FEATURE ARTICLES

Computer Algebra in Chemistry

CHARLES S. JOHNSON, JR.

Department of Chemistry 045A, University of North Carolina, Chapel Hill, North Carolina 27514

Received October 6, 1983

The importance of computer algebra (CA) in chemistry is discussed, and applications to a variety of chemical problems are presented. It is argued that CA can increase the efficiency of chemists in both research and educational activities. It is concluded that CA should be in every chemist's tool kit.

I. INTRODUCTION

Suppose that there were a computer program which would accept expressions as input and would manipulate these expressions on demand to product other expressions rather than simply performing numerical evaluations; e.g., the input expression (x + 1/2)↑3 with the command EXPAND would yield $x^3 + (3/2)x^2 + (3/4)x + (1/8)$. Furthermore, suppose that this program could obtain partial derivatives, indefinite and definite integrals, Taylor's series expansions, Laplace transforms, solutions to systems of linear equations and differential equations, matrix products, etc., all symbolically and with infinite precision and for the finale would present the results in convenient two-dimensional form with physical units where appropriate. What impact would this program have in chemistry? The fact is that such programs have existed for more than a decade, and most chemists know little about them! This is true even though one of the first uses of computer algebra (CA) or symbolic mathematics was in theoretical chemistry, and CA has been heavily exploited in various branches of physics.

Chemistry is a quantitative science, and mathematics is often required in its description. How can it be that CA has made so little headway in chemistry? The major reason is that research applications in chemistry seldom involve extremely tedious algebraic calculations such as those found in celestial mechanics and relativity theory. Thus, the driving force for learning about CA and using it has been much less. Until recently, powerful symbolic math systems have only been available at a few sites. For small mathematical manipulations encountered daily in chemistry, the use of CA has often not been worth the effort. Even so, the lack of appreciation of the capabilities of CA among chemists, even some theoretical chemists, is surprising.

However, the situation has changed markedly in the last 2 years. Powerful symbolic math programs are now available for large minicomputers such as those used in many chemistry departments, and one program (MACSYMA) is accessible through EDUNET. Less powerful but still impressive programs are available for 8-bit microcomputers, and demonstration programs have even been written in BASIC which can run on the least expensive home computers and even on some pocket computers. I believe that CA can and should be an important tool for chemists in their daily work. The purpose of this paper is to illustrate the kinds of things which can be done with CA and to stimulate thought about how these powerful systems can be used to improve efficiency in research and education. For chemists, the major impact of CA will be in increased accuracy and efficiency and a change in viewpoint concerning time-consuming mathematical manipulations rather than in extensive algebraic calculations.

There are numerous activities in chemistry in which mathematical manipulation is not central to the task but requires a disproportionate amount of time. For example, in reading a chemistry text or a paper, one needs to verify that each equation can be obtained from the preceeding equations through specified manipulations. This might require differentiation *n* times, integration, solution of simultaneous equations, etc., but once the result has been verified, the reader can proceed to the next step with confidence. Also, in chemical research fairly simple derivations are often required; e.g., sets of linear differential equations must be solved to relate the time dependence of concentrations to rate constants, and absolute magnitudes of expressions representing electric fields must be squared and simplified to obtain expressions for intensities. In these and similar cases CA can handle the mathematical steps or can be used to check the results including units even when CA was not used in the original derivation. This procedure clearly separates the purely mathematical part from the logical flow of the physical argument. The advantage of this mode of operation is obvious to those who must review scientific manuscripts containing sequences of equations, especially if the author(s) generated the equations with a CA program in the first place. The benefit of CA to students, who may need practice with mathematical manipulation, is still controversial; but these capabilities will become available whether teachers like it or not just as pocket calculators have. The only reasonable alternative is to design courses which take advantage of the availability of CA.

There are additional reasons for those involved with deviations and calculations to be interested in CA.[1] The most important one is that algebraic results may be easier to understand and may give "insight" into a problem beyond that which could be obtained from numerical results. CA may also expand frontiers and make calculations possible which were previously considered to be intractable. At a less profound level, CA may provide simplified algebraic expressions which can be evaluated more economically than by performing direct numerical calculations. Also, exact algebraic expressions may be valuable in algorithms used in numerical calculations. For example, when roots are obtained for an expression by Newton's method, an exact derivative can be used in each iteration for improved accuracy. Another practical effect of the spread of CA will be that reference tables for certain types on mathematical functions will become obsolete just as tables of numerical values of logarithms and trigonometric functions have in the wake of hand calculators. Expressions for Legendre functions, Hermite polynomials, etc. will be created as needed from the generating functions. This and other applications of CA are illustrated in section II.

History. How did the field of CA get to its present level of development? There are a number of review articles replete with references which cover history and applications in this fascinating field.[2-8] The review given here is not exhaustive but will point out items of particular interest to chemists and key references for additional study. The article in *Scientific American* by Pavelle et al.[1] gives impressive examples of the power of CA and is a good starting place for those interested in the principles of construction of CA programs. Also, the review article by Yun and Stoutemeyer can serve as a consumer's guide to the CA algebra systems available in 1980.[2]

The idea that computing machines can manipulate symbols as well as numbers predates modern electronic computers. In 1844 Ada Augusta, Countess of Lovelace and a friend and assistant of Charles Babbage, wrote, "The (analytical) engine can arrange and combine its numerical quantities exactly as if they were letters or other general symbols; and in fact it might bring out its results in algebraic notation, were provisions made accordingly."[9] The first use of symbolic math with digital computers apparently occurred during the 1950s when automatic differentiation was first programmed.[10] However, the development work which led to the present general purpose CA systems occurred mostly during the 1960s. The state of the art from the viewpoint of computer science was reviewed in *ACM Communications* for August and in the journal for October 1971. In 1972 a review article on applications of CA programs in physics contained almost 300 references.[6]

The development of general-purpose CA programs required considerable work in the areas of simplification, symbolic integration, and convenient mathematical input and output. The simplification problem is pervasive in algebraic manipulation and is controversial. There are no generally accepted rules for simplification. The nature of the problem and the options available have been discussed by Moses.[11] The aim is to transform expressions into forms which can be efficiently handled in subsequent calculations. However, some users may want to "understand" the output while others may simply require analytical expressions for efficient numerical evaluation. The question is whether the CA system should be designed so that it will automatically carry out radical transformations or whether the system should be conservative and require the user to specify the transformations. The systems now available represent various combinations of these extremes. Moses has defined *liberal* systems as those which carry out some generally desirable simplifications but avoid other transformations, e.g., expansions, unless there are explicit instructions. *Catholic* systems are those which use more than one representation and permit users to switch between representations. There are trade-offs to be made since the systems which are most efficient at certain manipulations may produce incomprehensible output, while systems which appear to be friendlier to the user may require very large amounts of memory. The effective use of the simplification options requires considerable practice on the part of the user, and this is especially true of systems designed to run on mini- and microcomputers.

Indefinite integration is an area in which CA systems easily out perform humans and standard integral tables. A powerful and convenient program known as SIN (Symbolic INtegrator) has been developed by Moses.[12] The capabilities of this program result from a combination of problem solving methods from the field of artificial intelligence and advances in the theory of integration. The initial strategy is similar to that used by humans. In the first stage, the integrand is expanded completely and tested to see if the integral or parts of the integral can be written in the form

$$\int f(u(x))u'(x)dx$$

where f(x) is an elementary function. A table of elementary integrals is then searched for f(x). If this attempt fails, stage two is entered. Here the type of expression occurring in the integrand is determined and is used to decide which of eleven standard integration methods will be attempted. Failure at this level to either complete the integration or to determine that no solution is possible causes SIN to enter stage three. In stage three Risch's powerful algorithm is used to determine the *form* of the integral from the integrand; i.e., a successful calculation produces an expression which contains undetermined coefficients.[13,14] The integral expression is then differentiated and compared with the integrand so that the coefficients can be determined. SIN and similar programs are incorporated in general purpose CA systems.

General Purpose CA Programs. It has been estimated that more than 60 CA programs are now in use. However, many of these are either special purpose packages or are not widely available. The most extensive summary and discussion of available systems is that given by Yun and Stoutemeyer.[2] Here I simply list a few well-known general purpose systems which are likely to be available to chemists, with comments and information about sources. This does not constitute any endorsement of these products.

MACSYMA: This is a large interactive system based on LISP, with a syntax which is mostly ALGOL60.[15,16] It was designed at the Mathlab at Massachusetts Institute of Technology (MIT) under the direction of J. Moses and has been under continuous development for more than a decade. The design strategy is essentially catholic, but the system permits liberal and conservative operation as well. In addition to general purpose features, there are special facilities for tensor manipulations. The system was originally implemented on a Digital Equipment Corp. (DEC) PDP-10 but is now available for the DEC VAX, Honeywell MULTICS, and Symbolics computers. In 1984 it will also be available for some microprocessors having virtual memories. MACSYMA has been accessible through ARPANET since 1972 and through EDUNET since 1981. The MACSYMA system is now being distributed by Symbolics, Inc.[17]

REDUCE: This is also a LISP-based system with ALGOL-like syntax.[18] REDUCE, which was designed by A. C. Hearn at the University of Utah, is apparently leaner than MACSYMA with less mathematical knowledge built in. It can be interactive if sufficiently large memory partitions are available to the user. Also, there are numerous options, including FORTRAN-compatible output. There are also special purpose facilities for quantum electrodynamics. REDUCE has been distributed in various forms since 1967. It is available for a number of computers including the IBM 360/370, DEC PDP-10 and -20, and DEC VAX with UNIX. A VAX/VMS version is also planned. Information about the system can be obtained from The Rand Corp.[19]

SMP: The Symbol Manipulation Program was written at the California Institute of Technology starting in 1979 by Cole and Wolfram.[20] The "C" language was used, and the aim was to improve on the capabilities of then-available CA programs for manipulating complex algebraic expressions. A special feature of this system is a knowledge of the mathematical functions of physics. Since SMP is written in C, the potential portability to other computers is high. Recently, Inference Corp.[21] announced that SMP is operational on the DEC VAX computer and that it will be released in the future for IBM mainframes and virtual-memory microprocessors such as the Motorola 68000 and the National Semiconductor NS16032.

MAPLE: The MAPLE project was begun at the University of Waterloo in 1980, and the system is currently being implemented.[22] The aim is to develop a system for algebraic computation which will take advantage of advances in software engineering technology and will be appropriate for the hardware of the 1980s. The basic system language belongs to the BCPL/B/C family, and a macroprocessor is used to generate versions of source code in B (for Honeywell TSS) and C (for VAX UNIX). For users, a surface language is provided which is similar to ALGOL68. Maple can now be ported to Honeywell, VAX with Unix, and a couple of new Motorola 68000 based microcomputers. The system is available from the Maple Lab in the Department of Computer Science at the University of Waterloo.

muMATH-80: This is an interactive CA system based on LISP for use with 8-bit microcomputers.[23] muMATH and the surface language muSIMP were designed and implemented by A. D. Rich and D. R. Stoutemeyer at The Soft Warehouse.[24] The features are similar to those of MACSYMA and REDUCE but with limited capabilities because of the small amount of memory available. The user must construct, from supplied modules, systems for special purposes such as solution of equations or integral calculus. A major disadvantage is the lack of two-dimensional output of mathematical expressions. However, the system is useful and is an impressive example of what can be done with the Radio Shack (RS) TRS-80 Model I and computers running the Digital Research operating system CP/M. An expanded version known as muMATH-83 has recently been developed for the IBM personal computer. The aim is to make CA available to microcomputer users in the home and laboratory. These programs are distributed by The Soft Warehouse and by Microsoft Corp.[25]

In addition to these general purpose CA systems, the reader should at least be aware of ALTRAN, which is available from Bell Telephone Laboratory, SCRATCHPAD, which is under development at IBM, and FORMAC, which was developed at IBM but now in available from the SHARE LIBRARY.[26] For other systems, review articles should be consulted.

The trend is clearly to make CA available on a wide variety of computers. The full size CA systems usually require at least superminicomputers, but implementations are becoming available for high-end microcomputers. At the other extreme, muMATH runs on 8- and 16-bit microcomputers and could operate on lap computers, e.g., the TRS-80 Model 100, if suitable ROMS were available. It is even possible for those only having access to minimum BASIC in pocket computers and home computers to experience some of the joys of CA. Stoutmeyer has discussed symbolic math using BASIC and has presented a program which expands polynomials.[27] However, it turns out that the program also generates power series expansions of functions other than polynomials! This program will run on the RS TRS-80 PC-4 pocket computer with extended memory. A set of demonstration CA programs in BASIC under the name PICOMATH-80 was developed, but these are no longer available, and all future development work will be devoted to more powerful systems.

II. APPLICATIONS OF COMPUTER ALGEBRA IN CHEMISTRY

The use of CA in chemistry apparently dates from 1956 when Boys et al.[28] described automatic calculations in quantum chemistry and wrote, "The machine has been caused to perform most of the organization of the calculations and a considerable portion of the mathematical analysis, as well as all of the arithmetic." These ideas were extended in subsequent work by Barnett and others.[29-31] The currently available special purpose programs for quantum chemical calculations such as GAUSSIAN-82[32] do, of course, include facilities for organizing the calculations and presenting nonnumeric information such as symmetry types. However, the chemical literature was silent about applications of general purpose CA until the early 1970s. In 1972 Benesch reported the use of FORMAC in molecular physics,[33] and in 1973 Underhill et al. used MACSYMA in the analysis of programmed chromatography.[34] Since that time there have been a few additional references to CA in chemistry, and in 1978 Ogilvie reviewed applications of CA in physical chemistry.[35] However, references to chemical and biological applications of CA have been so sparse that Calmet and van Hulzen[3] concluded in their 1982 review article that, "Here CA has almost no impact at all." A more realistic view is that applications of CA in chemistry, while potentially very important, seldom constitute more than a small part of a project. Yun and Stoutemyer wrote,[2] "Perhaps the most prevalent opportunity for computer algebra applications is for exploratory or confirmational symbolic computations that are supportive rather than central to the task." With this view in mind, we devote the remainder of the section to relatively modest applications of CA.

Illustrations of interactions with CA systems are found primarily in instruction manuals for the systems. However, there are a few examples of dialogues with CA systems in the literature. Fateman has illustrated the use of MACSYMA,[36] Stoutemyer demonstrated applications of muMATH in physics education,[37] and Howard devoted a section of his book to examples of CA in various fields.[38] Also, the extensive encyclopedia article by Yun and Stoutemyer provides examples.[2]

At this point we turn to a number of illustrations of the use of CA in chemistry. These examples range from elementary chemistry to graduate and research level programs. All are simple enough to be verified by hand, but it is easy to imagine extensions to extremely tedious manipulations where hand calculations, at least the first time through, would be certain to generate errors. Our hands-on experience with CA has been with MACSYMA on a VAX 11/780 and the extended version of muMATH for the TRS-80. MACSYMA (in a version called VAXIMA)[39] has been in use in this department for more than a year. It is available to students in some courses and to all research students through terminals to the departmental VAX. muMATH was available to us before MACSYMA, but during the past year it has been used primarily for comparison with the more powerful system.

All of the illustrations represent dialogues with MACSYMA which were printed out on line. Other full-sized CA systems would have different syntaxes but would give essentially the same results. We begin with simple, perhaps trivial, examples to illustrate some of the commands.[15,16] Figure 1 shows the dialogue a freshman chemistry student might have with the computer about the dissociation of acetic acid. The system types (c1) and waits for a command. The student types in an equation on the same line and assigns it the name eq1. Since the line ends with ";" the equation is echoed as output (d1) in 2-dimensional format. Using "$" in place of ";" would suppress this output. The student then asks for a solution in terms of x. The response (d2) is the general solution to a quadratic equation. The student then has the choice of either substituting numerical values into the positive solution (d2) or substituting the numerical values into (d1) and solving it a second time. The latter route is taken, and the student

```
(c1) eq1:ka = x^2/(c-x);

                                        2
                                       x
(d1)                             ka = -----
                                      c - x

(c2) solve(eq1,x);

                     2                               2
              sqrt(ka  + 4 c ka) + ka        ka - sqrt(ka  + 4 c ka)
(d2)    [x = - -----------------------, x = - -----------------------]
                        2                              2

(c3) eq2:subst(1.8*10^-5*mol/1,ka,eq1)$

(c4) eq3:subst(1.0*mol/1,c,eq2)$

(c5) ev(solve(eq3,x),numer:true);

              0.004251650233050092 mol      0.004233650233050092 mol
(d5)    [x = - ------------------------, x = ------------------------]
                         1                             1
```

Figure 1. Acid dissociation problem. Illustrations of the functions SOLVE and SUBST.

```
(c1) p:n*r*t/(v-n*b)-a*n^2/v^2;

                                               2
                                 n r t      a n
(d1)                            ------- - ----
                                v - b n     2
                                           v

(c2) eq1:diff(p,v) = 0;

                                  2
                             2 a n       n r t
(d2)                         ------ - ---------- = 0
                               3              2
                              v       (v - b n)

(c3) eq2:diff(p,v,2) = 0;

                                               2
                              2 n r t     6 a n
(d3)                         ---------- - ------ = 0
                                      3      4
                             (v - b n)      v

(c4) algsys([eq1,eq2],[v,t]);

                                                           8 a
(d4)              [[v = b n, t = 0], [v = 3 b n, t = ------]]
                                                        27 b r
```

Figure 2. van der Waals equation for the pressure of an imperfect gas. Illustrations of the functions DIFF and ALGSYS.

performs the substitutions with command (c3), which replaces K_a to create eq2, and command (c4), which replaces c in eq2 to create eq3. The command (c5) then requests the evaluation of the solution of eq3 and sets the NUMER flag so that rational fractions will be converted to floating point form. MACSYMA produces the answers in (d5). Notice that units have been manipulated as well as numbers.

A less trivial example concerning the van der Waals equation for an imperfect gas is shown in Figure 2. Here the aim is to determine the critical volume and temperature from the conditions that the first and second derivatives of the pressure with respect to volume vanish at the critical point.[40] The expression for the pressure is introduced in response to the prompt (c1) and is assigned the label p. In lines (c2) and (c3) the student specifies that the first and second derivatives are equal to 0 and that the resulting equations are labeled eq1 and eq2, respectively. Then in command line (c4) a solution of these simultaneous equations is requested for the variables V and T. MACSYMA supplies the possible solutions in the array (d4), where the desired set is assigned to d4[2].

Differentiation is a relatively easy operation in CA, and therefore Taylor's series expansions are readily available. As an example consider the Morse approximation to a vibrational potential function v(x). An expansion of this function about its minimum permits the force constant for the fundamental vibration to be determined.[41] In the dialogue listed in Figure 3, a Taylor's expansion is requested in (c1) without specifying the form of v(x). The arguments of the command TAYLOR specify that the expansion is with respect to x, about the point

```
(c1) taylor(v(x),x,0,2);

                                                 !
                                    2            !
                                   d             !      2
                                  (--- (v(x))!        ) x
                                     2           !
                                   dx            !
                          d          !           !x = 0
(d1)/T/    v(0) + (-- (v(x))!       ) x + -------------------- + . . .
                          dx         !                 2
                                     !x = 0

(c2) v:d*(1-%e^-(a*x))^2;

                                          - a x 2
(d2)                              d (1 - %e      )

(c3) solve(diff(v,x),x);

(d3)                                  [x = 0]

(c4) taylor(v,x,0,4);

                                                   4   4
                          2   2        3   3   (7 d a ) x
(d4)/T/                d a   x  - d a   x  + ----------- + . . .
                                                   12

(c5) forceconstant:2*coeff(d4,x,2);

                                        2
(d5)/R/                              2 a  d
```

Figure 3. Morse potential function for vibration. Illustrations of the functions TAYLOR and COEFF.

```
(c1) lplus:%e^(%i*phi)*('DIFF(psi,theta,1)+%i*cot(theta)*'DIFF(psi,phi,1));

                          %i phi      dpsi                    dpsi
(d1)                    %e       (%i ---- cot(theta) + ------)
                                      dphi                  dtheta

(c2) psi:%e^-(2*%i*phi)*sin(theta)^2;

                                - 2 %i phi     2
(d2)                          %e            sin (theta)

(c3) ev(lplus,diff);

       %i phi        - 2 %i phi                 2
(d3) %e         (2 %e             cot(theta) sin (theta)

                                                   - 2 %i phi
                                              + 2 %e             cos(theta) sin(theta))

(c4) ratsimp(d3);

          - %i phi                      2
(d4)    %e            (2 cot(theta) sin (theta) + 2 cos(theta) sin(theta))

(c5) trigreduce(d4);

                                - %i phi
(d5)                         2 %e            sin(2 theta)
```

Figure 4. "Raising" operator $\hat{L}_{+}$. Illustration of the functions EV, RATSIMP, and TRIGREDUCE, and the conversion of verbs to nouns by the QUOTE operation.

$x = 0$, and contains terms through second degree in x. The result is the general formula in (d1). Then in (c2) the Morse potential is entered and labeled v. Command (c3) tests for the minimum of this expression by setting the derivative of v with respect to x equal to zero and solving for x. Line (d4) shows that the minimum occurs at $x = 0$, and in (c4) a Taylor's expansion about $x = 0$ is requested. Finally, in (c5) 2 times the coefficient of x^2 is assigned to the label "forceconstant".

The differentiation operator can be converted from a verb to a noun by the introduction of a single quotation mark. This has the effect of postponing the evaluation of the derivative. An application of this feature is shown in Figure 4, where the "raising" operator $\hat{L}_{+}$ of angular momentum theory is defined in (c1).[41] The notation here is hard to follow since e appears as %e, the imaginary quantity $-1^{1/2}$ appears as %i, and the Greek letters ϕ, ψ, and θ are written out as phi, psi, and theta, respectively. The expression in (d1) should be

$$e^{i\phi}[i(\partial\psi/\partial\phi)\cot\theta + (\partial\psi/\partial\theta)]$$

In (c2) the wave function $\psi(\theta,\phi)$ is defined in terms of the angles ϕ and θ and is labeled psi. This function is proportional to spherical harmonic having $L = 2$ and $M_L = -2$. Then in (c3) the evaluation of lplus or (d1) is requested with the current expression for psi. The result given in (d3) obviously can be simplified. Therefore, the function RATSIMP, which rationally

```
(c1) h[n](x):=ratsimp((-1)^n*%e^(x^2)*diff(%e^(-x^2),x,n));

                                        2             2
                                   n   x           - x
(d1)          h (x) := ratsimp((- 1)  %e    diff(%e     , x, n))
               n

(c2) for i:0 thru 3 do display(h[i](z));
                                   h (z) = 1
                                    0

                                  h (z) = 2 z
                                   1

                                           2
                                h (z) = 4 z  - 2
                                 2

                                          3
                               h (z) = 8 z  - 12 z
                                3

(d2)                                 done
```

Figure 5. Generation of Hermite polynomials. Illustration of the definition of a function and the use of a program statement to display results.

```
(c1) 'DIFF(x(t),t,2) = -k/m*x(t);

                                  2
                                 d            k x(t)
(d1)                             --- (x(t)) = - ------
                                   2            m
                                 dt

(c2) [atvalue(x(t),t = 0,0),atvalue('DIFF(x(t),t),t = 0,v(0))]$

(c3) laplace(d1,t,s);

               2                                   k laplace(x(t), t, s)
(d3)          s  laplace(x(t), t, s) - v(0) = - ---------------------
                                                           m

(c4) linsolve(d3,laplace(x(t),t,s));

                                                    v(0) m
(d4)                     [laplace(x(t), t, s) = --------]
                                                      2
                                                   m s  + k

(c5) ilt(d4[1],s,t);

                                            sqrt(k m) t
                                  v(0) m sin(-----------)
                                                 m
(d5)                       x(t) = -----------------------
                                          sqrt(k m)
```

Figure 6. Simple harmonic oscillator as an example of a linear differential equation. Illustration of the functions ATVALUE, LAPLACE, LINSOLVE, and ILT.

simplifies its argument, is invoked in (c4). Additional simplification is obtained by using TRIGREDUCE, which attempts to remove denominators and introduces multiples of angles as arguments. The result given in (d5) is proportional to the spherical harmonic having $L = 2$ and $M_L = -1$, and the $\hat{L}_+$ operator has changed M_L from -2 to -1 as expected.

The generation of tables of special functions by means of CA was previously mentioned. Figure 5 gives an illustration of this application for Hermite polynomials. First, the definition of the polynomial $H_n(x)$ is used in line (c1) to define a function of x,[41] which includes the function RATSIMP. After line (d1), the polynomial can be printed out for any value of n. Line (c2) illustrates the use of a simple program loop statement to list the first four Hermite polynomials.

Now we turn to simple differential equations such as those which appear in introductory mechanics and chemical kinetics. The well-known harmonic oscillator equation and its solution by means of Laplace transforms is shown in Figure 6. The functions LAPLACE and ILT carry out the Laplace transformation and its inverse, respectively. Also, this example introduces ATVALUE, which is required for specifying the initial conditions, and LINSOLVE which solves a linear algebraic equation or set of equations for the unknowns. The great detail shown in this solution is neither necessary nor desirable in general, and a function called DESOLVE is available which combines all of the steps between the initial conditions and the solution.

```
(c1) eq1:'DIFF(e(t),t) = -k1*e(t)*s+k2*es(t)+k3*es(t);

                    d
(d1)                -- (e(t)) = k3 es(t) + k2 es(t) - k1 s e(t)
                    dt

(c2) eq2:'DIFF(es(t),t) = k1*e(t)*s-k2*es(t)-k3*es(t);

                   d
(d2)               -- (es(t)) = - k3 es(t) - k2 es(t) + k1 s e(t)
                   dt

(c3) [atvalue(e(t),t = 0,e(0)),atvalue(es(t),t = 0,0)]$

(c4) desolve([eq1,eq2],[e(t),es(t)]);

                         - (k1 s + k3 + k2) t
              e(0) k1 s %e                        e(0) k3 + e(0) k2
(d4) [e(t) = -------------------------------- + -----------------,
                      k1 s + k3 + k2               k1 s + k3 + k2

                                                      - (k1 s + k3 + k2) t
                          e(0) k1 s      e(0) k1 s %e
                es(t) = -------------- - ---------------------------------]
                        k1 s + k3 + k2           k1 s + k3 + k2
```

Figure 7. Michaelis–Menten enzyme kinetics as an example of a system of linear differential equations. Illustration of the function DESOLVE.

```
(c1) fcn:(1+cos(t)^2)*sin(t)/sin(t/2);

                                    2
                              (cos (t) + 1) sin(t)
(d1)                          --------------------
                                        t
                                    sin(-)
                                        2

(c2) integrate(fcn,t);

                    5 t          3 t            t
              3 sin(---) + 5 sin(---) + 30 sin(-)
                     2            2             2              t
(d2)          ----------------------------------- + 4 sin(-)
                              15                              2

(c3) ans:subst(%pi,t,d2)-subst(0,t,d2);

                                      88
(d3)                                  --
                                      15
```

Figure 8. Evaluation of an integral from light-scattering theory. Illustration of the function INTEGRATE.

As a more interesting example involving differential equations, consider enzyme kinetics as described by Michaelis and Menten.[40] The reaction scheme is

$$\mathrm{E} + \mathrm{S} \underset{k_2}{\overset{k_1}{\rightleftharpoons}} \mathrm{ES} \xrightarrow{k_3} \mathrm{E} + \mathrm{Z}$$

where E is the enzyme, S is the substrate, ES is a complex, and Z is the product. Since the rate of production of Z is just $k_3[\mathrm{ES}]$, one must determine the quantity [ES]. The appropriate equations are shown in Figure 7 where it has been assumed that the concentration of the substrate S is large and constant. The procedure used in all standard textbooks is to also assume steady-state conditions where the concentration of the enzyme–substrate complex is small and constant. The right-hand side of (d2) is then set equal to 0 and solved for ES. Here that assumption is not necessary. We simply specify the initial concentrations of the enzyme and the complex as E(0) and 0, respectively, and request a complete solution. The time dependence of ES(t) is shown in (d4), and it is easy to determine the amount of time required to reach the steady state if the rate constants and the concentration of the substrate are known.

As the last illustration we evaluate an integral. Algorithms for differentiation are easy, and those for integration in the general case are very difficult. The implementation of the Risch algorithm requires a sophisticated arithmetic package and a lot of memory. The function shown in line (c1) of Figure 8 arises in the derivation of the equation for the turbidity of a suspension of rod-shaped macromolecules.[42] Integration of this function, while not very difficult, exceeds the memory available to muMATH-80. We obtain the definite integral using

MACSYMA by evaluating the antiderivative (d2) at the upper and lower limits and taking the difference. This could be done in one step by specifying the limits as arguments in the INTEGRATE function. Note that π is represented by %pi. In the event that a closed form solution does not exist, as for example with Gaussian functions, MACSYMA can often evaluate definite integrals by numerical methods.

The illustrations shown here only scratch the surface of what is possible with CA systems. Vector and dyadic analysis can be performed, including coordinate transformations for the operators.[43,44] Series can be summed and can be used in calculations,[45] as for example in the solution of differential equations. Matrices can be added, multiplied, inverted, and diagonalized. Also, MACSYMA and SMP, and probably other systems as well, provide extensive facilities for plotting functions in two and three dimensions.

III. CAPABILITIES OF COMPUTER ALGEBRA SYSTEMS

What the Programs Do Not Know. All general purpose CA systems perform the operations of basic arithmetic, take derivatives, perform at least some integrations, factor and simplify expressions, solve equations including some differential equations, and compute with Taylor's series. These operations and the output are extremely impressive, even unbelievable, to beginning users. However, there is some danger of disillusionment when the user moves from demonstration packages to real problems. Often with extensive systems such as MACSYMA the capability is present for a particular type of manipulation, but the user must devote more time in learning to use the facility than is required to complete the problem by hand or with the use of tables.

In other cases the system does not "know" the required mathematics but can be programmed through the surface language to perform it. There is still a lot of undergraduate mathematics which the CA systems do not know. Steinberg has discussed this problem and has suggested that his readers select a favorite text from their favorite area of mathematics and see what fraction of the exercises can be done by their favorite CA system.[46] In most areas the CA systems would have to be given failing marks unless they received "help" from the user. This help might take the form of breaking the problem down into smaller parts or writing a procedure in the surface language.

In spite of the deficiencies in some basic areas, there are special purpose facilities in CA systems which are sophisticated indeed, e.g., the tensor package in MACSYMA.[47] Construction of such packages is, of course, labor intensive, and the work is not done unless there is a demonstrated need. Therefore, perceived gaps and unevenness in older CA systems are a reflection of historical development and pressure from users. Most chemists have come late to the field of CA and must devote considerable time and effort to finding out what the systems can do before giving thought to improvements. The most common questions are "How can I do..." or "Will it do...", and it may be difficult to reach a conclusion when the system documentation is incomplete or does not correspond to the latest version of the system. The design of documentation for CA systems leaves much to be desired for the beginner.

An effective procedure for beginners is to attempt to do every problem they encounter which involves mathematics, whether it is from homework, research, editing, etc., with CA either before or after it is done by hand. This not only gives practice but reveals weaknesses in the systems and indicates where new procedures may need to be written. It can also cure the temptation to make problems too general. Intermediate expressions can easily be generated which will overflow any memory allocation. Users quickly learn that algebraic expressions covering dozens of pages are often not extremely helpful.

How Can CA Programs Be Taught More? Most of us would prefer to select a CA system which has the desired capabilities and leave the programming to professional programmers. However, with any system some users will find important problems which cannot be done. The remedy, if there is one, depends on the CA system. There is usually a structured programming language, similar to ALGOL, which the user sees regardless of the system language. Sample procedures are listed in the system manual which can be used as models for constructing specialized procedures. At the lowest level these procedures permit the user to program loops for the evaluation or display of expressions as shown in Figure 5. The muMATH–muSIMP system comes with a tutorial in the form of a set of interactive lessons on program construction.

When extensive programming is required, the user is advised to review the extensions to the system which have been developed in other laboratories. For example, there are occasional publications of the MACSYMA users' conferences,[48,49] and The Soft Warehouse publishes a newsletter for the exchange of information between users of muMATH.[24] The last resort is to master the programming language, usually LISP or C, for the CA system and write the necessary routines.

IV. THE FUTURE

Powerful computer work stations are becoming standard features in research laboratories, and leading universities are making provisions for faculty and students to have access to such facilities. One of the ways this will change the lives of students and teachers is by providing easy access to CA. This can enhance the education of chemists by permitting students to quickly verify derivations and to experiment with many special cases which would otherwise require too much time. CA and other capabilities of work stations, e.g., high-resolution graphics, should be taken as challenges to instructors to produce more exciting and effective learning environments. It is also obvious that mathematics courses should be restructured to make students aware of the capabilities of CA systems and the principles on which they operate. The well-educated student should certainly know about Risch's contribution to the integration problem and should benefit from it.

ACKNOWLEDGMENT

This work was supported in part under NSF Grant CHE 8022198. I thank J. C. Marshall and E. M. Johnson for helpful comments.

REFERENCES AND NOTES

(1) Pavelle, R.; Rothstein, M.; Fitch, J. *Sci. Am.* **1981**, *245*, 136–152.
(2) Yun, D. Y. Y.; Stoutemyer, D. R. In "Encyclopedia of Computer Science and Technology"; Beizer, J., et al., Eds.; Marcel Dekker: New York, 1980; Vol. 15 Suppl., pp 235–310.
(3) Calmet, J.; van Hulzen, J. A. *Computing* **1982**, *Suppl. 4*, 245–258.
(4) van Hulzen, J. A.; Calmet, J. *Computing* **1982**, *Suppl. 4*, 221–243.
(5) Gerdt, V. P.; Tarasov, O. V.; Shirkov, D. V. *Sov. Phys.—Usp. (Engl. Transl.)* **1980**, *23*, 59–77.
(6) Barton, D.; Fitch, J. P. *Rep. Prog. Phys.* **1972**, *35*, 235–314.
(7) Fitch, J. P. In "Symbolic and Algebraic Computation"; Ng, E. W., Ed.; Springer-Verlag: New York, 1979; pp 30–41.
(8) d'Inverno, R. A. In "General Relativity and Gravitation"; Held, A., Ed.; Plenum Press: New York, 1980; Vol. 1, pp 491–534.
(9) Morrison, P.; Morrison, E. "Charles Babbage and His Calculating Engines"; Dover Publications: New York, 1961. (Note E, p 273, Notes by Ada Augusta, Countess of Lovelace, on the "Sketch of the Analytical Engine Invented by Charles Babbage" by L. F. Menabrea).
(10) For references to early work on automatic differentiation see: Hanson, J. W.; Caviness, J. S.; Joseph, C. *Commun. ACM* **1962**, *5*, 349–355.
(11) Moses, J. *Commun. ACM* **1971**, *14*, 527–537.
(12) Moses, J. *Commun. ACM* **1971**, *14*, 548–560.
(13) Risch, R. H. *A. M. S. Bull.* **1970**, *76*, 605–608.
(14) Risch, R. H. *A. M. S. Trans.* **1969**, *139*, 167–189.
(15) "MACSYMA Reference Manual"; MIT Mathlab Group: Cambridge, MA, 1977.

(16) "MACSYMA Primer"; MIT Mathlab Group: Cambridge, MA, 1982.
(17) Symbolics, Inc., 257 Vassar St., Cambridge, MA 02139.
(18) "REDUCE USER'S MANUAL"; Hearn, A. C., Ed.; The Rand Corp.: Santa Monica, CA, 1983; Rand Publication CP78(4/83), Version 3.0.
(19) The Rand Corp., 1700 Main Street, Santa Monica, CA 90406, Att: Dr. Anthony C. Hearn.
(20) Cole, C. A.; Wolfram, S. et al. "SMP Handbook"; Caltech: Pasadena, CA, 1981.
(21) Inference Corp., Computer Mathematics Group, Suite 203, 3916 South Sepulveda Boulevard, Culver City, CA 90230.
(22) Geddes, K. O.; Gonnet, G. H.; Char, B. W. "MAPLE User's Manual", 2nd ed.; University of Waterloo: Waterloo, Ontario, Canada N2L 3G1.
(23) Williams, G. *BYTE*, **1980**, *5*, 325–338.
(24) The Soft Warehouse, P. O. Box 11174, Honolulu, Hawaii.
(25) Microsoft Corp., 10700 Northrup Way, Bellevue, WA 98004.
(26) FORMAC/FORMAC73 Version A77 (Tape No. 360D-03.3.013), Share Program Library Agency, Triangle Universities Computation Center, P.O. Box 12076, Research Triangle Park, NC 27709.
(27) Stoutemyer, D. R. *BYTE*, **1980**, *5*, 232–246.
(28) Boys, S. F.; Cook, B. G.; Reeves, C. M.; Shavitt, I. *Nature* (*London*) **1956**, *178*, 1207–1209.
(29) Barnett, M. P. "Methods of Computational Physics"; Alder, B., et al., Eds.; Academic Press: New York, 1963; Vol. II, pp 95–153.
(30) Wactlar, H. D.; Barnett, M. P. *Commun. ACM* **1964**, *12*, 704–710.
(31) Shavitt, I. "Methods of Computational Physics"; Alder, B., et al., Eds.; Academic Press: New York, 1963; Vol. II, pp 1–45.
(32) "GAUSSIAN 82, Molecular Orbital Computer Program"; Distributed under license by the Carnegie-Mellon Chemistry Publication Unit, Carnegie-Mellon University: Pittsburgh, PA.
(33) Benesch, R. *J. Phys. B* **1971**, *4*, 1403–1414; *Phys. Rev. A* **1972**, *6*, 573–580.
(34) Underhill, D. W.; Reeds, J. A.; Bogen, R. *Anal. Chem.* **1973**, *45*, 2314–2316.
(35) Ogilvie, J. F. *Comp. Chem.* **1982**, *6*, 169–172.
(36) Fateman, R. J. *ACM-SIGSAM Bull.* **1981**, *15*, 21–32.
(37) Stoutemyer, D. R. *Am. J. Phys.* **1981**, *49*, 85–88.
(38) Howard, J. C. "Mathematical Modeling of Diverse Phenomena"; U.S. Government Printing Office: Washington, DC, 1979; Stock No. 033-000-00777-9.
(39) VAXIMA 2.04 was made available to the Department of Chemistry at the University of North Carolina by MIT under a test site agreement. The tapes were distributed by the University of California at Berkeley.
(40) Atkins, P. W. "Physical Chemistry", 2nd ed.; W. H. Freeman: San Francisco, 1982.
(41) Johnson, C. S., Jr.; Pedersen, L. G. "Problems and Solutions in Quantum Chemistry and Physics"; Academic Press: Reading, MA, 1976.
(42) Carr, M. E.; Hermans, J. *Macromolecules* **1978**, *11*, 46–50.
(43) Stoutemyer, D. R. *Comput. Math. Appl.* **1979**, *5*, 1–9.
(44) Wirth, M. C. *SIAM J. Comput.* **1979**, *8*, 306–319.
(45) Lafferty, E. L. in "Proceedings of the 1977 MACSYMA Users' Conference"; NASA Conference Publication 2012.
(46) Steinberg, S. *ACM-SIGSAM Bull.* **1982**, *16*, 11–15.
(47) Pavelle, R.; Wester, M. "Computer Programs for Research in Gravitation and Differential Geometry". "MACSYMA Reference Manual"; Massacheusetts Institute of Technology: Cambridge, MA, 1980; document MIT/LCS/TM-167, Chapter 11.
(48) "Proceedings of the 1977 MACSYMA Users' Conference"; Berkeley, CA, 1977; NASA Conference Publications, No. 2012 (available from National Technical Information Service).
(49) "Proceedings of the 1979 MACSYMA Users' Conference"; Lewis, E., Ed.; Massacheusetts Institute of Technology Laboratory for Computer Science: Cambridge, MA, 1979; OCLC 6681418.

ARTICLES

Nitrogen's Hydrido Oxo Acids

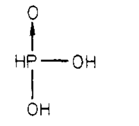
DAVID ECKROTH[†]

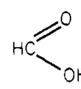
John Wiley & Sons, New York, New York 10158

Received September 29, 1982

The nomenclature of hydrido oxo acids of nitrogen is being considered by IUPAC for azinic acid (H_2NO_2H) and azonic acid (HNO_3H_2). The existing trivial names for methyleneazinic acid, *aci*-nitromethane and methylenenitronic acid, do not have systematic origins; i.e., derivatives cannot be named by the *aci* prefix, and "nitronic acid" is systematically related to the "inic" acids such as phosphinic, arsinic, and stibinic acids. The use of the name azinic acid provides both a parent name for derivatives and a systematic relationship to the names of the hydrido oxo acids of the other group 5A elements.

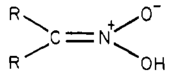
An oxo acid is a compound in which a central element is bonded to oxygen and hydroxyl groups. The acid hydrogen of an oxo acid is bonded to an oxygen, for example, nitric acid. A hydrido oxo acid is a compound in which the central element is bonded to oxygen(s) and the oxygen atom(s) of the hydroxide(s) as well as directly to hydrogen(s), for example, phosphonic acid and formic acid. Hydrido oxo acids can

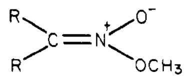

phosphonic acid


formic acid (methanoic acid)

alternatively have organic groups attached to the central element by replacement of the hydrido hydrogen by an alkyl or aryl group, e.g., methylenephosphinic acid.

The first nitrogenous example of a hydrido oxo acid to appear in the chemical literature was a tautomer of a nitroalkane called a nitronic acid by Bamberger.[1] From the original "nitronic acid" nomenclature proposal:[1]


(dialkyl) nitronic acid


methyl (dialkyl) nitronate

Nitronic acid can be used to name an alkylidene=NOOH group. The name nitronic acid is discussed below. The nitro tautomer of the name *aci*-nitro was originally proposed in a paper by Hantzsch.[2] He also described *aci*-aldehydes, *aci*-dibenzoylacetone, and the other compounds shown in Table I as well as *aci*-imides, *aci*-urethane, and *aci*-phthalimides, etc.

Table I. From the Original "*aci*" Nomenclature Proposal[2]

aci-aldehydes	*aci*-ketones
aci-dibenzoylacetone	*aci*-malonates
aci-diketopentamethylene	di*aci*-dihydroesorcinol
aci-formylcamphor	di*aci*-succinates

[†]Associate member of IUPAC's Commission on Nomenclature of Organic Chemistry.