# A scheme for manipulative algebra on a computer

*By* D. Barton*

**This paper describes a scheme of programs that have been written to perform elementary manipulation of multiple Fourier series. These are represented in their literal form in the computer and a wide variety of operations may be performed upon them. The paper describes the principles adopted while programming the scheme, lists a number of the principal operations that may be performed, and presents two examples to demonstrate how the scheme may be employed to solve certain types of equations.**

## Introduction

In this paper we shall describe a scheme of programs that have been written to perform a limited quantity of algebraical and trignometrical manipulation. This scheme has been implemented on the Titan computer in Cambridge and was originally designed to assist in a projected analytical development of the Main Problem of the Lunar Theory.

The scheme takes the form of a set of closed subroutines to perform various algebraical and trigonometrical operations, and the program has been written in Titan (Atlas 2) assembly code. A program that intends to perform manipulation using the scheme must first be written in an informal language that resembles the Titan Autocode, and subsequently hand-coded into assembly language. However, the hand-coding stage is very simple and it would be an elementary matter to write an interpretive program to accept the original "Autocode" and arrange to enter assembly code subroutines. Indeed the Strachey macrogenerator (Strachey, 1965) has been successfully used for this purpose.

## Representation of expressions

The basic operations that the scheme is able to perform are those of addition and multiplication upon functions of the form

$$\alpha = \sum_{\substack{ijk \\ i'j'k'}} P_{i'j'k'}^{ijk}(x_0, \ldots, x_7,) \begin{matrix}\cos \\ \sin\end{matrix} (iy_0 + jy_1 + \ldots + k'y_5) \qquad (1)$$

where $i, j, \ldots, k'$ are integers in the range $-63 \leqslant i \leqslant 63$

$y_0, \ldots, y_5$ are so-called *harmonic variables*

while $P_{i'j'k'}^{ijk}$ is a polynomial in the arguments $x_0, \ldots, x_7$. The latter variables are referred to as *polynomial-variables* and they may each occur to any degree in the range $0 \leqslant n \leqslant 63$. The polynomial $P_{i'j'k'}^{ijk}$ has rational coefficients and these are represented as the ratios of integers that in absolute magnitude must all be less than $2^{39}$.

The number of variables that the function $\alpha$ contains, together with the range of exponents and periods associated with the several variables, were all chosen with the particular problem of the lunar theory in mind. However, the methods that have been used to represent these functions on the computer and to manipulate the data internally are, in principle, independent of these restrictions.

In order to try to make efficient use of the machine store it was decided to represent the algebraic expressions as list structures. Accordingly the store was divided into units, two words in length,† that are all initially chained together to form a free list (see Woodward and Jenkins, 1961). An algebraic expression is then represented as an ordered list of units into each of which is packed the periods and phase of a periodic term together with two pointers. The first pointer is the address of the unit containing the next periodic term in the list while the second points to the head of a chain of units that each contain a term of the polynomial coefficient. Thus a polynomial coefficient is represented as an ordered list that branches from the main list of periodic terms, the branch denoting multiplication. When a unit is used to store a polynomial term it contains the exponents of the several variables that may occur in such a term together with a pointer to the next polynomial term and a pointer to a unit that contains the rational coefficient of the term.

Thus an algebraic expression is represented as an ordered list structure in the computer store as shown in **Fig. 1**. In order to ensure that there should be a unique correspondence between actual algebraic expressions and the list structures used by the computer, every expression used by the several manipulative subroutines must be presented in a standard form, and the output from all subroutines must also be in that form. The standard form is simply that in which the maximum amount of algebraic cancellation has taken place and in which every rational coefficient has been reduced to the ratio of mutually prime integers. Further, an expression in standard form has its periodic terms arranged in a well-defined order according to its periods, and each polynomial coefficient is similarly ordered according to the exponents of its terms.

† The Titan has 64K of 48-bit words.

* *St. John's College, Cambridge, and University Mathematical Laboratory, Cambridge.*

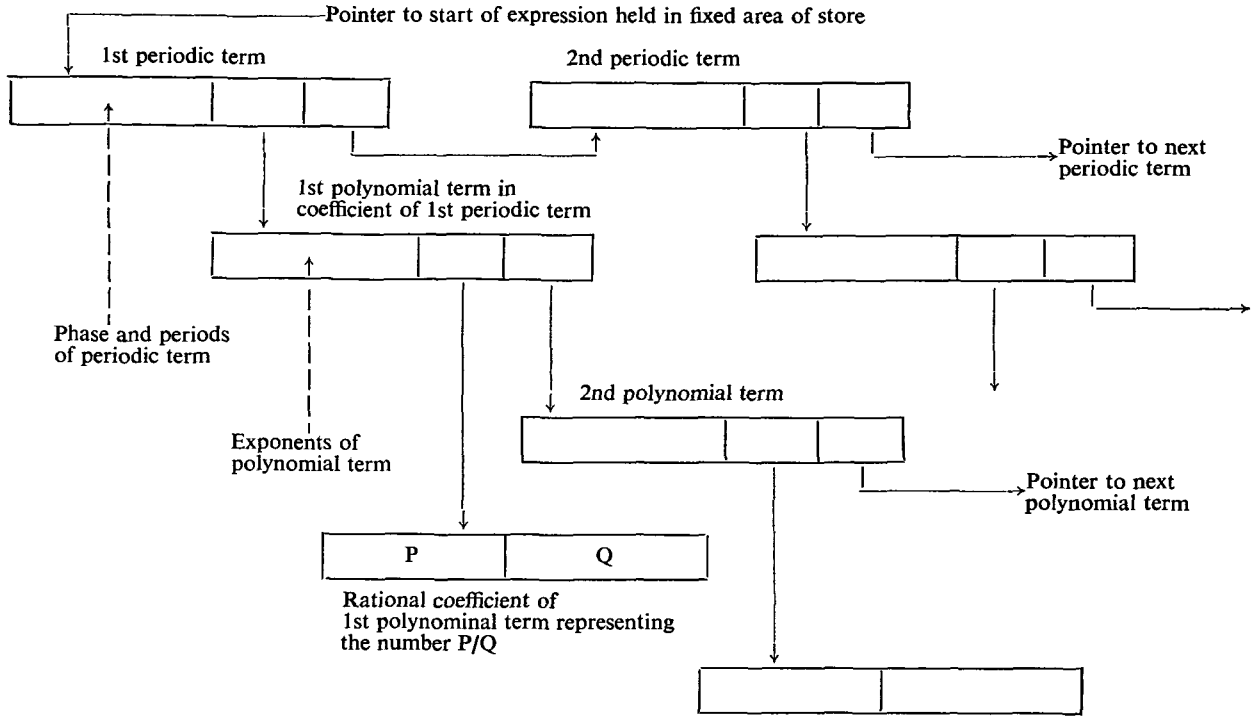


**Fig. 1.**

The above means of representation of our expressions ensures that whenever an expression is no longer required in the computer the space that it occupies may be immediately added to the free list of available space and later conveniently reissued. No form of "garbage collection" is necessary and complex shifting procedures to obtain continuous large expanses of empty store are entirely avoided.

**Programming details**

It has been mentioned above that the scheme was originally designed to assist with an analytical development of the lunar theory. The analysis required for this theory is so extensive that it was doubtful whether the scheme would be adequate to the task, and consequently the primary consideration that has taken precedence over all others while writing the program has been to conserve the machine's store. It was not considered important that the run-time system should operate quickly. In order to reduce the demands made for space at run time many of the manipulative subroutines have been written in a form that overwrites the operands that the subroutine uses. All subroutines that involve only addition or term-by-term differentiation or integration are written in this form, and consequently they make no demand at all upon the space allocation routines, indeed many of them will cause the list of free space to increase owing to cancellation.

However, any form of multiplication of the multiple Fourier series with which we must compute will in many cases give rise to a result that is substantially more complicated than either of the operands. Hence it is improbable that by overwriting the operands one would contribute significantly to the general balance of store during the multiplication procedure, and it was therefore decided to program the multiplication to conserve both the operands and the product. Nevertheless, experience of high-school algebra leads one to believe that on many occasions when it is desirable to form a product only part of the result is of particular interest. In any form of approximation procedure it is common to neglect in a result terms whose order is suitably large. Frequently we require to expand a function using Taylor's Theorem neglecting terms above a prescribed order. It would indeed be unfortunate if we should require to compute a result that could be easily stored but were unable to obtain it because it formed a part of another, more extensive, expression that we had no room to store.

In order to try to minimize the chance of this occurrence each manipulative subroutine that involves multiplication occurs in two forms. The first form gives the exact and complete result of the manipulation while the second form gives a restricted result. If the restricted routine is called for then the user must specify on entry to the routine the class of term that is required in the result. When this is done a dynamic check is incorporated into the manipulation and terms that are not required in the result are removed. It is most important to note that in this case no attempt is made to store the complete expression that would result from the use of the unrestricted routine.

The kind of restriction that may be imposed, of course, depends upon the particular piece of manipu-

lation being undertaken, but in all cases it is possible to restrict a result, including only those terms whose degree in some specified polynomial variables is suitably small, and consequently the scheme is able to undertake analytic calculations using approximation methods with great convenience. It is the author's view that the inclusion of the restricted routines has made the whole scheme usable and a great deal more powerful than it would otherwise have been.

It has been previously remarked that the basic subroutines are those that perform multiplication and addition. The addition subroutine must overwrite its operands, and consequently reduces to a subroutine to merge two ordered lists of periodic terms, or simply a sorting program. When two periodic terms are found having the same phase and periods then combination occurs, the free list is increased and the polynomial coefficients must be added. This addition again reduces to an elementary sorting procedure and again leads to cancellation. There are no difficulties in the coding and the available free space *cannot* decrease. Similar remarks apply to subroutines to perform formal differentiation and integration with respect to one variable.

The multiplication procedure is a little more complicated and proceeds as follows. To form the product $\alpha\beta$ where $\alpha$ and $\beta$ are multiple Fourier series of the form given in equation (1) the routine will take the first periodic term of $\alpha$ and multiply it in turn by each periodic term in $\beta$. As each pair are multiplied they are linearized to give two periodic terms of the product. At this stage the restricted form of the multiplication subroutine imposes its dynamic check upon the periods of these new terms to ensure that they meet the user's requirements, if not then they are abandoned and the multiplication continues. Provided that the periodic terms are retained their polynomial coefficients are next multiplied term-by-term and another dynamic check included on the resulting exponents in the case of the restricted product. Each polynomial term produced is added to the polynomial result so far obtained, by using a sorting program, and ultimately the complete product is associated with the two periodic terms by means of pointers. These periodic terms are then sorted into the sum of such terms so far, and the product continues.

Two sorting techniques are employed. An elementary merging program is used to sort the polynomial products since these appear in order to some extent, while a tree sorting technique is employed to sort the periodic terms since these are generated at random.

The remainder of this paper will be devoted to a more detailed specification of the several routines that make up the manipulative scheme and to some examples of their use.

## The manipulative routines

The routines that destroy their operands and hence cannot decrease the available free space are as follows. (We shall use $\alpha$, $\beta$ and $\gamma$ to denote multiple Fourier series in the standard form throughout.)

1. $\alpha := -\alpha$.
2. $\alpha := k\alpha$, $k$ is a rational constant.
3. $\alpha := \int \alpha dx_i$, no constant of integration is included.
4. $\alpha := \dfrac{d\alpha}{dx_i}$.
5. $\alpha := \int \alpha dy_i$. Those periodic terms that depend upon the harmonic variable $y_i$ are integrated with respect to that variable and their sum forms the first part of the result. Those terms that do not depend upon $y_i$ are collected together and their sum forms a subsidiary result. It is the responsibility of the user to take suitable action with such terms.
6. $\alpha := \iint \alpha dy_i dy_i$. This routine was included since repeated use of routine 5 causes the result to be sorted twice unnecessarily while routine 6 leads to no reordering, and hence it is faster.
7. $\alpha := \dfrac{d\alpha}{dy_i}$.
8. $\alpha := \dfrac{d^2\alpha}{dy_i^2}$.
9. $\alpha := \alpha + \beta$.
10. $\alpha := \alpha - \beta$.

Those routines that do not destroy their operands and may make substantial demands upon the available free space are as follows.

11. $\gamma := \alpha\beta$. This routine forms the exact product.
12. $\gamma :=$ the restricted product $\alpha\beta$.

When calling this routine the user may arrange to include in the result only those terms whose periods satisfy any condition that he may care to impose and also, similarly, to include only those terms whose exponents satisfy some similar condition.

13. $\gamma := \alpha(x_0, \ldots, \beta, \ldots, x_7 ; y_0, \ldots, y_5)$.

The multiple Fourier series $\beta$ is substituted for a polynomial variable into the expression $\alpha$.

14. $\gamma :=$ the restricted form of
$$\alpha(x_0, \ldots, \beta, \ldots, x_7 ; y_0, \ldots, y_5).$$

The restrictions that may be imposed upon the terms included in the result of this manipulation are considerably less general than is possible with routine 12. It is only possible to impose conditions of the form {total degree in specified polynomial variables} $< K$ where $K$ is some given constant.

15. $\gamma := \alpha(x_0, \ldots, x_7 ; y_0, \ldots, x, \ldots, y_5)$

where $x = (n_0y_0 + \ldots + n_5y_5) + \beta$ and $n_i$ are integers; $i = 0, \ldots, 5$.

This substitution is undertaken upon the assumption that $\beta$ is a small quantity and that Taylor series expansion in terms of $\beta$ is permissible. An example may help to clarify the action of the routine.

Let us assume that $\alpha = \sin(y_0 + 3y_1)$, $\beta = x_0 \sin y_2$ and $x = (y_0 + y_2) + \beta$. Then the result of using this routine to substitute $x$ for $y_0$ would be to calculate the expression

$$\sin(y_0 + 3y_1 + y_2)\left[1 - \frac{\beta^2}{2!} + \frac{\beta^4}{4!} - \ldots\right] + \cos(y_0 + 3y_i + y_2)\left[\frac{\beta}{1!} - \frac{\beta^3}{3!} + \ldots\right],$$

and finally to reduce it to the standard form. The user must specify on entry to the routine at what point the Taylor series in $\beta$ are to be terminated.

16. $\gamma :=$ the restricted form of

$$\alpha(x_0, \ldots, x_7 ; y_0, \ldots, x, \ldots, y_5).$$

Again the only condition that may be imposed on the terms that are included in the result of this manipulation must be of the form given for routine 14.

An input and output routine has been written for use with the scheme that will read and output the multiple Fourier series in a coded form that closely resembles the manner in which such expressions are usually written on paper. A program is available to evaluate numerically any expression given numerical values for the fourteen variables. A number of other housekeeping routines and filing facilities are also available.

**Examples of the use of the scheme**

It is not intended to describe here the details of any extensive calculation that has been undertaken using the scheme, but rather to present two examples of the manner in which the scheme may be employed to solve elementary problems simply by repeated approximation. Details of manipulations involving thousands of individual terms that have been successfully undertaken using the scheme have appeared elsewhere (Barton, 1966).

Let us consider the Kepler equation

$$l = E - e \sin E \qquad (2)$$

and see how it is possible to solve the equation to give $E = E(l)$.

It is well known that the solution to this equation is given by

$$E = l + 2\sum_{n=1}^{\infty}\left[\frac{J_n(ne)}{n}\right]\sin nl. \qquad (3)$$

However, we may construct the solution using the manipulative scheme in a very direct manner by the method of repeated approximation. The first approximation to the solution that is accurate to order zero in $e$ is clearly $E = l$. Substituting this result into the function $e \sin E$ using routine 16 and retaining terms of order $e$ we obtain the next approximation $E = l + e \sin l$. Again substituting this into $e \sin E$ and retaining terms of order $e^2$ we obtain $E = l + e \sin l + \frac{e^2}{2} \sin 2l$ and so on. The program required to perform this operation is evidently very simple and uses only the substitution routine and the addition routine to preserve the result so far. However, the result obtained is the analytic solution to the Kepler equation known correctly to some specified order in the variable $e$. The above example serves to illustrate a technique that has been found useful in many other applications. Using the manipulative scheme it has been possible to solve by repeated approximation simultaneous sets of equations similar to the Kepler equation when the right-hand sides were substantially more complicated than that of equation (2) and when no solution of the form given in (3) was known.

Our second example concerns the solution of the Mathieu Equation

$$y'' + (a - 2\alpha \cos 2\theta)y = 0$$

where $a$ is constant, $\alpha$ is a small parameter and the primes denote differentiation with respect to $\theta$. To construct periodic solutions to this equation is of course an eigenvalue problem, and we shall show how to construct these eigenvalues and the associated periodic solutions using an iterative algebraic technique on the computer. Now it is evident that a periodic solution to the equation exists in the form

$$y = \cos m\theta + O(\alpha),$$

$$a = m^2 + O(\alpha),$$

where $m$ is an integer.

Let us suppose that we have determined the solution accurately to the $n$th order in $\alpha$ and that this solution is $y_n$ and the approximate eigenvalue corresponding to the solution is $a_n$. To advance the solution to order $n + 1$ try $y = y_n + \epsilon$, $a = a_n + \eta$. Then clearly we must have

$$\epsilon'' + m^2\epsilon + \eta \cos m\theta = K_{n+1} \qquad (4)$$

where

$K_{n+1} =$ terms of order $n + 1$ in $-y_n\{a_n - 2\alpha \cos 2\theta\}$.

In order to avoid resonance that would give rise to a nonperiodic solution we must clearly choose $\eta$ to be the coefficient of the term in $\cos m\theta$ in $K_{n+1}$ and hence obtain the next approximation to the eigenvalue. It is then a simple matter to write a subroutine to obtain a particular integral to the resulting equation for $\epsilon$ and so

obtain the next approximation to $y$. The process may then be repeated.

It will be obvious that the examples given above make extensive use of the restricted forms of the multiplication and substitution programs, and in all the work that has been undertaken with the scheme these restricted forms have been more frequently used than the actual exact routines themselves.

**Acknowledgements**

The author would like to express his thanks to the Director of the University Mathematical Laboratory for the use of the Titan computer on which all the work has been done, and to Miss J. M. Wright who coded the input routine and the routine to perform the numerical evaluation of the expressions that are manipulated.

**References**

STRACHEY, C. (1965). "A general purpose macrogenerator", *The Computer Journal*, Vol. 8, p. 225.
BARTON, D. (1966). "Lunar disturbing function", *The Astronomical Journal*, Vol. 71, p. 438.
WOODWARD, P. M., and JENKINS, D. P. (1961). "Atoms and Lists", *The Computer Journal*, Vol. 4, p. 47.

# Correspondence

*To the Editor,*
*The Computer Journal.*

Sir,

**On finding the eigenvalues of real symmetric tridiagonal matrices**

By A. J. FOX and F. A. JOHNSON

*The Computer Journal* has published such significant papers on the Eigenvalue Problem that each new article on this topic is likely to arouse widespread interest. Since I consider the above paper to be misleading I hope you will permit me to make some rather critical remarks.

The Sturm Sequence method (SS) can be made completely reliable and as accurate as the word length permits. The code is brief and involves no *ad hoc* decisions, but it is rather slow. Almost any alternative is faster and several people have concocted rival algorithms. However, the increase of speed was usually bought at the cost of reliability, accuracy, or marked growth of the code.

Criticisms which I make might be considered academic (in the worst sense) were it not for the fact that there already exist published algorithms which are by no means optimal but which are superior to the algorithm proposed by the authors. Perhaps my chief criticism is that even after reading the references which they cite the authors persist in seeing $LL^t$ and QR, not as alternatives, but as supplements to the Sturm Sequence method.

Wilkinson has remarked that "often comparatively minor changes in the details of a practical process have a disproportionate influence on its effectiveness". To any theoretical method there correspond many computer implementations, some careful, some naive. In a comparison of performances who knows which has been used? Although FORTRAN and ALGOL serve well enough to define the order and arrangement of a calculation they are clumsy in treatment of underflow, overflow, and intermediate double precision.

The best of methods when poorly programmed can become a useless algorithm. Surely the following questions should nag any student of comparative algorithmics.

(*a*) Have I implemented the methods properly? Or am I comparing a brilliant realization of one method with a dim caricature of another?

(*b*) Are my comparisons fair? Have I unwittingly loaded the dice (parameters) in favour of one? Are my tests broad enough?

In my opinion these questions did not bother Fox and Johnson sufficiently. Consequently their results are misleading; different realizations have produced quite opposite results.

Let us examine a few aspects of the paper in some detail.

1. *Choice of parameters*

Fox and Johnson remark that the errors in their answers were usually about $10^{-8}$. In Table 2 there occurs an error of $10^{-7}$ (middle eigenvalue should be 0·42773454). This is easily dismissed or overlooked, yet it is a clue. To what?

The authors replace $b_i^2$ by 0 whenever

$$b_i^2 < \epsilon = 10^{-10}$$

This is equivalent to suppressing $b_i$ whenever

$$|b_i| < \epsilon^{1/2} = 10^{-5}$$

and this can cause changes in the eigenvalues up to $10^{-5}$.

Why did the authors not find errors of $10^{-5}$? The bound is certainly a realistic one.

The answer is given in a recent result of Kahan: for an $n \times n$ symmetric tridiagonal matrix if

$$\frac{b_n^2}{|a_n - a_{n-1}|} < \epsilon \text{ and } \frac{b_{n-1}^2}{|a_n - a_{n-1}|} < \epsilon$$

then $|\delta\lambda| < 3\epsilon$. Here $\delta\lambda$ is the change produced in any eigenvalue $\lambda$ by suppressing $b_n$.

On most of the matrices tested by Fox and Johnson $|a_n - a_{n-1}|$ was eventually greater than $10^{-2}$ and so their criterion happened to produce errors $< 3\,\epsilon/10^{-2} \doteqdot 10^{-8}$. However, in the case cited above the eigenvalue separation and hence $|a_n - a_{n-1}|$ was $10^{-3}$ and another iteration would have produced more accuracy. On the other hand for well-separated eigenvalues their criterion would provoke unnecessary iterations.

In the absence of Kahan's result the authors should have either (*a*) set $\epsilon = 10^{-16}$ and guaranteed 8 decimals, or (*b*) stated that with $\epsilon = 10^{-10}$ the user can only be sure of 5 decimals.

Now (*b*) could be disastrous if eigenvectors were desired as well and (*a*) would alter the time comparisons with other methods. See below.

(*continued on* page 420)