# Computers in Science and Mathematics

1984

*Computation offers a new means of describing and investigating scientific and mathematical systems. Simulation by computer may be the only way to predict how certain complicated systems evolve.*

Scientific laws give algorithms, or procedures, for determining how systems behave. The computer program is a medium in which the algorithms can be expressed and applied. Physical objects and mathematical structures can be represented as numbers and symbols in a computer, and a program can be written to manipulate them according to the algorithms. When the computer program is executed, it causes the numbers and symbols to be modified in the way specified by the scientific laws. It thereby allows the consequences of the laws to be deduced.

Executing a computer program is much like performing an experiment. Unlike the physical objects in a conventional experiment, however, the objects in a computer experiment are not bound by the laws of nature. Instead they follow the laws embodied in the computer program, which can be of any consistent form. Computation thus extends the realm of experimental science: it allows experiments to be performed in a hypothetical universe. Computation also extends theoretical science. Scientific laws have conventionally been constructed in terms of a particular set of mathematical functions and constructs, and they have often been developed as much for their mathematical simplicity as for their capacity to model the salient features of a phenomenon. A scientific law specified by an algorithm, however, can have any consistent form. The study of many complex systems, which have resisted analysis by traditional mathematical methods, is consequently being made possible through computer experiments and computer models. Computation is emerging as a major new approach to science, supplementing the long-standing methodologies of theory and experiment.

Originally published with illustrations under the title "Computer Software in Science and Mathematics" in *Scientific American*, volume 251, pages 188–203 (September 1984).

There are many scientific calculations, of course, that can be done by conventional mathematical means, without the aid of the computer. For example, given the equations that describe the motion of electrons in an arbitrary magnetic field, it is possible to derive a simple mathematical formula that gives the trajectory of an electron in a uniform magnetic field (one whose strength is the same at all positions). For more complicated magnetic fields, however, there is no such simple mathematical formula. The equations of motion still yield an algorithm from which the trajectory of an electron can be determined. In principle the trajectory could be worked out by hand, but in practice only a computer can go through the large number of steps necessary to obtain accurate results.

A computer program that embodies the laws of motion for an electron in a magnetic field can be used to perform computer experiments. Such experiments are more flexible than conventional laboratory experiments. For example, a laboratory experiment could readily be devised to study the trajectory of an electron moving under the influence of the magnetic field in a television tube. No laboratory experiment, however, could reproduce the conditions encountered by an electron moving in the magnetic field surrounding a neutron star. The computer program can be applied in both cases.

The magnetic field under investigation is specified by a set of numbers stored in a computer. The computer program applies an algorithm that simulates the motion of the electron by changing the numbers representing its position at successive times. Computers are now fast enough for the simulations to be carried out quickly, and so it is practical to explore a large number of cases. The investigator can interact directly with the computer, modifying various aspects of a phenomenon as new results are obtained. The usual cycle of the scientific method, in which hypotheses are formulated and then tested, can be followed much faster with the aid of the computer.

Computer experiments are not limited to processes that occur in nature. For example, a computer program can describe the motion of magnetic monopoles in magnetic fields, even though magnetic monopoles have not been detected in physical experiments. Moreover, the program can be modified to embody various alternative laws for the motion of magnetic monopoles. Once again, when the program is executed, the consequences of the hypothetical laws can be determined. The computer thus enables the investigator to experiment with a range of hypothetical natural laws.

The computer can also be used to study the properties of abstract mathematical systems. Mathematical experiments carried out by computer can often suggest conjectures that are subsequently established by conventional mathematical proof. Consider a mathematical system that can be introduced to model the path of a beam of electrons traveling through the magnetic fields in a circular particle accelerator. The transverse displacement of an electron as it passes a point on one of its revolutions around the accelerator ring is given by some fraction $x$ between 0 and 1. The value of the fraction corresponding to the electron's displacement on the next revolution is then $ax(1 - x)$, where $a$ is a number that can range between 0 and 4. The formula

gives an algorithm from which the sequence of values for the electron's displacement can be worked out.

A few trials show how the properties of the sequence depend on the value of $a$. If $a$ is equal to 2 and the initial value of $x$ is equal to .8, the next value of $x$, which is given by $ax(1 - x)$, is equal to .32. If the formula is applied again, the value of $x$ obtained is .4352. After several iterations the sequence of values for $x$ converges to .5. Indeed, when $a$ is small and $x$ is any fraction between 0 and 1, the sequence quickly settles down to give the same value of $x$ for each revolution of the electron.

As $a$ increases, however, a phenomenon called period doubling can be observed. When $a$ reaches 3, the sequence begins to alternate between two values of $x$. As $a$ continues to increase, first four, then eight and finally, when it reaches about 3.57, an entire range of values for $x$ appear. This behavior could not readily be guessed from the construction of the mathematical system, but it is immediately suggested by the computer experiment. The detailed properties of the system can then be established by a conventional proof.

The mathematical processes that can be described by a computer program are not limited to the operations and functions of conventional mathematics. For example, there is no conventional mathematical notation for the function that reverses the order of the digits in a number. Nevertheless, it is possible to define and apply the function in a computer program. The computer makes it practical to introduce scientific and mathematical laws that are intrinsically algorithmic in nature. Consider the chain of events set up when an electron accelerated to a high energy is fired into a block of lead. There is a certain probability that the electron emits a photon of a particular energy. If a photon is emitted, there is a certain probability that it gives rise to a second electron and a positron (the antiparticle of the electron). Each member of the pair can in turn emit more photons, so that a cascade of particles is eventually generated. There is no simple mathematical formula that can describe even the elements of the process. Nevertheless, an algorithm for the process can be incorporated into a computer program, and the outcome of the process can be deduced by executing the program. The algorithm serves as the basic law that describes the process.

The mathematical basis of most conventional models of natural phenomena is the differential equation. Such an equation gives relations between certain quantities and their rates of change. For example, a chemical reaction proceeds at a rate proportional to the concentrations of the reacting chemicals, and that relation can be expressed by a differential equation. A solution to the equation would give the concentration of each reactant as a function of time. In some simple cases it is possible to find a complete solution to the equation in terms of standard mathematical functions. In most cases, however, no such exact solution can be obtained, and one must resort to approximation.

The commonest approximations are numerical. Suppose one term of a differential equation gives the instantaneous rate of change of a quantity with time. The term

can be approximated by the total change in the quantity over some small interval and then substituted into the differential equation. The resulting equation is in effect an algorithm that determines the approximate value of the quantity at the end of an interval, given its value at the beginning of the interval. By applying the algorithm repeatedly for successive intervals, the approximate variation of the quantity with time can be found. Smaller intervals yield more accurate results. The calculation required for each interval is quite simple, but in most cases it must be repeated many times to achieve an acceptable level of accuracy. Such an approach is practical only with a computer.

The numerical methods embodied in computer programs have been employed to find approximate solutions to differential equations in a wide variety of disciplines. In some cases the solutions have a simple form. In many cases, however, the solutions show complicated, almost random behavior, even though the differential equations from which they arise are quite simple. For such cases experimental mathematics must be used.

In practical applications one often finds not only that differential equations are complicated but also that there are many of them. For example, the theoretical models of nuclear explosions employed in the design of weapons and the study of supernovas involve hundreds of differential equations that describe the interactions of many isotopes. In practice such models are always used in the form of computer programs: only a computer can follow the interrelations among so many quantities.

The results of some numerical calculations, such as the abundance of helium in the universe, can be stated as single numbers. In most cases, however, one is concerned with the variation of certain quantities as the parameters of a calculation are changed. When the number of parameters is only one or two, the results can be displayed as a graph. When there are more than two parameters, however, the results often can be stated succinctly only as a mathematical formula. Exact formulas usually cannot be found, but it is often possible to derive approximate formulas. Such formulas are particularly convenient because, unlike graphs or tables of numbers, they can be inserted directly into other calculations.

A common form for an approximate formula is a series of terms. Each term includes a variable raised to some power; the power is larger in each successive term. When the value of the variable is small, the terms in the series become progressively smaller; thus for small values of $x$ the sum of the first few terms in an infinite series such as $1 - x + x^2 - x^3 + \ldots$ gives an accurate approximation to the sum of the entire series, which is $1/(1+x)$. The first few terms in a series are usually easy to evaluate, but the complexity of the terms increases rapidly thereafter. In order to evaluate terms that include large powers of $x$ the computer becomes essential.

In principle computer programs can operate with any well-defined mathematical construct. In practice, however, the kinds of construct that can be used in a particular program are largely determined by the computer language in which the program is

written. Numerical methods require only a limited set of mathematical constructs, and the programs that embody such methods can be written in general-purpose computer languages such as C, FORTRAN or BASIC. The derivation and manipulation of formulas require operations on higher-level mathematical constructs such as algebraic expressions, for which new computer languages are needed. Among the languages of this kind now in use is the SMP language that I have developed.

SMP is a language for manipulating symbols. It operates not only with numbers but also with symbolic expressions that can represent mathematical formulas. For example, in SMP the algebraic expression $2x-3y+5x-y$ would be simplified to the form $7x-4y$. This transformation is a general one, valid for any possible numerical values of $x$ and $y$. The standard operations of algebra and mathematical analysis are among the fundamental instructions in SMP.

The SMP language also includes operations that allow higher-level mathematical constructs to be defined and manipulated, much as they are in ordinary mathematical work. Real numbers (which include all rational and irrational values) as well as complex numbers (which have both a real and an imaginary part) are fundamental in SMP. The mathematical constructs known as quaternions, which are generalizations of the complex numbers, are not fundamental. They can nonetheless be defined in SMP, and rules can be specified for their addition and multiplication. In this way the mathematical knowledge of SMP can be extended.

Some of the advantages of a language such as SMP can be compared to the advantages of using a calculator instead of a table of logarithms. By now the widespread availability of electronic calculators and computers has made such tables obsolete: it is far more convenient to call on an algorithm in a computer to obtain a logarithm than it is to look up the result in a table. Similarly, with a language such as SMP it has become possible to make the entire range of mathematical knowledge available in algorithmic form. For example, the calculation of integrals, conventionally done with the aid of a book of tables, can increasingly be left to a computer. The computer not only carries out the final calculations quickly and without error but also automates the process of finding the relevant formulas and methods.

In SMP an expanding collection of definitions is being assembled in order to provide for a wide variety of mathematical calculations. One can now find in SMP the definition of variance in statistics, and one can immediately apply the definition to calculate the variance in a particular case. Such definitions enable programs written in the SMP language to call on increasingly sophisticated mathematical knowledge.

Differential equations give adequate models for the overall properties of physical processes such as chemical reactions. They describe, for example, the changes in the total concentration of molecules; they do not, however, account for the motions of individual molecules. These motions can be modeled as random walks: the path of each molecule is like the path that might be taken by a person in a milling crowd. In the simplest version of the model the molecule is assumed to travel in a straight line

until it collides with another molecule; it then recoils in a random direction. All the straight-line steps are assumed to be of equal length. It turns out that if a large number of molecules are following random walks, the average change in the concentration of molecules with time can in fact be described by a differential equation called the diffusion equation.

There are many physical processes, however, for which no such average description seems possible. In such cases differential equations are not available and one must resort to direct simulation. The motions of many individual molecules or components must be followed explicitly; the overall behavior of the system is estimated by finding the average properties of the results. The only feasible way to carry out such simulations is by computer experiment: essentially no analysis of the systems for which analysis is necessary could be made without the computer.

The self-avoiding random walk is an example of a process that can apparently be studied only by direct simulation. It can be described by a simple algorithm that is similar to the ordinary random walk. It differs in that the successive steps in the self-avoiding random walk must not cross the path taken by any previous steps. The folding of long molecules such as DNA can be modeled as a self-avoiding random walk.

The introduction of the single constraint makes the self-avoiding random walk much more complicated than the ordinary random walk. Indeed, there is no simple average description, analogous to the diffusion equation, that is known for the self-avoiding random walk. In order to investigate its properties it seems one has no choice but to carry out a direct computer experiment. The procedure is to generate a large number of sample random walks, choosing a random direction at each step. The properties of all the walks are then averaged. Such a procedure is an example of the Monte Carlo method, so called because its application depends on the element of chance.

Several examples have been given of systems whose construction is quite simple but whose behavior is extremely complicated. The study of such systems is leading to a new field called complex-systems theory, in which the computational method plays a central role. The archetypal example is fluid turbulence, which develops, for example, when water flows rapidly around an obstruction. The set of differential equations satisfied by the fluid can easily be stated. Nevertheless, the patterns of fluid flow to which the equations give rise have largely defied mathematical analysis or description. In practice the patterns are found either through observation of the actual physical system or, as far as possible, through computer experiment.

It is suspected there is a set of mathematical mechanisms common to many systems that give rise to complicated behavior. The mechanisms can best be studied in systems whose construction is as simple as possible. Such studies have recently been done for a class of mathematical systems known as cellular automata. A cellular automaton is made up of many identical components; each component evolves according to a simple set of rules. Taken together, however, the components generate behavior of essentially arbitrary complexity.

The components of a cellular automaton are mathematical "cells," arranged in one dimension at a sequence of equally spaced points along a line or in two dimensions on a regular grid of squares or hexagons. Each cell carries a value chosen from a small set of possibilities, often just 0 and 1. The values of all the cells in the cellular automaton are simultaneously updated at each "tick" of a clock according to a definite rule. The rule specifies the new value of a cell, given its previous value and the previous values of its nearest neighbors or some other nearby set of cells.

Consider a one-dimensional cellular automaton in which each cell can have the value 0 or 1. Even in such a simple case the overall behavior of the cellular automaton can be quite complex; the most effective way to investigate the behavior is by computer experiment. Most of the properties of cellular automata have in fact been conjectured on the basis of patterns generated in computer experiments. In some cases they have later been established by conventional mathematical arguments.

Cellular automata can serve as explicit models for a wide variety of physical processes. Suppose ice is represented on a two-dimensional hexagonal grid by cells with the value 1 and water vapor is represented by cells with the value 0. A cellular-automaton rule can then be used to simulate the successive stages in the freezing of a snowflake. The rule states that once a cell is frozen it does not thaw. Cells exposed at the edge of the growing pattern freeze unless they have so many ice neighbors that they cannot dissipate enough heat to freeze. Snowflakes grown in a computer experiment from a single frozen cell according to this rule show intricate treelike patterns, which bear a close resemblance to real snowflakes. A set of differential equations can also describe the growth of snowflakes, but the much simpler model given by the cellular automaton seems to preserve the essence of the process by which complex patterns are created. Similar models appear to work for biological systems: intricate patterns of growth and pigmentation may be accounted for by the simple algorithms that generate cellular automata.

Simulation by computer is the only method now used for investigating many of the systems discussed so far. It is natural to ask whether simulation, as a matter of principle, is the most efficient possible procedure or whether there is a mathematical formula that could lead more directly to the results. In order to address the question the correspondence between physical and computational processes must be studied more closely.

It is presumably true that any physical process can be described by an algorithm, and so any physical process can be represented as a computational process. One must determine how complicated the latter process is. In cellular automata the correspondence between physical and computational processes is particularly clear. A cellular automaton can be regarded as a model of a physical system, but it can also be regarded as a computational system closely analogous to an ordinary digital computer. The sequence of initial cell values in a cellular automaton can be understood as abstract data or information, much like the sequence of binary digits in the memory

of a digital computer. During the evolution of a cellular automaton the information is processed: the values of the cells are modified according to definite rules. Similarly, the digits stored in the memory of the digital computer are modified by rules built into the central processing unit of the computer.

The evolution of a cellular automaton from some initial configuration may thus be viewed as a computation that processes the information carried by the configuration. For cellular automata exhibiting simple behavior the computation is a simple one. For example, it may serve only to pick out sequences of three consecutive cells whose initial values are equal to 1. On the other hand, the evolution of cellular automata that show complicated behavior may correspond to a complicated computation.

It is always possible to determine the outcome of a given number of steps in the evolution of a cellular automaton by explicitly simulating each step. The problem is whether or not there can be a more efficient procedure. Can there be a short cut to step-by-step simulation, an algorithm that finds the outcome after many steps in the evolution of a cellular automaton without effectively tracing through each step? Such an algorithm could be executed by a computer, and it would predict the evolution of a cellular automaton without explicitly simulating it. The basis of its operation would be that the computer could carry out a more sophisticated computation than the cellular automaton could and so achieve the same result in fewer steps. It would be as if the cellular automaton were to calculate 7 times 18 by explicitly finding the sum of seven 18's, while the computer found the same product according to the standard method for multiplication. Such a short cut is available only if the computer is able to carry out a calculation that is intrinsically more sophisticated than the calculation embodied in the evolution of the cellular automaton.

One can define a certain class of problems called computable problems that can be solved in a finite time by following definite algorithms. A simple computer such as an adding machine can solve only a small subset of these problems. There exist universal, or general-purpose, computers, however, that can solve any computable problem. A real digital computer is essentially such a universal machine. The instructions that can be executed by the central processing unit of the computer are rich enough to serve as the elements of a computer program that can embody any algorithm. A number of systems in addition to the digital computer have been shown to be capable of universal computation. Several cellular automata are among them: for example, universal computation has been proved for a simple two-dimensional cellular automaton with a 0 or a 1 in each cell. It is strongly suspected that several one-dimensional cellular automata are also universal computers. The simplest candidates have three possible values at each cell and rules of evolution that take account only of the nearest-neighbor cells.

Cellular automata that are capable of universal computation can mimic the behavior of any possible computer; since any physical process can be represented as a computational process, they can mimic the action of any possible physical system as

well. If there were an algorithm that could work out the behavior of these cellular automata faster than the automata themselves evolve, the algorithm would allow any computation to be speeded up. Because this conclusion would lead to a logical contradiction, it follows there can be no general short cut that predicts the evolution of an arbitrary cellular automaton. The calculation corresponding to the evolution is irreducible: its outcome can be found effectively only by simulating the evolution explicitly. Thus direct simulation is indeed the most efficient method for determining the behavior of some cellular automata. There is no way to predict their evolution; one must simply watch it happen.

It is not yet known how widespread the phenomenon of computational irreducibility is among cellular automata or among physical systems in general. Nevertheless, it is clear that the elements of a system need not be very complicated for the overall evolution of the system to be computationally irreducible. It may be that computational irreducibility is almost always present when the behavior of a system appears complicated or chaotic. General mathematical formulas that describe the overall behavior of such systems are not known, and it is possible no such formulas can ever be found. In that case, explicit simulation in a computer experiment is the only available method of investigation.

Much of physical science has traditionally focused on the study of computationally reducible phenomena, for which simple overall descriptions can be given. In real physical systems, however, computational reducibility may well be the exception rather than the rule. Fluid turbulence is probably one of many examples of computational irreducibility. In biological systems computational irreducibility may be even more widespread: it may turn out that the form of a biological organism can be determined from its genetic code essentially only by following each step in its development. When computational irreducibility is present, one must adopt a methodology that depends heavily on computation.

One of the consequences of computational irreducibility is that there are questions that can be asked about the ultimate behavior of a system but that cannot be answered in full generality by any finite mathematical or computational process. Such questions must therefore be considered undecidable. An example of such a question is whether a particular pattern ever dies out in the evolution of a cellular automaton. It is straightforward to answer the question for some definite number of steps, say 1,000: one need only simulate 1,000 steps in the evolution of the cellular automaton. In order to determine the answer for any number of steps, however, one must simulate the evolution of the cellular automaton for a potentially infinite number of steps. If the cellular automaton is computationally irreducible, there is no effective alternative to such direct simulation.

The upshot is that no calculation of any fixed length can be guaranteed to determine whether a pattern will ultimately die out. It may be possible to tell the fate of a particular pattern after tracing only a few steps in its evolution, but there is no general

way to tell in advance how many steps will be required. The ultimate form of a pattern is the result of an infinite number of steps, corresponding to an infinite computation; unless the evolution of the pattern is computationally reducible, its consequences cannot be reproduced by any finite computational or mathematical process.

The possibility of undecidable questions in mathematical models for physical systems can be viewed as a manifestation of Gödel's theorem on undecidability in mathematics, which was proved by Kurt Gödel in 1931. The theorem states that in all but the simplest mathematical systems there may be propositions that cannot be proved or disproved by any finite mathematical or logical process. The proof of a given proposition may call for an indefinitely large number of logical steps. Even propositions that can be stated succinctly can require an arbitrarily long proof. In practice there are many simple mathematical theorems for which the only known proofs are very long. In addition the cases that must be examined to prove or refute conjectures are often quite complicated. In number theory, for example, there are many cases in which the smallest number having some special property is extremely large; the number can often be found only by testing each whole number in turn. Such phenomena are making the computer an essential tool in many mathematical investigations.

Computational irreducibility implies many fundamental limitations on the scope of theories for physical systems. It may be possible to model a system at many levels, from simulating the motions of individual molecules to solving differential equations for overall properties. Computational irreducibility implies there is a highest level at which abstract models can be made; above that level results can be found only by explicit simulation.

When the level of description becomes computationally irreducible, undecidable questions also begin to appear. Such questions must be avoided in the formulation of a theory, much as the simultaneous measurement of the position and velocity of an electron—impossible according to the uncertainty principle—is avoided in quantum mechanics. Even if such questions are eliminated, there is still the practical difficulty of answering questions that in principle can be answered. The degree of difficulty depends strongly on the nature of the objects involved in the simulation. If the only way to predict the weather were to simulate the motions of every molecule in the atmosphere, no practical calculations could be carried out. Nevertheless, the relevant features of the weather can probably be studied by considering the interactions of large volumes of the atmosphere, and so useful simulations should be possible.

The efficiency with which a computationally irreducible system can be simulated depends on the computational sophistication of each step in its evolution. The steps in the evolution of the system can be simulated by instructions in a computer program. The fewer the instructions needed to reproduce each step, the more efficient the simulation. Higher-level descriptions of physical systems typically call for more sophisticated steps, much as single instructions in higher-level computer languages

correspond to many instructions in lower-level ones. One time step in the numerical approximation of a differential equation that describes a jet of gas requires a computation more sophisticated than the one needed to follow a collision between two molecules in the gas. On the other hand, each step in the higher-level description given by a differential equation accounts for an immense number of steps in the lower-level description of molecular collisions. The resulting gain in efficiency more than makes up for the fact that the individual steps are more sophisticated.

In general the efficiency of a simulation increases with higher levels of description, until the operations needed for the higher-level description are matched with the operations carried out directly by the computer doing the simulation. It is most efficient for the computer to be as close an analogue to the system being simulated as possible.

There is one major difference between most existing computers and physical systems or models of them: computers process information serially, whereas physical systems process information in parallel. In a physical system modeled by a cellular automaton the values of all the cells are updated together at each time step. In a standard computer program, however, the simulation of the cellular automaton is carried out by a loop that updates the value of each cell in turn. In such a case it is straightforward to write a computer program that performs a fundamentally parallel process with a serial algorithm. There is a well-established framework in which algorithms for the serial processing of information can be described. Many physical systems, however, seem to require descriptions that are essentially parallel in nature. A general framework for parallel processing does not yet exist, but when it is developed, more effective high-level descriptions of physical phenomena should become possible.

The introduction of the computer in science is comparatively recent. Already, however, computation is establishing a new approach to many problems. It is making possible the study of phenomena far more complex than the ones that could previously be considered, and it is changing the direction and emphasis of many fields of science. Perhaps most significant, it is introducing a new way of thinking in science. Scientific laws are now being viewed as algorithms. Many of them are studied in computer experiments. Physical systems are viewed as computational systems, processing information much the way computers do. New aspects of natural phenomena have been made accessible to investigation. A new paradigm has been born.