MATHEMATICS IN COMPUTER SCIENCE AND VICE VERSA

by

Michael Cowling*

I propose to describe how computer science depends heavily on mathematics, and how mathematics has been revolutionised by the advent of the computer. The moral of my discussion is that to be a good mathematician, some familiarity with computing is nowadays vital, and conversely, to be a good computer scientist, a solid grounding in mathematics is equally important.

Areas in computer science which depend heavily on mathematical ideas include complexity theory and program verification. Two of the areas of mathematics which have developed remarkably because of the computer are called fluid dynamics and topological dynamics. I will describe each of these very briefly.

First let us look at how mathematics plays a role in computer science. I want to describe briefly the mathematical ideas of countability and of combinatics, and then indicate how they fit into computing.

Last century, Georg Cantor introduced the idea of cardinality, to count infinite sets. Recall that a "set" is an object made up of "elements" - we write $x \in S$ to indicate "x is an element of S" - and usually some explicit rule is given for describing the set in terms of its elements. For instance $\{3,4,5\}$ denotes the set whose elements are the numbers 3,4 and 5, $\mathbb{N}$ denotes the set of natural numbers $\{0,1,2,3,4,\ldots\}$, where the dots are supposed to indicate that the other elements of the set are obvious, and $(0,1)$ denotes the set of all (real) numbers between 0 and 1, exclusive. Cantor observed that a finite set S has n elements exactly when there is a function from $\{1,2,3,\ldots,n\}$ to S which assumes each element of S as a value exactly once. Thus, the function $f(x) = x + 2$ has the property that, as x varies over $\{1,2,3\}$ the values $f(x)$ vary over $\{3,4,5\}$: the cardinality of $\{3,4,5\}$ is 3. Cantor said that a set S has cardinality $\aleph_0$ (aleph-null) when there is a function f from $\mathbb{N}$ to S with the same property (we usually say f is a bijection (or an isomorphism)) Here are some examples of sets with cardinality $\aleph_0$:

Michael Cowling is Professor of Pure Mathematics at the University of New South Wales.

1) N itself: take $f(x) = x$, and as x varies over N, $f(x)$ does too;

2) the even positive integers $\{2,4,6,8,10,12,\ldots\}$: take $f(x) = 2x + 2$, and as x varies over $\{0,1,2,3,\ldots\}$, $f(x)$ varies over the even positive integers.

3) all the integers $\{\ldots,-4,-3,-2,-1,0,1,2,3,4,\ldots\}$: take

$$f(x) = \left\{ \begin{array}{l} x/2 \text{ for } x \text{ even} \\ \\ -(\frac{x+1}{2}) \text{ for } x \text{ odd} \end{array} \right\} = (-1)^n \frac{x}{2} + \frac{(-1)^n+(-1)}{4}.$$

These three examples indicate that there are infinite sets with cardinality $\aleph_0$ which are smaller sets than N, and others which are larger than N. After Cantor found Q, the set of all rational numbers, which seems much bigger than N, also has cardinality $\aleph_0$, i.e. there is a function f from N to Q which is a bijection (though it's difficult to write explicitly), it seemed reasonable to suppose that all infinite sets have cardinality $\aleph_0$. But then he showed that (0,1) (which is clearly infinite, since all the numbers $\frac{1}{n}$ $(n>2)$ lie in (0,1), for positive integral n) does not have cardinality $\aleph_0$. His argument goes as follows: he assumes (0,1) has cardinality $\aleph_0$, and then argues that something incorrect happens; this means his assumption of cardinality $\aleph_0$ must be wrong. Indeed, if we could find a bijection f from N to (0,1), then we could list the values f(0), f(1), f(2), etc. as decimal expansions:

$$f(0) = 0.a_1a_2a_3a_4\ldots$$
$$f(1) = 0.b_1b_2b_3b_4\ldots$$
$$f(2) = 0.c_1c_2c_3c_4\ldots$$

and so on. We then write down another decimal expansion:

$$r = 0.\dot{a}_1\dot{b}_2\dot{c}_3\dot{d}_4\ldots .$$

where, for $k = 0,1,2,\ldots,9$, $\dot{k} \neq k$; more precisely, $\dot{k}$ means 1 unless $k = 1$, and $\dot{1} = 2$. The decimal number r lies between 0 and 1, and differs from $f(0)$ in the 1st decimal place, from $f(1)$ in the 2^{nd} decimal place, from $f(2)$ in the 3^{rd} decimal place, and so on. Then r is not equal to any of the numbers $f(0)$, $f(1)$, $f(2)$, etc.; this means that not every element of $(0,1)$ is taken as a value $f(n)$ for some n. But this contradicts the assumption that f is a bijection from $\mathbb{N}$ to $(0,1)$. It follows that $(0,1)$ has cardinality different from $\aleph_0$.

What, you may well ask, does this have to do with computer science? There are many things we may expect of a computer: to do word-processing, to store and regurgitate information on demand, and even to do mathematics. These look different to us, but to the computer it's all just a question of turning semiconductors on and off, of sending pulses along wires, and so on. And any limitations on what computers can do which will show up in database manipulation will show up in a slightly different form in doing mathematics, and vice versa. One of these limitations shows up in mathematics because Cantor did mathematics last century - had he worked on databases, it would now be formulated in different terms. This limitation is called computability: If I write

$$f(x) = x^2 + 40x + 41$$

then this is a "computable function": one can design a computer program which given (say) any rational number x, will regurgitate $f(x)$ after some finite amount of time. (This will probably be sooner if $x = 1$ than if $x = 10^{100}$, but never mind). However, the set of all computable functions has cardinality $\aleph_0$, while the set of all functions has larger cardinality. In other words, there are "non-computable functions". It took a long time for people to find an example of a non-computable function; it has now been done (See Scientific American, May 1984, pp 70-80).

The next example I want to consider is called the "travelling salesman problem". When a new product comes on the market, a salesman tries to show it to all his clients as fast as possible (and using as little petrol as possible). For example, a Sydney-based salesman with clients in Sydney, Camden, Mittagong, Goulburn and Gundagai will visit them in that order, because they are all in the same direction, but if he must go to Sydney, Windsor, Penrith, Camden, Campbelltown, Picton, Mittagong and Wollongong it's not so obvious. We have a map like Figure 1:

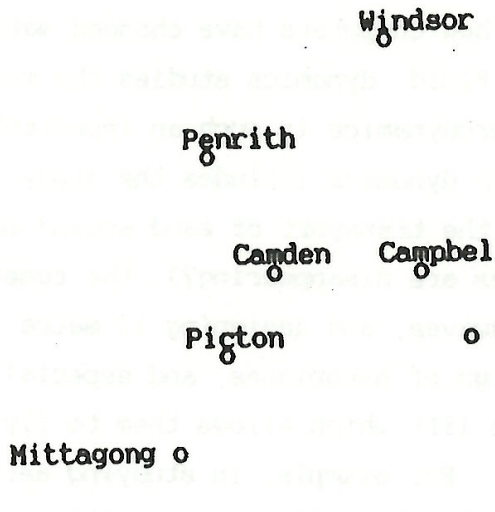

Figure 1

and it's not clear if it's better to go Sydney-Windsor-Penrith-Camden-Campbelltown-Picton-Mittagong-Wollongong-Sydney or Sydney-Windsor-Penrith-Camden-Picton-Mittagong-Wollongong-Campbelltown-Sydney, or if there's a better route. What we need is a list of places and the distances between them, and then we can work out the best route. This is called combinatorial optimisation: of the different combinations (routes) we seek the optimum (shortest). Mathematicians have been fretting about this sort of problem since travelling salesmen got wheels, and have a lot to say about it. They know that the problem gets very complicated very quickly as the number of places to visit goes up: if there are N places to go to, there are N! possible routes; so in the above example, the number of possibilities (starting and ending at Sydney) are 5020, though lots of these can be discarded immediately. A recent case study treated over 300 places to visit - the mind boggles at the number of possibilities.

How does this affect computing (other than travelling computer salesman)? Many problems in computer science are of a similar nature to the travelling salesman problem, because the order of doing things affects the time taken to run a program, and the ideas of combinatorial optimization have been used successfully in looking for faster search algorithms (find a mis-spelt word in twenty pages of document) and sort algorithms (arrange a list of people first in alphabetical order, then in order of age). The idea of algorithmic complexity (which means how long will it take to solve the travelling salesman problem, or how long will it take to sort a long list) are important in large scale operations. One U.S. Government agency spent many millions of dollars updating computer equipment when a particularly fast algorithm was shown mathematically to be fastest possible.

Now let us look at two examples of how computers have changed mathematics, fluid dynamics and topological dynamics. Fluid dynamics studies the motion of fluids, including gases, such as air, but aerodynamics is such an important part of fluid dynamics that it has its own name. Fluid dynamics includes the study of water waves, which sheds light on such questions as the transport of sand around beaches (did you know that all around the world, beaches are disappearing?), the construction of sea walls and breakwaters, predicting tidal waves, and designing 12 metre yacht hulls and keels. Aerodynamics includes the design of aeroplanes, and especially their wings, which are the part responsible for the lift which allows them to fly. All of these are mathematically very complicated. For example, in studying aerofoils (a fancy name for aeroplane wings), we usually imagine that we are seated on the aeroplane, and that the air is rushing past us, as in the diagram:

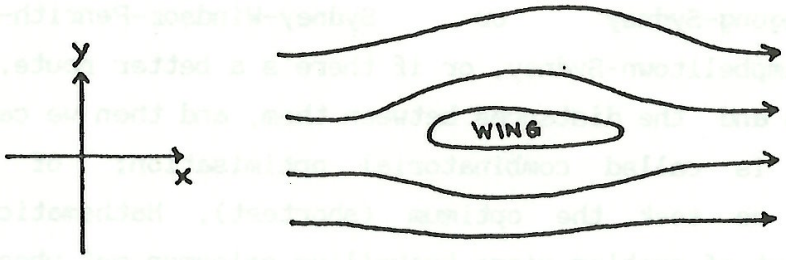

Figure 2: Section of aeroplane wing, with air flowing past

This is what actually happens in wind tunnels, which are used to test aerofoils experimentally.

To describe such a flow, we need to take into account the pressure and temperature of the air, which may vary from point to point and in time, and its velocity of motion, which also varies from point to point and in time. If we consider just the two dimensional picture of Figure 2, we need to work with functions $p(x,y,t)$ [the pressure] $T(x,y,t)$ [the temperature], $v_x(x,y,t)$ [the x-component or horizontal component of the velocity] and $v_y(x,y,t)$ [the y-component or vertical component of the velocity] All of these will be linked by various equations, (which express such physical principle such as the conservation of mass or of energy), usually involving derivatives of the other quantities. In order to describe such a flow in three dimensions, we need functions of three variables and a third velocity

component, and the equations became even more complex. We would probably simplify matters by assuming constant temperature and constant velocity, but things are still difficult.

In theory, if we could solve such equations, we could design the best (in the sense of most efficient) aeroplane or the best 12 metre yacht in the world. In practice, we cannot solve the equations involved. However, with the aid of very powerful computers, we are beginning to be able to say something about the solutions of these equations. For only $15 million, you too can have a computer which is powerful enough to tackle this sort of mathematics with. It is a sobering thought that, while Ben Lexcen was dragging model yachts through the water for months on end in order to come up with the Australia II keel which won the America's Cup, the Lockheeds, the Boeings and the McDonnell-Douglases of this world were designing their next generation of aircraft with the aid of supercomputers, and shutting down their windtunnels. It seems that a lot of money is spent on yacht design, but it is trivial compared to the investment in aircraft development. Nevertheless it is likely that America's Cup winners in the 2000's will be designed by computers, if only because computers are getting cheaper and cheaper.

Our last example is topological dynamics. This is a fancy name for the study of $f(x)$, $f(f(x))$, $f(f(f(x)))$, and so on. Here is an example. Suppose that, on a tropical island in the middle of the ocean, enough fruit grows each year to feed 10,000 fruit bats, but no more. Suppose also that each spring fruit bats reproduce, so that, in ideal conditions, N fruit bats give birth to αN baby fruit bats [α is a constant of proportionality] and hence after one spring, there are a total of $N + \alpha N$ fruit bats. A simple model of population growth says that if there are N bats one year, then there are $(1+\alpha)N$ the following year, $(1+\alpha)^2N$ the next year, $(1+\alpha)^3N$ the year later, and so on. This corresponds to letting $f(n) = (1+n)N$, and considering $f(N)$, $f(f(N))$, $f(f(f(N)))$, and so on. A slightly less simple model of population growth takes into account the limits to growth posed by the food supply, and says that if the population is close to 10,000, then some bats will not reproduce, and many others will die of starvation, so that if the population of bats one year is N, then the next population the following year will be $g(N)$, where

$$g(N) = (1+\alpha)N(1 - \frac{N}{10000}).$$

If instead of considering the number of bats present, N, we consider the fraction $x = N/10,000$ of the maximum population present, we find ourselves discussing the function

$$h(x) = \lambda x(1-x),$$

where $\lambda = 1 + \alpha$. It turns out that the behaviour of $h(x)$, $h(h(x))$, $h(h(h(x)))$,... depends in a remarkable way on the parameter λ. If λ is very small and positive $h(h(h...(h(x))...))$ becomes almost constant as the number of h's increases; this corresponds to a stable population. If λ is a little bigger, then $h(h(h...(h(x))...))$ tends to a "limit cycle": one year the population is N, the next year something different, N' say, and the year after is N again, and then N' again, and so on. As λ gets larger, the "limit cycle" gets more complicated, and repeats itself every 4 years, or every 8 years, or every 16 years, ... As λ gets close to 4, the behaviour of $h(h(h(...(h(x))...))$ becomes "chaotic", that is, there is no discernible pattern to it. This phenomenon, and in particular the "period doubling" and the "onset of chaos" was discovered experimentally; subsequently theory confirmed what had been found with the computer. See Scientific American, Nov. 1981, 16-29 and Aug, 1985, 8-14.

And now I conclude. In this day and age, you cannot do mathematics without some familiarity with computing. And, increasingly, you cannot do computer science without a very solid base in mathematics. If you are planning to go to the University soon to do either mathematics or computer science, think again. Do both!

<> <> <> <>