACT(1)*IA(1)+KFACT(2)*IA(2)
      KH=KH+KFACT(3)*IA(3)+KFACT(4)*IA(4)
      KH=KH+KFACT(5)*IA(5)+KFACT(6)*IA(6)
      KH=KH+KFACT(7)*IA(7)+KFACT(8)*IA(8)
    6 IF (ITRIG(K)) 10,30,30
   10 IF (ITRIG(J)) 40,50,50
   30 IF (ITRIG(J)) 60,70,70
   40 CALL INDICES(IEX(J),IB)
      IH=KH+KFACT(1)*IB(1)+KFACT(2)*IB(2)
      IH=IH+KFACT(3)*IB(3)+KFACT(4)*IB(4)
      IH=IH+KFACT(5)*IB(5)+KFACT(6)*IB(6)
      IH=IH+KFACT(7)*IB(7)+KFACT(8)*IB(8)
      IF (IH.GT.MORDER) GO TO 300
      IF ((IA(8)+IB(8)).GT.KFACT(10)) GO TO 300
      I=LZERO $ II=NEXT(I)
      IF (II.GT.0) GO TO 41
      PRINT 1003 $ CALL QUIT $ STOP
   41 LZERO=NEXT(II) $ NEXT(II)=0
      IF (LZERO.GT.0) GO TO 42
      PRINT 1003 $ CALL QUIT $ STOP
   42 CONTINUE
      CALL RFACTOR(N(J),M(J),N(K),M(K),-1,+2,N(I),M(I),1)
      N(II)=-N(I)
      M(II)=M(I)
```

```
      ITRIG(II)=ITRIG(I)=+1
      IVA=IEX(K)-INDEX
      IEX(II)=IEX(I)=IEX(J)+IVA
      IVB=IRG(K)-INDEX
      IRG(I)=IRG(J)+IVB
      IVC=IRG(K)-IRG(J)
      IRG(II)=IVC+INDEX
      NUTIL(8)=NUTIL(8)+1
      IF (IRG(I)-INDEX) 43,44,44
   43 IRG(I)=IRG(I)-INDEX
      IRG(I)=INDEX-IRG(I)
   44 IF (IRG(II)-INDEX) 45,90,90
   45 IRG(II)=IRG(II)-INDEX
      IRG(II)=INDEX-IRG(II)
      GO TO 90
   50 CALL INDICES(IEX(J),IB)
      IH=KH+KFACT(1)*IB(1)+KFACT(2)*IB(2)
      IH=IH+KFACT(3)*IB(3)+KFACT(4)*IB(4)
      IH=IH+KFACT(5)*IB(5)+KFACT(6)*IB(6)
      IH=IH+KFACT(7)*IB(7)+KFACT(8)*IB(8)
      IF (IH.GT.MORDER) GO TO 300
      IF ((IA(8)+IB(8)).GT.KFACT(10)) GO TO 300
      I=LZERO $ II=NEXT(I)
      IF (II.GT.0) GO TO 51
      PRINT 1003 $ CALL QUIT $ STOP
   51 LZERO=NEXT(II) $ NEXT(II)=0
      IF (LZERO.GT.0) GO TO 52
      PRINT 1003 $ CALL QUIT $ STOP
   52 CONTINUE
      CALL RFACTOR(N(J),M(J),N(K),M(K),+1,+2,N(I),M(I),1)
      N(II)=+N(I)
      M(II)=M(I)
      ITRIG(II)=ITRIG(I)=-1
      IVA=IEX(K)-INDEX
      IEX(II)=IEX(I)=IEX(J)+IVA
      IVB=IRG(K)-INDEX
      IRG(I)=IRG(J)+IVB
      IVC=IRG(K)-IRG(J)
      IRG(II)=IVC+INDEX
      NUTIL(9)=NUTIL(9)+1
      IF (IRG(I)-INDEX) 53,54,55
   53 IRG(I)=IRG(I)-INDEX
      IRG(I)=INDEX-IRG(I)
      N(I)=-N(I) $ GO TO 55
   54 N(I)=0
   55 IF (IRG(II)-INDEX) 56,57,90
   56 IRG(II)=IRG(II)-INDEX
      IRG(II)=INDEX-IRG(II)
      N(II)=-N(II)
      GO TO 90
   57 N(II)=0
      GO TO 90
   60 CALL INDICES(IEX(J),IB)
      IH=KH+KFACT(1)*IB(1)+KFACT(2)*IB(2)
      IH=IH+KFACT(3)*IB(3)+KFACT(4)*IB(4)
      IH=IH+KFACT(5)*IB(5)+KFACT(6)*IB(6)
      IH=IH+KFACT(7)*IB(7)+KFACT(8)*IB(8)
      IF (IH.GT.MORDER) GO TO 300
      IF ((IA(8)+IB(8)).GT.KFACT(10)) GO TO 300
```

```
      I=LZERO $ II=NEXT(I)
      IF (II.GT.0) GO TO 61
      PRINT 1003 $ CALL QUIT $ STOP
   61 LZERO=NEXT(II) $ NEXT(II)=0
      IF (LZERO.GT.0) GO TO 62
      PRINT 1003 $ CALL QUIT $ STOP
   62 CONTINUE
      CALL RFACTOR(N(J),M(J),N(K),M(K),+1,+2,N(I),M(I),1)
      N(II)=-N(I)
      M(II)=M(I)
      ITRIG(II)=ITRIG(I)=-1
      IVA=IEX(K)-INDEX
      IEX(II)=IEX(I)=IEX(J)+IVA
      IVB=IRG(K)-INDEX
      IRG(I)=IRG(J)+IVB
      IVC=IRG(K)-IRG(J)
      IRG(II)=IVC+INDEX
      NUTIL(10)=NUTIL(10)+1
      IF (IRG(I)-INDEX) 63,64,65
   63 IRG(I)=IRG(I)-INDEX
      IRG(I)=INDEX-IRG(I)
      N(I)=-N(I)
      GO TO 65
   64 N(I)=0
   65 IF(IRG(II)-INDEX) 66,67,90
   66 IRG(II)=IRG(II)-INDEX
      IRG(II)=INDEX-IRG(II)
      N(II)=-N(II)
      GO TO 90
   67 N(II)=0
      GO TO 90
   70 CALL INDICES(IEX(J),IB)
      IH=KH+KFACT(1)*IB(1)+KFACT(2)*IB(2)
      IH=IH+KFACT(3)*IB(3)+KFACT(4)*IB(4)
      IH=IH+KFACT(5)*IB(5)+KFACT(6)*IB(6)
      IH=IH+KFACT(7)*IB(7)+KFACT(8)*IB(8)
      IF (IH.GT.MORDER) GO TO 300
      IF ((IA(8)+IB(8)).GT.KFACT(10)) GO TO 300
      I=LZERO $ II=NEXT(I)
      IF (II.GT.0) GO TO 71
      PRINT 1003 $ CALL QUIT $ STOP
   71 LZERO=NEXT(II) $ NEXT(II)=0
      IF (LZERO.GT.0) GO TO 72
      PRINT 1003 $ CALL QUIT $ STOP
   72 CONTINUE
      CALL RFACTOR(N(J),M(J),N(K),M(K),+1,+2,N(I),M(I),1)
      N(II)=+N(I)
      M(II)=M(I)
      ITRIG(II)=ITRIG(I)=+1
      IVA=IEX(K)-INDEX
      IEX(II)=IEX(I)=IEX(J)+IVA
      IVB=IRG(K)-INDEX
      IRG(I)=IRG(J)+IVB
      IVC=IRG(K)-IRG(J)
      IRG(II)=IVC+INDEX
      NUTIL(11)=NUTIL(11)+1
      IF (IRG(I)-INDEX) 73,74,74
   73 IRG(I)=IRG(I)-INDEX
      IRG(I)=INDEX-IRG(I)
```

```
 74 IF (IRG(II)-INDEX) 75,90,90
 75 IRG(II)=IRG(II)-INDEX
    IRG(II)=INDEX-IRG(II)
    GO TO 90
 90 CONTINUE
    L=MSC=MSTART(ISC)
    IF (MSC.NE.0) GO TO 94
    IF (IRG(I)-IRG(II)) 91,92,93
 91 NEXT(II)=I
    NEXT(I)=0
    MSTART(ISC)=II
    MAX(ISC)=I
    KOUNT=2
    GO TO 300
 92 CALL RFACTOR(N(I),M(I),N(II),M(II),1,1,N(I),M(I),0)
    M(II)=ITRIG(II)=1
    IEX(II)=IRG(II)=INDEX
    NEXT(II)=LZERO
    N(II)=NEXT(I)=0
    LZERO=II
    MSTART(ISC)=MAX(ISC)=I
    KOUNT=1
    NUTIL(5)=NUTIL(5)+1
    GO TO 300
 93 MSTART(ISC)=I
    MAX(ISC)=II
    KOUNT=2
    GO TO 300
 94 CONTINUE
    IF (NUMPAR.GE.256) GO TO 100
    IPAR=16*NUMPAR
    IF (KOUNT.LE.IPAR) GO TO 100
    KSCAN=.FALSE.
    NUMPAR=2*NUMPAR
    KS=MSTART(ISC)
    DO 98 MX=1,NUMPAR,1
    LMAP(MX)=KS
    DO 98 MY=1,8,1
    KS=NEXT(KS)
    IF (KS.LE.0) GO TO 450
 98 CONTINUE
100 L=MSC=MSTART(ISC) $ LAST=0
    IF (KSCAN) GO TO 125
    MX=NUMPAR/2 $ LX=LMAP(MX)
    MDX=NUMPAR/4
102 IF (IRG(LX)-IRG(I)) 105,103,107
103 IF (ITRIG(LX)-ITRIG(I)) 105,104,107
104 IF (IEX(LX)-IEX(I)) 105,106,107
105 IF (MDX.EQ.0) MDX=1
    MX=MX-MDX $ MDX=MDX/2
    IF (MX.LE.0) GO TO 118
    LX=LMAP(MX) $ GO TO 102
106 L=LX $ GO TO 160
107 IF (MDX.EQ.0) GO TO 120
    MX=MX+MDX $ LX=LMAP(MX)
    MDX=MDX/2 $ GO TO 102
118 LX=MSC
120 L=LX
125 IF (IRG(L)-IRG(I)) 130,140,170
```

```
140 IF (ITRIG(L)-ITRIG(I)) 130,150,170
150 IF (IEX(L)-IEX(I)) 130,160,170
130 IF (LAST.NE.0) GO TO 131
    MSTART(ISC)=I
    NEXT(I)=L
    KOUNT=KOUNT+1
    GO TO 200
131 NEXT(LAST)=I
    NEXT(I)=L
    KOUNT=KOUNT+1
    GO TO 200
160 CALL RFACTOR(N(I),M(I),N(L),M(L),1,1,N(L),M(L),0)
    N(I)=0
    M(I)=ITRIG(I)=1
    IEX(I)=IRG(I)=INDEX
    NEXT(I)=LZERO
    LZERO=I
    NUTIL(5)=NUTIL(5)+1
    GO TO 200
170 LN=NEXT(L)
    IF (LN) 450,171,172
171 NEXT(L)=MAX(ISC)=I
    NEXT(I)=0 $ KOUNT=KOUNT+1
    GO TO 200
172 LAST=L
    L=LN
    NUTIL(15)=NUTIL(15)+1
    GO TO 125
200 L=MSC=MSTART(ISC) $ LAST=0
    IF (KSCAN) GO TO 225
    MX=NUMPAR/2 $ LX=LMAP(MX)
    MDX=NUMPAR/4
202 IF (IRG(LX)-IRG(II)) 205,203,207
203 IF (ITRIG(LX)-ITRIG(II)) 205,204,207
204 IF (IEX(LX)-IEX(II)) 205,206,207
205 IF (MDX.EQ.0) MDX=1
    MX=MX-MDX $ MDX=MDX/2
    IF (MX.LE.0) GO TO 218
    LX=LMAP(MX) $ GO TO 202
206 L=LX $ GO TO 260
207 IF (MDX.EQ.0) GO TO 220
    MX=MX+MDX $ LX=LMAP(MX)
    MDX=MDX/2 $ GO TO 202
218 LX=MSC
220 L=LX
225 IF (IRG(L)-IRG(II)) 230,240,270
240 IF (ITRIG(L)-ITRIG(II)) 230,250,270
250 IF (IEX(L)-IEX(II)) 230,260,270
230 IF (LAST.NE.0) GO TO 231
    MSTART(ISC)=II
    NEXT(II)=L
    KOUNT=KOUNT+1
    GO TO 300
231 NEXT(LAST)=II
    NEXT(II)=L
    KOUNT=KOUNT+1
    GO TO 300
260 CALL RFACTOR(N(II),M(II),N(L),M(L),1,1,N(L),M(L),0)
    N(II)=0
```

```
      M(II)=ITRIG(II)=1
      IEX(II)=IRG(II)=INDEX
      NEXT(II)=LZERO
      LZERO=II
      NUTIL(5)=NUTIL(5)+1
      GO TO 300
  270 LN=NEXT(L)
      IF (LN) 450,271,272
  271 NEXT(L)=MAX(ISC)=II
      NEXT(II)=0 $ KOUNT=KOUNT+1
      GO TO 300
  272 LAST=L $ L=LN
      NUTIL(15)=NUTIL(15)+1
      GO TO 225
  300 CONTINUE
      NJ=NEXT(J)
      NUTIL(16)=NUTIL(16)+1
      IF (NJ.EQ.0) GO TO 400
      J=NJ
      GO TO 6
  400 NK=NEXT(K)
      IF (NK.EQ.0) GO TO 500
      K=NK
      J=MSB
      GO TO 5
  450 PRINT 1005 $ CALL QUIT $ STOP
  500 CALL PACK(ISC)
      NPRINT=PNPRINT
      IF (NPRINT) RETURN
      CALL TIME(MH,MM,MS)
      NB=3600*MH+60*MM+MS
      NT=NB-NA
      PRINT 1002,ISA,ISB,ISC,LORDER,MORDER
      PRINT 1004,NUMPAR,KOUNT,NTOTAL
      PRINT 1006,NT
      RETURN
      END
```

```
      SUBROUTINE POWER(ISA,IJKL,ISB)
  100 FORMAT (I2,16H RAISED TO POWER,I3,17H RESULT STORED IN,I3)
  101 FORMAT (44H **** SERIES RAISED TO A NEGATIVE POWER ****)
  102 FORMAT (14H  TOTAL TIME =,I4,4H SEC)
      COMMON/1/NEXT(6500)/2/N(6500)/3/M(6500)
      COMMON/4/IEX(6500)/5/ITRIG(6500)/6/IRG(6500)
      COMMON/7/LFACTL(30),NUTIL(30),MSTART(125),MAX(125)
      COMMON/8/KFACT(10),KMAX,MORDER,LORDER,LZERO,INDEX
      COMMON/9/NOFACT,NORDER,NOSIMP,NPRINT,LPRINT
      TYPE LOGICAL NOFACT,NORDER,NOSIMP,NPRINT,LPRINT,PNPRINT
      CALL TIME(MH,MM,MS)
      NA=3600*MH+60*MM+MS
      PNPRINT=NPRINT
      IF (LPRINT) NPRINT=.TRUE.
      PRINT 100, ISA,IJKL,ISB
      IF (IJKL.GE.0) GO TO 5
      PRINT 101 $ CALL QUIT $ STOP
    5 CONTINUE
      ISC=107 $ CALL ZERO(ISC)
      IJ=IJKL-1
      IF (IJ) 6,7,8
    6 CONTINUE
      CALL ZERO(ISB)
      MSTART(ISB)=LZERO
      MSB=LZERO
      N(MSB)=1
      LZERO=NEXT(MSB)
      MAX(ISB)=MSB
      NEXT(MSB)=0
      GO TO 12
    7 CONTINUE
      CALL TRANSF(ISA,ISB)
      GO TO 12
    8 CONTINUE
      CALL TRANSF(ISA,ISB)
      DO 10 K=1,IJ,1
      CALL MULTIPLY(ISA,ISB,ISC)
      CALL SWITCH(ISC,ISB)
   10 CONTINUE
   12 CONTINUE
      CALL ZERO(ISC)
      IF (LPRINT) NPRINT=PNPRINT
      CALL TIME(MH,MM,MS)
      NB=3600*MH+60*MM+MS
      NT=NB-NA
      PRINT 102,NT
      RETURN
      END
```

```
      SUBROUTINE EXPAND(ISA,IORDER,IN,IM,ISB)
  100 FORMAT (12H EXPAND (1 +,I3,4H)**(,I2,1H/,I2,1H))
  101 FORMAT (10H  TO ORDER,I3,3H IN,I3)
  102 FORMAT (18H  RESULT STORED IN,I3)
  103 FORMAT (14H  TOTAL TIME =,I4,4H SEC)
      COMMON/1/NEXT(6500)/2/N(6500)/3/M(6500)
      COMMON/4/IEX(6500)/5/ITRIG(6500)/6/IRG(6500)
      COMMON/7/LFACTL(30),NUTIL(30),MSTART(125),MAX(125)
      COMMON/8/KFACT(10),KMAX,MORDER,LORDER,LZERO,INDEX
      COMMON/9/NOFACT,NORDER,NOSIMP,NPRINT,LPRINT
      TYPE LOGICAL NOFACT,NORDER,NOSIMP,NPRINT,LPRINT,PNPRINT
      CALL TIME(MH,MM,MS)
      NA=3600*MH+60*MM+MS
      PNPRINT=NPRINT
      IF (LPRINT) NPRINT=.TRUE.
      PRINT 100, ISA,IN,IM
      PRINT 101, IORDER,ISA
      PRINT 102, ISB
      CALL ZERO(ISB)
      ISC=108 $ CALL ZERO(ISC)
      ISD=109 $ CALL ZERO(ISD)
      MSTART(ISB)=LZERO
      MSB=LZERO
      N(MSB)=1
      LZERO=NEXT(MSB)
      MAX(ISB)=MSB
      NEXT(MSB)=0
      MSTART(ISC)=LZERO
      MSC=LZERO
      N(MSC)=1
      LZERO=NEXT(MSC)
      MAX(ISC)=MSC
      NEXT(MSC)=0
      IF (IN.EQ.0) GO TO 30
      IF (IORDER.EQ.0) GO TO 30
      JN=IN
      JM=IM
      DO 20 K=1,IORDER,1
      JN=IN-(K-1)*IM
      JM=K*IM
      CALL FACTOR(JN,JM)
      CALL MULTIPLY(ISA,ISC,ISD)
      CALL ZERO(ISC)
      CALL MULCONS(ISD,JN,JM)
      CALL ADD(ISB,ISD,ISB)
      CALL SWITCH(ISD,ISC)
   20 CONTINUE
   30 CONTINUE
      CALL ZERO(ISC)
      CALL ZERO(ISD)
      IF (LPRINT) NPRINT=PNPRINT
      CALL TIME(MH,MM,MS)
      NB=3600*MH+60*MM+MS
      NT=NB-NA
      PRINT 103,NT
      RETURN
      END
```

```
      SUBROUTINE SUBAB(ISA,ISB,IVA,IVB,NUM)
  100 FORMAT (11H SUBSTITUTE,5I4)
  101 FORMAT (36H **** STORAGE OVERFLOW IN SUBAB ****)
  102 FORMAT (28H **** MSB ZERO IN SUBAB ****)
  103 FORMAT (14H  TOTAL TIME =,I4,4H SEC)
      COMMON/1/NEXT(6500)/2/N(6500)/3/M(6500)
      COMMON/4/IEX(6500)/5/ITRIG(6500)/6/IRG(6500)
      COMMON/7/LFACTL(30),NUTIL(30),MSTART(125),MAX(125)
      COMMON/8/KFACT(10),KMAX,MORDER,LORDER,LZERO,INDEX
      COMMON/9/NOFACT,NORDER,NOSIMP,NPRINT,LPRINT
      TYPE LOGICAL NOFACT,NORDER,NOSIMP,NPRINT,LPRINT,PNPRINT
      DIMENSION IA(8)
      PNPRINT=NPRINT
      CALL TIME(MH,MM,MS)
      NA=3600*MH+60*MM+MS
      IF (LPRINT) NPRINT=.TRUE.
      PRINT 100,ISA,ISB,IVA,IVB,NUM
      CALL TRANSF(ISA,ISB)
      CALL PACK(ISB)
      DO 50 J=1,NUM,1
      ISC=110 $ CALL ZERO(ISC)
      K=MSB=MSTART(ISB)
      L=MSC=MSTART(ISC)=LZERO
      IF (MSB.GT.0) GO TO 10
      PRINT 102
      IF (LPRINT) NPRINT=.FALSE.
      RETURN
   10 CONTINUE
      CALL INDICES(IEX(K),IA)
      IF (IA(IVA).LT.2) GO TO 30
      IA(IVA)=IA(IVA)-2
      CALL IMPACT(IA,IEX(K))
      N(L)=-N(K)
      M(L)=+M(K)
      IA(IVB)=IA(IVB)+2
      CALL IMPACT(IA,IEX(L))
      ITRIG(L)=ITRIG(K)
      IRG(L)=IRG(K)
      LN=NEXT(L)
      IF (LN.GT.0) GO TO 20
      PRINT 101
      CALL QUIT
      STOP
   20 CONTINUE
      L=LN
   30 CONTINUE
      KN=NEXT(K)
      IF (KN.EQ.0) GO TO 40
      K=KN
      GO TO 10
   40 CONTINUE
      MXC=MAX(ISC)=L
      LZERO=NEXT(MXC)
      NEXT(MXC)=0
      CALL ADD(ISB,ISC,ISB)
      CALL ZERO(ISC)
   50 CONTINUE
      IF (LPRINT) NPRINT=PNPRINT
```

```
CALL TIME(MH,MM,MS)
NB=3600*MH+60*MM+MS
NT=NB-NA
PRINT 103,NT
RETURN
END
```

```
      SUBROUTINE DERIV(ISA,IV,ISB)
  100 FORMAT(14H DIFFERENTIATE,I3,16H WITH RESPECT TO,I3)
  101 FORMAT(18H  RESULT STORED IN,I3)
  102 FORMAT (28H **** MSB ZERO IN DERIV ****)
      COMMON/1/NEXT(6500)/2/N(6500)/3/M(6500)
      COMMON/4/IEX(6500)/5/ITRIG(6500)/6/IRG(6500)
      COMMON/7/LFACTL(30),NUTIL(30),MSTART(125),MAX(125)
      COMMON/8/KFACT(10),KMAX,MORDER,LORDER,LZERO,INDEX
      COMMON/9/NOFACT,NORDER,NOSIMP,NPRINT,LPRINT
      COMMON/12/LPMAX/13/LPRIME(20)/14/EPSILON/15/NOFIX
      TYPE LOGICAL NOFACT,NORDER,NOSIMP,NPRINT,LPRINT,NOFIX
      DIMENSION IA(8),II(8)
      IF (NPRINT) GO TO 2
      PRINT 100,ISA,IV
      PRINT 101,ISB
    2 CONTINUE
      CALL TRANSF(ISA,ISB)
      MSB=MSTART(ISB)
      IF (MSB.GT.0) GO TO 10
      IF (NPRINT) RETURN
      PRINT 102
      RETURN
   10 CONTINUE
      K=MSB
   20 CONTINUE
      NUTIL(12)=NUTIL(12)+1
      IF (IV.GE.9) GO TO 40
      IF (IV.LE.8) GO TO 30
   30 CONTINUE
      CALL INDICES(IEX(K),IA)
      KCON=IA(IV)
      IA(IV)=IA(IV)-1
      CALL IMPACT(IA,IEX(K))
      CALL RFACTOR(N(K),M(K),1,1,KCON,1,N(K),M(K),1)
      GO TO 50
   40 CONTINUE
      JV=IV-8
      CALL INDICES(IRG(K),II)
      KCON=-II(JV)*ITRIG(K)
      ITRIG(K)=-ITRIG(K)
      CALL RFACTOR(N(K),M(K),1,1,KCON,1,N(K),M(K),1)
      GO TO 50
   50 CONTINUE
      KN=NEXT(K)
      IF (KN.EQ.0) GO TO 60
      K=KN
      GO TO 20
   60 CONTINUE
      CALL PACK(ISB)
      RETURN
      END
```

```
      SUBROUTINE NDERIV(ISA,IV,NUM,ISB)
      COMMON/9/NOFACT,NORDER,NOSIMP,NPRINT,LPRINT
      TYPE LOGICAL NOFACT,NORDER,NOSIMP,NPRINT,LPRINT,PNPRINT
  100 FORMAT (10H PDERIV OF,I3,4H WRT,I3,I8,6H TIMES)
  101 FORMAT (18H  RESULT STORED IN,I3)
      PNPRINT=NPRINT
      IF (LPRINT) NPRINT=.TRUE.
      PRINT 100,ISA,IV,NUM
      PRINT 101,ISB
      CALL TRANSF(ISA,ISB)
      DO 50 K=1,NUM,1
      CALL DERIV(ISB,IV,ISB)
   50 CONTINUE
      IF (LPRINT) NPRINT=PNPRINT
      RETURN
      END
```

```
      SUBROUTINE BRACKET(ISA,ISB,IP,IQ,ISC)
  100 FORMAT (19H POISSON BRACKET OF,I3,1H,,I3,4H WRT,I3,1H,,I3)
  101 FORMAT (18H  RESULT STORED IN,I3)
      COMMON/9/NOFACT,NORDER,NOSIMP,NPRINT,LPRINT
      TYPE LOGICAL NOFACT,NORDER,NOSIMP,NPRINT,LPRINT
      PRINT 100,ISA,ISB,IP,IQ
      PRINT 101,ISC
      ISD=111 $ CALL ZERO(ISD)
      ISE=112 $ CALL ZERO(ISE)
      ISF=113 $ CALL ZERO(ISF)
      CALL DERIV(ISA,IP,ISC)
      CALL DERIV(ISB,IQ,ISD)
      CALL MULTIPLY(ISC,ISD,ISE)
      CALL DERIV(ISB,IP,ISC)
      CALL DERIV(ISA,IQ,ISD)
      CALL MULTIPLY(ISC,ISD,ISF)
      CALL SUB(ISE,ISF,ISC)
      CALL ZERO(ISD)
      CALL ZERO(ISE)
      CALL ZERO(ISF)
      RETURN
      END
```

```
      SUBROUTINE INTEG(ISA,IV,ISB)
  100 FORMAT (10H INTEGRATE,I3,16H WITH RESPECT TO,I3)
  101 FORMAT (18H  RESULT STORED IN,I3)
  102 FORMAT (28H **** MSB ZERO IN INTEG ****)
      COMMON/1/NEXT(6500)/2/N(6500)/3/M(6500)
      COMMON/4/IEX(6500)/5/ITRIG(6500)/6/IRG(6500)
      COMMON/7/LFACTL(30),NUTIL(30),MSTART(125),MAX(125)
      COMMON/8/KFACT(10),KMAX,MORDER,LORDER,LZERO,INDEX
      COMMON/9/NOFACT,NORDER,NOSIMP,NPRINT,LPRINT
      COMMON/12/LPMAX/13/LPRIME(20)/14/EPSILON/15/NOFIX
      TYPE LOGICAL NOFACT,NORDER,NOSIMP,NPRINT,LPRINT,NOFIX
      DIMENSION IA(8),II(8)
      IF (NPRINT) GO TO 2
      PRINT 100,ISA,IV
      PRINT 101,ISB
    2 CONTINUE
      CALL TRANSF(ISA,ISB)
      MSB=MSTART(ISB)
      IF (MSB.GT.0) GO TO 10
      IF (NPRINT) RETURN
      PRINT 102
      RETURN
   10 CONTINUE
      K=MSB
   20 CONTINUE
      NUTIL(13)=NUTIL(13)+1
      IF (IV.GE.9) GO TO 40
      IF (IV.LE.8) GO TO 30
   30 CONTINUE
      CALL INDICES(IEX(K),IA)
      IA(IV)=IA(IV)+1
      KCON=IA(IV)
      CALL IMPACT(IA,IEX(K))
      CALL RFACTOR(N(K),M(K),1,1,1,KCON,N(K),M(K),1)
      GO TO 50
   40 CONTINUE
      JV=IV-8
      CALL INDICES(IRG(K),II)
      KCON=II(JV)*ITRIG(K)
      ITRIG(K)=-ITRIG(K)
      CALL RFACTOR(N(K),M(K),1,1,1,KCON,N(K),M(K),1)
      GO TO 50
   50 CONTINUE
      KN=NEXT(K)
      IF (KN.EQ.0) GO TO 60
      K=KN
      GO TO 20
   60 CONTINUE
      CALL PACK(ISB)
      RETURN
      END
```

```
      SUBROUTINE TAYLOR(ISA,ISB,IV,NUM,ISC)
  100 FORMAT (24H TAYLOR SERIES EXPANSION,5I5)
  101 FORMAT (36H  TIME FOR TAYLOR SERIES EXPANSION =,I4,4H SEC)
      COMMON/1/NEXT(6500)/2/N(6500)/3/M(6500)
      COMMON/4/IEX(6500)/5/ITRIG(6500)/6/IRG(6500)
      COMMON/7/LFACTL(30),NUTIL(30),MSTART(125),MAX(125)
      COMMON/8/KFACT(10),KMAX,MORDER,LORDER,LZERO,INDEX
      COMMON/9/NOFACT,NORDER,NOSIMP,NPRINT,LPRINT
      TYPE LOGICAL NOFACT,NORDER,NOSIMP,NPRINT,LPRINT,PNPRINT
      CALL TIME(MH,MM,MS)
      NA=3600*MH+60*MM+MS
      PNPRINT=NPRINT
      PRINT 100,ISA,ISB,IV,NUM,ISC
      ISD=114 $ CALL ZERO(ISD)
      ISE=115 $ CALL ZERO(ISE)
      ISF=116 $ CALL ZERO(ISF)
      IF (LPRINT) NPRINT=.TRUE.
      MS=LZERO
      MSTART(ISD)=MAX(ISD)=MS
      N(MS)=1
      LZERO=NEXT(MS)
      NEXT(MS)=0
      CALL TRANSF(ISA,ISE)
      CALL TRANSF(ISA,ISC)
      DO 10 K=1,NUM,1
      CALL DERIV(ISE,IV,ISE)
      CALL MULCONS(ISE,1,K)
      CALL MULTIPLY(ISB,ISD,ISF)
      CALL SWITCH(ISF,ISD)
      CALL MULTIPLY(ISD,ISE,ISF)
      CALL ADD(ISC,ISF,ISC)
   10 CONTINUE
      CALL ZERO(ISD)
      CALL ZERO(ISE)
      CALL ZERO(ISF)
      IF (LPRINT) NPRINT=PNPRINT
      CALL TIME(MH,MM,MS)
      NB=3600*MH+60*MM+MS
      NT=NB-NA
      PRINT 101,NT
      RETURN
      END
```

```
      SUBROUTINE ERASE(ISA,IV)
  100 FORMAT (6H ERASE,I3,4H WRT,I3)
      COMMON/1/NEXT(6500)/2/N(6500)/3/M(6500)
      COMMON/4/IEX(6500)/5/ITRIG(6500)/6/IRG(6500)
      COMMON/7/LFACTL(30),NUTIL(30),MSTART(125),MAX(125)
      COMMON/8/KFACT(10),KMAX,MORDER,LORDER,LZERO,INDEX
      COMMON/9/NOFACT,NORDER,NOSIMP,NPRINT,LPRINT
      TYPE LOGICAL NOFACT,NORDER,NOSIMP,NPRINT,LPRINT
      DIMENSION IAA(8)
      IF (NPRINT) GO TO 5
      PRINT 100,ISA,IV
    5 CONTINUE
      K=MSA=MSTART(ISA)
      IF (MSA.EQ.0) RETURN
   10 CONTINUE
      CALL INDICES(IEX(K),IAA)
      IAA(IV)=0
      CALL IMPACT(IAA,IEX(K))
      KN=NEXT(K)
      IF (KN.EQ.0) GO TO 20
      K=KN
      GO TO 10
   20 CONTINUE
      RETURN
      END
```

```
      SUBROUTINE MULVAR(ISA,IV,NUM)
  100 FORMAT (9H MULTIPLY,I3,11H BY VARIBLE,I3,9H TO POWER,I3)
      COMMON/1/NEXT(6500)/2/N(6500)/3/M(6500)
      COMMON/4/IEX(6500)/5/ITRIG(6500)/6/IRG(6500)
      COMMON/7/LFACTL(30),NUTIL(30),MSTART(125),MAX(125)
      COMMON/8/KFACT(10),KMAX,MORDER,LORDER,LZERO,INDEX
      COMMON/9/NOFACT,NORDER,NOSIMP,NPRINT,LPRINT
      TYPE LOGICAL NOFACT,NORDER,NOSIMP,NPRINT,LPRINT
      DIMENSION IAA(8)
      IF (NPRINT) GO TO 5
      PRINT 100,ISA,IV,NUM
    5 CONTINUE
      K=MSA=MSTART(ISA)
      IF (MSA.EQ.0) RETURN
   10 CONTINUE
      CALL INDICES(IEX(K),IAA)
      IAA(IV)=IAA(IV)+NUM
      CALL IMPACT(IAA,IEX(K))
      KN=NEXT(K)
      IF (KN.EQ.0) GO TO 20
      K=KN
      GO TO 10
   20 CONTINUE
      RETURN
      END
```

```
      SUBROUTINE ORDER(ISA)
  100 FORMAT (28H **** OVERFLOW IN ORDER ****)
  101 FORMAT (14H TIME TO ORDER,I4,2H =,I4)
      COMMON/1/NEXT(6500)/2/N(6500)/3/M(6500)
      COMMON/4/IEX(6500)/5/ITRIG(6500)/6/IRG(6500)
      COMMON/7/LFACTL(30),NUTIL(30),MSTART(125),MAX(125)
      COMMON/8/KFACT(10),KMAX,MORDER,LORDER,LZERO,INDEX
      COMMON/9/NOFACT,NORDER,NOSIMP,NPRINT,LPRINT
      TYPE LOGICAL NOFACT,NORDER,NOSIMP,NPRINT,LPRINT
      IF (NORDER) RETURN
      CALL TIME(MH,MM,MS)
      NA=3600*MH+60*MM+MS
      ISB=101 $ CALL ZERO(ISB)
      MX=MSA=MSTART(ISA)
      IF (MSA.EQ.0) RETURN
      K=NEXT(MX)
      IF (K.EQ.0) RETURN
      LB=MSB=MSTART(ISB)=LZERO
      LZERO=NEXT(MSB) $ NEXT(MSB)=0
      IF (LZERO.GT.0) GO TO 3 $ PRINT 100
      CALL QUIT $ STOP
    3 LASTK=MX $ LASTMX=0
    5 IF (IEX(MX)-IEX(K)) 7,8,10
    8 IF (IRG(MX)-IRG(K)) 7,9,10
    9 IF (ITRIG(MX)-ITRIG(K)) 7,10,10
    7 MX=K $ LASTMX=LASTK
   10 KN=NEXT(K)
      IF (KN.EQ.0) GO TO 20
      LASTK=K $ K=KN
      GO TO 5
   20 IF (MX.EQ.MSA) GO TO 22
      NEXT(LASTMX)=NEXT(MX)
      GO TO 24
   22 NMSA=NEXT(MSA)
      IF (NMSA.EQ.0) GO TO 26
      MSA=NMSA
   24 NEXT(LB)=MX
      NEXT(MX)=0
      LB=MX
      LASTMX=0
      MX=MSA
      NK=NEXT(MX)
      IF (NK.EQ.0) GO TO 26
      LASTK=MX
      K=NK
      GO TO 5
   26 NEXT(LB)=MSA
      NEXT(MSA)=0
      CALL SWITCH(ISA,ISB)
      MSTART(ISB)=MAX(ISB)=0
      CALL PACK(ISA)
      IF (NPRINT) RETURN
      CALL TIME(MH,MM,MS)
      NB=3600*MH+60*MM+MS
      NT=NB-NA
      PRINT 101,ISA,NT
      RETURN
      END
```

```
      GO TO 60
   60 IF (.NOT.NOFACT) PRINT 103
      KN=NEXT(K)
      IF (KN.EQ.0) GO TO 70
      K=KN $ GO TO 10
   70 PRINT 101
      RETURN
      END
```

```
      SUBROUTINE OUTAA(ISA,IVA,IVI)
  100 FORMAT (6H WRITE,I3)
  101 FORMAT (///)
  102 FORMAT(I35)
  103 FORMAT (1H )
  104 FORMAT (21H **** FILE EMPTY ****)
  142 FORMAT (I35,4H*E**,I2,5H*SIN(,I2,2HM))
  143 FORMAT (I35,4H*E**,I2)
  144 FORMAT (I35,4H*E**,I2,5H*COS(,I2,2HM))
  152 FORMAT (F35.8,4H*E**,I2,5H*SIN(,I2,2HM))
  153 FORMAT (F35.8,4H*E**,I2)
  154 FORMAT (F35.8,4H*E**,I2,5H*COS(,I2,2HM))
      COMMON/1/NEXT(6500)/2/N(6500)/3/M(6500)
      COMMON/4/IEX(6500)/5/ITRIG(6500)/6/IRG(6500)
      COMMON/7/LFACTL(30),NUTIL(30),MSTART(125),MAX(125)
      COMMON/8/KFACT(10),KMAX,MORDER,LORDER,LZERO,INDEX
      COMMON/9/NOFACT,NORDER,NOSIMP,NPRINT,LPRINT
      TYPE LOGICAL NOFACT,NORDER,NOSIMP,NPRINT,LPRINT
      DIMENSION IA(8),II(8)
      PRINT 100,ISA $ PRINT 101
      K=MS=MSTART(ISA)
      IF (MS.GT.0) GO TO 10
      PRINT 104 $ RETURN
   10 IF (N(K).EQ.0) GO TO 60
      NUTIL(14)=NUTIL(14)+1
      IF (M(K).LT.0) GO TO 20
      GO TO 25
   20 N(K)=-N(K) $ M(K)=-M(K)
   25 CALL INDICES(IEX(K),IA) $ CALL INDICES(IRG(K),II)
      IF (IRG(K).EQ.INDEX) ITRIG(K)=0
      IF (NOFACT) GO TO 50
      SM=M(K) $ SM=ABS(SM)
      IF (SM.LT.1.0E6) GO TO 40
      SM=ALOG10(SM)+0.0001
      SM=AMOD(SM,1.0)
      IF (SM.LT.0.001) GO TO 50
   40 IF (ITRIG(K)) 42,43,44
   42 PRINT 142,N(K),IA(IVA),II(IVI)
      IF (M(K).EQ.1) GO TO 60
      PRINT 102, M(K)
      GO TO 60
   43 PRINT 143,N(K),IA(IVA)
      ITRIG(K)=1
      IF (M(K).EQ.1) GO TO 60
      PRINT 102, M(K)
      GO TO 60
   44 PRINT 144,N(K),IA(IVA),II(IVI)
      IF (M(K).EQ.1) GO TO 60
      PRINT 102, M(K)
      GO TO 60
   50 COEFF=N(K) $ COEFF=COEFF/M(K)
      IF (ITRIG(K)) 52,53,54
   52 PRINT 152,COEFF,IA(IVA),II(IVI)
      GO TO 60
   53 PRINT 153,COEFF,IA(IVA)
      ITRIG(K)=1
      GO TO 60
   54 PRINT 154,COEFF,IA(IVA),II(IVI)
```

Security Classification

DOCUMENT CONTROL DATA - R & D

(Security classification of title, body of abstract and indexing annotation must be entered when the overall report is classified)

1. ORIGINATING ACTIVITY *(Corporate author)* Naval Research Laboratory Washington, D.C. 20375	2a. REPORT SECURITY CLASSIFICATION Unclassified 2b. GROUP

3. REPORT TITLE

ALGEBRAIC MANIPULATION BY COMPUTER

4. DESCRIPTIVE NOTES *(Type of report and inclusive dates)*

Final report on one phase of a continuing problem.

5. AUTHOR(S) *(First name, middle initial, last name)*

R. R. Dasenbrock

6. REPORT DATE June 29, 1973	7a. TOTAL NO. OF PAGES 71	7b. NO. OF REFS 10
8a. CONTRACT OR GRANT NO. NRL Problem B01-10 b. PROJECT NO. RR 003-02-41-6152 c. d.	9a. ORIGINATOR'S REPORT NUMBER(S) NRL Report 7564 9b. OTHER REPORT NO(S) *(Any other numbers that may be assigned this report)* Operations Research Group Report 73-1	

10. DISTRIBUTION STATEMENT

Approved for public release; distribution unlimited.

11. SUPPLEMENTARY NOTES	12. SPONSORING MILITARY ACTIVITY Department of the Navy (Office of Naval Research) Arlington, Va. 22217

13. ABSTRACT

This report describes a program which has the capability of performing literal algebraic manipulations on a high speed electronic computer. Emphasis is concentrated on the manipulation of the Poisson series occurring in the classical theory of lightly perturbed dynamical systems. The program is written in Fortran and is now operational on a CDC-3800. With little or no modification it could be implemented on any computer possessing a Fortran compiler.

DD FORM 1 NOV 65 1473 (PAGE 1)

S/N 0101-807-6801

Security Classification

Security Classification

14. KEY WORDS	LINK A ROLE	LINK A WT	LINK B ROLE	LINK B WT	LINK C ROLE	LINK C WT
Machine algebra						
Algebraic manipulation						
Literal expansions						

DD FORM 1 NOV 65 1473 (BACK)
(PAGE 2)

Security Classification

☆U.S. GOVERNMENT PRINTING OFFICE:1973 542-163/ Z-4 1-3