Computer Calculus

Long perceived as merely 'number crunchers,' computers are now moving into the realm of elegant mathematics

BY LYNN ARTHUR STEEN

"Awesome ... invaluable ... unbelievable...." These are the assessments by normally taciturn research scientists of symbolic computer algebra, a group of programs that allows computers to carry out theoretical (rather than merely numerical) calculations. These programs do in a few brief minutes virtually all mathematics that most engineers and scientists know; their ability to slog through theoretical solutions to large systems of equations has already led to advances in gravitation and high energy physics. "It is only a matter of time before these programs provide major breakthroughs," says physicist Richard Pavelle of Massachusetts Institute of Technology's Lincoln Labs in Lexington, Mass.

Although computer prophets have been touting the potential of artificial (machine) intelligence for several decades, the major triumphs of this research endeavor have been limited to the narrow fields of checkers and chess. Major commercial and scientific uses of the computer—data processing, word processing and "number crunching" — have little to do with intelligent thought. Computer algebra systems, in contrast, manipulate abstract symbolic mathematical expressions. They do algebra, and calculus, and linear algebra; indeed, they do virtually everything taught in the first two years of university mathematics.

Traditional scientific computing deals with numbers; computer algebra deals with symbols. Asked to solve the quadratic equation $4x^2 - 8x + 1 = 0$, traditional computer programs will yield the two numerical answers: .133975 and 1.866025. But a computer algebra program will respond, like any good high school algebra student, with the quadratic formula:

$$x = \frac{8 \pm \sqrt{64 - 16}}{8} = 1 \pm \frac{1}{2}\sqrt{3}$$

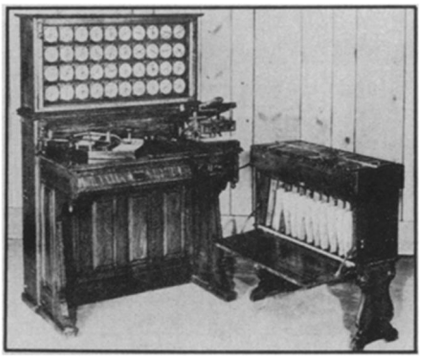
Moreover — and this is what sets computer algebra apart from previous scientific programing—it can equally well solve the generic equation $ax^2 + bx + c = 0$:

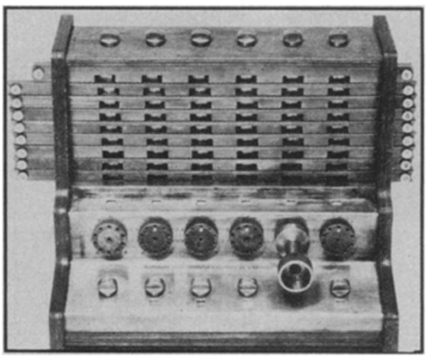
$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

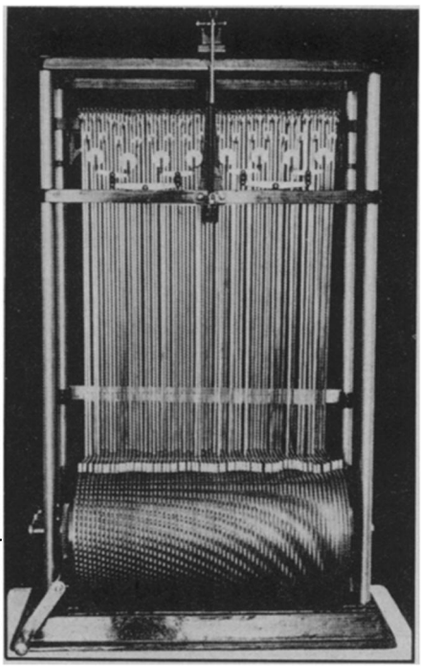
The quadratic formula is, of course, a rather elementary part of mathematics. Yet computer algebra systems can do higher mathematics with equal ease. They can factor polynomials, simplify expressions, differentiate functions, solve equations, expand functions into Taylor series, invert matrices and—the coup de grâce—integrate functions. It is this latter ability that inspires awe in the knowledgeable beholder.

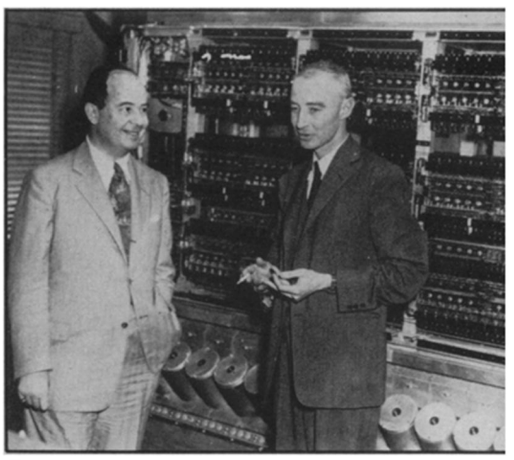
Anyone who has studied university calculus knows that there is a fundamental difference between such activities as differentiating functions or inverting matrices and integrating functions. The former procedures, like virtually all parts of high school algebra, are entirely algorithmic. There is a recipe for each of these techniques that, if carefully followed, will guarantee success. But calculus books contain no recipe for success in integration. Beyond a few simple cases and common patterns, there is only a murky never-never land of trial and failure. Success with problems of integration — and consequently with differential equations, whose solutions require integrals—seems to be as much art as science.

That computers have mastered this art is due to the successful completion in the 1960s of a two-hundred-year research effort into the mysteries of symbolic integration. Early in the nineteenth century Pierre Simon de la Place formulated a conjecture









Photos: The Computer: From PASCAL to von Neumann

They could calculate or they could analyze harmonic phenomena. Even the Institute for Advanced Study Computer of 1952 was billed as a "tool for heuristic investigations in pure mathematics where the burden of algebraic computation is prohibitive by normal human methods." Now computers are beginning to do the algebra and analysis themselves. These historic ones are (counterclockwise from top right) Wilhelm Schickard's calculator of 1623, the Hollerith tabulating machine that crunched the U.S. 1890 census returns, the Michelson-Stratton harmonic analyzer of 1898 that solved Fourier series and the IAS *shown with John von Neumann (left) and J. Robert Oppenheimer.*

about the integral of algebraic functions. This conjecture, which was proved around 1830 by Niels Henrik Abel, led Joseph Liouville to formulate a general theorem about the integral of any elementary function — those built up from the standard scientific functions such as sin, cos, log, exp, etc. Liouville's theorem tells which functions can be integrated and what form the answers can take, but it does not contain any useful clue about how the answer can be found.

The missing ingredient — an algorithm for integration — was finally completed in 1968 by Robert H. Risch of the System Development Corp. in Santa Monica, Calif., just as work was beginning at MIT and other places on symbolic computer algebra. Risch's algorithm is the Rosetta Stone of elementary calculus: It replaced mystery with mechanism, thus solving the last major puzzle in the syllabus of university calculus.

Translating the theoretical algorithms into computer programs took another several years. One of the largest and best known of these systems is MACSYMA, developed at MIT between 1968 and 1971 by Carl Engleman, Bill Martin and Joel Moses. MACSYMA now consists of nearly 500,000 words of computer code, and represents about 50 man-years of effort. Similar projects have been developed at other institutions (e.g., SCRATCHPAD at IBM, ALTRAN at Bell Labs), and use by the scientific community is spreading rapidly as word gets around concerning the power of these systems.

Perhaps the greatest popular impact of computer algebra will come as a result of its translation into the language of microcomputers. About a year ago the major parts of MACSYMA were condensed and rewritten (under the name muMath) to fit on a 5¼" floppy disc and to run on the ubiquitous TRS-80 Radio Shack computer. By now muMath is available for virtually any personal computer with a Z80 or 8080A microprocessor. Recently a still more stripped down version (called Picomath) has been written in Basic to run on just about any computer. These systems will soon bring into every research lab and every classroom a powerful tool that will do for mathematical analysis what the hand calculator has done for calculation and arithmetic.

The fact that computer algebra systems deal with symbols rather than with numbers gives them an enormous advantage over traditional scientific computer calculations. Heretofore mathematical calculations on a computer were carried out in decimal (actually in binary) arithmetic with a limited number of digits — usually about 16 decimal places for so-called "extended precision" work. Sixteen digits may appear to afford adequate precision for most work, and it usually does. However, the errors caused by rounding off the seventeenth place do not go away. In any calculation requiring millions or billions of steps — as many calculations do — the accumulated roundoff error can be as large as the answer, thus invalidating the entire calculation.

The computer algebra packages avoid this problem by working with symbols rather than with numbers. They keep track of each and every symbol, no more thinking of rounding off numbers than words. This means that answers — whether symbolic or numerical — are exact, not approximate.

If you ask a typical computer program in BASIC or FORTRAN to calculate the number 30! (which is $30 \times 29 \times 28 \times 27 \ldots \times 2 \times 1$), it will respond with something like 2.6525286×10^{32}. But if you ask a computer algebra system to calculate this, it will yield

265252859812191058636308480000000.

For some problems the 8 digits of the first answer are quite adequate; but for other problems (e.g., factoring) you do need the complete accuracy of the exact result.

What's more important, though, is the ability of a computer algebra system to save calculation to the last stages in its analysis—thus avoiding the accumulation of roundoff error as well as the need for remembering and manipulating enormous numbers. This is in fact the way scientists and mathematicians have traditionally solved problems: It is the essence of *algebra* — to reason symbolically rather than numerically.

Even if scientists begin a problem with specific numbers, they first rephrase things by substituting letters for the numbers, and then solve the problem. Once they have the solution expressed in letters, they can then substitute the corresponding numbers to calculate the numerical answer. This reduces significantly the number of arithmetic steps (and the chance of human error). More important, it makes it possible for the scientist to use the answer to explore variations in the original problem, to answer the "what if" questions that inevitably follow the first solution. What if this coefficient were twice as big? What if the price of oil goes up by 12 percent rather than by 10 percent? True problem solving requires the ability to explore easily and quickly such natural variations on the original problem.

Traditional scientific software can do this, but only at the cost of recalculating the entire solution each time. If this solution happens to be lengthy and expensive in computer time, it reduces or eliminates the ability of the investigator to explore these options. Computer algebra systems, in contrast, do just what human problem solvers do: They express their answers in symbolic form, then merely substitute various numerical values to quickly produce answers to innumerable "what if" questions.

It is this ability to mimic so closely patterns of human problem solving that makes observers feel that these programs are very near to true "artificial" intelligence. While human intervention and direction is still needed — none of these programs can determine which equations need to be solved, nor can they complete their analysis without hints from their human masters — the computer algebra systems represent a major advance in the evolution of computer software toward realistic and useful problem solving. □

Examples of Symbolic Computer Algebra

Factor $x^{25} + 1$:

$$(x + 1)(x^4 - x^3 + x^2 - x + 1)(x^{20} - x^{15} + x^{10} - x^5 + 1)$$

Integrate $\dfrac{2x}{x^3 + 1}$:

$$2\,\frac{\log(x^2 - x + 1)}{6} - \frac{\log(\text{abs}(x + 1))}{3} + \frac{\text{atan}\left(\dfrac{2x - 1}{\sqrt{3}}\right)}{\sqrt{3}}$$

Expand $x \prod_{n=1}^{\infty} (1 - x^n)^{24}$:

$$x - 24x^2 + 252x^3 - 1472x^4 + 4730x^5 + \ldots$$

Determinant of $\begin{bmatrix} x^3 & x^2 & x & 1 \\ y^3 & y^2 & y & 1 \\ z^3 & z^2 & z & 1 \\ w^3 & w^2 & w & 1 \end{bmatrix}$:

$$-(x-w)(y-w)(y-x)(z-w)(z-x)(z-y)$$

These are examples of certain standard functions that high-school and college students learn to work by hand: factoring, integration (one of two main procedures of calculus), expansion of a power series, and taking the determinant of a matrix.