SCIENTIFIC APPLICATIONS OF SYMBOLIC COMPUTATION

Anthony C. HEARN*

Abstract : This paper reviews the use of symbolic computation systems for problem solving in scientific research. The nature of the field is described, and particular examples are considered from celestial mechanics, quantum electrodynamics and general relativity. Symbolic integration and some more recent applications of algebra systems are also discussed.

FEBRUARY 1976

76/P.817

* Centre de Physique Théorique, CNRS Marseille
and the University of Utah

POSTAL ADDRESS : Centre de Physique Théorique - C.N.R.S.
31, Chemin Joseph Aiguier
F-13274 MARSEILLE CEDEX 2 (France)

SCIENTIFIC APPLICATIONS OF SYMBOLIC COMPUTATION

Anthony C. Hearn

C.N.R.S., Marseilles, France
and the University of Utah

ABSTRACT

This paper reviews the use of symbolic computation systems for problem solving in scientific research. The nature of the field is described, and particular examples are considered from celestial mechanics, quantum electrodynamics and general relativity. Symbolic integration and some more recent applications of algebra systems are also discussed.

1. INTRODUCTION

The ability of the computer to perform algebraic and more general symbolic computations has been exploited by researchers in several fields for over a decade now. As a result, previously intractable problems are now solved routinely often by fairly elementary algebraic systems. This paper is intended as a review of some of these successful applications in order to acquaint the general reader with the potential of today's available systems as tools for scientific problem solving. However, as there are now over 500 papers which consider some aspect or application of symbolic computation, this particular paper cannot itself hope to be a complete review of the field. So examples will be chosen which illustrate some important aspectsof the subject, and references will be made to published reviews in particular areas so that the reader can fill in the gaps. As a starting point in this direction, a review which comes closest to covering the whole applications area is that of Barton and Fitch (1) to which we shall often refer.

The plan of this paper is as follows. In Section 2 we shall discuss the nature of symbolic computation so that the reader unfamiliar with this area can understand the scope of the subject. Particular applications are considered in some detail in Section 3 , and we conclude in Section 4 with a discussion of future trends as predicted from today's research in the area.

2. THE NATURE OF SYMBOLIC COMPUTATION

In order to introduce the nature and scope of symbolic computation, let us begin by comparing a simple FORTRAN calculation with a similar algebraic one. The example to be used is the computation of Legendre polynomials by the usual recurrence relation, for which the following FORTRAN program segment could be used :

```
      DIMENSION P(11)
      P(1)=1
      P(2)=X
      DO 10 N=3,11
      P(N)=((2*N-3)*X*P(N-1)-(N-2)*(P(N-2))/(N-1)
      N1=N-1
   10 WRITE (6,20)N1,P(N)
   20 FORMAT (3HOP(,I1,4H)=,F5.1)
```

Given that X is assigned a value in the program, and apart from a possible change in the PRINT statement, this program will run correctly under most FORTRAN systems, and produce output for, say, X = 2.0 of the form :

```
P(2)=  5.5
P(3)= 17.0
   ...
```

On the other hand, if we forgot to give X a value, then our program would terminate with an error. However, suppose our programming system was sufficiently general that it could represent the variable X as an undetermined symbol and carry out polynomial operations in the unknown X as it iterates through the loop. Then with an appropriate change in the format statement our results might come out like this :

```
P(2)= 3/2*X**2-1/2
P(3)= 5/2*X**3-3/2*X
   ...
```

This example shows one of the basic differences between a numerical calculation using FORTRAN, for example, and an algebraic calculation in which indeterminates such as X with no pre-assigned numerical value can appear, and expressions themselves have algebraic rather than numerical values. Furthermore, by the use of integer and rational number arithmetic, the results are exact rather than approximate. Systems exist which can do such polynomial manipulation using either a FORTRAN or ALGOL-like syntax, with of course additional declarations and syntatical constructs to handle the

more complicated data-types. For example in ALTRAN (2), a FORTRAN embedded system, the above program segment might take the form :

```
ALGEBRAIC (X:10) ARRAY (0:10) P
INTEGER N
P(0)=1
P(1)=X
DO N=2,10
   P(N)=((2*N-1)*X*P(N-1)-(N-1)*P(N-2))/N
   WRITE P(N)
DOEND
```

What else can these systems do ? At the other end of the spectrum, we have the following possible expression (ZE being e)

```
INTEGRATE (X * ZE ** X/(X+1) ** 2, X) ,
```

which if given to the algebraic system MACSYMA (3) will result in the printed result

$$-\frac{ZE^{X}}{X+1} \quad .$$

Between these two extremes of simple polynomial manipulation on the one hand and the sophisticated analytic integration of complicated expressions on the other there exists a wide range of facilities now offered by algebraic systems. Many of these facilities will be described as we consider various application examples of the available systems. One might ask whether symbolic computation is limited to algebraic simplification. After all, parsing and compilation of a FORTRAN program are symbolic operations, so are they included ? The consensus on this point is that they are not and that one limits symbolic computation to techniques which solve problems outside of computer science, rather than say program compilation which is a problem of computer science itself. However, even this definition includes subjects other than algebraic manipulation such as analysis of graphs, which we shall touch on briefly later. However, our main concerns will be with algebraic manipulation.

Our discussion has mentioned the existence of algebraic systems, and it is true that if you want to do such calculations at present, you must use a special program for this purpose. One might therefore ask why symbolic computation systems cannot be made available as subroutines to be loaded into a standard FORTRAN job, so that they can be used as needed as

part of a combined numerical and algebraic calculation. There are several reasons why this hasn't happened. First, although important systems such as ALTRAN are based on FORTRAN, a user cannot write an efficient numerical program which includes a symbolic computation step in it because FORTRAN is normally a compile, load and go system. Therefore, since one generates algebraic expressions during the go step, one has no way of compiling them into efficient code in the same job step. The only way to do this is to pass such expressions on to another job or job step, at which point the whole purpose of doing the algebraic calculation in a numerical system is lost. In fact, the main reason such systems are implemented in FORTRAN is to provide for portability and standardization or to utilize the numerical subroutine library and I/O capabilities of the language. However, there is to be a growing feeling among algebraic system designers that such systems should provide more efficient numerical computations than they now allow, thus possibly bypassing the use of numerical programming systems entirely. To alleviate this problem for the time being, most available systems have the ability to produce output in a FORTRAN compatible format which can then be used in a later job or job step for numerical calculations.

Another deterrent to using a purely numerical programming system for symbolic calculations is that the data structures used in these calculations, such as polynomials and power series, are quite different from those of numerical computation where integers and floating point numbers are the principal concern. Algebraic data structures clearly vary in size considerably during expression evaluation and therefore the storage management can be quite complicated, although the techniques for doing this are well understood (4). Unfortunately, FORTRAN does not provide the necessary tools for doing this in an efficient manner, and so they must usually be provided in assembly language. Many systems have therefore bypassed the use of numerical systems for embedding and utilized languages with storage management facilities already built in, such as LISP (5), or the majority of the system has been written in assembly language.

So we are dealing with systems or programs essentially independent from numerical systems but with languages similar in form to their numerical counterparts. One notable exception to the latter statement is SCRATCHPAD (6) which has incorporated a completely new language design which is intended to provide a more natural mathematical notation for problems. For example,

in SCRATCHPAD, we could compute and print the first eleven Legendre polynomials by the statements :

```
p<0>=1
p<1>=x
n in (2,3,...,10)
p<n> = ((2*n-1)*x*p<n-1> -(n-1)*p<n-2>)/n
n in (0,1,...,10)
p<n>
```

Other examples of programming styles in algebraic systems are given in Ref. (1).

The computational characteristics of symbolic calculations are also quite different from numerical calculations. Roundoff is not a problem since, as was mentioned earlier, exact results may be obtained. The stability and convergence problems of numerical approximations are also not relevant. On the other hand, symbolic computation has its set of own difficulties. A common one in practice is that even though most calculations start from short expressions, the expressions formed during intermediate stages of the calculations are often quite large even if the answer isn't, and may in fact exceed the available storage, aborting the computation. This problem of intermediate expression swell as it has been called remains of constant concern in symbolic calculations, and often requires much innovation to overcome. Another difficulty is the fact that many of the necessary algorithms, such as those for polynomial factorization, have potentially exponential growth rates associated with them. Much of the theoretical work in the field during the past five years has been devoted to overcoming such problems. Considerable progress has in fact occurred, the benefits of which should be passed on to users in the near future.

A third problem relates to the fact that it is only possible theoretically to put upper bounds on the computation times associated with symbolic computations, unlike numerical calculations where it is often possible to determine average computing times. To put this problem in the right perspective, consider the evaluation of a symbolic determinant of order 10. If every element of the matrix is distinct, the determinant will have in general 10!, or over 3,000,000 terms in the answer. Apart from being incomprehensible, such a result would take a nontrivial time to derive. However, most scientific problems have various symmetries in their specification which put constraints on the form of the matrices with which one works. For example, the matrix may

in fact be singular, and the recognition of this may take an algebra program negligible time. To get the value 0 for such a determinant may have very important consequences in a calculation, and so a researcher may attempt to evaluate the determinant for this reason. However, he must be prepared to get an answer as large as 3,000,000 terms if his guess isn't right, and so the estimate of the time required to discover whether the determinant is really zero may be prohibitive in the worst case, although in practice such an upper bound may rarely be reached.

With these thoughts in mind, we can now look at some of the typical calculations which have been completed in this area, and use these to gain further insight into the nature of symbolic computation.

3. SPECIFIC APPLICATIONS

A perusal of essentially all successful applications of symbolic computation so far reveals that they involve problems solved by means of perturbation theories. Hand calculations which can be tractably taken to a given order are then extended one or two more orders by computer. Because of the exponential nature of such techniques, at this point expressions grow so large using current systems that the computational time and sheer incomprehensibility of the output limit any further extension. In addition, with a few notable exceptions, most of the calculations have been an adjunct to a numerical calculation. In other words, part of the calculation was done algebraically and these results then used to determine a final numerical solution for the problem.

Consider for example the field of celestial mechanics where researchers have made much use of algebraic systems in the past decade. Here the most tedious task to be faced is in the construction of an analytic perturbation theory ; in other words a formula for calculating the position or rotational properties of a body at any instant in time. The body might be the Moon, a planet or more topically an artificial satellite or space station. To construct such theories requires solving the differential equations which describe the motion of the body. Since solutions rarely exist in closed form, one solves them by a technique of repeated approximation and expresses the solution as a power series in terms of the small parameters of the theory, such as the body eccentricities.

The most famous such calculation is surely Delaunay's work in

the Lunar Theory (7), in which the coordinates of the Moon are computed to the seventh order in the small quantities. Like most hand calculations of this type, this investigation involved working with thousands of terms. In Delaunay's case the computations took 20 years. When Deprit, Henrard and Rom (8) repeated this calculation by computer in 1970, they found only one error in Delaunay's computation of the secularized Hamiltonian up to the ninth order, a major part of the work ; certainly a great tribute to Delaunay's skill and dedication.

What types of manipulation are involved in these calculations ? The most frequently considered series in celestial mechanics turn out not to be power series, but Poisson series, which have the form :

$$Q(\underset{\sim}{x}, \underset{\sim}{y}) = \sum_{\underset{\sim}{j}} P_{\underset{\sim}{j}} \cos(\underset{\sim}{j} . \underset{\sim}{y}) + Q_{\underset{\sim}{j}} \sin(\underset{\sim}{j} . \underset{\sim}{y}) , \qquad (1)$$

where $\underset{\sim}{x}$ is a vector of polynomial variables, $\underset{\sim}{y}$ a vector of trigonometrical variables and $\underset{\sim}{j}$ is a vector of integers. $P_{\underset{\sim}{j}}$ and $Q_{\underset{\sim}{j}}$ are polynomials, indexed by $\underset{\sim}{j}$, in the polynomial variables $\underset{\sim}{x}$ whose coefficients may be rational or floating point numbers. It is clear that by the use of the trigonometrical identities

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

such series are closed under the operations of addition, subtraction, multiplication and differentiation. By making suitable assumptions about the size of the parameters in the series, we can also define operations of division, integration and substitution which do not lead outside this class. Finally, an important theoretical consideration is that Poisson series can be written in a unique canonical form, in other words a form to which all equivalent expressions can be uniquely reduced. This form can then be used to define the simplification of Poisson series expressions. This means we can avoid many of the problems associated with general expression manipulation which we shall return to later. Such problems arise for example in recognizing that

$$\log \tan(x + \tfrac{1}{4}\pi) - \sin h^{-1} \tan 2x = 0 \qquad (3)$$

The importance and mathematical simplicity of Poisson series has led to the construction of many systems for their manipulation, including

that of Rom (9) used in repeating the Delaunay calculations, Jeffrey's system TRIGMAN (10) which is written in FORTRAN and is fairly portable, and CAMAL (11) which has also been used extensively for calculations in general relativity, is well documented and easily obtainable. Further examples are given in the reviews by Jeffreys (12) and Barton and Fitch (1).

To see how Poisson series arise in a natural way in celestial mechanics calculations, let me repeat Barton and Fitch's discussion of the computation of the series solution to Kepler's equation by repeated approximation, which is the starting point for many of these calculations. We are interested in obtaining the solution for E, as a function of u and e, of the implicit equation

$$E = u + e \sin E \tag{4}$$

where e is to be regarded as a small quantity, typically the eccentricity of an elliptic orbit. Although the problem is capable of a formal solution in terms of Bessel functions, since we normally require the solution correct to a given order in e (say tenth) it is computationally easier if we adopt a repeated approximation procedure. From the Kepler equation, it is obvious that $E = u$ to zero order in e. Then, supposing that $E = u + A_k$ is the solution correct to order k in e, we can easily see that A_k satisfies the equation

$$A_{k+1} = e \sin(u + A_k) , \tag{5}$$

where the right hand member of Eq (5) is to be taken to order $k+1$ in e. Thus an approximation algorithm may be stated as follows :

$$E = u + \lim_{n\to\infty} A_n$$

where

$$A_0 = 0$$
$$A_{k+1} = \left[e \sin u \left\{ 1 - \frac{A_k^2}{2!} + \frac{A_k^4}{4!} - \ldots \right\} + e \cos u \left\{ A_k - \frac{A_k^3}{3!} + \ldots \right\} \right]_{k+1} \tag{6}$$

The outer brackets in the above expression indicate that terms of degree greater than $k+1$ are to be ignored in the calculation. By using the trigonometrical relations given in Eq (2), we can express the result of this computation as a linear expression in the trigonometrical functions ; that is in the form of a Poisson series.

This computation is clearly only a small part of say the complete

Delaunay result, but does illustrate the types of expressions to be manipulated. Barton and Fitch (1) go on to consider the steps involved in the remainder of the Delaunay calculation, as well as looking at other examples of the application of Poisson series processors. The reader is encouraged to study these calculations in order to gain some idea of the sheer enormity of the computations involved.

The second application I would like to consider is in the field of quantum electrodynamics (or QED) , an area where one tries to understand the detailed interactions of electrons and photons by means of relativistic quantum mechanics. The most commonly used calculational method is due to Feynman (13) who showed that one could develop a perturbation theory of such interactions with an expansion parameter of about 1/137 (the so-called fine structure constant). The starting point for this method is a diagrammatic representation of the particular process under study.

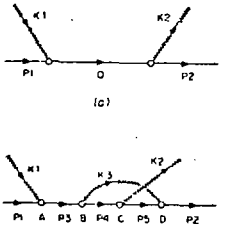

Fig.1 Possible interactions between electrons and photons.

For example, in Figure 1 we have two such diagrams which occur in the representation of the process which occurs when a quantum of radiation (that is, a photon) interacts with an electron. From these diagrams, Feynman developed a method of calculating the contribution of any given diagram to the determination of a physical quantity. The order of perturbation is related very simply to the number of vertices in a diagram. Therefore since the most important contributions come from the simplest diagrams one can quickly get a rough approximation to the answer. However, as the accuracy of experiments has increased, the need for more and more precise calculations has required more and more complicated diagrams to be studied. To understand the complexity of such calculations it is necessary to look in some detail at the way they are done.

Feynman's method gives you a one-to-one mapping between diagrams like those in Figure 1 and expressions like

$$\sum_{\mu,\nu} \text{Trace}\{(\gamma.p2+M)\gamma_\mu(\gamma.p2 + \gamma.k2 +M)\gamma_\nu (\gamma.p1+M)\gamma_\nu (\gamma.p2 + \gamma.k2 + M)\gamma_\mu\} , \quad (7)$$

which, apart from some simple multiplicative factors, is the contribution of Figure 1(a) to the experimentally measurable cross-section. The rules for this mapping are quite straightforward and may be found in any suitable textbook on relativistic quantum mechanics, e.g. (14).

In the above expression, γ_μ represents a set of four 4 X 4 noncommuting matrices known as Dirac matrices which satisfy the relation

$$\gamma_\mu\gamma_\nu + \gamma_\nu\gamma_\mu = 2\delta_{\mu\nu} . \quad (8)$$

The Greek indices range from 1 to 4 , and $\delta_{\mu\nu}$ is the Kronecker delta function defined by

$$\delta_{\mu\nu} = \begin{cases} 1 & \text{if } \mu = \nu \\ 0 & \text{if } \mu \neq \nu . \end{cases} \quad (9)$$

M is the electron mass multiplied by a unit 4 X 4 matrix, and $p1$, $p2$, $k1$ and $k2$ are four-component vectors representing the electron and photon momenta as in Figure 1 . Thus $\gamma.p1$ represents the sum

$$\gamma_1 p1_1 + \gamma_2 p1_2 + \gamma_3 p1_3 + \gamma_4 p1_4 . \quad (10)$$

However, because of the relativistic invariance of the result, one rarely has to study the individual components of the vectors, but only their scalar products, so calculations are less complicated than they might be.

Thus the result of taking the trace of the 4 X 4 matrix expression in the curly brackets in Eq (7) and summing over the tensor indices is a scalar. In principle, this computation is straightforward. Equation (8) implies that the trace of a product of an odd number of gamma matrices is zero, and the trace of the product of 2n such matrices may be expressed in terms of the trace of 2n-2 products by a simple recurrence relation, given for example in Ref (14) . However, a naive calculation can be disastrous. For example, the trace of a single product of 12 gamma matrices generates over 10,000 terms, many of which prove to be equal when contraction over the tensor indices is taken. This problem has motivated the development of more sophisticated algorithms, most of which involve summation over the

repeated tensor indices before taking the trace. A description of one such algorithm, due to Kahane (15) , may be found in Ref (1) . Fairly complete reviews of the symbolic calculations which have been done in QED may be found by studying Refs (16), (17), (18) and (19).

From a computational point of view, such calculations involve quite straightforward though lengthy symbol manipulation. Like Poisson series, the rules for simplification are well defined although a little more exotic than those for polynomial manipulation. In addition, one needs a mechanism for substitution for variables and simple expressions. Although denominators occur in some expressions, rational algebra is not necessary, because the denominators have a simple structure and can therefore be replaced by variables in the numerator. The most widely used systems for such calculations have been REDUCE (20) and SCHOONSCHIP (21). The former is a LISP-based system which is also well suited to more general calculations than just QED and has a large number of published applications in many fields to its credit. The latter system is written in COMPASS for the CDC 6000 and 7000 series computers and also has a long application list, though all essentially in QED. A third system of some interest is a FORTRAN based program ASHMEDAI (22), which has some important calculations to its credit.

To give one a feel for the scope of current calculations in this area, it is worth looking at the most important such calculation undertaken during the past few years. This is undoubtedly the computation of the sixth order corrections to the anomalous magnetic moment of the electron. The analytic form for the fourth order contribution was completed by hand in 1957 (23). Because of the sheer magnitude of the calculation, the sixth order result, which involves looking at 72 diagrams, has been completed by a group effort involving many different independent researchers. To be sure that the result is correct, each diagram has by now been computed independently by two different groups, usually with different algebra programs. The complete answer is still not known analytically because some of the integrations involved were performed numerically.

The need for integration in these calculations is apparent if we look again at Figure 1 (b) . An electrical analogy is often used to explain the problem ; just as current flows through an electrical circuit, momentum flows through these diagrams. If we assume that the initial momenta are known, then we know the momentum in line AB by momentum conservation at

vertex A by analogy to current conservation in the electrical case. However, at vertex B the momentum splits into two paths, so we no longer know the values of the individual momenta in lines BC and the wavy line BD but only their sum. So an integration over this unknown momentum is necessary to get the required result.

The integrals which arise in these calculations are messy and tedious, rather than being sophisticated, as one usually starts with rational functions. However, as integration over several variables is required, the results come out in terms of generalized Spence functions in most cases. Examples of such integrations are given in Ref (24) and (25), which describe programs in SCHOONSCHIP and REDUCE for evaluating integrals of the form

$$I(k,m,n,\lambda,\delta) = \int_0^1 dy \int_0^1 dx \int_0^1 d\omega \int_0^1 dz \; z^n \omega^\delta x^{2k-3+n-m} \times y^{k-2+n} \Delta^m / \beta^{n\lambda} \Gamma^k$$

where

$$\begin{aligned} \beta &= 1 - xy(1-\omega z + \omega^2 z^2) \\ \Delta &= 1 - x(1-\omega z(1-\omega)) \\ \Gamma &= \omega^2 x\beta + y\Delta^2 \end{aligned} \qquad (11)$$

which arise in some fourth order computations in QED. For example

$$I(1,0,2,2,3) = \frac{1}{6}\pi^2 - \frac{1}{3}\pi^2 \log 2 + \frac{1}{2}\zeta(3) + \frac{1}{3}, \qquad (12)$$

where $\zeta(3)$ is the Riemann zeta function. Currently, two groups (26, 27) are trying to find the whole sixth order result analytically by computer and considerable progress has been made. Although their methods are different, the procedure involves expanding expressions in partial fractions and integrating term by term. To give you an idea of the relative times involved in this work, Levine, Perisho and Roskies (26) give statistics for the analytic evaluation of one of the more difficult of the 72 graphs in the sixth order calculation. Their method involved trace evaluation, two straightforward integrations, a partial fraction expansion of the result followed by two further integrations. Using the ASHMEDAI program, the times on a Univac 1108 were 2 minutes for the traces and first two integrals, 18 minutes for the partial fraction expansion and 15 minutes for the two final integrations. It is interesting to note that the trace taking, the

original stumbling block for the hand calculations which motivated the development of such computer programs in the first place, is now only a small part of the whole computer calculation.

In fact, it is interesting to see that once the basic trace problem was solved, researchers in this area started looking at other ways of using symbolic computation in their work. Analytic integration is one such example. Another application involved looking at ways to actually generate and manipulate the diagrams which occur in a given problem. Once such a diagram is written down, it is a fairly simple procedure to generate the matrix elements which were the starting point of the previous discussion (28) . The most difficult problem here is to recognize topologically equivalent diagrams, but a recently developed program (29) seems to have solved this problem in an efficient manner. Some earlier programs are described in Ref (19). If one is ever interested in looking at eighth order calculations, where over 1000 diagrams contribute to the magnetic moment correction, such graphical programs should be essential even if only used as a bookkeeping device.

These two very successful applications of symbolic computation, celestial mechanics and quantum electrodynamics, are prototypical of a large number of the successes in this field. Both are problems with well-defined rules of simplification, even if the non-standard algebra was responsible for the early development of special purpose systems to handle this. Major application successes were already in evidence five years ago, and the work during the past five years has been concerned with extending the earlier successes to harder and harder problems, or tackling new problems, such as the integration example in QED. At the same time, problems in other areas such as fluid mechanics, plasma physics, electrical network theory and queuing theory, which were amenable to solution by polynomial or power series manipulation were being solved by systems such as ALTRAN (and its predecessor ALPAK), FORMAC (30) and REDUCE which possessed the necessary facilities for performing these tasks. A considerable number of applications of these programs have been described in the literature. Another convenient system for such calculations is Engeli's SYMBAL (31).

There was, however, another group of important calculations being carried out at the same time which involved the manipulation of more

general expressions than the examples so far. Since there is no canonical form for a sufficiently large class of expressions (32) the designers of algebra systems for such applications must either try to solve a very complicated simplification problem for which no clear cut solution exists, or provide sufficiently general substitution facilities that users can input their own set of rules for simplification. A particularly good example of such calculations, and one for which good reviews exist (1, 33) is general relativity, where the calculations considered are concerned with a very large set of elementary functions for which many complicated simplification rules exist.

Applications in this area have been mainly concerned with the field equations which express the relationship between the geometrical structure of the four dimensional space and the distribution of mass and energy within it. These field equations are very complicated even for physically simple cases and involve exponential and circular functions in their description. An account of the systems which have been used to solve such problems is given in the cited references. Among the most widespread in use are FORMAC, CAMAL and ALAM and its predecessors (34). Each of these systems, as is true for the other systems mentioned earlier in this paper, take a different approach to the problem of simplification, and an amusing review of this subject may be found in Ref (35).

Another application of algebraic manipulation involving general expressions which deserves consideration in its own right is the problem of analytic integration. This is different from our previous applications in that it is in mathematics rather than physics or engineering, and it is certainly not a perturbation problem. However its importance to the latter fields is obvious. Considerable progress has been made in solving this problem during the past decade and it is now possible to integrate a wide variety of expressions automatically. Most of the effort has been directed toward the problem of indefinite integration which we shall consider here, although some work on definite integration has also been reported (36, 37). Of course, given the result of an indefinite integral we can always find the equivalent definite integral by substitution. However, as is well known, closed form solutions exist for many definite integrals where the indefinite form does not, and special methods such as contour integration are applicable in these cases.

Although earlier programs existed, the first truly effective program in this area was Moses' SIN program, which in an improved form provides the basis for the MACSYMA integration facility, an example of whose use was shown earlier. The structure of this program is described in some detail in Moses' review of the field (38) . SIN works very much like a scientist when confronted with an expression to be integrated. It first of all tries a cheap general method which, if it succeeds, gives the result very quickly. This general method involves the so-called "derivative divides" test which determines by pattern matching whether the integral can be written in the form

$$\int c\, f(u(x))\, u'(x)\, dx .$$

If this test succeeds, then the integral can be immediately written down if the integral of f is known. This part of the program also expands the integral to its fullest extent, and then expresses the integral of a sum as the sum of integrals in order to reduce the complexity of the integrands. For example,

$$\int (\sin x + \cos x)^3 dx$$

would be written in the form

$$\int 3 \sin^2 x \cos x \, dx + \int 3 \cos^2 x \sin x \, dx + \int \sin^3 x \, dx + \int \cos^3 x \, dx ,$$

the first two terms of which satisfy the derivative divides test. If this general method fails, as for the last two terms in the above, special methods which are specific to a certain class of integrals are tried. For example, if the integrand contains trigonometrical functions, the substitution $t = \tan \theta/2$ would be attempted, if it is a rational function then the well known Hermite's method (39) would be used. Again, even though these methods are narrow in scope, they provide efficient solutions when they are applicable. In addition, the integration examples we mentioned in QED are of this type because they are amenable to special case algorithms and Hermite's method.

Clearly a large class of interesting integrals will not be evaluated by the above techniques and, at this point, an entirely different approach to the problem is taken based on an algorithmic approach to integration developed by Risch (40) . This particular method requires a radical transformation of the integrand ; for example, sines and cosines are converted

to complex exponentials. Since, as pointed out by Moses (38) , a user prefers to see his answer in terms of the same sort of functions as the problem was presented such transformations should only be used when there is no other more straightforward way to evaluate the integral.

The complete Risch algorithm is of extreme importance because it provides a procedure for deciding if an integral exists within a given class of functions. If the integral does exist, the algorithm finds it. If it does not exist, the algorithm states this. It is very important to understand the difference between such a decision procedure and the usual methods for integration which either find the integral or leave us *not knowing if the integral exists or not.* In other words the search for a "closed form" solution to an integration problem is a search for the form of the integral if it exists in the class of functions we are considering. The Risch algorithm can tell us this and in the affirmative case provide the result. This is, after all, all we can reasonably hope for in integration ; if the integral does not exist in the current class of functions under consideration a new function must be introduced anyway. Risch's algorithm is quite complicated, and beyond the scope of this paper to describe, but a readable account of it may be found in Refs (1) and (38). In broad terms, it is based on a generalization of Liouville's theorem, well known to every student of mathematical analysis, and uses a method similar to the Hermite algorithm to determine the parts of the integral if they exist in closed form. In principle, a computer program could now be written which provides for the indefinite integration of any expression in a class of functions sufficiently large to satisfy all physicists and engineers. However, there remain several theoretical and practical difficulties in the implementation of this algorithm and at present programs only exist for functions which are so-called monomial transcendental extensions of the rational functions (that is, exponentials or logarithms of the rational functions) or integrals over these (such as the error function or Spence functions). However, there is considerable current research in progress aimed at enlarging this class, and some results should be forthcoming shortly.

4. FUTURE TRENDS

It should be clear from the examples shown in this paper that symbolic computation can play a useful role in a wide range of scientific

applications, and there is no doubt that this range will increase as more and more researchers discover the possibilities inherent in this approach. However, I see three main problems which limit the applicability of today's available systems, but fortunately current research is seeking solutions to these, so that no doubt they will be overcome in the next few years.

The first problem is concerned with the production of large expressions. As was mentioned earlier, most symbolic calculations can only carry a hand perturbation calculation further by one or two orders at most. At that point, expressions become so large and unwieldy that not only does the computing time increase dramatically, but the results are thoroughly intractable. However, one of the reasons for the explosive growth of expressions is that many algorithms used internally by existing systems require expressions in an expanded form. So any structure in the input expressions, especially factor structure, is immediately destroyed by this expansion. However, as had been pointed out most succinctly by Brown (41) there are ways of consistently handling expressions in a factored form, and the use of these and similar ideas should shortly have an impact on the field as they are incorporated into available systems. The recognition of common sub-expressions in an expression and their renaming by a single parameter, or the deferred expansion of a function or variable in an expression can also be very important. Systematic ways of handling this problem are being studied by several groups, and some practical experience with such techniques in a plasma physics calculation has recently been published (42).

A second problem from the average user's point of view is that as programming systems, algebra programs are not always designed in the most transparent manner, and so the user is often frustrated in trying to understand what is going on in a calculation. Finally, the range of facilities offered by available systems is limited, and it is not easy for the average user to extend them in a straightforward manner.

There are now trends in computing methodology which can help solve these latter two problems and which will, I believe, cause symbolic computation systems to have an even greater impact in the years ahead. The consistent introduction of types at a very basic level, software portability, modularity and structured programming are all examples of such

developments. The result in the near future will be algebraic systems which have well designed modular construction, easily understandable programming language syntax and semantics and robustness and portability far in excess of today's systems. Efforts towards machine-independent implementations of such systems will find immediate use on the new inexpensive hardware which is being predicted.

Another point to consider is that symbolic computation and algebraic manipulation are now recognized as having much wider application than previously considered. For example, algebraic manipulation methods find an obvious place in such tasks as program verification (43) , optimization (44) and error analysis (45) . Although these applications are only just beginning, we should see more such activity in the next few years.

Another approach in the use of algebraic manipulation systems detectable during the past few years has been the increasing use of such systems for making a quick check of the validity of some physical idea which would otherwise be difficult or impossible to resolve in a reasonable time by hand. An example of such use was given in a paper by Gibbons and Russell-Clark (46) , which showed, using CAMAL, that a solution of Einstein's vacuum field equations put forward as a possible description of a black hole lacked several of the necessary properties. Another example was reported recently by Cohen (47) who used REDUCE to look for valid perturbation expansion techniques for solving a problem in fluid mechanics. In this case, he could learn in an evening at an interactive console whether a particular method was applicable by seeing if the secular terms were eliminated in second order. Such calculations would have taken many months by hand and would therefore probably have never been taken to that order. Calculations of this type are often not reported in the literature unless the results are dramatic ; the researcher accepts such techniques as another problem solving tool and uses them when appropriate. This is especially true for users with access to interactive systems like MACSYMA and REDUCE which encourage casual use because of the ease of access to the program. In batch oriented environments, this access is less easy, especially if a user has to see the results of one step before going on the next as was the case in the fluid mechanics calculation discussed above.

In the long-term, I believe that we shall see the emergence of computing systems which can provide algebraic manipulation at very low cost for a wide spectrum of users, including even high school and college students in addition to scientists. We should have programming systems sufficiently well designed and understood to support tasks far beyond our present capabilities. For example, the problems of simplification of general expressions should also be better understood in a few years. I believe that we shall then have in time a complete programmed solution of the classical indefinite and definite integration problems. Once such tasks are successfully accomplished, we shall begin to see systems developed which can provide formal solutions for more complicated problem representations such as integral and differential equations.

Even more interesting is the possibility that since many scientific problems are formulated in terms of fundamental conservation laws, we may be able to attempt solutions directly in terms of these laws rather than to follow the traditional practice of producing approximate solutions by linearization and perturbation methods. This is but one example of the possible impact of such work on scientific problem solving, and there are no doubt countless others which will emerge in the years ahead.

ACKNOWLEDGMENTS

The author's work was supported in part by the National Science Foundation under Grant No. GJ-32181. The figure was taken from Ref.(17) and reprinted by permission of the Association for Computing Machinery.

REFERENCES

(1) BARTON, D. and FITCH, J.P., *Rep. Prog. Phys.* 35 (1972) 235-314.

(2) BROWN, W.S., *ALTRAN User's Manual*, Third Edition, Bell Laboratories, 1973.

(3) BOGEN, R. et al., *MACSYMA Reference Manual*, Project MAC M.I.T. Cambridge, Mass., 1974.

(4) KNUTH, D.E., *The Art of Computer Programming, Vol 1, Fundamental Algorithms* and *Vol 2, Seminumerical Algorithms*, Addison Wesley, Reading, Mass. (1968).

(5) McCARTHY, J. et al., *LISP 1.5 Programmer's Manual* M.I.T. Press, Cambridge, Mass. (1965).

(6) JENKS, R.D., *The SCRATCHPAD Language, SIGPLAN Notices*, ACM, New York, 9 (1974)

(7) DELAUNAY, C., *Théorie du Mouvement de la Lune* (Extraits des Mém.Acad. Sci.), Mallet-Bachelier, Paris, 1860.

(8) DEPRIT, A., HENRARD, J. and ROM, A., *Science 168* (1970) 1569-1570.

(9) ROM, A., *Celest. Mech.* 3 (1971) 331-345.

(10) JEFFREYS, W.H., *Celest. Mech.* 2 (1970) 474-480.

(11) FITCH, J.P., *CAMAL User's Manual*, Computer Lab., Cambridge, U.K. (1975).

(12) JEFFREYS, W.H., *Comm. ACM* 14 (1971) 538-541.

(13) FEYNMAN, R.P., *Phys. Rev.* 76 (1949) 769-789.

(14) BJORKEN, J.D. and DRELL, S.D., *Relativistic Quantum Mechanics*, McGraw-Hill, New York, 1964.

(15) KAHANE, J., *J. Math. Phys.* 9 (1968) 1732-1738.

(16) CAMPBELL, J.A., *Comp. Phys. Comm.* 1 (1970) 251-264.

(17) HEARN, A.C., *Comm. ACM* 14 (1971) 511-516.

(18) HEARN, A.C., *Computer Solution of Symbolic Problems in Theoretical Physics, Computing as a Language of Physics*, IAEA, Vienna (1972) 567-596.

(19) CAMPBELL, J.A., *Acta Phys. Austriaca Suppl.* XIII (1974) 595-647.

(20) HEARN, A.C., *REDUCE User's Manual*, Second Edition, University of Utah, 1973.

(21) STRUBBE, H., *Comp. Phys. Comm. 8* (1974) 1-30.

(22) PERISHO, R.C., *ASHMEDAI User's Guide*, U.S.A.E.C. Rep. No COO-3066-44 (1975).

(23) PETERMANN, A., *Helv. Phys. Acta 30* (1957) 407-408.

(24) MAISON, D. and PETERMANN, A., *Comp. Phys. Comm. 7* (1974) 121-134.

(25) FOX, J.A. and HEARN, A.C., *J. Comp. Phys. 14* (1974) 301-317.

(26) LEVINE, M.J., PERISHO, R.C. and ROSKIES, R., *Would You Believe More Graphs for g-2 ?* , Univ. Pittsburgh Preprint PITT-153 (1975).

(27) BARBIERI, R., CAFFO, M. and REMIDDI, E., *Phys. Lett. 57B* (1975) 460-462.

(28) CAMPBELL, J.A. and HEARN, A.C., *J. Comp. Phys. 5* (1970) 280-327.

(29) SASAKI, T., *Automatic Generation of Feynman Graphs in QED*, Rikagaku Kenkyusho, Wako-Shi, Saitama, Japan, Preprint 1575.

(30) TOBEY, R.G. et al., *PL/1-FORMAC Symbolic Mathematics Interpreter*, SHARE Contributed Program Library, No 360 D-0.3.3.004 (1969).

(31) ENGELI, M., *An Enhanced SYMBAL System*, SIGSAM Bulletin, ACM, New York, No 36 (1975) 21-29.

(32) RICHARDSON, D., Ph. D. Thesis, Univ. of Bristol (1966).

(33) BARTON, D. and FITCH, J.P., *Comm. ACM 14* (1971) 542-547.

(34) D'INVERNO, R.A., *Comp. J. 12* (1969) 124-127.

(35) MOSES, J., *Comm. ACM 14* (1971) 527-537.

(36) WANG, P., *Symbolic Evaluation of Definite Integrals by Residue Theory in MACSYMA, Proc. IFIP Congress 74* (1974) 823-827.

(37) CAMPBELL, J.A., *Applications of Symbolic Programs to Complex Analysis, Proc. ACM Annual Conf. 72* (1972) 836-839.

(38) MOSES, J., *Comm. ACM 14* (1971) 548-560.

(39) HARDY, G.H., *The Integration of Functions of a Single Variable*, Second Edition, CUP, Cambridge, England (1916).

(40) RISCH, R., *Trans. AMS 139* (1969) 167-189.

(41) BROWN, W.S., *On Computing with Factored Rational Expressions, SIGSAM Bulletin*, ACM, New York, No 31 (1974) 27-34.

(42) KERNER, W. and STEUERWALD, J., *Comp. Phys. Comm. 9* (1975) 337-349.

(43) LONDON, R. and MUSSER, D.R., *The Application of a Symbolic Mathematical System to Program Verification*, Proc. ACM Annual Conf. *74* (1974) 265-273.

(44) STOUTEMYER, D., *Trans. Math. Software 1* (1975) 147-164.

(45) STOUTEMYER, D., *Trans. Math. Software* (to be published).

(46) GIBBONS, G.W. and RUSSELL-CLARK, R.A., *Phys. Rev. Lett.* 30 (1973) 398-399.

(47) COHEN, I., *Perturbation Calculations in Fluid Mechanics Using REDUCE*, *SIGSAM Bulletin*, ACM, New York, No 36 (1975) 8 (abstract only).