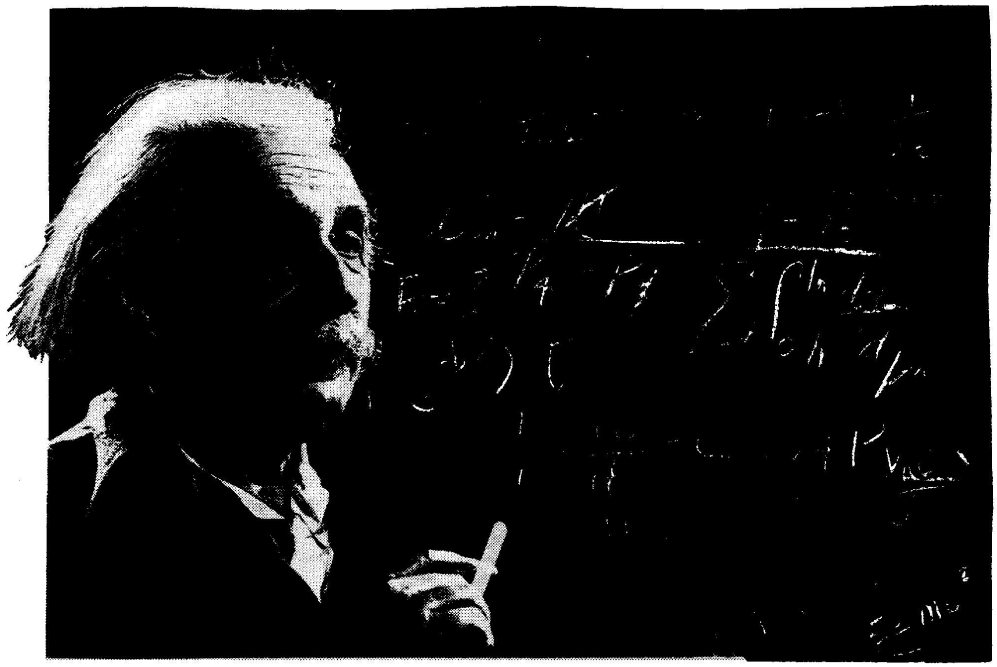


Computer maths: could do better

COMPUTERS can add and subtract more nimbly than Albert Einstein ever could, but they are still mere beginners when it comes to "real" mathematics. Nevertheless they are proving to be adept pupils. There is a glimmer of hope that from their new-found skills a new technique—call it maths processing—will bring to engineers and scientists the sort of benefits that word processing brought to writers and secretaries. But there is a long way still to go.

Most of the mathematics that computers now do is unskilled labour. Scientists devise, say, an elegant new equation to calculate the size of the universe, which unfortunately takes unthinkably many multiplications to reach an answer. Thirty years ago, that would have made the equation useless; now computers can be programmed to make such boring tasks more manageable.

Programming computers to do these calculations—a skill known as numerical analysis—is not straightforward. Some problems do not lend themselves to solution by computer; even those that do are sometimes plagued by other irritations, such as the fact that decimal numbers do not always translate perfectly into the binary numbers which computers use. Both of these problems are yielding to treatment, but numerical analysis still has a fundamental flaw: it cannot pick the right equation to do a job if you do not already know which one to use.

Many of even the most day-to-day tasks of science and engineering involve recasting old equations into new forms. Most of this work takes little more than the algebra that 12-year-olds learn in school. In theory, rearranging equations is the sort of repetitive and tedious work at which computers excel. Yet in the 20 or so years that researchers have been trying to teach computers how to do symbolic mathematics (the parts of mathematics in which symbols, like x, are used to stand for things) success has remained elusive.

The grandfather of symbolic-mathematics programs is Macsyma, a project begun in the late 1960s at the Massachusetts Institute of Technology (MIT). Algebra is a snap for Macsyma. So is most of calculus, and many branches of mathematics too obscure to mention. Though Macsyma can do more mathematics than most undergraduates and many engineers, it suffers from several flaws. One that it shares with many other computer-maths programs is that it can be awkward to display a formula on a computer screen. Computers simply cannot print many of the squiggles that mathematicians use.

Another problem is inconsistency. Because Macsyma evolved over the years as a test-bed for whatever new mathematical technique was sweeping MIT that year, various parts of the program make different assumptions. One Macsyma user, a young physicist called Mr Stephen Wolfram, thought he could do better, and did so—twice. His first effort, SMP, was created at the California Institute of Technology and licensed to a Californian software company called Inference, which sold about 200 copies.

Mr Wolfram's second-generation symbolic-maths program, called Mathematica, caught the eye of Mr Steve Jobs, a founder of the computer maker, Apple. A copy of Mathematica comes free with Mr Jobs's new NeXT computer. Inference was so pessimistic about the market for symbolic-maths programs that it got rid of SMP to concentrate on expert systems. Mathematica, however, is enjoying popularity and some commercial success.

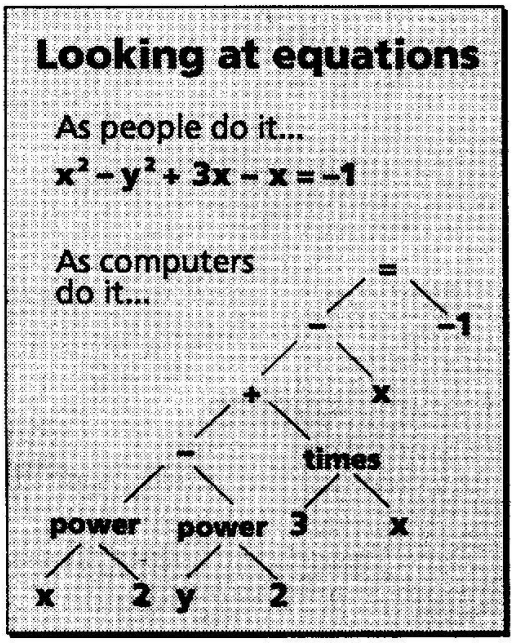
Mathematica is by no means the last word in symbolic mathematics. Its critics point out that it will occasionally do things to an equation that are plain wrong. Most of the errors are in obscure niches of mathematics. Mathematica's supporters argue that anyone treading in such dangerous areas should realise that there are bound to be pitfalls.

The problem lies in the way symbolic-maths programs keep track of what they are doing. The first steps in, say, solving for y in the equation $x^2 - y^2+3x - x=-1$ is to make sure that the equation is not a nonsense, and to get it into a form that the computer can cope with. Both can be taken by parsing the equation into a hierarchical structure called a tree (see diagram on next page).

The answer to an equation represented as a tree can be calculated by applying the operator (eg, + or −) at each node of the tree to the values of the branches hanging down from it. If those branches have values that are numbers, you get a numeric answer. But the real advantage of the tree is that it also allows the computer to manipulate symbols—to simplify an equation or to transform one equation into another. The trick here is to give the program a set of rules which state how parts of an equation can be rewritten, then to search through the tree for places to apply them.

One rule for simplifying an equation might say that any minus node with two of

the same variables attached to it can be transformed into a single node whose value is the left-hand variable minus the right-hand variable. Thus $x - x$ could be transformed to 0. The snag is that even this apparently trivial transformation is sometimes—albeit very rarely—wrong. What if x stands for a function which yields a random number? One random number minus another is probably not zero. Other exceptions can be found.



Mr Richard Fateman, one of the creators of Macsyma and now a professor at the University of California at Berkeley, points out that it quickly gets unwieldy to keep track of the exceptions to the rules needed to transform equations in a computer. So some programs let a few errors slip by. Others—notably at the University of Waterloo in Ontario and at IBM's Thomas J Watson Laboratories—are trying to help computers control such complexity by building more knowledge of mathematics into their programs.

The idea is to keep a sharper eye on what sort of things a symbol can stand for. Using a branch of mathematics called modern or abstract algebra, mathematicians can show what transformations can consistently be wrought on what numbers, functions and so on. In theory, if the maths program can keep track of which "domain" it is working in, it can avoid illegal transformations. Unfortunately, keeping track of this knowledge is tricky. People flit from the rules of mathematics involving fractions to those of whole numbers and on to those of mathematical functions without even thinking about it. Literal-minded computers cannot yet keep up.

Sceptics will say that even if researchers succeed in creating this next generation of symbolic-maths programs, computers will still not be able to do what many mathematicians consider to be their real work: to find and prove new theorems. Computers can prove any theorem that a person can—though it may take more than one person's lifetime to do it. But a computer cannot distinguish between a breakthrough and trivia. Although it is more mundane, rearranging equations is thus a more likely chore for computers—so long as they can learn to do even that right. ■