Computer Physics Communications 17 (1979) 207–215

APPLICATIONS OF SYMBOLIC ALGEBRAIC COMPUTATION *

W.S. BROWN
Bell Laboratories, Murray Hill, NJ 07974, USA

and

A.C. HEARN
University of Utah, Salt Lake City, UT 84112, USA

This paper is a survey of applications of systems for symbolic algebraic computation. In most successful applications, calculations that can be taken to a given order by hand are then extended one or two more orders by computer. Furthermore, with a few notable exceptions, these applications also involve numerical computation in some way. Therefore, we emphasize the interfaces between symbolic and numerical computation, including:

1. Computations with both symbolic and numerical phases.
2. Data involving both the unpredictable size and shape that typify symbolic computation and the (usually inexact) numerical values that characterize numerical computation.
3. Applications of one field to the other.

We conclude that the fields of symbolic and numerical computation can advance most fruitfully in harmony rather than in competition.

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 survey 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, we could not hope to present a complete review of the field. Instead we have chosen a variety of examples that are familiar to us and illustrative of important aspects of the field, and we have cited several reviews that describe additional applications.

An examination of nearly all successful applications of symbolic algebraic computation so far reveals that they involve problems solved by means of perturbation theories. Calculations that can be taken to a given order by hand are then extended one or two more orders by computer. Because of the exponential nature of such techniques, expressions tend to grow so large that the computational time and sheer incomprehensibility of the output preclude 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 were then used to determine a final numerical solution for the problem.

One might therefore ask why the whole calculation was not done numerically. There are two general reasons why symbolic computation is necessary in these problems. First, some intermediate steps may be so ill-conditioned numerically that a complete numerical calculation would be impossible. Secondly, the algebraic result can be used for determining numerical results over a wide range of values of the parameters in the expression (for example, in a multidimensional integral), whereas the numerical derivation of each point might be prohibitively expensive.

* Work supported in part by the National Science Foundation under Grant No. MCS76-15035.

Alternatively, one might ask whether symbolic computation should replace numerical computation in most applications, since the insight provided by an analytical solution would be of greater value than any number of numerical solutions. During the early 1960s, when general purpose symbolic algebra systems were in their infancy, some enthusiastic partisans believed that indeed this would happen. Although these hopes have not proven entirely groundless, it has become clear that the two fields can advance most fruitfully in harmony rather than in competition.

We shall therefore begin by considering some general classes of calculations that have been performed using symbolic techniques, and then explore in more detail the interface between symbolic and numerical computation in these areas.

2. Popular applications areas

A very popular area for the application of symbolic algebra systems is the field of celestial mechanics. Here the most tedious task to be faced is in the construction of an analytic perturbation theory; that is, a formula for calculating the position, velocity and rotation of a body at any instant in time. The body might be the moon or a planet, or more often these days an artificial satellite or space station. To construct such a theory requires solving the differential equations that describe the motion of the body. Since a solution rarely exists in closed form, one expresses it as a power series in terms of the small parameters of the theory, such as the orbital eccentricities, and then solves by a technique of repeated approximation.

Although the relevant physics is well understood, the parameters cannot be known with sufficient accuracy until the results of the computation are compared with observations. Thus it is imperative to obtain the coefficients of the perturbation series first as symbolic functions of the parameters. Then, as the observational evidence accumulates, the parameter values can be refined, and numerical predictions will become increasingly reliable.

The most famous such calculation is surely Delaunay's lunar theory [1], in which the secularized Hamiltonian is computed to ninth order in the small quantities, yielding the coordinates of the moon to seventh order. Like most hand calculations of this type, this investigation involved working with thousands of terms. In Delaunay's case the computation took 20 years. When Deprit, Henrard and Rom [2] repeated this calculation by computer in 1970, they found only one error in Delaunay's computation of the Hamiltonian, a major part of the work. Certainly this is a great tribute to Delaunay's skill and dedication.

The moons of Saturn pose a more challenging problem because of phase locking among their orbits. In support of the United States program of space exploration, Jefferys and Ries [3] have developed a theory of Enceladus and Dione, which can also be applied (with minor modifications) to Mimas–Tethys and Titan–Hyperion. To test the theory and refine its parameters, the series are generated symbolically and then fitted numerically to the observational data.

What types of manipulation are involved in these calculations? The most frequently considered series in celestial mechanics are Poisson series which have the form

$$S(\boldsymbol{x}, \boldsymbol{y}) = \sum_{\boldsymbol{j}} P_{\boldsymbol{j}}(\boldsymbol{x}) \cos(\boldsymbol{j} \cdot \boldsymbol{y}) + Q_{\boldsymbol{j}}(\boldsymbol{x}) \sin(\boldsymbol{j} \cdot \boldsymbol{y}) ,$$

where $\boldsymbol{x}$ is a vector of polynomial variables, $\boldsymbol{y}$ is a vector of trigonometrical variables, and $\boldsymbol{j}$ is a vector of integers. $P_{\boldsymbol{j}}(\boldsymbol{x})$ and $Q_{\boldsymbol{j}}(\boldsymbol{x})$ are polynomials, whose coefficients may be rational or floating-point numbers. By the use of standard trigonometric identities, such series are closed under the operations of addition, subtraction, multiplication and differentiation. By making suitable assumptions about the size of the parameters involved, we can also define operations of division, integration and substitution that 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, to which we shall return later. Such problems arise for example in recognizing that

$$\log \tan\left(x + \frac{\pi}{4}\right) - \sinh^{-1} \tan 2x = 0 .$$

The importance and mathematical simplicity of

Poisson series has led to the construction of many systems for their manipulation, including Rom's [4], the TRIGMAN system [5], which was used in studying the moons of Saturn, and CAMAL [6], which has been used extensively for calculations in both celestial mechanics and general relativity. Further examples are discussed in the review by Jefferys [7] and the reviews by Barton and Fitch [8,9].

To see how Poisson series arise in a natural way in celestial mechanics calculations, let us review Barton and Fitch's discussion of the computation of the series solution to an implicit equation that is the starting point for many of these calculations. We are interested in solving the Kepler equation

$$E = u + e \sin E$$

for the moon's eccentric anomaly E in terms of its mean anomaly u and orbital eccentricity e, where e is to be regarded as a small quantity. Although the problem can be solved formally in terms of Bessel functions, we normally require the solution correct to a given order in e (say, tenth), and therefore it is computationally easier to 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

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

where the right hand side 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

$$A_0 = 0 ,$$

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

The outer brackets in the above expression indicate that terms of degree greater than $k + 1$ are to be ignored in the calculation. Using standard trigonometric relations, we can express the result of this computation as a linear expression in the trigonometric functions; that is, in the form of a Poisson series. Such a computation is clearly only a small part of a complete result, such as Delaunay's, but it does illustrate the types of expressions to be manipulated. Barton and Fitch 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 second application we shall consider is in the field of quantum electro–dynamics, an area where one tries to understand the detailed interactions of electrons and photons by means of relativistic quantum mechanics. The most commonly used calculation method is due to Feynman [10], 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. From this representation, Feynman developed a method for calculating the contribution of any such diagram to the determination of a physical quantity. Since the order of perturbation is simply the number of vertices in the diagram, it follows that the most important contributions come from the simplest diagrams, and therefore 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.

From a computational point of view, calculations in this area involve quite straightforward though lengthy symbol manipulation. Feynman's method uses matrices and tensors for its description. As in the case of Poisson series, the rules for simplification are well defined, but 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 [11] and SCHOONSCHIP [12]. The former is a general LISP-based system, which has a large number of published applications in many fields to its credit. The latter system, written in COMPASS for the CDC 6000 and 7000 series compu-

ters, also has a long list of applications, essentially all of which are in the field of quantum electro–dynamics. A third system of some importance is the FORTRAN-based ASHMEDAI system [13].

To give one a feel for the scope of current calculations in this area, it is worth looking at the most important one to be 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 [14]. 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. However, the complete answer is not yet known analytically, because some of the integrations must still be performed numerically.

The need for integration in these calculations is apparent if we look again at the diagrams for the interaction of a photon with an electron. 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 momenta of some internal lines by momentum conservation. However, if a known momentum splits into two paths, we may no longer know the individual momenta but only their sum. In this case, an integration over one unknown momentum is necessary to get the desired result. The integrals which arise in these calculations are messy and tedious, rather than sophisticated, as one usually starts with rational functions. However, as integration over several variables is required, the results usually come out in terms of generalized Spence functions [15,16].

Currently, two groups [17,18] are trying to find the whole sixth order result analytically by computer, and considerable progress has been made. Although their methods are different in detail, both involve 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 [17] 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, and two further integrations. Using the ASHMEDAI program, the times on a Univac 1108 were 2 min for the traces and first two integrals, 18 min for the partial fraction expansion and 15 min for the two final integrations. It is interesting to note that the trace-taking, the original stumbling block for the hand calculations that motivated the development of such computer programs in the first place, is now only a small part of the whole computer calculation.

These two very successful applications areas for 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 but nontrivial rules of simplification Major applications were already in evidence in the 1960s, and the work of this decade has been concerned with extending the earlier successes to harder and harder problems. At the same time, problems in other areas, such as fluid mechanics, plasma physics, electrical network theory and queuing theory, that were amenable to solution by polynomial or power series manipulation were being solved by systems like ALTRAN [19] (and its predecessor ALPAK [20]), FORMAC [21] and REDUCE that possessed the necessary facilities for performing these tasks. A considerable number of applications of these programs have been described in the literature. Other convenient systems for such calculations are MACSYMA [22] and SYMBAL [23].

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 [24], the designers of algebra systems for such applications must try either to solve a very complicated simplification problem to which no clearcut solution exists, or to provide sufficiently general substitution facilities so that users can input their own simplification rules.

A particularly good example of calculations that are concerned with a large set of elementary functions with many complicated simplification rules is general relativity [8,9]. Applications in this area have been mainly concerned with the field equations,

which express the relationship between the geometrical structure of four-dimensional space-time and the distribution of mass and energy within it. Such equations are very complicated even for physically simple cases, involving tensor notation and exponential and circular functions. Among the most used systems for these problems are FORMAC, CAMAL and LAM [25]. Each of these systems, as is true for the others mentioned earlier in this paper, takes a different approach to the problem of simplification, and an amusing review of this subject has been written by Moses [26].

Having seen some of the general applications of symbolic algebra systems, let us now look in more detail at the interfaces with numerical calculations.

3. Numerical evaluation of symbolic results

When a problem in applied mathematics can be solved completely in symbolic form, the results may provide both qualitative insight and a basis for quantitative predictions. To facilitate such predictions, most symbolic algebra systems permit the replacement of the indeterminates and parameters in a symbolic expression by floating-point numbers. However, these general *substitution* mechanisms tend to be quite inefficient, and it is best to generate a special numerical evaluation program whenever a symbolic expression must be evaluated many times.

For this purpose, several systems include auxiliary processors which can compile accurate and efficient subroutines for the numerical evaluation of their symbolic output. For example, ALTRAN includes the POLGEN processor [27] for rational expressions, and TRIGMAN includes a processor [28] for Poisson series.

The design of such a processor poses an interesting and challenging problem in numerical analysis, since the given expression must be transformed symbolically into a form that minimizes execution time and roundoff error, and also avoids overflows and unnecessary underflows. Furthermore, the compiled subroutine should be able to warn the user whenever the expression is inherently ill-conditioned. Further research in this area will be very worthwhile, especially in the context of larger classes of expressions.

4. Hybrid problems

Many problems require a combination of symbolic and numerical techniques for their solution, and we shall call such problems *hybrid*. Several examples are discussed in section 2, but we give a few more here to illustrate further aspects of the interface.

Since the accurate and efficient numerical evaluation of a large symbolic expression is rarely trivial, it is important to start with the best possible expression for the purpose, and this may well be only an approximation to the solution of the original problem. For example, if one has to evaluate the Bessel function $Y_\nu(x)$, the definition

$$Y_\nu(x) = \frac{J_\nu(x)\cos\pi\nu - J_{-\nu}(x)}{\sin\pi\nu}$$

is numerically useless for small ν and x. However, Cody, Motley and Fullerton [29] had no difficulty in developing the power series with the aid of an ALTRAN program.

Another example arose in a recent study by Fullerton and Rinker [30] of the lowest-order electron–positron vacuum-polarization potential around a point charge. Since the well known closed-form solution was computationally unsatisfactory, they used ALTRAN to develop a pair of rational approximations to replace it.

Sometimes it is prohibitively expensive, or even impossible, to solve an essentially numerical problem by purely numerical means because it involves too many variables, requires too much accuracy, or is presented in an ill-conditioned or intractable form. However, a symbolic transformation may reduce the dimensionality, evade a large dose of roundoff, finesse the ill-conditioning, and otherwise change the problem into one that can be solved by standard numerical methods.

For example, in an error analysis of Goertzel's (Watt's) method for computing Fourier coefficients, Gentleman [31,32] was obliged to find the eigenvalues and eigenfunctions of three integral equations. Although a purely numerical approach would encounter the difficulties mentioned above, Loos [33] solved the problem accurately and efficiently, using a REDUCE program that involved symbolic differentiation of the integral equations followed by a standard numerical computation.

At about the same time, Gentleman [34] was also studying the possibility of a stable bound state of a positron with a helium atom, and needed to evaluate a large number of integrals from a three-parameter family. The integrals cannot be done numerically, because they are badly conditioned in some regions of interest. However, they have been evaluated analytically with the aid of ALTRAN. More recently, an analytical expression has been derived for their denominators, and a very efficient ALTRAN program has been written to obtain their numerators.

Finally, in studying a problem in the theory of magneto–hydrodynamics, Kerner and Grimm [35] needed to evaluate a family of three-dimensional Galerkin integrals. To obtain a high degree of accuracy, the integrations with respect to two of the coordinates were done analytically using REDUCE, and only the final (analytically intractable) integration was performed numerically.

In branches of physics where the theory is less well understood, computations of a similar nature may be useful for exploratory research. An interesting example is a study by Feldman [36] of a tentative piezoelectric theory of the fine-structure splitting of donor–acceptor-pair luminescence spectra of gallium phosphide. To test the theory, he used ALTRAN to construct a series approximation to the interaction energy as an analytical expression in five parameters. The output of the ALTRAN program was converted by POLGEN into a FORTRAN program, which was used in an attempt to find optimal values for the parameters. Unfortunately, a good fit between theory and experiment proved to be impossible, thus demonstrating the need for a more elaborate theory.

So far, all of our examples of hybrid computations have involved problems with purely numerical solutions. As an important contrast, we conclude this subsection with an opposite case. In numerical studies of the nonlinear interaction of two Taylor-unstable waves in a plasma, White [37] observed a phenomenon that could not be fully confirmed except by rigorous theoretical analysis. Although his subsequent analysis [38] did confirm the conjecture, it involved a large burden of tedious algebra, which was later verified by an ALTRAN program written by L.W. Fullerton. This example illustrates a vital computational symbiosis between numerical experiment and symbolic theory, which is likely to spread throughout science and engineering as appropriate languages and software become increasingly available.

5. Hybrid methods for numerical computation

Having considered a number of specific applications, let us turn our attention for a while to general hybrid methods, starting with some examples from the realm of differential equations.

To solve a system of partial differential equations by the finite element method, one first transforms the given continuum equations into an approximating set of algebraic equations, whose coefficients are integrals of certain "shape functions" and their derivatives. While these integrals are usually evaluated numerically, Andersen [39] has found that symbolic integration, performed by the MACSYMA system, is often considerably faster.

In the late 1960s, Barton, Willers and Zahar [40] programmed the classical Taylor series method for one-dimensional ordinary differential equations, using a special-purpose compiler based on the CAMAL system. Starting from an ordinary differential equation in symbolic form, this compiler performs all the required symbolic computations and produces a machine-language subroutine that becomes a key component of the overall numerical program.

Since the Taylor series method breaks down in the vicinity of a singularity, Willers [41] generalized the preceding algorithm to approximate poles of small integer order by continued fractions derived symbolically from the Taylor series. In the case of a branch point or an essential singularity, the subroutine generated by this algorithm approaches the singularity more accurately, but (of course) cannot get past it.

The idea of using symbolic computation to generate a needed numerical subroutine is far more broadly applicable. For example, in solving a system of nonlinear equations, each step requires a numerical approximation to the Jacobian matrix (i.e., the matrix of first partial derivatives). When numerical differentiation is used, it is generally considered too costly to evaluate the Jacobian afresh at each step, so one settles for rank-one updates, with perhaps an occasional restart in case of trouble. However, when the symbolic Jacobian matrix is available, it can often be evaluated together with the function vector at little extra cost,

and thus one may achieve substantial improvements in both speed and accuracy.

Recognizing a widespread need for symbolic derivatives in numerical applications, Warner [42] developed a program, PDGEN, that accepts a set of functions in symbolic form, symbolically differentiates them, and compiles a FORTRAN subroutine to evaluate both the functions and their Jacobian matrix. High level numerical languages incorporating such symbolic capabilities would be very desirable, and in fact, the developers of the proprietary non-portable PROSE system [43] claim that it meets this need.

6. Hybrid data

Usually, symbolic computation involves mathematical expressions of unpredictable size and shape in which the coefficients are exact integers or rational numbers, while numerical computation involves fixed-precision floating-point numbers. If a computation involves data with attributes of both of these customary forms, we shall call the data and the operators that manipulate them *hybrid*.

Several symbolic-algebra systems are able to operate on symbolic expressions with floating-point coefficients. Unfortunately, the presence of symbolic structure severely aggravates the already vexing problem of analyzing roundoff error, while the presence of roundoff error (no matter how small) undermines many symbolic algorithms. Nevertheless, the alternatives (introducing symbolic parameters to represent the approximate coefficients, or doing the entire computation numerically) may be quite costly, and any improvement in our ability to handle this kind of hybrid data would be very valuable.

The other kind of hybrid data consists of real numbers represented in "symbolic" form as

(1) floating-point numbers of arbitrary (dynamically determined) precision,

(2) floating-point intervals of arbitrary (dynamically determined) precision, or

(3) exact rational intervals.

The purpose of such representations is to permit real computations to yield results of guaranteed accuracy, sufficient to meet pre-specified requirements or to answer the user's questions.

A general method for the efficient evaluation of a given rational expression to within a given error tolerance, using numbers in class (1) and intervals in class (2), is given by Richman [44], who envisioned an implementation based on something like Fateman's Big-Floating-Point Arithmetic System [45] in MACSYMA.

The problem of finding isolating intervals for the distinct real roots of a polynomial poses an application for such techniques, since the required accuracy (just enough to separate the distinct roots) cannot be known in advance. One approach, based on Sturm sequences and interval bisection, requires the construction and repeated evaluation of a polynomial remainder sequence. If exact rational arithmetic is used, as discussed by Heindel [46], this method may be quite slow because of the very large integers that must be generated. However, Pinkert [47] has observed that only the signs of the generated integers are needed in this algorithm, and hence the coefficients can be represented approximately as intervals in class (2). When this is done, the computation of relevant sign sequences is apparently two orders of magnitude faster, even for relatively small problems.

Prior to this discovery, Yun [48] implemented Heindel's theory as a part of his ALGSYS package for solving systems of polynomial equations. If the solution to a given system is a set of points in (real or complex) n-dimensional Euclidean space, then ALGSYS obtains each of these points as a vector in which each coordinate is either an exact number or an isolating interval. If the solution is a set of algebraic manifolds, ALGSYS obtains a canonical representation for them. Finally, if the equations are inconsistent, ALGSYS discovers the inconsistency.

In seeking the extrema of a function of several variables, a numerical optimization program may lose accuracy, converge too slowly, or fail altogether because of ridges, narrow curved valleys, or plateaus. To avoid such difficulties, Stoutemyer [49] developed a package for analytical optimization. His package first symbolically computes a system of equations defining the stationary points, then invokes ALGSYS to solve these equations, and finally uses a combination of symbolic and numerical methods to classify the stationary points into maxima, minima, and a variety of saddles.

7. Applications of numerical methods to symbolic computation

Perhaps the most interesting and sensitive part of the interface between symbolic and numerical computation consists of the applications of one of these fields to the other. While little has been done in applying numerical methods to symbolic computation, this subsection suggests an interesting possibility.

Although the problem of finding the irreducible factors of a polynomial with integer coefficients is conceptually quite different from the problem of finding its real and complex roots, there is an obvious relationship. If there were a very fast algorithm for finding the roots to any specified accuracy, then one could easily obtain each irreducible factor from the appropriate subset of approximate roots. The available algorithms for polynomial root-finding may well be fast enough to make this method competitive for polynomials of low degree, but the requirement for *guaranteed accuracy,* which is an axiom for the root-finding work discussed in section 6, still poses obstacles.

8. Applications of symbolic computation to numerical analysis

Of the many disciplines to which symbolic computation has been applied, one of the most active and fruitful is numerical analysis. We conclude our survey of the interface with a brief discussion of this vital area.

We have already discussed (see section 4) Gentleman's application of hybrid computation to the error analysis of a well known numerical algorithm. More recently, Stoutemyer [50] has developed a general package that uses the REDUCE system to analyze the inherent error and the roundoff error of a given expression.

Unfortunately, such error analysis can be quite expensive. However, Miller [51] has shown that roundoff errors can be represented algebraically at the nodes of the graph of an expression, and then moved from node to node with appropriate algebraic transformations. In many cases the overall analysis can be simplified substantially by these techniques, and in some cases all the roundoff errors can be represented as small increments to the errors in the initial data. The automation of such analyses might prove to be very fruitful.

Finally, in the realm of partial differential equations, Cloutman and Fullerton [52] have used symbolic multi-dimensional Taylor series expansions, computed by the ALTRAN system, to analyze the discretization and roundoff errors of various numerical methods, and also, more importantly, to eliminate inaccurate or unstable methods prior to coding and testing, and to develop methods in which the lowest order errors cancel each other out.

9. Summary

We have surveyed the most popular areas of application for symbolic algebra systems, and we have discussed a rich assortment of specific examples. While symbolic methods may yield more insight than numerical techniques, and also avoid roundoff and explicitly reveal the dependence of results on parameters, there are nevertheless very few successful applications that do not also involve numerical computation in some way. Hence we have paid particular attention in this survey to the interfaces between symbolic and numerical computation. These interfaces include:

(1) Computations with both symbolic and numerical phases.
(2) Data involving both the unpredictable size and shape that typify symbolic computation and the (usually inexact) numerical values that characterize numerical computation.
(3) Applications of one field to the other.

The impending marriage between these two ungainly disciplines will not be based on either love or convenience, but on the mathematical foundation that they share and the rich variety of important practical problems that neither can solve without the help of the other. In short, this is an exciting research frontier with brilliant prospects.

References

[1] C. Delaunay, Mém. Acad. Sci. 28 (1860) 883; 29 (1867) 931.
[2] A. Deprit, J. Henrard and A. Rom, Science 168 (1970) 1569.

[3] W.H. Jefferys and L.M. Ries, Astron. J. 80 (1975) 876.
[4] A. Rom, Celestial Mech. 3 (1971) 331.
[5] W.H. Jefferys, Celestial Mech. 2 (1970) 474.
[6] J.P. Fitch, CAMAL User's Manual, Comput. Lab., Cambridge, UK (1975).
[7] W.H. Jefferys, Commun. ACM 14 (1971) 538.
[8] D. Barton and J.P. Fitch, Commun. ACM 14 (1971) 542.
[9] D. Barton and J.P. Fitch, Rep. Progr. Phys. 35 (1972) 235.
[10] R.P. Feynman, Phys. Rev. 76 (1949) 769.
[11] A.C. Hearn, REDUCE User's Manual, Second Edition, University of Utah (March 1973).
[12] H. Strubbe, Comput. Phys. Commun. 8 (1974) 1.
[13] R.C. Perisho, ASHMEDAI User's Guide, Rep. No. C00-3066-44, U.S.A.E.C. (February 1975).
[14] A. Petermann, Helv. Phys. Acta 30 (1957) 407.
[15] D. Maison and A. Petermann, Comput. Phys. Commun. 7 (1974) 121.
[16] J.A. Fox and A.C. Hearn, J. Comput. Phys. 14 (1974) 301.
[17] M.J. Levine, R.C. Perisho and R. Roskies, Would you believe more graphs for g-2?, Preprint PITT-153, University of Pittsburgh (1975).
[18] R. Barbieri, M. Caffo and E. Remiddi, Phys. Lett. 57B (1975) 460.
[19] W.S. Brown, ALTRAN User's Manual, Fourth Edition, Bell Laboratories, Murray Hill, New Jersey (1977).
[20] W.S. Brown, B.A. Tague and J.P. Hyde, Bell System Tech. J. 42 (1963) 2081; 43 (1964) 785; 43 (1964) 1547.
[21] R.G. Tobey et al., SHARE Contributed Program Library, No. 360, D-O.3.3.004 (1969).
[22] R. Bogen et al., MACSYMA Reference Manual, Project MAC, M.I.T., Cambridge, MA (1974).
[23] M. Engeli, ACM SIGSAM Bull. (Nov. 1975) 21.
[24] D. Richardson, J. Symbolic Logic 33 (1968) 514.
[25] R.A. D'Inverno, Comput. J. 12 (1969) 124.
[26] J. Moses, Commun. ACM 14 (1971) 527.
[27] S.I. Feldman and J. Ho, Computing Science Technical Report No. 34, Bell Laboratories, Murray Hill, N.J. (September 1975).
[28] P.J. Shelus and W.H. Jefferys, III, Celestial Mech. 11 (1975) 75.
[29] W.J. Cody, R.M. Motley and L. Wayne Fullerton, ACM Trans. Math. Software 3 (1977) 232.
[30] L. Wayne Fullerton and G.A. Rinker, Jr., Phys. Rev. A13 (1976) 1283.
[31] W.M. Gentleman, Comput. J. 12 (1969) 160.
[32] W.M. Gentleman, ACM SIGSAM Bull. (April 1971) 4.
[33] R. Loos, ACM SIGSAM Bull. (January 1972) 32.
[34] W.M. Gentleman, Proc. ACM Nat. Conf., Boston, August 1972 (Association for Computing Machinery, New York, 1972) p. 840.
[35] W. Kerner and R.C. Grimm, Proc. 7th Conf. on Numerical simulation of plasmas (Courant Institute, New York, June 1975).
[36] S.I. Feldman, Computing Science Technical Report No. 15, Bell Laboratories, Murray Hill, NJ (May 1974).
[37] G.N. White, Report LA-5575-MS, Los Alamos Scientific Laboratory (1974).
[38] G.N. White, Report LA-5717-MS, Los Alamos Scientific Laboratory (September 1974).
[39] C.M. Andersen, Proc. ACM Nat. Conf., San Diego, CA, November 1974 (Association for Computing Machinery, New York, 1974) p. 554.
[40] D. Barton, I.M. Willers and R.V.M. Zahar, Mathematical software, ed. John Rice (Academic Press, New York, 1971) p. 369.
[41] I.M. Willers, Commun. ACM 17 (1974) 504.
[42] D.D. Warner, Computing Science Technical Report No. 28, Bell Laboratories, Murray Hill, NJ (April 1975).
[43] Calculus Programming, Datamation (March 1975) p. 138.
[44] P.L. Richman, Commun. ACM 15 (1972) 813.
[45] R. Fateman, Proc. ACM Symp. on Symbolic and algebraic computation, Yorktown Heights, New York, August 1976 (Association for Computing Machinery, New York, 1976) p. 209.
[46] L.E. Heindel, J. ACM 18 (1971) 533.
[47] J.R. Pinkert, Proc. ACM Symp. on Symbolic and algebraic computation, Yorktown Heights, New York, August 1976 (Association for Computing Machinery, New York, 1976) p. 214.
[48] D.Y.Y. Yun, SIGSAM Bull. (Sept. 1973) 19.
[49] D.R. Stoutemyer, ACM Trans. Math. Software 1 (1975) 147.
[50] D.R. Stoutemyer, ACM Trans. Math. Software 3 (1977) 26.
[51] W. Miller, SIAM J. Comput. 5 (1976) 204.
[52] L.D. Cloutman and L.W. Fullerton, Report LA-6885-MS, Los Alamos Scientific Laboratory (July 1977).