Computers & Chemistry Vol. 6, No. 4, pp. 169–172, 1982
Printed in Great Britain.

0097-8485/82/040169-04$03.00/0
Pergamon Press Ltd.

APPLICATIONS OF COMPUTER ALGEBRA IN PHYSICAL CHEMISTRY

J. F. OGILVIE
Research School of Chemistry, Australian National University, P.O. Box 4, Canberra, A.C.T. 2600, Australia

(*Received* 25 *February* 1982; *in revised form* 23 *March* 1982)

Abstract—After a brief discussion of the nature of computer algebra, the facilities of some algebraic or symbolic processors are outlined, followed by some instances of applications of important features to problems in physical chemistry.

1. INTRODUCTION

Among the variety of methods that modern chemists apply in their work, numerical computing is well established. Through the use of such unstructured languages (compilers or interpreters) as BASIC or FORTRAN or such structured languages as ALGOL or PL-1, or Iverson's array language APL, one can readily make numerical calculations to perform such diverse tasks as to compute the electronic density distribution of a molecule according to a quantum theory, to find the thermodynamic properties of a liquid substance by simulation of the motions of the molecular particles, by a Monte-Carlo sampling technique, or to find the velocity coefficient of a chemical reaction by regression of some function of reactant concentrations with time. In principle, all such computations (i.e. the execution of the programs) can be performed by a (patient) human being on an abacus. In the history of the development of human numeracy, there was long ago discovered the capacity to think in a purely symbolic manner.

Many of the branches of mathematics, such as algebra, differential and integral calculus, group theory, etc. are founded on such symbolic representations, although of course purely arithmetic operations are an essential component of any such practice. Perhaps many chemists, even those familiar with numerical computations, may be unaware of both the existence of symbolic processors (or languages) to conduct mechanically these mathematics (as opposed to primarily arithmetic) and the potentially powerful applications of such processors in their scientific work. In physics, already extensive reviews (Barton & Fitch, 1972; Campbell, 1974; anonymous, 1981; Pavelle *et al.*, 1981) have been published with much discussion of applications, and recently a book (Howard, 1980) with FORMAC and MACSYMA programs for aeronautical applications, particle dynamics, fluid mechanics, cosmological models and trajectory calculations has appeared.

In this article we outline some exemplary uses of computer algebra in physical chemistry after describing briefly some processors and their facilities for such applications. So utilised, the computer can be regarded as having not only the numerical capabilities of traditional scientific computing, but also a word-processing facility and knowledge of rules of higher mathematics that permit it still to execute purely numerically while simulating an algebraic or symbolic capacity.

2. SURVEY OF PROCESSORS

In fact the first application of symbolic computation (Kahrimanian, 1953) dates from the earliest years of modern electronic digital computers, at a time that preceded active development of FORTRAN in U.S.A. and ALGOL in Europe. However the really concerted development of algebraic processors started about 1960. FORMAC, a formula manipulation compiler, is a superset of PL-1; because its usage is closely similar to that of numerical PL-1, it is easy to use by programmers with knowledge of this language. Furthermore this formulation has potentially enormous power due to the combination of algebraic and numerical functions. Unfortunately development of FORMAC was terminated by IBM before a completely satisfactory processor had been achieved; some improvements and enhancements were however made during further development in Europe that resulted in FORMAC-73 (Bahr, 1975). Nevertheless both versions are useful. FORMAC has been distributed in the form of macros that restricted its implementation to IBM (or compatible) computers.

LISP, a list processor, was designed not only as a language for instructing computers but also as a formal mathematical language primarily for symbolic data processing (Berkeley & Dobrow, 1964). In fact the first application of computer algebra to physical chemistry was in LISP, where wavefunctions were formulated both algebraically and numerically in order to check the results (Chandler *et al.*, 1968). This language, developed at Massachusetts Institute of Technology, is characterised by a proliferation of parentheses in its typical expressions; thus the formal construction of such expressions became an impediment to casual use by non-professional programmers. But the importance of LISP has far transcended its visible applications, at least for physical scientists, because it has been used as a preprocessor or interpreter of programs through its ability to generate programs for further execution.

One processor based on LISP was MATHLAB(68), also developed at M.I.T. (Engleman, 1969). This interactive processor enabled such common procedures as simplification, substitution, differentiation, polynomial factorisation, indefinite integration (of a restricted class of functions), direct and inverse Laplace transforms, solution of linear differential equations with constant coefficients, solution of simultaneous linear equations, and a collection of matrix functions for the introduction

or modification of matrices, extraction of sub-matrices or elements, matrix arithmetic (addition, multiplication or inversion), and extraction of significant properties (rank, determinant, characteristic polynomial and trace). Development of MATHLAB has been superseded by that of MACSYMA, a much more advanced and powerful processor, only recently released for use outside M.I.T.

A product of the University of Wisconsin (Madison) was SAC-1, for symbolic and algebraic calculations. This system consists of several FORTRAN subroutines and functions that can be called in a user's program to facilitate these types of calculations. All the subroutines and functions form thirteen different sets of programs, called subsystems, each of which is designed for the manipulation of a special type of mathematical object. The philosophy of the producers of SAC-1 was to regard the mathematical objects as lists, so that in principle the programs containing SAC-1 extend FORTRAN to become a specialised list processing language. The availability of SAC-2 has recently been announced.

A language and system for performing symbolic computations on algebraic data originating in Bell Laboratories, U.S.A., ALTRAN, algebra translator, has the basic capability to perform rational operations on rational expressions in one or more indeterminates (algebraic quantities or variables), with integer coefficients, and is designed to handle very large problems involving such data with considerable efficiency. The syntax of the language is similar to that of FORTRAN. An attractive feature of the language is that it has been itself prepared in a very transportable form of FORTRAN, although some assembly language primitives for several popular computer systems are included on the system tape. Thus it is readily installed on almost any computer having both a FORTRAN compiler and sufficient core memory.

Another popular processor for computer algebra has been REDUCE (Hearn, 1973). This programming system, built on LISP, is capable of expansion and ordering of polynomials and rational functions, symbolic differentiation, substitution and pattern matching in a wide variety of forms, calculation of greatest common divisor of two polynomials, automatic and user-controlled simplication of expressions, matrix and tensor operations, and spin-$\frac{1}{2}$ and spin-1 algebra for calculations in high-energy physics. REDUCE is still in active development and has recently incorporated a general integration capability.

Although really satisfactory languages for computer algebra require both a word length of at least 32 bits and a machine size of at least 32000 words of core memory, recently two packages, MUMATH and PICOMATH, for computer algebra have become available for popular microcomputers (Stoutemyer, 1980). Despite not being capable of extensive computations, these programs demonstrate the nature of computer algebra, and are appropriate to the context of high-school mathematics.

There are of course other algebraic processors designed for more specialised applications, such as SCHOONSCHIP (C.E.R.N., Geneva) for quantum electrodynamics, CAMAL (Cambridge University, U.K.) for general relativity, TRIGMAN for celestial mechanics etc., but these are unlikely to be generally useful for chemical applications.

3. APPLICATIONS IN PHYSICAL CHEMISTRY

Having outlined some available algebraic processors, we can now discuss some applications in physical chemistry. Perhaps one should first comment that these processors are likely to be useful for any symbolic computation of which the method is well understood but for which the operations are tedious to perform. In the following paragraphs, we specify some facilities that any generally useful algebraic processor must provide and give instances of how these have been used. Some typical times for the central processor unit, of moderately large computers common during the mid 1970's, are given as an indication of machine requirements, but of course these will vary depending on the particular processor and the machine on which it is implemented.

(1) *All types of manipulation of polynomials or multinomials should be possible, although division may be difficult*

Consider the Dunham (1932) potential-energy function $V(x)$ in terms of the reduced internuclear separation x:

$$V(x) = a_0 x^2 \left\{ 1 + \sum_{i=1}^{10} a_i x^i \right\}.$$

Sandeman (1940) has given expressions for energy coefficients Y_{kl} in terms of c_j defined in terms of the inverse relationship:

$$x = \pm \left(\frac{V}{a_0}\right)^{1/2} \left\{ 1 + \sum_{j=1}^{10} c_j \left(\frac{V}{a_0}\right)^{j/2} \right\}.$$

In order to use Sandeman's formulae in the usual form $Y_{kl}(a_i)$, it is necessary to revert the first series to obtain c_j as a function of a_i. This task is easily done in ALTRAN, for instance, and the expressions for c_7–c_{10}, to check and extend Woolley's (1962) results, require only a few seconds of CPU time.

(2) *Differentiation should be available*

If we wish to determine the discrete energy states of Ar_2 from the potential-energy function of Barker *et al.* (1971) from thermodynamic measurements, for instance, of the form

$$V(x) = \epsilon \left[e^{-\alpha x} \sum_{i=0}^{5} A_i x^i - \sum_{j=0}^{2} C_{2j+6} / (\delta + (1+x)^{2j+6}) \right],$$

we might solve the Schrödinger equation numerically, but the accuracy may be inadequate. Alternatively we could differentiate numerically this $V(x)$ function at $x = 0$ and equate the derivatives to the Dunham potential-energy coefficients a_i as:

$$\left(\frac{d^n V(x)}{dx^n}\right)_{x=0} = a_0 a_{n-2} n!, \text{ for } n > 2.$$

Numerical accuracy is almost certainly a problem if one wanted to determine the twelfth derivative in order to obtain a_{10}. Perhaps the most precise approach is to conduct the twelve differentiations analytically, with "infinite" precision, and then to substitute the values of the known A_i and C_i to obtain the desired numbers. The latter procedure requires about a minute of CPU time in FORMAC, for instance, to derive a_i up to a_{10}, whereas manually even the first eight derivatives constitute a day's work.

Another example can be found in determination of standard errors of parameters from the standard deviations of observable quantities. For instance if we wish to estimate the significance (Ogilvie & Koo, 1976) of some set of potential-energy coefficients, $a_i \pm \sigma_i^a$, resulting from the experimental error of some set of $Y_{kl} \pm \sigma_{kl}^Y$, then the procedure for error propagation (Clifford, 1973) requires the knowledge of values of the derivatives, dY_{kl}/da_i. Such analytic expressions for the derivatives can be readily generated, in REDUCE for instance, in a few minutes of CPU time; in this case furthermore, the output can be expressed as statements directly acceptable for inclusion in FORTRAN programs in which the actual numerical computations are better carried out.

(3) *Substitution*

Some of the most important operations in computer algebra are substitutions: these can be local or global, they can occur in general circumstances, or an exact pattern-matching criterion might be applied.

As an instance of this type of operation, we can consider the generation of the energy coefficients Y_{kl} from the rotationless Y_{k0}. The latter quantities can be determined from solution of the Schrödinger equation for a vibrational potential-energy function, that of Dunham (1932), expressed as

$$V(x) = B_e J\gamma^2\left\{1 + \sum_{i=1}^{10} a_i x^i\right\}$$

where $\gamma = 2B_e/\omega_e$. The Y_{kl} can be obtained from the corresponding Y_{k0} by making J-dependent all the parameters of the $V(x)$ function, such that $\gamma \to \gamma^J$, $x \to x^J$ and $a_i \to a_i^J$ (where the superscript J denotes a dependence upon the quantum number J of rotational angular momentum, not an exponent). For instance,

$$a_1^J = a_1 + \gamma^2[J(J+1)]\{4(a_2+1) - 3a_1(a_1+1)\} + \gamma^4[J(J+1)]^2\{\ldots$$

Substitution of these J-dependent parameters into the Y_{k0} has been used to determine all the 84 expressions (containing up to a_{10}) of Y_{kl}, up to $[J(J+1)]^{12}$. The expression for $Y_{1,10}$ for instance contains 139 terms, such as $87091200000a_1^2a_3a_4$. In a few minutes of CPU time with REDUCE, it was possible to double the number of expressions found since Dunham's work (1932). A smaller collection of Y_{kl} required nearer an hour of CPU time in a numerical computation according to perturbation theory (Bouanich, 1978), and derivation of $Y_{0,11}$ would probably require a year of manual effort.

(4) *General matrix operations*

Although some operations on matrices are trivial to program, such as addition or multiplication, others like inversion or evaluation of the determinant are less simple.

As an example from crystal chemistry, consider the transformation from reduced coordinates (in terms of lattice parameters a, b, c, α, β, γ) to orthogonal cartesian coordinates (x, y, z) according to a laboratory framework. While it is convenient to perform symmetry operations upon the lattice points expressed in reduced coordinates, all calculations of internuclear distances rely upon the cartesian coordinates. Therefore we require the following transformation:

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} = \left(\left(\begin{matrix} R \end{matrix}\right)\right)\begin{pmatrix} a \\ b \\ c \end{pmatrix},$$

where the elements of R involve the angular parameters as follows:

$$R_{11} = 1; \quad R_{12} = \cos\gamma; \quad R_{13} = \cos\beta;$$
$$R_{21} = R_{31} = R_{32} = 0;$$
$$R_{22} = \sin\gamma; \quad R_{23} = (\cos\alpha - \cos\beta\cos\gamma)/\sin\gamma;$$
$$R_{33} = k^{1/2}/\sin\gamma,$$

where

$$k = 1 - \cos^2\alpha - \cos^2\beta - \cos^2\gamma - 2\cos\alpha\cos\beta\cos\gamma$$

and the unit cell volume

$$V = k^{1/2}abc.$$

In REDUCE, the statements

$$RIN := R ** (-1) \quad \text{or} \quad RIN := 1/R$$

produce the results:

$$R_{11}^{-1} = 1; \quad R_{12}^{-1} = -\cos\gamma/\sin\gamma;$$
$$R_{13}^{-1} = (\cos\alpha\cos\gamma - \sin^2\gamma\cos\beta - \cos\beta\cos^2\gamma)/(k^{1/2}\sin\gamma);$$
$$R_{22}^{-1} = 1/\sin\gamma; \quad R_{23}^{-1} = (-\cos\alpha + \cos\beta\cos\gamma)/k^{1/2}\sin\gamma);$$
$$R_{33}^{-1} = \sin\gamma/k^{1/2}; \quad R_{21}^{-1} = R_{31}^{-1} = R_{32}^{-1} = 0.$$

Because the transformation matrix is inverted symbolically there can be no problems of numerical ill-conditioning to affect adversely the precision of the results.

(5) *Integration*

Perhaps many chemists labour under the mistaken impression that there is no general algorithm for indefinite integration. In fact such an algorithm was discovered by Risch (1969) and implemented in relation to the CAMAL system by Norman (Norman & Moore, 1977). Now this scheme has been incorporated into the REDUCE system, to enable use of expressions comprising polynomials, logarithmic functions, exponential functions, tan and arctan functions, and also to attempt to integrate expressions involving error functions, dilogarithms and other trigonometric expressions. For instance the expression INT(LOG(X), X) will return the result $X*(\text{LOG}(X)-1)$. Of course the result can always be checked by differentiation. Without this explicit integration facility, one can still include the definition of integration operators for specific classes of functions. Thus integration stages can be incorporated into computer algebra programs just as easily and naturally as differentiation.

4. CONCLUSIONS

Computer algebra makes possible a new approach to machine solution of problems in physical chemistry. Because calculations are exact (of "infinite" precision), with no approximations (in principle), and because of the direct algebraic form of the results, thus conveying more analytical or physical meaning than a mere table of

numbers, this approach complements traditional numerical computing.

A distinction as to mode of use of numerical and symbolic computing should be stated. Typically one develops an extensive numerical program with the intention of making many production runs with different data sets in order to generate lesser or greater quantities of numerical results of perhaps uncertain accuracy or validity. In contrast, a program of computer algebra is typically run once successfully, to produce exact symbolic expressions probably checked easily (to some extent) by inspection, although the resulting expressions might later be incorporated into numerical programs to produce several sets of meaningful numbers. Computer algebra is thus best conducted in an interactive fashion, as the user engages in transformations or substitutions on the basis of intermediate results in order to produce the final output in the most readable form. Although algebraic computing may sometimes be relatively less efficient than purely numerical computing if only the CPU time to yield a given numerical result is considered, it is likely that a judicious combination of algebraic and numerical computing can produce more reliable and meaningful results for many problems of interest to physical chemists, accompanied by both greater physical insight of the mathematical expressions and a significant decrease of programming effort and computer time, than can be achieved by numerical programming alone.

REFERENCES

Anonymous (1981), *Nature* **290**, 198.

Bahr, K. (1975), *SIGSAM Bull. ACM* **9**(1), 21.

Barker, J. A., Fisher, R. A. & Watts, R. O. (1971), *Mol. Phys.* **21**, 657.

Barton, D. & Fitch, J. P. (1972), *Rep. Prog. Phys.* **35**, 235.

Berkeley, E. C. & Dobrow, D. G. (1966), *The Programming Language LISP*, Cambridge, U.S.A., M.I.T. Press.

Bouanich, J. P. (1978), *J. Quant. Spectrosc. Radiat. Trans.* **19**, 381.

Campbell, J. A. (1974), *Acta Phys. Austr.*, suppl. XIII, 595.

Chandler, G. S., Thirunamachandran, T. & Campbell, J. A. (1968), *J. Chem. Phys.* **49**, 3640.

Clifford, A. A. (1973), *Multivariate Error Analysis*. London, Applied Science Publishers.

Dunham, J. L. (1932), *Phys. Rev.* **41**, 721.

Engleman, C. (1969), *Information Processing* 68 (Edited by Morrell, A. J. H.), pp. 462–467, Amsterdam, North Holland.

Hearn, A. C. (1973), *Proc. Third Int. Colloq. on Advanced Computing Methods in Theoretical Physics*, p. AV1-19. Marseilles, C.N.R.S.

Howard, J. C. (1980), *Practical Applications of Symbolic Computation (Mathematical Modelling of Diverse Phenomena)*. Guildford, U.K., IPC Science and Technology Press.

Kahrimanian, H. G. (1953), *Analytic Differentiation by a Digital Computer*. M.A. Thesis, Temple University, Philadelphia, U.S.A.

Norman, A. C. & Moore, P. M. A. (1977), Implementing the New Risch Integration Algorithm. *Fourth Int. Colloq. on Advanced Computing Methods in Theoretical Physics*, Marseilles.

Ogilvie, J. F. & Koo, D. (1976), *J. Mol. Spectrosc.* **61**, 332.

Pavelle, R., Rothstein, M. & Fitch, J. (1981), *Sci. Am.* **245**(6), 102.

Risch, R. H. (1969), *Trans. AMS* **139**, 167.

Sandeman, I. (1940), *Proc. Roy. Soc. Edinburgh* **60**, 210.

Stoutemyer, D. R. (1980), *SIGSAM Bull. A.C.M.* **14**(3), 5.

Woolley, H. W. (1962), *J. Chem. Phys.* **37**, 1307.