Equation-Solving Programs for the Personal Computer

KENNETH R. FOSTER

Many engineering and scientific tasks require small computations: varying parameters in a model to study its properties, developing error models for an experiment, or solving equations numerically. Four programs for the IBM PC and compatible computers that are well suited for such purposes are reviewed. These programs are diverse: one, TK Solver Plus (*1*), resembles a cross between a computer language and a spreadsheet; a second, MathCAD (*2*), might be called a word processor that computes. Two others, Eureka (*3*) and Solver-Q (*4*), solve equations, but with quite different approaches. Both MathCAD and TK Solver are available in inexpensive student versions distributed by major textbook publishers (*5*). The programs were chosen for their wide availability, low price, and range of functions broadly centered about equation-solving.

These programs offer convenience. Models or sets of equations can be quickly set up, solved, revised, and solved again with little or no programming. These programs do not do things that other software cannot do; their value depends on whether they offer sufficient convenience to justify their use. Three of these programs are intended as productivity programs for scientists and engineers, a new class of software for the personal computer.

These programs can numerically solve equations by iteration. For example, equations such as

$$\lambda \tan \lambda = B \qquad (1)$$

frequently occur in the solution of partial differential equations; in this case B is known and λ is to be determined (*6*). Such an equation must be solved numerically. More generally, an equation such as $y = f(x)$ (where y is known and x is unknown) can be rewritten as

$$g(x) = y - f(x) \qquad (2)$$

and solved by iteration to locate the root of $g(x)$, provided that an initial guess x_0 is supplied.

This task is not trivial. In addition to the possibility that several roots may exist, the iterations may converge slowly, or not at all. Some authorities assert that no good general methods exist for solving systems of more than one nonlinear equation (*7*). The best known method is Newton's method, which is equivalent to linearizing $g(x)$ in Eq. 2 about the current estimate x_k to yield the next estimate x_{k+1} of the root. Newton's method can be extended to systems of equations by calculating the derivatives of each equation with respect to each variable. In contrast, the method of steepest descent chooses successive estimates of a root so as to progress down the steepest slope in multidimensional parameter space toward the minimum.

Both methods have advantages as well as serious limitations. Newton's method is fast (successive estimates double their number of significant digits as they converge on a root), but it is notoriously apt to fail if the initial guess is far from the root. The method of steepest descent is much slower, but is far more tolerant of bad initial guesses (*7*, *8*). Solver-Q and Eureka use variants of Newton's method and the method of steepest descent, respectively. MathCAD uses two different methods: a hybrid approach [the Levenberg-Marquardt method (*9*)] for systems of equations and a different method entirely (the secant method) to find the root of a single equation. For systems of equations, TK Solver Plus uses a sophisticated method of direct substitution followed by an iterative method based on Newton's method. The result is an enormous variability among these programs in their ability to solve systems of equations.

An equation-solving program can be used for other tasks as well. For example, the maxima or minima of a function can be found from the roots of its derivatives. Thus Eureka and Solver-Q (which are basically equation-solving programs) allow the user to solve various problems involving optimization, such as fitting data to equations by minimizing the sum-of-squares error. More efficient algorithms surely exist for such tasks, but for small problems that are infrequently encountered, computational efficiency may be an inconsequential factor compared with the convenience of setting up the problem and displaying the results. The maximum size of a problem that can be conveniently handled depends on the speed of the equation-solving routine, which is relatively fast for Solver-Q and relatively slow for Eureka. TK Solver Plus and MathCAD have many other applications that are unrelated to equation-solving.

This review will not provide speed benchmarks or exhaustive performance comparisons of these various programs. The programs are so different (in both design philosophy and features) that any fair comparison of them would be difficult. Rather, some attempt will be made to compare their design philosophies, styles of performance, and their potential usefulness to scientists or engineers.

To illustrate the use of these programs, two simple problems will be solved. First, the roots will be calculated for the system of equations:

$$x = \sin(x + y); \quad y = \cos(x - y) \qquad (3)$$

A second problem is the solution by iteration of a complex nonlinear equation

$$\frac{(z - z_1)}{(z_2 - z_1)} \left(\frac{z_2}{z}\right)^{1/3} = q \qquad (4)$$

which arises in the theory of the electrical properties of mixtures (*10*). The parameters z_1 and z_2 are complex numbers representing the electrical properties of the constituents of the mixture, and q is the volume fraction of the suspension whose property is z_2. These programs all claim to be able to handle complex numbers, which are a frequent source of trouble in scientific software. Equation 4 was solved for several values of q from 0 to 1, which represents a typical calculation that a scientist might need to do.

All of the programs solved these equations with essentially identical results; however, pronounced differences were apparent in their performance. More interesting applications of two of the programs (MathCAD and TK Solver Plus) are also described.

MathCAD

MathCAD is aptly described by MathSoft as an "engineer's scratch pad." The user is presented initially with a screen that is blank except for a single message line and a cursor.

Department of Bioengineering, University of Pennsylvania, Philadelphia, PA 19104.

The user can move the cursor around the screen and enter equations or text or obtain output in the form of plots or tables. Thus a complex problem can be set up and solved in a logical sequence that might extend over many screens. The file can be printed to yield a highly readable document that integrates text, equations, results, tables, and plots.

The most attractive feature of the program is its user interface, which is naturally adapted to scientific or engineering calculations. Equations are entered and displayed (with the graphics display of the computer) much as if they were written on a scratch pad. Plots can be created or revised in format and size with a few simple commands; tables can be created with a single keystroke. This encourages the user to play around with solutions or to experiment with different ways to represent data graphically in a way that scientists of an earlier generation (with their large assortments of graph paper) would have easily appreciated. The program has an extensive menu of commands and help messages and is supplied with a 288-page spiral-bound manual and 28 tutorial programs.

Several features of the program contribute to its extraordinary power, some of which are illustrated in Figs. 1 and 2. One is its use of range variables and iteration. For example, a variable can be defined to run over a range of values (0 to 10 for q in Fig. 1). Each subsequent equation containing that variable is evaluated for each different value of that variable, much as in a FOR-NEXT loop in a conventional programming language. In MathCAD 2.0 (but not earlier versions) the iterations can extend over sets of coupled equations, which allows, for example, the user to implement one of the various standard algorithms to solve an ordinary differential equation.

MathCAD supports an enormous variety of built-in functions, 70 of which are listed in the manual. In addition to the usual logarithmic and trigonometric functions, MathCAD includes Bessel functions, real and complex fast Fourier transforms, the error function, spline functions, linear regression, numerical integration and differentiation, and matrix operations such as multiplication and inversion. Calculations can be done with real, hexadecimal, octal, or complex numbers or with vectors or matrices. Equation-solving is accomplished by two functions: the **root** function to obtain roots of a single equation and the **find** function for roots of systems of equations (as many as 50 of them).

MathCAD solved Eqs. 3 and 4 in a time only slightly longer than that required to type in the equations and data (Fig. 1). Figure 2 illustrates its use in a simple demonstration of Fourier series.

MathCAD has a great appetite for memory and at times its response can be sluggish. The program requires nearly 350 kbytes of memory to load, but can fill up much more memory with results of its calculations, particularly if many range variables are used. The slow response arises in part from the large number of calculations that can be required; for example, the sine function had to be evaluated more than 5000 times to prepare Fig. 2 (yet the program still ran in a minute or so). Scrolling through a document with many graphs and tables can be painfully slow, as the program continually rewrites the graphics display. (MathCAD 2.0 is considerably faster than the student version in this respect.) Although the program will run on an IBM PC computer with 512 kbytes of memory (the minimum recommended by MathSoft) and a clock speed of 4.77 MHz, few users are likely to be satisfied with its performance; an AT-compatible computer (with an 80286 processor) and 640 kbytes of memory are preferable. The use of a numeric coprocessor chip (such as the 80287) is strongly recommended for all of the programs considered here.

Iterative Solution of a Complex Equation

constants: z1 := 10 - 100i

z2 := 100 - 10i

TOL := 10^{-9} tolerance for iterations

z := 50 - 50i Initial guess - needed for iterative solution

now let q vary from 0 to 1 in increments of 0.1:

q := 0,.1 ..1

$$z(q) := \text{root}\left[\left[\frac{z - z1}{z2 - z1}\right] \cdot \left[\frac{z2}{z}\right]^{\frac{1}{3}} - q, z\right]$$

Root function - to solve Eq. 4 for the unknown variable z.

Output tables

q	z(q)
0	10 - 100i
0.1	21.764 - 95.44i
0.2	32.957 - 89.966i
0.3	43.569 - 83.568i
0.4	53.585 - 76.232i
0.5	62.987 - 67.932i
0.6	71.752 - 58.631i
0.7	79.854 - 48.276i
0.8	87.27 - 36.792i
0.9	93.984 - 24.079i
1	100 - 10i

Plot - real and imaginary parts of z vs. q

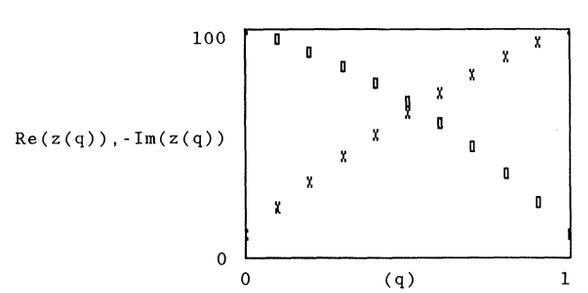

Fig. 1. MathCAD printout describing the solution of Eq. 4, illustrating the combination of text, equations, tables, and a graph to form a complete document.

MathCAD Student Edition

MathCAD Student Edition is an earlier version of the program (Version 1.1) that has been handicapped by limiting its output to a document 80 characters wide and 120 lines long. The program lacks a host of features added to Version 2.0, such as the ability to solve sets of coupled equations and perform vector and matrix operations. These limitations are probably insignificant for instructional use. The student version is accompanied by a well-produced 228-page spiral-bound manual. (MathSoft has announced that a Student Edition of Version 2.0 will be released in August 1988.)

I used MathCAD extensively (and productively) in a recent study that required a complex error analysis of a microwave measurement (*11*). This year I assigned the student version to two engineering courses with a combined enrollment of 40 students from the sophomore to graduate level. With it, students designed digital and analog filters and studied the response of transducers, the convolution operation, and the discrete Fourier transform. The students quickly began using it for other projects, including the graphing of data for a biology course. Their only frequent complaint is its sluggish response with 8086-based machines.

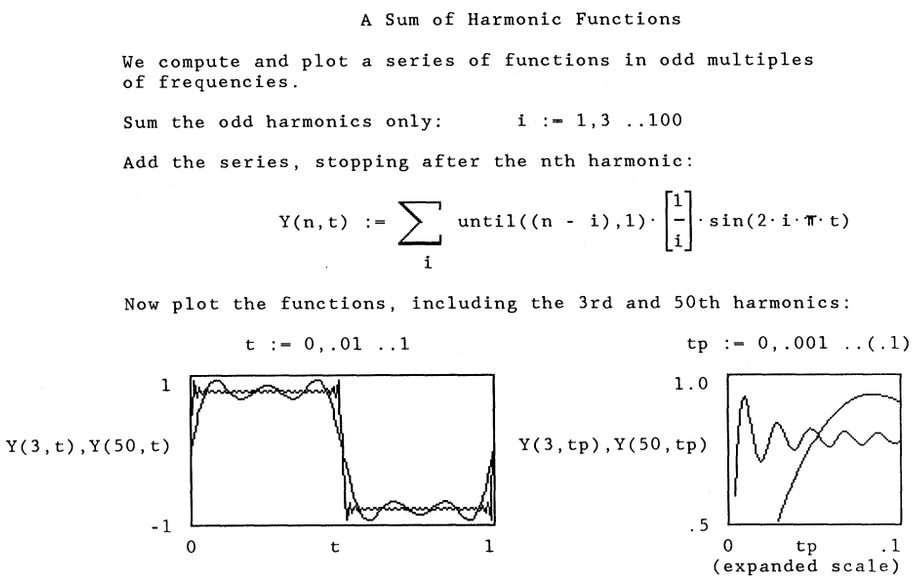

Fig. 2. MathCAD printout showing the partial sums of the Fourier series for a square wave. This illustrates the graphics capabilities of the program.

TK Solver Plus

TK Solver Plus is a greatly enhanced version of TK!Solver, which was first introduced in 1983. The initials TK refer to "toolkit," and the program is intended to serve as a comprehensive problem-solving environment for scientific and technical users. Its origins go back to the efforts by Konopasek in the 1970s to develop a very high-level language for solving equations (*12*); its present owners, working with Konopasek, have developed it into a major professional and educational tool.

TK Solver Plus is a complex program that combines elements of a spreadsheet and a programming language. The program is structured about nine sheets and eleven subsheets, each serving for entering and displaying information about the model that one is working with (data, relations, parameters related to the operation of the program, and so forth). The most important of these are the variable sheet (listing input and output values for variables) and the rule sheet (listing relations or equations governing the variables). Other sheets contain parameters for the program, unit conversion factors, data to generate plots, lists of data, and user-defined functions and procedures. The program is accompanied with three lengthy manuals and many sample programs. It is run with menu-driven commands and has an extensive and well-designed array of help screens that can be called up from the program.

The heart of TK Solver Plus is its equation-solving routine. The user enters equations on the rule sheet, and values for the known parameters on the variable sheet. The program will then solve for as many unknowns as possible with the provided data, and the results are entered on the variable sheet.

The program has two modes of solution. In its "direct mode," direct substitution is used that employs a sophisticated method of parsing the equations in a way that is transparent to the user (*12*). Thus, for example, given a rule such as

$$A + B = C * D$$

the program will find any of A, B, C, or D if it is given (or can compute) values of the remaining three parameters. In its direct mode, the program will invert functions such as $\sin(x)$ if necessary. This flexibility, which is absent in a programming language such as BASIC or Pascal, spreadsheets, and MathCAD, resembles that found in some artificial intelligence applications.

The "indirect mode," in contrast, solves equations by iteration, and is entered when a user supplies a guess for one or more variables in the variable sheet. This mode is used, for example, when an equation contains an expression that cannot be inverted. The method is a variation of Newton's method, with numerical evaluation of the necessary derivatives.

This combination—extensive substitution in the direct mode followed by iteration—is extremely efficient. The user can enter all of the pertinent equations (up to 1000 of them) for a model into the rule sheet without regard for known or unknown parameters. The set of equations can be underdetermined (with more unknowns than equations), and the program will still find the variables that can be unambiguously determined; no other program that was reviewed could do this.

The program's 71 built-in functions can be supplemented by several kinds of user-defined functions and procedures, which greatly enhance the program. For example, **list** functions allow the program to search tables for needed data, interpolating values if necessary. Users can associate variables with lists, which effectively turn them into arrays, and perform repetitive solutions of the model and then generate plots and tables of the results. User-defined procedures are equivalent to subroutines in a conventional programming language; more than 100 application programs are supplied that implement standard algorithms for solving differential equations, statistical analysis, matrix manipulation, and so forth.

This approach provides a convenient alternative to writing programs in a conventional language such as BASIC or Pascal. A user can enter a few rules in the rule sheet, give values for known parameters in either the variable sheet, user-defined functions, or procedures (if necessary), and then perform the calculation. TK Solver Plus will handle the input and output, create graphs, and do other chores.

TK Solver Plus has an inexpensive College Edition that is functionally identical to TK Solver Plus (but is supplied with only a single introductory manual and without many of the application programs). Seven "TK Solverpacks" that cover engineering and scientific subjects are available for use with TK Solver Plus. The two Solverpacks that were provided for review (introductory science and mechanical engineering) each contained a dozen models representing a broad range of topics. For example, the mechanical engineering Solverpack included a model for three rotating disks connected by two shafts. Its rule sheet includes 15 equations for the moments of inertia, resonant frequencies, and other properties; the student is asked to find the radius of an axle that gives a wheel assembly a specified resonant frequency.

More interesting educational applications of TK Solver Plus have been developed for chemistry, using models for reactions that might consist of dozens of coupled nonlinear equations (*13*). For example, a model for acid rain developed by Smith (*14*) predicts the *p*H of a droplet of cloud water in the presence of reactive gases such as sulfur dioxide. While few instructors would develop such models for their students, many potential student projects can easily be imagined. I suspect, however, that most students will use the program in the same manner as their instructors, as a convenient computational tool.

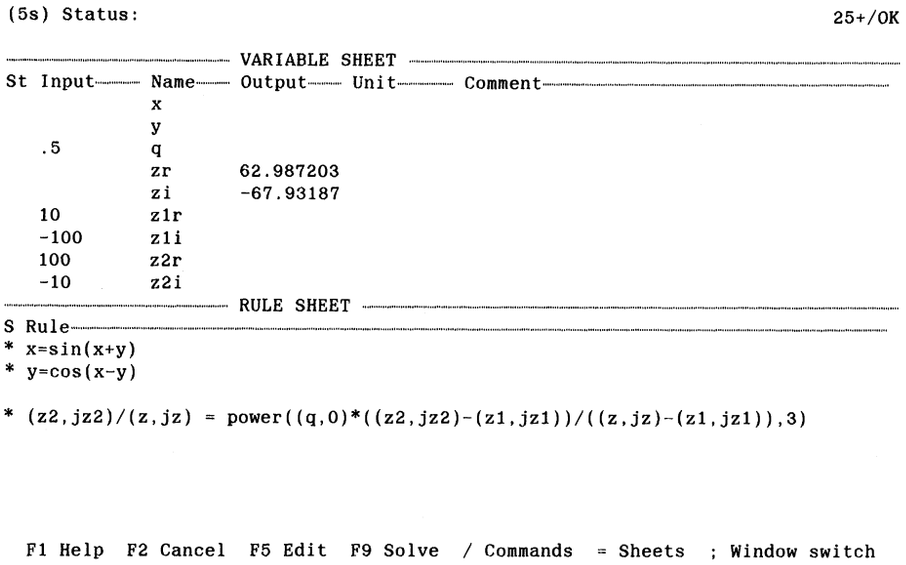

Fig. 3. Screen display from TK Solver Plus showing the solution to Eqs. 3 and 4. The parameters z_1 and z_2 in Eq. 4 were divided into real and imaginary parts [(z1r, z1i) and (z2r, z2i), respectively]. Also shown is the menu for the command keys (bottom line).

TK Solver Plus easily solved Eq. 3. Initial attempts to solve Eq. 4 were unsuccessful, because the complex power function requires an integer power; Eq. 4 was cubed and reentered, and the solution was then quickly found. The program's method of handling complex numbers as pairs of expressions is effective, but somewhat clumsy (Fig. 3). The inability of this program to handle fractional powers of complex numbers is a potential source of difficulty; for example, cubing Eq. 4 introduced additional roots.

Eureka

Eureka has been widely promoted as a general tool for scientists. It is primarily an equation-solving program, but (for reasons indicated above) it can also solve problems involving optimization and curve-fitting. It offers various other functions including financial calcuations, numerical integration, and finding all roots (real and complex) of a polynomial. (In contrast, the other programs will only find roots one at a time depending on the initial guess that is supplied.) The program is run with commands listed on the screen with the use of pull-down windows.

Eureka has a pretty face, with windows that can be separately "zoomed" (expanded to fill the entire screen), "tiled" (arranged in blocks on the screen), or "stacked." Some windows contain the equations to be solved, results, and plots. Others contain "reports" (a running record of the calculations that can be edited with a simple text editor) or results of the "verify" operation (this last operation involves the substitution of a solution back into the original equations to verify its accuracy). Crude but effective plots can be displayed in the plot window, even with a text-only display. The program is supplied with a lengthy, 243-page manual, which is largely devoted to elaborately described test problems, but has little technical information about the numerical methods used.

Eureka is easy (and even fun) to use, at least for simple problems. The user types in the equations to be solved with the text editor, adds initial guesses, and enters the **solve** command. Eureka easily solved Eq. 3, and it solved the complex equation (Eq. 4) for one value of q with the least fuss of all the programs considered here (Fig. 4). These calculations were performed in a few seconds with the computer that was used (an 80286-based machine with numeric coprocessor chip).

However, Eureka has severe limitations. Perhaps as a result of its slow algorithm, it is designed to handle a maximum of 20 equations, or 10 equations with complex parameters. Optimization and fitting applications are limited to no more than 19 data values, fewer if complex parameters are present. Users are likely to find the program uncomfortably slow for problems that approach these limits. Attempts to solve Eqs. 3 and 4 together were unsuccessful; the program treated x and y in Eq. 3 as complex variables and the solution did not converge.

Moreover, Eureka is extremely clumsy to use for repeated calculations. The user must manually change parameters and later sort through the Report file to collect the results; to solve Eq. 4 with 11 values of q required far more keystrokes than with any other program considered here. For curve-fitting problems, each data value must be explicitly written into the equation file as a separate constraint equation, an extremely clumsy approach. Of all of the programs considered here, Eureka is least well suited for problems requiring extensive calculations.

Eureka has a more serious problem: it will occasionally terminate its iterations with in-

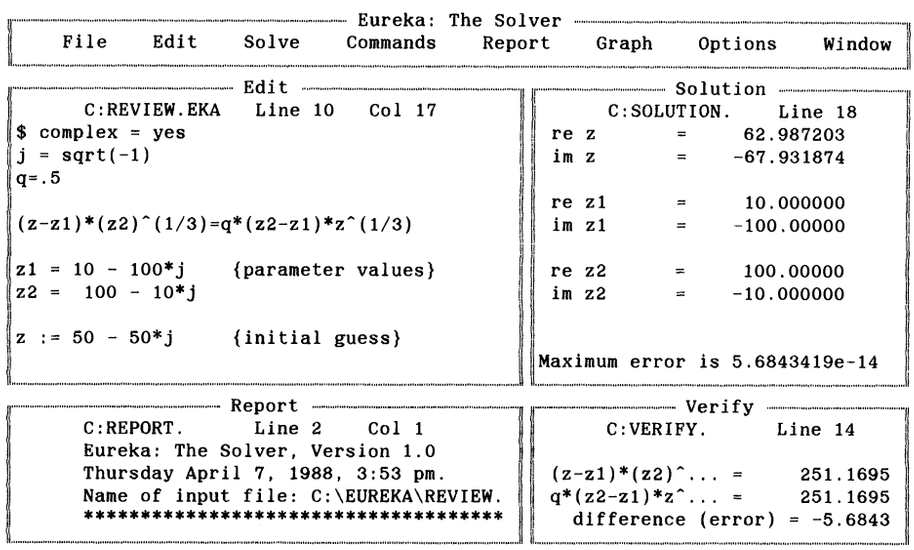

Fig. 4. Screen output from Eureka for the solution of Eq. 3. Portions of four windows are shown: the Edit window contains the equations to be solved and initial guesses; the Solution window displays the solution; the Verify window shows the results when the equations are evaluated using the solution (to verify its accuracy); and the Report window. Also shown is the command line (top) and list of function keys (bottom).

correct or misleading results but with no error message to indicate that something is wrong. This unfortunate situation arises when the program is asked to solve underdetermined sets of equations (with more variables than equations) and is a clear hazard to a careless user. (The other programs include safeguards against such errors.)

Solver-Q

This equation-solving program is to my mind more interesting than Eureka. Unlike the other programs considered here, Solver-Q uses extensive symbolic manipulation of equations in a way that is apparent to the user. The program was written by Fernando L. Alvarado of the University of Wisconsin, which distributes it at low cost. Like Eureka, it uses menu-driven commands and windows for equations and results. However, it has no graphics capabilities (but it can write its output to a file that can be read by a spreadsheet and plotted).

Solver-Q symbolically separates complex equations into real and imaginary parts, and (in its default mode) it symbolically calculates the derivatives needed to apply Newton's method. In fitting data to a function, it will symbolically calculate the sum-of-squares error and then minimize it. This process can be very time-consuming, but once completed, a system of equations can be solved repeatedly with little delay (after one iteration, if all of the equations are linear). Solver-Q's capability of symbolic manipulation is reminiscent of that of programs such as Macsyma, but its implementation is far simpler, and is chiefly limited to finding derivatives and simple algebraic maneuvers.

Repeated calculations can be performed for data contained in a file, a great convenience. Moreover, initial guesses and other parameters of the equations can be modified during a calculation, which allows seeded iterations (and thus, for example, the iterative solution of differential equations). Solver-Q will handle as many as 1000 simultaneous equations, subject to further limitations by available memory and the patience of the user. Alvarado reported that he and his co-workers frequently use it for models with 50 to 100 equations (*15*).

In many respects, Solver-Q is more powerful than Eureka. It is clearly aimed at a more sophisticated audience. It offers alternative methods of solution, including Newton's method with numerical evaluation of derivatives and an advanced technique based on homotopy methods. The 104-page manual is supplemented with 27 tutorial programs, considerable technical information about numerical methods, and references to advanced literature on numerical analysis. However, Solver-Q requires version 3.0 or later of PC/MS-DOS, a likely inconvenience to many potential users.

Solver-Q easily solved Eq. 3. Initial attempts to solve Eq. 4 were unsuccessful; the program aborted because of its inability to compute the fractional power of a complex number. Equation 3 was cubed and reentered. After a few minutes the program had separated it into its real and imaginary parts, in which all parameters (including q) were assumed to be complex. The imaginary part of q was manually set to zero, after which the program simplified the equations to yield those shown in Fig. 5. These equations were quickly solved for several values of q.

Table 1. Specifications of the software packages reviewed. The programs were tested on a Hewlett-Packard Vectra computer with a numerical coprocessor chip, compatible with an IBM AT, with MS-DOS 3.2. Eureka is also available for the Macintosh. All of these programs are delivered compiled. Solver-Q requires PC/MS-DOS 3.0 or a later version; all of the other programs require DOS 2.0 or a later version. None of these programs is copy protected.

	MathCAD	TK Solver Plus	Eureka	Solver-Q
Version tested	2.0*	1.0	1.0	1.10B
Cost†	$349	$395‡	$167§	$40/$90‖
Special hardware requirements¶	Graphics display#	**	**	
Memory required (kbyte)††	512	384	384	128

*The MathCAD Student Version ($37.75) is a limited version of MathCAD Version 1.1. †The prices shown are list prices. All of these programs except Solver-Q are available at substantial discounts from the list price. MathCAD Version 2.0 and TK Solver Plus are available to owners of previous versions for modest upgrade costs. A student version of TK Solver Plus is available for $44.95. ‡Application modules ("Solverpacks") are available for $60 to $70. §Substantial educational discounts of Borland products are available. ‖$40 for educational institutions, $90 for commercial users. ¶All of the programs support mathematics coprocessor chips (8087, 20287, or 80387), which are highly recommended. #Requires graphics [IBM CGA, VGA, EGA, Hercules Graphics Card, Toshiba T3100, or AT&T 6300 series (monochrome monitor)]. **Graphics [IBM CGA, EGA, Hercules Graphics Card, Toshiba T3100, or AT&T 6300 series (monochrome monitor)] optional, but recommended. ††Minimum memory recommended; many applications will require more money.

Discussion

Comparing these programs is difficult because of their great diversity. Problems can easily be constructed for which one or another is clearly superior. In particular, Solver-Q's capability for symbolic manipulation gives it several distinctive characteristics. The use of all these programs is straightforward; all are equipped with pull-down menus for commands and extensive help files. Still, they differ considerably in their potential usefulness to an average user.

Eureka and Solver-Q are primarily equa-

```
Solve Edit Update Results Options FileOp AdvOp Quit  Robust Step 1 Iter 6
Solve Equations, Display Results                     Tolerance -8 in Eqn 1
"F1" for Help   hoose command  "Enter" to Execute Solution Found

 Equation 4 separated into real and imaginary parts

 real part
z2r*((zr-z1r)*((zr-z1r)^2-(zi-z1i)^2)-2*(zr-z1r)*(zi-z1i)^2)-z2i*((zi-
z1i)*((zr-z1r)^2-(zi-z1i)^2)+2*(zi-z1i)*(zr-z1r)^2)-zr*q^3*((z2r-z1r)
*((z2r-z1r)^2-(z2i-z1i)^2)-2*(z2r-z1r)*(z2i-z1i)^2)+zi*q^3*((z2i-z1i)
*((z2r-z1r)^2-(z2i-z1i)^2)+2*(z2i-z1i)*(z2r-z1r)^2)=0;

 imaginary part
z2r*((zi-z1i)*((zr-z1r)^2-(zi-z1i)^2)+2*(zi-z1i)*(zr-z1r)^2)+z2i*((zr-
z1r)*((zr-z1r)^2-(zi-z1i)^2)-2*(zr-z1r)*(zi-z1i)^2)-zr*q^3*((z2i-z1i)
*((z2r-z1r)^2-(z2i-z1i)^2)+2*(z2i-z1i)*(z2r-z1r)^2)-zi*q^3*((z2r-z1r)
*((z2r-z1r)^2-(z2i-z1i)^2)-2*(z2r-z1r)*(z2i-z1i)^2)=0;

z1r=10; z1i=-100;  real and imaginary parts of z1,z2
z2r=100;   z2i=-10;
q=.5;

zr≈50;  zi≈-50;         initial guess for real and imag. parts of z
      "zr"        "zi"     "tol"
 62.987203 -67.931874        -8
```

Fig. 5. Equation window of Solver-Q showing Eq. 4, which had been cubed and symbolically manipulated by the program to separate its real from complex parts (see text). At the bottom of the figure is the solution window.

tion-solving routines, but they provide other functions of potential use to scientists and engineers. Eureka is probably too limited for most professional applications. However, it is easy to use, effective for simple problems, and could be valuable for educational purposes (provided that students can be taught to be skeptical of their results). Solver-Q is more useful for larger problems, but it makes heavier demands on the user. Its approach of using symbolic manipulations to facilitate numerical computations is innovative.

MathCAD and TK Solver Plus are tools of much broader usefulness. Their styles and capabilities are quite different, and the programs are complementary rather than competing. MathCAD's free-form style encourages its use for ad hoc calculations; it has the most natural user interface of any of the programs considered here.

TK Solver Plus is the more powerful program of the two. Its ability to efficiently solve many coupled equations without regard for known and unknown parameters will be a great asset for some users. But it can also serve as a convenient alternative to languages such as BASIC or Pascal for many other kinds of problems. Excellent sources of standard algorithms are available [see, for example, (*7*)], which can be easily implemented with this program. If MathCAD is a super calculator, the TK Solver Plus is a super programming language. Both programs offer exciting possibilities for education. MathCAD is ideal for students, who can concentrate on the problem without having to spend much time learning the tool. Educational applications can be developed for TK Solver Plus (such as the acid rain problem described above) that involve interesting phenomena modeled by systems of equations. In short, both programs deserve large followings among professional scientists and engineers and their students.

REFERENCES

1. TK Solver Plus, Universal Technical Systems, Inc., 1220 Rock Street, Rockford, IL 61101.
2. MathCAD, MathSoft, Inc., One Kendall Square, Building 200, Cambridge, MA 02139.
3. Eureka, Borland International, Inc., 4585 Scotts Valley Drive, Scotts Valley, CA 95066.
4. Solver-Q, Software Development and Distribution Center, University of Wisconsin–Madison, 1210 West Dayton Street, Madison, WI 53706.
5. MathCAD, Student Edition, Addison-Wesley, Reading, MA 01867; TK Solver Plus, College Edition, McGraw-Hill, New York, NY 10020.
6. H. S. Carslow and J. C. Jaeger, *Conduction of Heat in Solids* (Oxford Univ. Press, Oxford, ed. 2, 1959).
7. W. H. Press, B. P. Flannery, S. A. Teukolsky, W. T. Vetterling, *Numerical Recipes* (Cambridge Univ. Press, Cambridge, 1986).
8. F. van Zeggeren and S. H. Storey, *The Computation of Chemical Equilibria* (Cambridge Univ. Press, New York, 1970); A. M. Ostrowski, *Solution of Equations and Systems of Equations* (Academic Press, New York, 1966).
9. J. J. More, B. S. Garbow, K. E. Hillstrom, "User's Guide to Minipack I," *Argonne Natl. Lab. Publ. ANL-80-74* (1980).
10. T. Hanai, in *Emulsion Science*, P. Sherman, Ed. (Academic Press, New York, 1968), chap. 5.
11. K. R. Foster, B. R. Epstein, M. A. Gealt, *Biophys. J.* **52**, 421 (1987).
12. M. Konopasek and C. Papaconstadopoulos, *Comput. Lang.* **3**, 145 (1978); M. Konopasek and S. Jayaraman, *Proc. IEEE* **73**, 1791 (1985).
13. A. L. Smith and M. Konopasek, *Thermodynamics of Aqueous Systems with Industrial Applications: Second International Conference*, M. L. Rafal, Ed., (Hemisphere, Washington, DC, in press).
14. A. L. Smith, *Acad. Comput.* **2** (No. 3), 36 (1987).
15. F. L. Alvarado, personal communication.