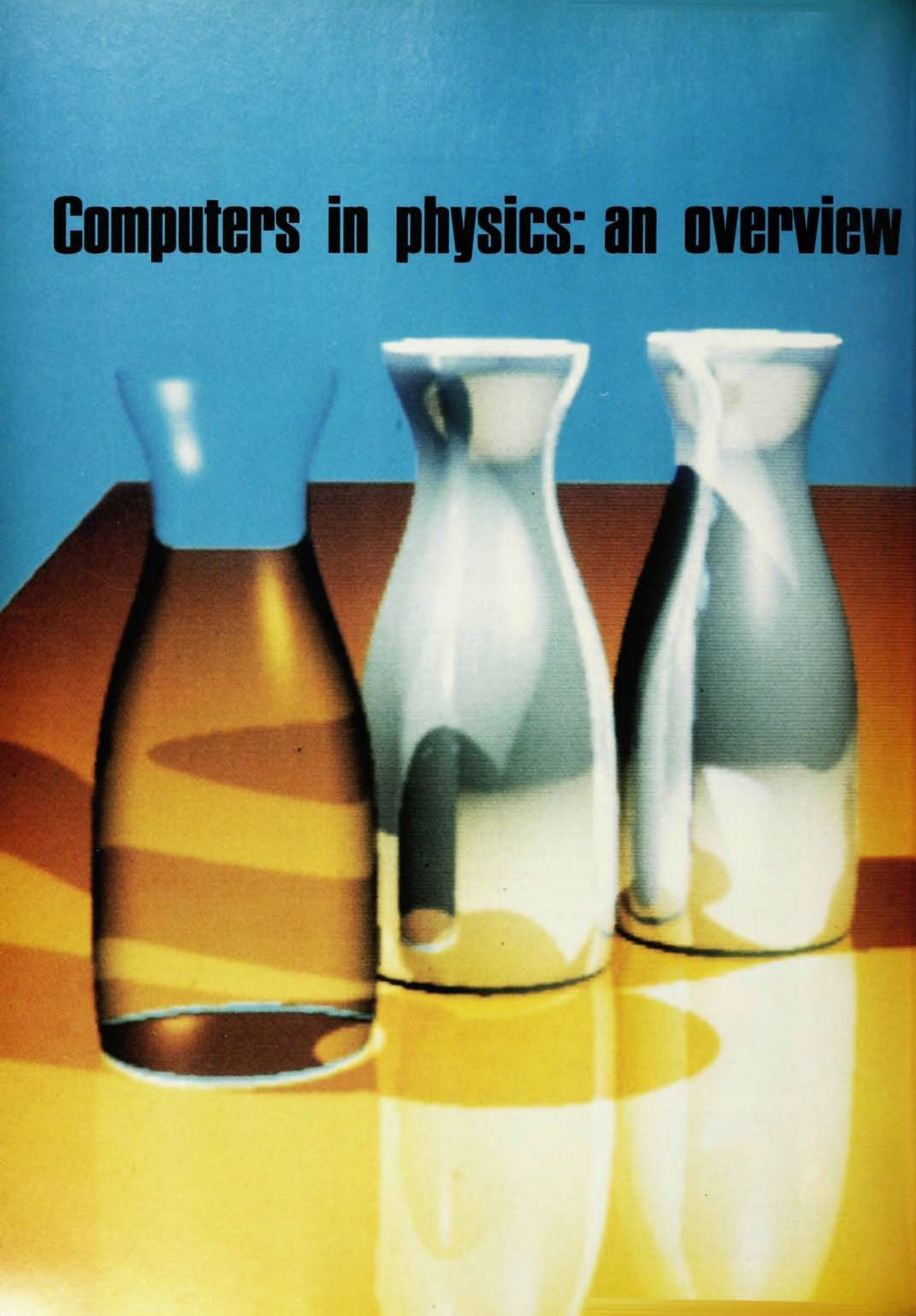
Computers in physics: an overview



While physicists debate which computer language is best and which computing philosophy is right, they are busy using computers to study subjects ranging from quarks to models of the Universe.

Donald R. Hamann

In the eyes of the general public, the computer has changed from a remote and recalcitrant source of error on monthly bills to a friendly and increasingly common household appliance. *Time* magazine, which names a "Man of the Year," in 1982 named the computer instead, and *The New York Times* runs a weekly computer column. With the popular press full of the subject, it hardly seems necessary to tell an audience of physicists that a computer consists of a central processing unit, a high-speed memory, mass-storage disks and so on. A large fraction of the total number of scientists active in research or development have ready access to computers.[1] I can safely assume that my readers have some level of familiarity with the basic operation of modern computers and with a few uses of computers in their fields of specialization. In this article, then, I will address a set of topics of special concern to physicists in the hope that doing so will stimulate further discussion. One indisputable fact is that the presence of computers will continue to expand in our science as in the rest of our lives, and some thoughtful reflection on where we are going and how we shall get there should be time well spent.

The physics literature is filled with the results of experimental and theoretical research performed with the help of computers. It is often difficult to find just how computers are being used, however, or just how users regard computation as a part of their overall contribution. In the course of gathering material for this article, it became apparent that some rather divergent views exist on the role of computation in theory. We will begin by addressing some questions of the philosophy of computer use that cut across various specialties. Then we will discuss a series of examples to make some of these distinctions more concrete. The examples range from one of the earliest computations to current work, and from widely known to highly specialized results. The distribution of these examples among fields of physics is not intended to reflect the distribution of computational efforts or contributions.

Software, the various codes that cause the machine to do what we want, is an area in which there are some divergent practices, if not viewpoints, between physicists and computer scientists. We will examine some of these issues before concluding with a look at the computational hardware available today and what one can anticipate.

Computed still life. This is an example of state-of-the-art computer graphics incorporating the principles of physical optics for reflective and refractive objects. Increasingly sophisticated visual displays are helping researchers to identify key features of large quantities of data. (Image by Michael Potmesil, Bell Laboratories.) Figure 1

Philosophy

I discern two schools of thought regarding the appropriate ways of approaching computation in theoretical physics. Both approaches require a model—a set of basic laws known or at least assumed to describe the physical system of interest. In one approach, simplifying physical principles governing key qualitative aspects of the system's behavior are to be discovered prior to the computation and embodied in the computer program to as great an extent as possible. The goal of the computation, then, is to produce quantitative predictions that can be compared with experiment to determine the range of validity of the principles, the accuracy of the model, or properties of the experimental system that are not subject to direct measurement. I will call this the "numerical analysis" or simply the "analysis" school.

The second school tries to build the basic laws of the model into the computer program in as direct a way as possible. The goals of the computation, in addition to those stated above, include the discovery of new simplifying physical principles by observing the computed behavior of the model. Because members of this school customarily refer to their computations as simulations (or boldly, on occasion, as experiments), I will refer to it as the "simulation" school.

Both approaches, of course, require the use of simplifying approximations to make the problem computationally feasible. There is no particular physical significance attached to such approximations, and ideally the error introduced can be bounded analytically or tested numerically on a simpler member of the class of problems. At worst, one can test effects of approximations only when substantially greater computational power becomes available, or by comparison to real-world experiments. In the latter case, of course, it is difficult to distinguish failures of the computational simplification from those of the models or principles.

The distinction between the two schools serves to categorize not only computations but also certain attitudes of the practitioners and their critics. One group maintains that a problem must be well understood before one can formulate the questions best answered by numerical computation. Analytic knowledge of as many pieces of a problem as possible will lend efficiency to the program and help one generalize from the specific cases worked out numerically. Expectations regarding qualitative features of the results will aid in testing the program; unexpected results will be highly suspect. The idea of an omniscient "black box," which would accurately compute any desired property from the basic parameters of a system, is regarded as pernicious by some members of this group. Too much of the physics is hidden from the user in the machinery of the box.

There are many adherents to the

Donald R. Hamann is head of the surface-physics research department at Bell Laboratories in Murray Hill, New Jersey.

0031-9228 / 83 / 0500 24-10 / $01.00

alternative view that new insights, as well as tests of conjectured mechanisms and, simply, useful numbers can flow from simulations of problems in a relatively "raw" form. In this view the box is not black but transparent—one can follow details of the internal dynamics of the simulated system that are not experimentally accessible in real systems. Furthermore, imposing too much analytic structure can condition the output to reflect incomplete or incorrect preconceived ideas. Some from this group propose that computational physics be regarded as a new discipline cutting across traditional subject lines on the basis of a common methodology.

An undercurrent in the debate between these schools is driven by differing perceptions of the intellectual respectability of computation in physics. There is undoubtedly a generational bias in the range of opinion. Some views of those who received their training when computer science was unknown as an academic discipline may not be shared by a recent physics graduate who seriously considered this as an alternative career. Those who decry a generation of intellects dulled by tv may sense an analogy in the video terminal. Viewpoints on the issue have serious consequences for the training of students, for the allocation of research support and for scientific communication.

Intellectual respectability may, in fact, only be tangential to a somewhat more elusive core issue. After all, computers don't conceptualize; it is physicists who do physics. Critics may really be saying that they think too little physics is being done *well* with computers, that the process is not sufficiently subject to peer scrutiny, or that too much time is being spent programming instead of thinking. The computer is a tool, and like any other, it can be used well or badly. It is intellectually neutral.

Numerical analysis

Computations in my category of analysis do not fall into any easily discernible subcategories, so I will begin with an example. The current–voltage characteristics of superconducting tunnel junctions contain a good deal of structure, and one of the early computer calculations in condensed-matter many-body physics demonstrated that one can relate this structure quantitatively to the phonon spectrum of the superconductor. The physical principle underlying this calculation came from the BCS theory of superconductivity. When this principle was embodied in a set of integral equations derived using an analytic many-body-theory approach, an approximate analytic solution gave encouraging qualitative agreement with the tunneling data.[2]

William McMillan, working at Bell Labs, introduced a computational procedure for solving the integral equations and iteratively refining the model so that the phonons fit the tunneling measurements quantitatively. The results were so successful at matching photon spectra measured by neutron scattering spectroscopy that the tunneling technique turned into an important alternative form of phonon spectroscopy, and McMillan's programs were widely distributed and used for this purpose.[3] This example satisfies my criteria for the category of analysis because the underlying simplifying principles of superconductivity were central to the calculation. The use of a "first-principles" approach to try to simulate the superconducting state would obviously have been futile.

Another good example is found in the theory of phase transitions in condensed matter. It was long known experimentally that the so-called critical behavior of many systems near their transition points (for example, ferromagnets near their Curie temperatures T_c) was incorrectly described by mean-field theory, an approximation that gave qualitatively correct analytic solutions far from the transition region. To be more specific, physical properties vary in a power-law fashion, as $(T - T_c)^x$, and the critical exponent x seldom has the "classical" value predicted by mean field theory. Lars Onsager's tour-de-force exact analytic solution of the two-dimensional Ising model showed that one could obtain non-classical exponents even for a highly idealized model. Unfortunately, the method of this analytic solution could not be generalized, nor could a general simplifying principle be extracted.

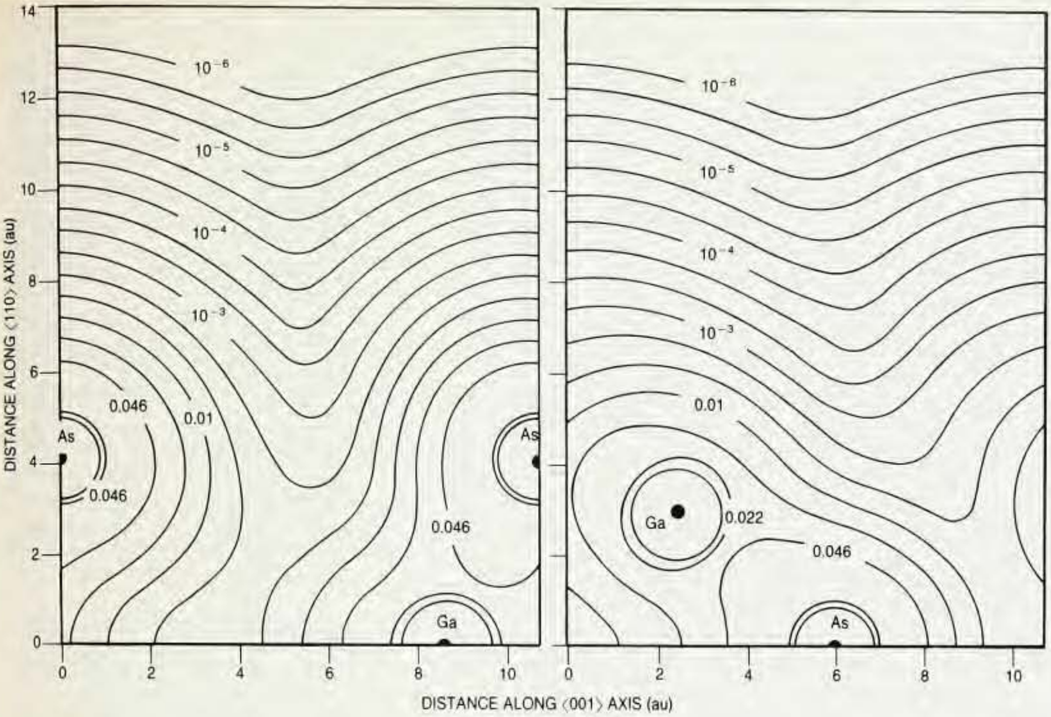
The theory advanced with the observation that the critical exponents describing several properties of a system obey simple interrelationships derivable from an assumed scale invariance of the free energy of the system at the critical point. To be more graphic, this suggested that if a microscope were available that could image the order parameter of the ordered phase (the magnetization in a ferromagnet, for example), the fluctuating order parameter at the critical point would look the same at any magnification. Kenneth Wilson, in the brilliant advance recognized by last year's Nobel prize in physics, used these ideas to derive recursion relations relating successive "coarse-grainings" of the free-energy functional. He developed a computer program to apply an approximate recursion relation numerically for a model closely related to the three-dimensional Ising model of a ferromagnet and succeeded in calculating explicitly its non-classical critical exponents.

Physicists have generalized this renormalization-group method and applied it to a wide variety of phase-transition problems, in the form of approximate analytic solutions and more exact numerical computations. In this example it was a new physical principle that permitted computations capable of solving a previously intractable set of problems. The initial computational test of the principle played a major role in establishing its utility.[4]

Electronic structure, studied in condensed-matter physics and in quantum chemistry, is one of the largest areas of computation in which the analysis approach is mandated. Suppose, for example, that we take the Schrödinger equation for half a dozen nuclei and a few dozen electrons interacting via the Coulomb interaction, along with boundary conditions based on the Pauli exclusion principle, and present this to a general-purpose program for solving multidimensional partial differential equations. The program would be hard pressed to discover the existence of atoms and molecules within this system, even given computational power many orders of magnitude beyond that achievable today.

Any practical calculation builds upon a large amount of prior knowledge, such as the adiabatic approximation, reasonable nuclear positions, the central-field form of wave functions near particular nuclei, and the dominant effect of the self-consistent electrostatic potential. A computation based on these principles gives a reasonable account of chemical bonding geometries and already constitutes a sizable computation for modest-sized molecules. A well-converged calculation on a benzene molecule, for example, requires 5 minutes on a Cray-1 computer. To achieve the accuracy necessary for useful bond-energy comparisons, one must include the effects of the dynamical correlations of the electrons, which can be done in a systematic manner for molecules. The computational complexity of current methods grows rapidly with the level of convergence and the size of the molecule because the dynamical correlations of the electrons are treated globally over the entire molecule. Such bond-energy calculations for benzene use about 30 minutes of Cray time.

For crystalline solids and similar extended systems, one can exploit the periodicity of the lattice to make the complexity of the calculation proportional to the number of atoms in a unit cell. The systematic treatment of electron correlation used in quantum chemistry does not generalize to solids, however, and researchers make wide use of an approximate treatment of the exchange and correlation energies



Surface charge density on two planes through a gallium arsenide crystal. These calculated contours of constant electron density are logarithmically spaced, three to the decade, with the progression 1, 2.2, 4.6. Charge density is given in units of electrons/au^3. A 20 meV helium atom has a classical turning point near the contour at 10^{-4}, and atom diffraction experiments have confirmed the corrugation of this contour. (From reference 6.) Figure 2

based on the local-density-functional method.[5] These calculations are notably accurate in their results for ground state properties such as cohesive energies, equilibrium geometries, and vibrational force constants. They are less successful in predicting, for example, the energies of excitations to unoccupied states, and there is no systematic means of improvement. We need a new principle, perhaps a theory for treating correlation in a semi-local way, to improve significantly the state of the art in this field.

One can study systems with substantially less symmetry than full crystalline order by making accurate calculations within the above framework. Examples of such systems are surfaces that are periodic in two dimensions, and crystals that are perfect except for isolated defects. Some of my own studies on surfaces illustrate the use of analytic knowledge to simplify complex calculations. Figure 2 shows the calculated electron charge density on the (110) surface of GaAs. The wave functions for this system are dominated by the spherical part of the attractive potential near each atom and by the one-dimensional part of the surface barrier potential in the vacuum region. The Schrödinger equation separates for spherical and one-dimensional potentials, and numerical solutions for these dominant potentials are easy to calculate at a few energies in the range of interest. The wave functions in the interstitial regions vary slowly, and an expansion in plane waves both converges rapidly and provides the desired translational symmetry. One joins these three types of wave functions smoothly on appropriate boundaries to form a variational basis set that gives accurate solutions for the full potential. Comparison of the calculated charge distribution in figure 2 with results from helium-beam diffraction experiments shows[6] that this method gives accurate wavefunctions at least 3 Å to 5 Å above the surface atoms.

Simulation

The majority of simulations fall roughly into three classes.

▶ Continuum models are used widely in plasma physics, fluid dynamics and astrophysics. In simulations based on these models, one solves numerically the coupled nonlinear partial differential equations describing the relevant force and matter fields, usually after introducing discrete meshes of points to represent the space and time variables.

▶ The second group of models deals with finite collections of particles interacting via given force laws. The "particles" can range from galaxies to molecules to electrons as one goes from astrophysics to condensed-matter physics to plasma physics. The simulations, variously called "particle codes" and "molecular dynamics," proceed by integrating the Newtonian equations of motion of each particle acting under the force of the others. One can introduce damping or viscosity and a random force proportional to the damping and temperature of a real or hypothetical thermal reservoir.[7] As Bruce Cohen and John Killeen explain in their article on page 54 of this issue, plasma physicists are making wide use of simulations that fall into the first two classes.

▶ The third class of simulations is identified by the methods rather than by the models. These simulations attack problems with many variables through the use of random-sampling techniques, known as "Monte Carlo" methods. (These methods are named after the site of their use in a historic proof of the irreversibility of cash flow.) The version of the Monte Carlo method used in statistical mechanics generates a random walk in the configuration space of the simulated system using an algorithm that enforces the principle of detailed balance at a given temperature. According to this principle, transition probabilities between states of the system are related by a Boltzmann factor, and the algorithm biases the choice of each step in the random walk accordingly. After the system equilibrates, averages over the paths converge to thermodynamical averages. Physicists have applied variants of this method to homogeneous quantum-mechanical systems such as liquid helium, and to a few highly idealized quantum models,[8] including, most recently, high-energy quantum field-theory models.

All these simulations share a set of related concerns that tie their accuracy closely to the computer power available. More spatial grid points or more particles or spins will always give a better approximation to the continuum or thermodynamic limit. The shortest important spatial scale or longest correlation length may not be known in advance. The shortest time step is set by the fastest propagating wave or vibrational period in the problem, and to have a valid sample or ensure equilibrium the running time must allow the slowest hydrodynamic flow or domain-wall diffusion to make several circuits of the system. It may be difficult to get close enough to convergence to see the form of the convergence from the simulation itself.

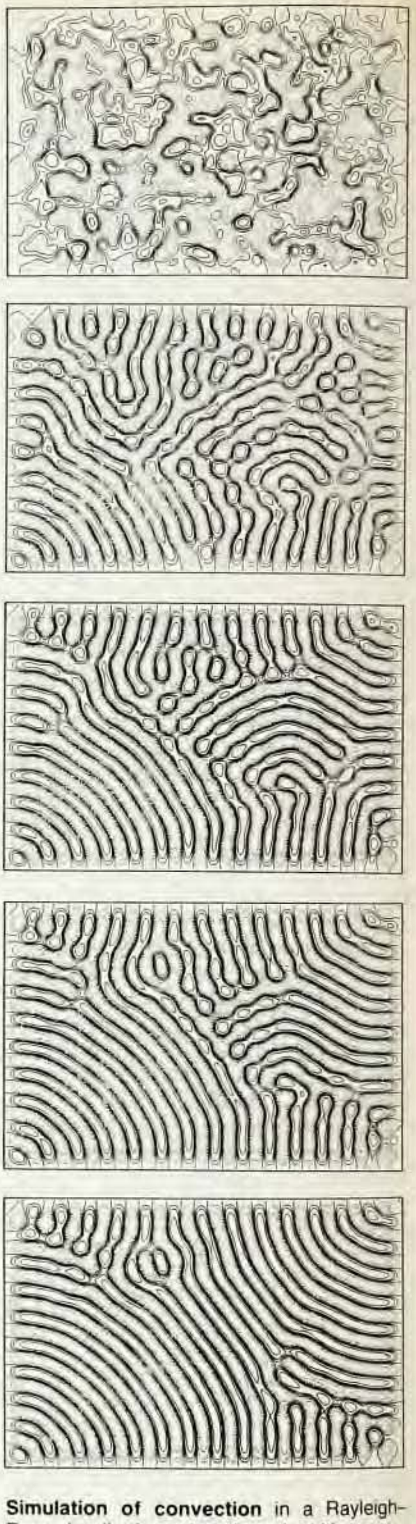
A key historical example of a breakthrough due to simulation cuts across several fields. In the initial step in this development, Enrico Fermi, John Pasta and Stanislaw Ulam undertook what we would call a molecular dynamics study of the motion of a chain of masses coupled by nonlinear springs. Their goal was to help develop a theory for the long-standing problem of the finite thermal conductivity of solids. In their calculations, a long-wavelength disturbance, imposed as an initial condition, was seen to decay into a quasi-random mix of shorter wavelength modes as expected. They found, however, that after a sufficient time, simple states nearly identical to the initial condition recurred. This surprising result stimulated further work, including the approximate reduction of the original problem to a nonlinear partial differential equation well known in hydrodynamics, the Korteweg–deVries equation. This equation, which describes shallow water waves, was known to possess "solitary" solutions—pulse-like waves that propagate at constant velocity with unchanged shape. In studying more-complex solutions of this equation numerically, Martin Kruskal and Norman Zabusky discovered the remarkable new principle that one can classify any complicated solution by a set of "solitons" that resemble the solitary solutions when well separated and that retain their identities as they pass through each other. A period of rapid development followed in which researchers proved analytically many new properties of this and large classes of other nonlinear partial differential equations important in physics. Zabusky has written a detailed account[9] of the "synergetic" interplay of simulation and analysis in this case.

Simulations also stimulated an important conceptual advance in the area of turbulence. Motivated by a problem in atmospheric physics related to weather prediction, Edward Lorenz of MIT approximated the relevant hydrodynamic equations by a set of coupled nonlinear ordinary differential equations describing a limited number of modes. He found that this set displayed chaotic behavior in certain parameter ranges and pursued this unexpected observation by further reducing the number of modes. He finally found[10] the minimum number of modes necessary for chaotic behavior to be three, reversing previous thinking that true chaotic behavior required an infinity of degrees of freedom.

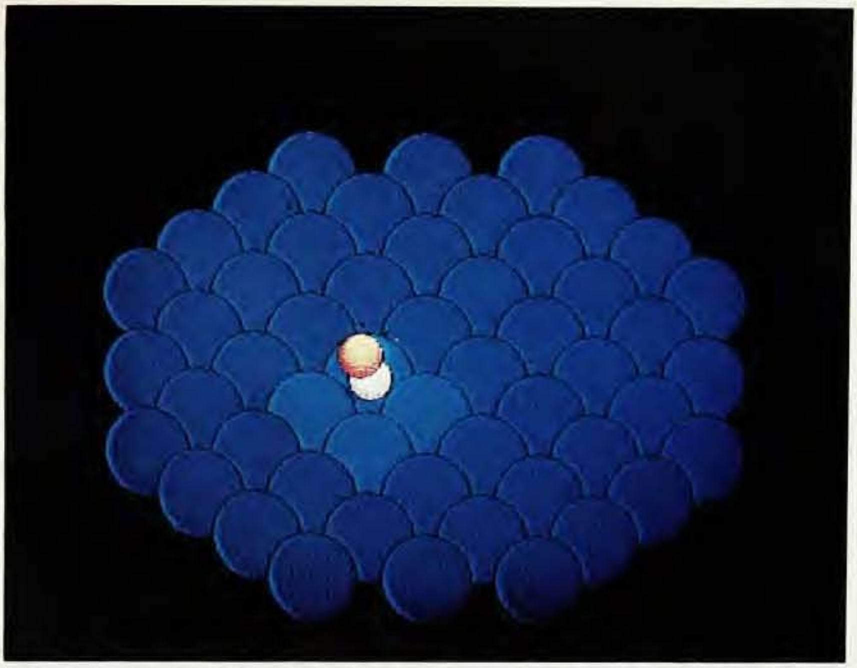
The preceding examples represent completely unexpected results for very simple systems, results with broad implications through mathematical generalization. Simulations are usually more complicated and more specific, and have results anticipated to at least some degree. A recent study of a hydrodynamics problem, the so-called Rayleigh–Benard convection, embodies some characteristics that are useful to consider here. A Rayleigh–Benard cell consists of a fluid-filled box (rectangular in this case), with a fixed temperature difference maintained between the top and bottom. When the lateral dimensions are large compared to the height, complicated convective roll patterns form and evolve for temperature differences not too far above that required for the onset of convection. Researchers found the full 3-dimensional time-dependent fluid equations much too complicated to attack numerically. However, the solution for the fluid flow in a laterally infinite cell just at the onset of convection was known to be a particular periodic function. Solutions near onset could be represented as this analytic result multiplied by a slowly varying amplitude function, which obeys a certain nonlinear biharmonic partial differential equation in the two lateral dimensions and time.

Even this simplified problem could be solved only because a highly efficient new algorithm was available for the biharmonic equation. In a simulation, researchers at Bell Labs followed the evolution of the fluid flow from random initial conditions by generating a series of contour plots of the midplane vertical fluid velocity. (See figure 3.) The excellent pattern-recognition capabilities of the human brain immediately permitted the long-time-scale relaxation of the system to be identified as the "annealing" of defects in the roll pattern, a process apparent in the sequence in figure 3. Three elements—analytic simplification based on known physics, good algorithms and good graphics—characterize many successful simulations.[11]

A final example, also from Bell Labs, illustrates a simulation technique that might be termed, "Put your money where the action is." Physicists wanted to study gas–surface interactions using molecular dynamics. A large number of atoms would be required to represent the semi-infinite solid, and most of them would simply be performing uninteresting small thermal vibrations most of the time. John Tully developed a technique for treating these atoms in the analytically tractable harmonic approximation and doing detailed numerical trajectory calcula-



Simulation of convection in a Rayleigh–Benard cell whose length and width are respectively 29.2 and 19.5 times its height. Shown here are contours of constant vertical velocity at the midplane of the cell. The solid and dotted contour lines represent negative and positive velocities, respectively, at $1/2$, $1/4$, $1/8$ and $1/16$ of the maximum. The temperature difference driving the convection is 1.1 times that required for the onset of convection in a laterally unbounded cell. The random pattern imposed at time zero (top frame) evolves into a complex but repeatable stable pattern of convection rolls. Figure 3 tions only for the incident molecule and a cluster of surface atoms closest to it. The cluster is redefined dynamically as the molecule or its constituents move laterally over the surface, so the technique treats the full effect of the anharmonic potential wherever large energy transfers can take place. In addition to extracting average quantities of interest for comparison with experiment, Tully prepared a movie using sophisticated computer graphics with perspective, color, shading and shadows. Figure 4 is a frame from the film. The Maxwell-demon's-eye view provided by the movie shows many new aspects of the interplay of kinetic, vibrational and rotational energy in scattering and chemisorption.[12]



Surface-physics simulation. This is a single frame from a computer-generated movie illustrating dissociative adsorption of nitric oxide on platinum. The reddish atom is oxygen; the lighter one is nitrogen. The four light-blue platinum atoms near the nitric oxide molecule comprise the "active" surface zone. "Shadows" are cast directly below the gas atoms. (From reference 12.) Figure 4

Experiment. Computers are important tools today in experimental physics, where there is little debate about their appropriate role. I have never seen a defense of the intellectual aesthetics of copying a series of readings into a lab notebook by hand! Computers often enter all phases of an experiment: designing the apparatus, controlling it during experimental runs, and collecting and analyzing data. The scale of usage goes with the scale of the experiment, from a large high-energy-accelerator center with a collection of powerful mainframes and hundreds of thousands of lines of specialized program code to a one-person solid-state lab with a minicomputer and a few small programs. Some of the tasks involved are similar to those encountered in theoretical computation, such as simulating the behavior of particles in an accelerator, or solving a model of the system under study to find the values of parameters that fit the data. Whether this latter activity qualifies as theory or experiment in a given case is largely a matter of judgment.

The tasks involved in control and data collection are qualitatively different. They involve real-time programming, and often the handling of many inputs and outputs simultaneously as in a time-sharing system. Some of the needed skills are more akin to those of a systems programmer than to those of an applications programmer, and require knowledge of machine architecture and a system-level language. Computers permit many otherwise-impossible things to be done in experiments today, such as simultaneously scanning several variables, real-time pre-processing of data and interactive data analysis during a run. Unfortunately, all this gets very short shrift in the physics literature, and will get the same here, with apologies from your theorist author.

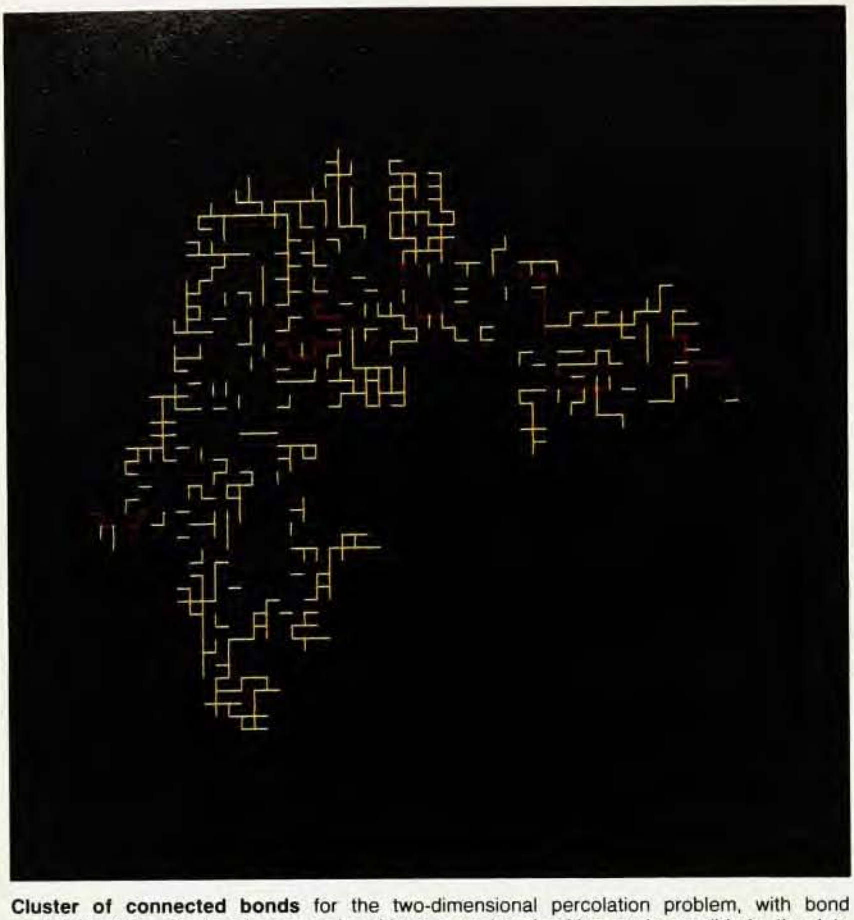
Software

When computational results are reported in the physics journals, the software involved in producing those results is seldom given even the briefest attention. This is a major problem for a reader who wants to assess the reliability of the results, reproduce them, extend the methods or apply them to other problems. The tradition of not publishing computational procedures probably stems from the issue of intellectual respectability, and tends to be self-sustaining because assessing the reliability of results is a part of that issue. This practice is not consistent with the usual practice of publishing experimental procedures or analytic calculations in verifiable detail. Publishing the methods in journals devoted to computation serves a function in some cases, but is not a satisfactory substitute.

The design of any complex computation is in fact a challenging task and comprises a significant portion of the creative part of the research. In addition, the computer program contains a large amount of information about the problem it is designed to solve, much of it not reported in the text of the usual journal article. The resolution of this issue must come from a change in attitude on the part of editors, referees, and above all, authors themselves.

This is not to suggest that it would be useful to publish a complete listing of the program. A schematic description of each major step in the program, identification and discussion of the numerical algorithms used, and discussion of special considerations in which the physics of the problem influenced the design of the program would be a valuable addition to a physics publication. The discipline of writing out such a description should serve the same role in clarifying thought that is found in writing up any research. In fact, writing it out before starting to write any code constitutes a much more professional approach to programming than most physicists practice and would undoubtedly save much grief in the process. This suggestion need not lead to a large increase in the number of journal pages. Much of the discussion organized around the program could substitute for other text of the traditional paper.

FORTRAN a curse? The question of what language to use to code a physics problem, if asked, usually has just one answer—FORTRAN. While this first high-level programming language certainly played a major role in stimulating scientific computation when it was introduced in the fifties, some feel that it is a curse on our house today.[13] The basis of the complaint is that we have made great progress since then, not just in programming languages, but in understanding models of computation and how we use computers. FORTRAN does not reflect this progress. The most commonly cited specific problems are FORTRAN's poor control flow, its lack of facilities for structuring data, and its lack of recursion. Control flow refers to the sequence in which portions of the program are executed, and many FOR-



Cluster of connected bonds for the two-dimensional percolation problem, with bond probability 0.45. The bonds were colored by a recursive algorithm made possible by the data structure representing the cluster. If we consider the graph as a resistor network carrying current from the left to the right, the red bonds carry the full current, the blue bonds share the current and the yellow bonds are "dead ends" carrying no current. (After reference 14.) Figure 5

TRAN programs resemble bowls of spaghetti in this respect. Recursion is the very useful ability of a subroutine to call itself. These failings are corrected in more modern languages such as ALGOL, PASCAL and C, but these have not gained much popularity among physicists.

The control-flow problem can make it difficult or impossible to understand, modify or even use a FORTRAN program written by someone else (or yourself a few years ago). I can best illustrate this point by a few personal anecdotes. I once obtained a copy of a classic program for calculating the electronic structure of atoms, which was written in the earliest FORTRAN. I wished to change the kind of mesh used for the radial variable, but found this nominally small change so difficult to make that I elected instead to rewrite the entire program. The second example stemmed from my need for a personalized graphics program to drive a recently acquired printer. I found it easy to modify a system-library C program for the purpose. My last example, modifying a set of library matrix diagonalization routines to take advantage of vector processing on a Cray-1, seems at first glance to contradict the others. To my surprise, even though the routines were written in FORTRAN, it was easy to understand them enough to make the modifications. However, the introductory comments indicated that the routines were FORTRAN transcription of originals written in ALGOL!

Two examples serve to illustrate the other limitations of FORTRAN. The first concerns a percolation problem in statistical mechanics. Bonds joining neighboring points on a lattice are present with probability p, and the statistics of clusters of connected points are sought as p approaches the critical probability p_c, where infinitely large clusters of bonds first occur. In the "standard" approach, bonds are placed randomly throughout a box. This is easily represented as an array, the only data structure generally familiar to FORTRAN programmers. A research team with computer-science expertise introduced a novel approach in which only bonds that are part of a cluster are generated, using a recursive procedure that examines sites on the perimeter of the cluster and at random either adds a bond or marks the site empty. It terminates when the entire perimeter is marked empty. The bonds are stored as a data structure called a doubly linked graph, which reflects the essential character of the problem: connectivity. The new approach handles much larger clusters with a given amount of memory than does the box approach; it avoids spurious effects from the box boundary, and the computation time grows much more slowly with cluster size.[14] Figure 5 shows a sample cluster, colored by the program to identify its "backbone."

The second example, from astrophysics, concerns the evolution of a large collection of galaxies under the force of gravity. The straightforward approach integrates the equations of motion of N point-mass galaxies, N^2 calculations being required to obtain the forces at each time step. A new approach,[15] devised at Princeton University, introduces a binary-tree data structure to organize the galaxies into a hierarchy of spatially disjoint "clumps," so that a recursive procedure can compute only center-of-mass forces between sufficiently separated clumps. The result is $N \log N$ computations per step with a bonus of much longer time steps for the whole system, because the fast internal motion of the few very small clumps is easy to identify and handle separately. Results using this program show the tendency of an initially random distribution of galaxies to cluster, as figure 6 shows. This problem was coded in PASCAL, and the percolation problem in C. I have seen the code for the latter, and although it is small and easy understand, I would hate to try it in FORTRAN.

These examples support the generally held opinion of computer scientists that knowing only FORTRAN makes it hard to write efficient algorithms and understandable programs. Two reasons most frequently given by physicists for retaining FORTRAN are that it produces more efficient machine language code and that there are valuable libraries of FORTRAN subroutines in many fields. The efficiency issue actually concerns compilers, not languages. FORTRAN compilers have been refined for such a long time that some of them do produce very efficient code. However, compilers for better languages are improving and will improve much more, especially if the demand grows. In terms of the productivity of a research program, the staff hours that could be saved in program development might easily be worth the cost in object code efficiency. A factor of two reduction in efficiency is certainly not going to make a qualitative difference in the physics that can be done. This assumes the same algorithm. If a more efficient algorithm is achieved in another language, the machine-time cost could *decrease* substantially.

The library question is a more substantial one. In the long run we can expect libraries to be replaced by new ones written in a new dominant language. A currently available alternative is to use a programming language that can be automatically translated into FORTRAN. The languages EFL and RATFOR, which are in this class, offer many of the features of other modern languages and can take advantage of FORTRAN compilers and libraries.

Special-purpose languages. Despite their many improvements over FORTRAN, even the newer languages do not permit one to write a program in such a way that it resembles the problem it is designed to solve. Ideally, conceptually distinct pieces, such as the basic equations, the boundary conditions and the discretization of variables, might be grouped in independent modules of the source code. Kenneth Wilson and others see the development of such programming languages as an important goal for scientific computing. Constructing compilers for such special-purpose languages is becoming a much easier task thanks to the development of "compiler compilers." Nevertheless, an extensive effort by physicists and computer scientists will be necessary to design languages useful beyond very narrowly defined areas.

There already are a number of special-purpose languages for doing another kind of task very useful in physics: symbolic manipulation, or, more popularly, algebra. One example is MACSYMA, developed by the Mathlab group at MIT. It is a large and powerful language that enables the user to differentiate, integrate, take limits, solve systems of linear or polynomial equations, factor polynomials, expand functions in Laurent or Taylor series, solve differential equations, compute Poisson series, plot curves and manipulate matrices and tensors.

For an electronic-structure calculation, I once needed a set of a few hundred integrals that could all be evaluated analytically and expressed as sums of multiple partial derivatives of a reasonably simple "core" function of twelve variables. The chances of my ever completing the job, let alone getting it all correct, were zero. It took only a month to learn enough of an "algebra" language available at the time, ALTRAN, to produce the needed algebraic expressions in a form that could go directly into a FORTRAN program, undefiled by human hands. MACSYMA itself is written in LISP, a list-processing language, and ALTRAN is written in, of all things, FORTRAN.

Before even getting to the language of his choice, the user must deal with another piece of software, the operating system. Operating systems completely lack the standarization of the major programming languages and present an annoying hurdle to anyone attempting to use different computers. The rapid spread of the UNIX [16] operating system on many small, medium-sized and even a few large computers is a hopeful step toward *de facto* standardization. UNIX offers many nice features for program development and text processing as well as computing. It even supports programs to protect users from other operating systems. For example, by typing a few UNIX-like commands, I can send program and input data files from my local UNIX-run VAX computer through a network to our computer center's Honeywell and on to their Cray. After the numbers are appropriately crunched, output is shipped back to my file space on the VAX for my perusal, and I can remain blissfully ignorant of the Honeywell and Cray operating-system control statements that were necessary to actually accomplish these tasks. The UNIX system is itself written in C, so that one can use this high-level language to handle the real-time control and data acquisition tasks associated with experimental applications.

Clustering of galaxies. Simulation shows the evolution of a random distribution of 10 000 galaxies (top) to a distribution showing significant clustering (bottom). This is a two-dimensional projection of the actual three-dimensional simulation. The mass density of the universe is assumed to be the minimum value necessary for it to be closed according to general relativity. The relativistic expansion of the distance scale is 7.1 between the two frames. (From reference 15.) Figure 6

Algorithms. The issue of algorithms has come up peripherally in some of the preceding discussion. The importance of development in this area is well known to computer scientists but much less appreciated among physicists. Figure 7 dramatically compares the evolution of hardware with that of software for solving a typical simulation problem, a 3-dimension partial differential equation in a box. The left axis gives the intrinsic gate switching times for the devices used to make computers. (This overestimates the increase in computer speed in recent years because transmission time between elements has become a limiting factor.) The right axis gives the number of arithmetical operations necessary to solve the equation to a specified accuracy. Including pre-1950 algorithms, which no one could in fact use in a practical case, or considering an irregularly shaped boundary instead, would give a line that was steeper yet. The user, of course, benefits from the product of the hardware and software curves.

What is the magic behind such advances? The Cooley–Tukey algorithm for the fast Fourier transform introduced in 1965 is fairly well known. For N mesh points it reduces the computational effort in the asymptotic large-N limit from the order of N^2 operations for the straightforward algorithm to $\mathcal{O}(N \log N)$. Less well known are the facts that two N-digit numbers can be multiplied in $\mathcal{O}(N \log N \log\log N)$ operations—not $\mathcal{O}(N^2)$, the way we learned in grade school—and that two general $N \times N$ matrices can be multiplied in $\mathcal{O}(N^{2.81})$. These and many other improvements on the asymptotic computational complexity of mathematical tasks, beyond that of the obvious approach (the way the operation is conventionally written in an equation), are neither accidental nor isolated discoveries. They stem from the application of known principles that can also be applied to more-complex tasks such as the percolation and galaxy problems.

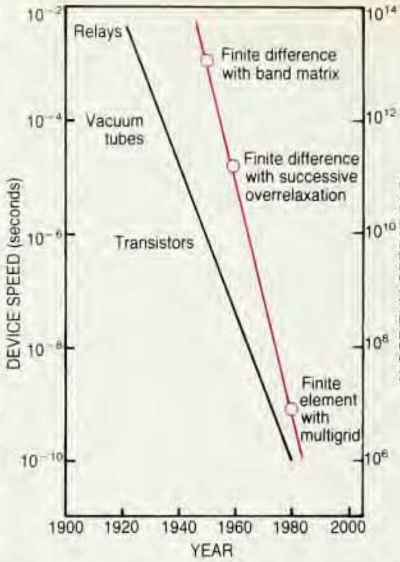
The bottom line on physicists and software may well be education. Every physics student today should have some courses in computer science, as distinctly opposed to a course in FORTRAN programming. Ideally, there would be a one- or two-term course specially tailored to physics or science students.[13] For those beyond the student stage, the awareness that there is something to be caught up with is at least a beginning. The programming environments in which we must function may change slowly, but as one computer scientist told me, it's not too bad to program in FORTRAN as long as you don't *think* in FORTRAN.

Hardware

At the top of the current hardware line are two general-purpose supercomputers, the Cray-1 built by Cray Research, Inc. (see the photograph on page 23), and the Cyber 205 built by Control Data Corporation. These can achieve peak speeds of around 80 and 400 million floating-point operations per second, or "megaflops," respectively, for algorithms that permit vector processing. Practical average speeds on a Cray-1 are reported to be around 20 megaflops. (There is great variability in all such estimates, and I will not stand by any of the performance numbers numbers I quote.) The number of these machines is small, and time on them is hard for most physicists to obtain. The CDC 7600 and IBM 360/195, with average speeds around 2 megaflops, are more widely available.

The Cray-2, with 800 megaflops (peak, presumably), is planned for introduction in late 1983 or early 1984. The Japanese government has announced a supercomputer project aimed at creating a 10-gigaflop machine by 1989. Achieving this goal will require the development of alternatives to silicon semiconductor technology for high-speed logic and memory elements and of highly parallel processing systems. The Department of Defense and the National Science Foundation recently sponsored a panel study of large-scale computing in science and engineering, with Peter Lax as chairman. The panel report[18] stressed in its conclusions the need for a comparable US initiative to maintain our leadership. The panel anticipates government sponsorship and university involvement in early research and development to get such a long-range project to a state where it will be attractive for commercial development.

What do these speed numbers imply? There seems to be a general consensus that a factor of 10 change in computer speed will have no qualitative effect on scientific computation, but that a factor of 10^3 will. One exception is real-time control tasks, where a smaller factor may make the difference in being able to keep up with a physical process. The total computational power available is the product of the speed of the computer and the time one has use of it, so the speed factor really translates into tolerable (10) and intolerable (10^3) extensions of time needed to obtain results. We must consider another factor to normalize megaflop-based comparisons. Existing supercomputers get their big advantage from vector, or parallel, processing, and while this mode can usually be used in



Evolution of hardware and software, showing the development of faster circuit components and more efficient algorithms. The two lines compare the decrease in the intrinsic gate-switching time of computer logic elements with the decrease in the number of arithmetical operations necessary to solve a particular 3-dimensional partial differential equation. The user benefits according to the product of the two curves. (Reference 21.) Figure 7

executing "obvious" numerical algorithms, it often cannot for optimum algorithms. We may need to do much research to generalize current optimization strategies to highly parallel processing.

The computers most affected by present developments in electronics are in the mid-, mini- and personal-size range. By mid-sized I mean a computer such as Digital Equipment Corporation's VAX 11/780, which has an average speed of 0.2 megaflops. Silicon very-large-scale integrated-circuit technology is rapidly bringing down sizes and prices in this performance range. Current industry development should put VAX computing power in a box on your desk for \$10 000—including a hard disk—within the next five years. Meanwhile, add-on vector processors in the \$500 000 range can already give mid-sized computers large main-frame power for appropriate number-crunching problems.

A last option increasingly discussed today is the special-purpose processor such as Belle, the champion chess computer,[19] and the Santa Barbara Ising Monte Carlo processor.[20] These can match or surpass supercomputer power for highly specialized problems at a fraction of supercomputer cost. The decision to invest time and money in developing and building a special-purpose machine has some parallels to that for a new experimental facility.

The hardware environment we shall face in the future will have several components. Access to supercomputers will expand, especially if the government heeds the Lax panel's recommendations[18] for an expanded national network generalizing the existing Department of Defense Advanced Research Project Agency computer network and the plasma fusion computer network. The most predictable change will be the expansion of distributed computer power, which should have significant effects on physics research. Having a machine of VAX power with a good graphics display all to oneself will encourage freer and more creative patterns of use. Kruskal and Zabusky discovered their solitons and Wilson found his first critical exponents using modest amounts of computer power. In a network with bigger machines, this "superpersonal" computer will allow interactive analysis and display of stored output from large-scale simulations. If a local run does take a solid weak of cpu time, there are no other users to compete or complain, and there is no bill from the computer center at the end of the month.

What role will the first few 10-gigaflop machines, or Cray-2s for that matter, play when they arrive? My opinion is that the most appropriate computational physics problems for such a valuable resource should be of a developmental and not a research nature. A development problem is characterized by a deadline, and it is necessary to plan the best job possible with known techniques and available resources. To pick an example from the Lax report, the aerodynamic efficiency of recent airframe designs could only be optimized in pieces using available hydrodynamic algorithms and a Cray-1. If a Gig-10 had been available, it could have been used to optimize the airframe as a whole, presumably yielding efficiency gains of substantial economic significance. No responsible project manager would have adopted the strategy of first hiring a team of mathematicians and computer scientists to improve the algorithms by 10^3 and then optimizing the whole airframe on a Cray-1.

Research physics is different and requires different strategies. If brute force Monte Carlo simulation had been the only method pursued to obtain the theoretical critical exponents of phase transitions, that field would still be languishing. The galaxy simulation could have been done in original form with an estimated 30 hours of Cray-1 time. However, the strategy of achieving a factor of 400 improvement and running 20 hours on a VAX was not only cost-effective (the program was developed as an undergraduate thesis), but left room for extensions. Luckily, neither critical-exponent nor galacticclustering researchers face deadlines.

* * *

Much of the material for this article was developed from unpublished reports and extended conversations. I would like especially to thank A. V. Aho, P. W. Anderson, A. W. Appel, W. D. Arnett, W. Fichtner, H. S. Greenside, P. D. Lax, A. Ogielsky, R. Pike, N. L. Schryer, J. C. Tully, V. A. Vyssotsky, J. D. Weeks and K. G. Wilson.

References

1. P. H. Abelson, Science **215**, 751 (1982). (The issue is devoted to computers.)
2. J. R. Schrieffer, D. J. Scalapino, J. W. Wilkins, Phys. Rev. Lett. **10**, 336 (1963).
3. W. L. McMillan, J. M. Rowell in *Superconductivity*, R. D. Parks, ed., Vol. **1**, Dekker, New York (1969), page 561.
4. K. G. Wilson, Phys. Rev. B **4**, 3184 (1971); Rev. Mod. Phys. **47**, 773 (1975).
5. L. J. Sham, M. Schlüter, PHYSICS TODAY, February 1982, page 36.
6. D. R. Hamann, Phys. Rev. Lett. **46**, 1227 (1981).
7. K. Binder, ed., *Monte Carlo Methods in Statistical Physics*, Springer–Verlag, Berlin (1979).
8. R. W. Hockney, J. W. Eastwood, *Computer Simulation Using Particles*, McGraw–Hill, New York (1981).
9. N. J. Zabusky, J. Comp. Phys. **43**,195 (1981).
10. E. N. Lorenz, J. Atmos. Sci. **20**, 130 (1963).
11. H. S. Greenside, W. M. Coughran Jr, N. L. Schryer, Phys. Rev. Lett. **49**, 726 (1982).
12. J. C. Tully, Acc. Chem. Res. **14**, 188 (1981); Computers and Chemistry **5**, 159 (1981).
13. R. Pike, *CERN Workshop on Software in High Energy Physics Proceedings*, CERN, Geneva (1982), page 13.
14. R. Pike, H. E. Stanley, J. Phys. A **14**, L169 (1981).
15. A. W. Appel, *An Investigation of Galaxy Clustering Using an Asymptotically Fast N-body Algorithm*, Undergraduate Thesis, Princeton University, Princeton, New Jersey (April 1981). An independent method for beating N^2 using a particle–field approach is reported in reference 8, page 440.
16. UNIX is a trademark of Bell Laboratories.
17. A. V. Aho, J. E. Hopcroft, J. D. Ullman, *The Design and Analysis of Computer Algorithms*, Addison–Wesley, Reading, Massachusetts (1974).
18. *Report of the Panel on Large Scale Computing in Science and Engineering*, P. D. Lax, Chairman, US Department of Defence and National Science Foundation (1983).
19. J. H. Condon, K. Thompson, *Advances in Computer Chess 3*, M. R. B. Clarke, ed., Pergamon, New York (1982), page 45.
20. R. B. Pearson, J. L. Richardson, D. Toussaint, J. Comp. Phys., to be published.
21. After original by N. Schryer; data from R. W. Keyes, IEEE Trans. on Elect. Devices **26**, 271 (1979) and J. R. Rice, ACM Signum News **11** (4), 6 (1976). □