33. G. Felsenfeld *et al.*, *J. Am. Chem. Soc.* **79**, 2023 (1957); A. G. Letai *et al.*, *Biochemistry* **27**, 9108 (1988).
34. M. Riley, *Microbiol. Rev.* **57**, 862 (1993).
35. Supported in part by Department of Energy Cooperative Agreements DE-FC02-95ER61962 (J.C.V.) and DEFC02-95ER61963 (C.R.W. and G.J.O), NASA grant NAGW 2554 (C.R.W.), and a core grant to TIGR from Human Genome Sciences. G.J.O. is the recipient of the National Science Foundation Presidential Young Investigator Award (DIR 89-57026). M.B. is supported by National Institutes of Health grant GM00783. We thank M. Heaney, C. Gnehm, R. Shirley, J. Slagel, and W. Hayes for software and database support; T. Dixon and V. Sapiro for computer system support; K. Hong and B. Stader for laboratory assistance; and B. Mukhopadhyay for helpful discussions. The *M. jannaschii* source accession number is DSM 2661, and the cells were a gift from P. Haney (Department of Microbiology, University of Illinois).

RESEARCH ARTICLES

Universal Quantum Simulators

Seth Lloyd

Feynman's 1982 conjecture, that quantum computers can be programmed to simulate any local quantum system, is shown to be correct.

Over the past half century, the logical devices by which computers store and process information have shrunk by a factor of 2 every 2 years. A quantum computer is the end point of this process of miniaturization—when devices become sufficiently small, their behavior is governed by quantum mechanics. Information in conventional digital computers is stored on capacitors. An uncharged capacitor registers a 0 and a charged capacitor registers a 1. Information in a quantum computer is stored on individual spins, photons, or atoms. An atom can itself be thought of as a tiny capacitor. An atom in its ground state is analogous to an uncharged capacitor and can be taken to register a 0, whereas an atom in an excited state is analogous to a charged capacitor and can be taken to register a 1.

So far, quantum computers sound very much like classical computers; the only use of quantum mechanics has been to make a correspondence between the discrete quantum states of spins, photons, or atoms and the discrete logical states of a digital computer. Quantum systems, however, exhibit behavior that has no classical analog. In particular, unlike classical systems, quantum systems can exist in superpositions of different discrete states. An ordinary capacitor can be either charged or uncharged, but not both: A classical bit is either 0 or 1. In contrast, an atom in a quantum superposition of its ground and excited state is a quantum bit that in some sense registers both 0 and 1 at the same time. As a result, quantum computers can do things that classical computers cannot.

Classical computers solve problems by using nonlinear devices such as transistors to perform elementary logical operations on the bits stored on capacitors. Quantum computers can also solve problems in a similar fashion; nonlinear interactions between quantum variables can be exploited to perform elementary quantum logical operations. However, in addition to ordinary classical logical operations such as AND, NOT, and COPY, quantum logic includes operations that put quantum bits in superpositions of 0 and 1. Because quantum computers can perform ordinary digital logic as well as exotic quantum logic, they are in principle at least as powerful as classical computers. Just what problems quantum computers can solve more efficiently than classical computers is an open question.

Since their introduction in 1980 (*1*) quantum computers have been investigated extensively (*2–29*). A comprehensive review can be found in (*15*). The best known problem that quantum computers can in principle solve more efficiently than classical computers is factoring (*14*). In this article I present another type of problem that in principle quantum computers could solve more efficiently than a classical computer—that of simulating other quantum systems. In 1982, Feynman conjectured that quantum computers might be able to simulate other quantum systems more efficiently than classical computers (*2*). Quantum simulation is thus the first classically difficult problem posed for quantum computers. Here I show that a quantum computer can in fact simulate quantum systems efficiently as long as they evolve according to local interactions.

Feynman noted that simulating quantum systems on classical computers is hard. Over the past 50 years, a considerable amount of effort has been devoted to such simulation. Much information about a quantum system's dynamics can be extracted from semiclassical approximations (when classical solutions are known), and ground state properties and correlation functions can be extracted with Monte Carlo methods (*30–32*). Such methods use amounts of computer time and memory space that grow as polynomial functions of the size of the quantum system of interest (where size is measured by the number of variables—particles or lattice sites, for example—required to characterize the system). Problems that can be solved by methods that use polynomial amounts of computational resources are commonly called tractable; problems that can only be solved by methods that use exponential amounts of resources are commonly called intractable. Feynman pointed out that the problem of simulating the full time evolution of arbitrary quantum systems on a classical computer is intractable: The states of a quantum system are wave functions that lie in a vector space whose dimension grows exponentially with the size of the system. As a result, it is an exponentially difficult problem merely to record the state of a quantum system, let alone integrate its equations of motion. For example, to record the state of 40 spin-½ particles in a classical computer's memory requires $2^{40} \approx 10^{12}$ numbers, whereas to calculate their time evolution requires the exponentiation of a $2^{40} \times 2^{40}$ matrix with $\approx 10^{24}$ entries. Feynman asked whether it might be possible to bypass this exponential explosion by having one quantum system simulate another directly, so that the states of the simulator obey the same equations of motion as the states of the simulated system. Feynman gave simple examples of one quantum system simulating another and conjectured that there existed a class of universal quantum simulators capable of simulating any quantum system that evolved according to local interactions.

The answer to Feynman's question is, yes. I will show that a variety of quantum systems, including quantum computers, can be "programmed" to simulate the behavior of arbitrary quantum systems whose dynamics are determined by local interactions. The programming is accomplished by inducing interactions between the variables of the simulator that imitate the interactions between the variables of the system to be simulated. In effect, the dynamics of the properly programmed simulator and the dynamics of the system to be simulated are one and the same to within any desired accuracy. So, to simulate the time evolution of 40 spin-½ particles over time t requires a simulator with 40 quantum bits evolving

The author is at the D'Arbeloff Laboratory for Information Systems and Technology, Department of Mechanical Engineering, Massachusetts Institute of Technology, Cambridge, MA 02139, USA. E-mail: slloyd@mit.edu

over a period of time proportional to t (Fig. 1). Quantum computers are Feynman's universal quantum simulators.

To factor a 100-digit number using Shor's algorithm, a quantum computer would have to perform millions of quantum logic operations coherently and without errors. As a result, despite the recent invention of quantum error–correcting routines (*29*), constructing a quantum computer that can factor large numbers is likely to prove difficult (*11*, *19*, *25–28*). With the methods detailed here, in contrast, a quantum computer need only perform a few tens or hundreds of operations to simulate a quantum system, such as the set of spins described above, that would take a classical computer Avogadro's number of operations to simulate. Nor do the quantum operations need to be performed entirely coherently or without errors. In fact, decoherence and thermal effects in the quantum computer can be exploited to mimic decoherence and thermal effects in the system to be simulated. Consequently, the set of quantum effects that might be exploited to construct a quantum computer capable of simulating other quantum systems is larger than the set appropriate for performing Shor's algorithm: Essentially any nonlinear interaction between quantum systems that an experimentalist can modulate in a controlled fashion could be used for quantum simulation.

Simulation

Simulation is a process by which one system is made to mimic another. To simulate an engine on a classical analog computer, for example, one patches up an electrical circuit whose dynamics mimic the engine's dynamics. To simulate one quantum system using another, one needs methods for controlling precisely the dynamics of quantum systems, so that the dynamics of the simulator can be made to mimic the dynamics of the system to be simulated.

Atomic physics, quantum optics, solid-state physics, and quantum chemistry all supply methods for controlling the behavior of quantum systems. Each operation that an experimenter can perform on a quantum system—for example, the turning on and off of a laser pulse or the modulation of a magnetic field—is in effect a "tool" that can be applied to the system. Formally, experimental operations can be divided into two categories: those that approximately preserve quantum coherence, corresponding to Hamiltonian time evolution, and those that do not, corresponding to time evolution governed by a superscattering operator or master equation. Practically, of course, no operation entirely preserves quantum coherence, and operations vary considerably in the amount of decoherence they cause. Simulation of a closed system that evolves according to the Schrödinger equation requires operations that approximately preserve coherence. (The more general case of simulating open systems with both coherence-preserving and coherence-destroying operations will be addressed below). An experimenter who applies such operations can be thought of as turning on and turning off Hamiltonians from a set $\{\tilde{H}_1, \tilde{H}_2 \ldots \tilde{H}_\ell\}$. Essentially, each experimental operation moves the system a controlled distance along one out of a set of predetermined directions in Hilbert space. The experimenter moves the system first one way, then another, like a driver parking a car. A familiar example of this technique is the use of extended sequences of pulses in nuclear magnetic resonance to drive a set of spins to a desired state (*33*).

All techniques that involve the repeated application of coherence-preserving operations possess a common algebraic structure that allows the specification at an abstract level of where the system of interest can be driven by using such operations. The following result, derived independently in the context of quantum control theory and of quantum computation, determines just what quantum systems can be simulated by the repeated application of such operations. The straightforward application of the Campbell-Baker-Hausdorff formula shows that by judiciously turning on and off Hamiltonians, the experimenter can drive the system along any time evolution corresponding to a unitary operator $U = e^{iAt}$, where A lies in the algebra $\mathcal{A}$ generated from the set $\{\tilde{H}_1, \ldots, \tilde{H}_\ell\}$ by commutation (*20*, *21*, *34*). The number of operations required to generate an arbitrary $m \times m$ U is on the order of m^2—the number of parameters required to specify U in the first place. The number of operations that need to be applied to construct a desired U can be thought of as the quantum computational complexity of the construction.

A particularly useful case occurs when the experimenter can make different quantum variables interact according to a particular Hamiltonian. Such an interaction realizes a quantum logic gate (*8*, *16–21*, *23*, *24*, *26*, *35*). In this case it can be seen that almost any nonlinear interaction allows the construction of arbitrary unitary transformations on the 2^N-dimensional Hilbert space (*20*, *21*). For example, Kimble *et al.* have suggested that nonlinear interactions between photons and atoms in small-cavity quantum electrodynamics could be used to "dial up" arbitrary unitary transformations of the photons (*23*, *36*). The results above imply that a variety of quantum systems to which a set of simple experimental operations can be applied can simulate other quantum systems. For example, the quantum variables in the simulator could be photons, as described above, or spins that are made to interact by means of double resonance methods, or quantum dots that interact by dipole or exchange forces. In particular, to the extent that they can be expanded to include more quantum bits, ion-trap quantum logic devices of the sort that are currently being constructed by Monroe *et al.* could act as universal quantum simulators (*24*).

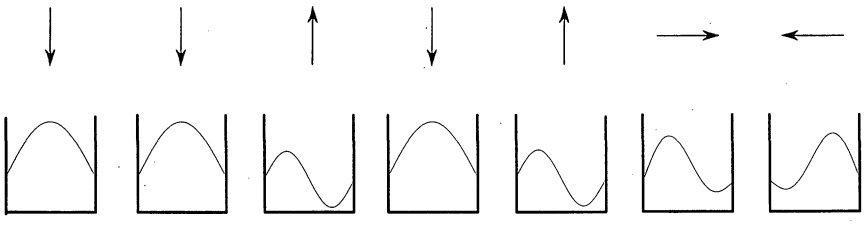

Fig. 1. Electrons in quantum dots simulating a set of quantum spins such as protons. The upper line shows the spins. An arrow pointing down represents a proton spinning clockwise, and an arrow pointing up represents a proton spinning counterclockwise. The lower line shows the states of electrons in quantum dots. The longer wavelengths correspond to electrons in the ground, or lowest energy, state, and shorter wavelengths correspond to higher energy states. The simulation is set up so that spin-down corresponds to a dot in its ground state, and spin-up corresponds to a dot in its first excited state. "Spin-sideways," a superposition of spin-up and spin-down, corresponds to a superposition of ground and excited states, a state with no classical analog. By modulating the interactions between the dots, one can make their dynamics mimic the dynamics of the spin system.

Simulation Efficiency

As noted above, the efficiency of a simulation depends on how hard it is to set up the simulator-system correspondence, to control the simulator to perform the simulation, and to extract its results. A simulation on a classical digital computer of an arbitrary unitary transformation of N quantum variables (for example, spin-½ particles) involves the multiplication of a 2^N-dimensional state vector by a $2^N \times 2^N$ matrix and requires memory space and computational time on the order of 2^{2N}. As noted by Deutsch (*6*), to perform the same simula-

tion on a quantum computer requires only N quantum bits. But the results above imply that 2^{2N} applications of the elementary tools are required to construct an arbitrary $2^N \times 2^N$ unitary U. That is, although a quantum computer gives an exponential compression of the amount of memory space required to construct an arbitrary U, the number of elementary logic steps or applications of quantum logic gates required to construct U on a quantum computer is of the same order as the number of elementary logic steps required to compute the action of U on a classical digital computer. For the case of N spin-½ particles, both simulations require the solution of $2^N \times 2^N$ matrix equations, a difficult task for $N \geq 40$.

The result just presented might seem at first glance to contradict the claim that quantum analog computers can simulate quantum systems much more efficiently than can classical computers. For arbitrary unitary operators U, quantum and classical simulations both require m^2 operations simply because m^2 numbers are required to characterize an arbitrary $m \times m$ U. The time evolution of real quantum systems is not arbitrary, however. Feynman's conjecture was that quantum computers could provide efficient simulations not of arbitrary quantum systems but of systems that evolve according to local interactions. Examples of local systems include hard-sphere and van der Waals gases, Ising and Heisenberg spin systems, strong and weak interactions, and lattice gauge theories. In fact, any system that is consistent with special and general relativity evolves according to local interactions. As will now be demonstrated, quantum simulation is potentially much more efficient than classical digital simulation for local quantum systems.

The method for performing the simulation is conceptually straightforward, if mathematically involved. The goal is to get the simulator from point A to point B along a particular route. But the simulator can only be driven in certain directions—the operations that can be applied experimentally are limited—so it is usually not possible to go from point A to point B directly. But by moving the simulator first a little bit in one direction, then a little bit in another, then a little bit in another, and so on, it is possible to move from A to B. A car can only be driven forward and backward—it cannot be driven sideways. But it is still possible to parallel park. The following construction demonstrates a quantum analog of a familiar classical fact: By going forward and backing up a sufficiently small distance a large enough number of times, it is possible to parallel park in a space only ε longer than the length of the car.

More precisely, consider a quantum system composed of N variables with Hamiltonian $H = \sum_{i=1}^{\ell} H_i$, where each H_i acts on a space of dimension m_i encompassing at most k of the variables. H can depend on time. Any Hamiltonian system with local interactions can be written in this form. Many nonlocal systems such as nonlocal spin glasses also have Hamiltonians of this form and can be efficiently simulated. As with simulations on classical computers (*30–32*), the quantum simulation works by evolving the system forward locally over small, discrete time slices. Because $e^{iHt} \approx (e^{iH_1 t/n} \ldots e^{iH_\ell t/n})^n$, e^{iHt} can be simulated by simulating the local time evolution operators $e^{iH_1 t/n}$, $e^{iH_2 t/n}$, and so on, up to $e^{iH_\ell t/n}$, and repeating n times.

To ensure that the simulation takes place to within some desired accuracy, one needs to regulate the time-slicing as in (*30–32*)

$$e^{iHt} = (e^{iH_1 t/n} \ldots e^{iH_\ell t/n})^n + \sum_{i>j} [H_i, H_j] t^2/2n + \sum_{k=3}^{\infty} E(k) \qquad (1)$$

where the higher order error terms $E(k)$ are bounded by $||E(k)||_{\sup} \leq n||Ht/n||_{\sup}^k /k!$ (where $||A||_{\sup}$ is the supremum, or maximum expectation value, of the operator A over the states of interest), and the total error in approximating $e^{iHt} \approx (e^{iH_1 t/n} \ldots e^{iH_\ell t/n})^n$ is less than $||n(e^{iHt/n} - 1 - iHt/n)||_{\sup}$, which can be made as small as desired by taking n sufficiently large. As a result, for any $\varepsilon > 0$, n can always be picked sufficiently large to ensure that the simulator always tracks the correct time evolution e^{iHt} to within ε.

Once the accuracy to within which the simulation is to take place is fixed, the quantum computational complexity of performing the simulation can be estimated. As noted above and in (*20, 21*), because each H_j acts on a local Hilbert space of only m_j dimensions, the number of operations needed to simulate $e^{iH_j t/n}$ is $\approx m_j^2$ (*37*). Because each local operator is simulated n times, the total number of operations needed to simulate the time evolution e^{iHt} to an accuracy ε is $\approx n(\sum_{i=1}^{\ell} m_i^2) \leq n\ell m^2$ where $m = \max\{m_i\}$. In order for the overall error in building up the product of exponentials to be less then ε, the actual experimental error in implementing each of the elementary operations must be less than $\varepsilon/n\ell m^2$, so that the cumulative experimental error is less than ε. In conventional computational complexity, a simulation is said to be efficient if to simulate a system with N variables takes computer time that is polynomial in N (that is, if the simulation is a tractable problem). Here, the quantum simulation is efficient as long as $\ell = \ell(N)$ is a polynomial function of N. For typical local interactions such as nearest neighbor or next-nearest neighbor, ℓ is proportional to N.

Equation 1 implies that the minimum number of steps n required to simulate the system to accuracy ε over time t is proportional to t^2/ε. However, the duration of each coherent operation required to produce the local Hamiltonian evolutions $e^{iH_j t/n}$ is proportional to t/n, that is, to $1/t$. As a result, the total duration of time required to simulate the system over time t is just proportional to t. In other words, just as with a classical analog computer, the quantum simulation takes an amount of time proportional to the time over which the system to be simulated evolves.

As an example, consider a lattice of N spin-½ particles with pairwise interactions between neighboring particles. $H = \sum_{i=1}^{pN/2} H_i$, where p is the number of neighbors per particle and each H_i acts on a local two-particle Hilbert space of dimension $2^2 = 4$. Determination of the correct set of operations to apply to simulate the local H_i requires the solution of a 4×4 matrix equation in 16 variables. (A simulation of a lattice Hubbard model, in contrast, requires two qubits per site to indicate the number and spin of fermions present; a 16×16 matrix equation must be solved to determine the proper sequence of operations.) The total number of operations needed to simulate the spins to an accuracy ε requires $\approx 2pN/\varepsilon$ steps [if the spins' spatial wave functions overlap, an extra factor of at most N is required to keep track of Fermi statistics (*38*)].

Next consider simulation of the same system on a classical computer. Monte Carlo techniques, which are polynomial in N, can extract some information about the ground state of such a local system by applying relaxational techniques, or estimate the time evolution of such a system by Green's function techniques, but they typically do so by dealing with a restricted class of wave functions (fermions are also problematic in more than one dimension) (*30–32*). In general, merely to specify an arbitrary wave function for the particles requires 2^N memory sites on a classical computer, and to compute its time evolution requires the exponentiation of $2^N \times 2^N$ matrices. For directly simulating the time evolution of an arbitrary state, the quantum simulation is exponentially faster than simulation by the classical computer. This result holds in general. Simulation of a quantum system with N variables that evolves according to a local Hamiltonian on a quantum computer requires a number of steps proportional to N. Simulation of the same system on a classical computer requires a number of steps exponential in N.

The quantum simulation becomes even

faster if the intrinsic parallelizability of the simulation can be exploited, as in the quantum cellular automata described in (*13*). Terms in the sum $\Sigma_i H_i$ that commute can be enacted simultaneously by applying operations to different variables in the simulator at once. The terms in H can be divided up into groups such that within each group, all terms commute. All terms within a group can be enacted in parallel. But because H is local, the number of groups is independent of N. In this case, the time required to perform the simulation is independent of the size of the local system simulated. On a parallel classical computer it still takes a number of steps exponential in N merely to write out the matrix H, let alone to exponentiate it.

In summary, a quantum computer can simulate closed quantum systems with local Hamiltonians efficiently. The simulation operates by inducing interactions between the quantum variables of the simulator that mimic the interactions between the variables of the system to be simulated. The simulator must use a number of quantum variables such as spin-½ particles that is proportional to the number of variables of the system to be simulated. Several bits in the simulator may be allocated to simulate a given local variable; for example, two quantum bits are required to specify the number and type of fermions at a lattice site in the Hubbard model. Although continuous variables complicate the computation, they can still be approximated discretely: N quantum bits are required to simulate a continuous local variable such as the position of a particle or the value of a field in a lattice gauge theory to N bits of accuracy. The simulation takes an amount of time proportional to the time over which the system is to be simulated. The number of basic experimental operations needed is proportional to the number of variables of the system. This scaling holds true for both time-independent and time-dependent Hamiltonians. That is, the quantum simulation takes resources of quantum computer time and memory space directly proportional to the time and space taken up by the system to be simulated. This contrasts with simulation of the system on a classical digital computer, which requires exponential resources in time and space.

Simulating Open Systems

All physical systems interact with their environment to a greater or lesser extent. Up to this point, the systems to be simulated have been assumed to undergo a Hamiltonian time evolution. Such systems are either closed and do not interact with their surroundings, or their interaction with the environment is effectively classical and can be described by a time-varying potential. However, preparing a system in a desired state and making measurements on the system necessarily subject the system to external influences such as dissipation and decoherence, effects that are conventionally treated with master equations or superscattering operators. As will now be demonstrated, preparation and measurement as well as environmental effects such as dissipation and decoherence can be easily included in a quantum analog simulation.

Consider a quantum system interacting with its environment. If the system has Hilbert space $\mathcal{H}_S$ and the environment has Hilbert space $\mathcal{H}_E$, then the Hilbert space for the system and environment together is $\mathcal{H}_S \otimes \mathcal{H}_E$. The joint time evolution for the system and environment is given by a unitary operator $U(t)$ acting on $\mathcal{H}_S \otimes \mathcal{H}_E$. The system and environment together can be described by a joint density matrix $\rho_{SE}(t) = U(t)\rho_{SE}(0)U^\dagger(t)$ (the dagger indicates Hermitian conjugation), and the state of the system on its own is given by the reduced density matrix $\rho_S(t) = \mathrm{tr}_E\rho_{SE}(t)$ obtained by taking the trace tr_E over the states of the environment. If the system and environment are initially uncorrelated, so that $\rho_{SE}(0) = \rho_S(0) \otimes \rho_E(0)$, then the time-evolved reduced density matrix can be written $\rho_S(t) = \mathcal{S}(t)[\rho_S(0)]$ where $\mathcal{S}(t)$ is a trace-preserving linear operator called a superscattering operator that depends only on $U(t)$ and $\rho_E(0)$.

To simulate an open system it is not necessary to simulate the entire behavior of the environment, but only the aspects of the environment's behavior that affect the system. In general, the effect of the environment can be mimicked by using at most as many variables as are required for the system. For some cases the effect of the environment can be encompassed by using a single quantum bit. (Below, it will be shown that sometimes not even a single bit is required; the effect of the environment on the system can be mimicked by the effect of the computer's environment on the computer.) The reason is as follows: The goal of the simulation is to make the variables in the simulator corresponding to the system evolve according to some desired superscattering operator $\mathcal{S}(t)$. $\mathcal{S}(t)$ is a strictly positive trace-preserving linear operator acting on an $m \times m$ Hermitian density matrix and so requires $m^4 - m^2$ parameters to define. It is straightforward to verify that

$$\mathcal{S}(t)[\rho_{\tilde{S}}(0)] = \mathrm{tr}_{\tilde{E}}\tilde{U}(t)\rho_{\tilde{S}}(0) \otimes \rho_{\tilde{E}}(0)\tilde{U}^\dagger(t) \quad (2)$$

for some unitary operator $\tilde{U}(t)$ acting on an m^2-dimensional simulated system-environment Hilbert space $\mathcal{H}_{\tilde{S}} \otimes \mathcal{H}_{\tilde{E}}$ and for some initial simulated environment state $\rho_{\tilde{E}}(0)$ (*39*). Finding the $m^4 - m^2$ parameters that define $\tilde{U}(t)$ and the m^2 parameters that define $\rho_{\tilde{E}}(0)$ requires the solution of an $m^2 \times m^2$ matrix equation [$\tilde{U}(t)$ is unique only up to arbitrary unitary transformations on $\mathcal{H}_{\tilde{E}}$]. Once the proper $\tilde{U}(t)$ has been determined, the simulation proceeds as in the Hamiltonian case above.

For example, consider a spin-½ particle evolving under an applied magnetic field and interacting with its environment according to the Bloch equation. Simulation of the time evolution of such a particle on a quantum analog computer requires two qubits. Approximately $2^4 - 2^2 = 12$ operations are required to take the particle from an initial state to a state to which it evolves under the Bloch equation.

System Preparation and Measurement

Consider an open-system operation that puts one of the simulator variables in a known state, such as cooling an ion in an ion-trap computer, or measuring the polarization of a photon in quantum optical logic device. Such an operation is either dissipative (as in cooling) or decohering (as in measurement). This operation can be combined with the sort of coherent operations used above to simulate Hamiltonian systems to prepare the remaining variables in any desired state. First, the variable is specified in a known state, then a unitary operation is effected that "pumps" entropy to the variable from the remaining variables, and then this procedure is repeated. If the variable is a quantum bit, then with each cycle of the pump the entropy of the remaining variables is decreased by one bit. Repeated pumping reduces their entropy to any desired value. By adjusting the unitary "pumping" operation, the remaining variables can be prepared in any desired state.

A closely analogous technique can be used to make arbitrary measurements on the simulator. Consider an operation that extracts some information from one of the simulator variables (the "read-out variable") while leaving the other variables unaffected. For example, photon polarizations can be detected with beam splitters and photodetectors, and the states of ions can be read by using fluorescence. The read-out operation need not be perfectly accurate, nor need it leave the read-out variable undisturbed.

An arbitrary measurement on the state of the simulator using this read-out operation is made by labeling with binary numbers the set of orthogonal states between which the measurement is to distinguish. A unitary transformation on the simulator is

then effected that correlates the states of the read-out variable with the value of the first bit of the number labeling the states; next, the read-out operation is performed and these steps are repeated until the first bit has been identified to the desired degree of certainty. At this point the same read-out process is performed on the second bit, and so on, until all bits required to identify the state have been determined. For example, the Kimble quantum logic gate could be used to correlate arbitrary polarization states of a single photon with the left- and right-circular polarization states of a sequence of read-out photons. By increasing the number of read-out photons, one can accurately measure the polarization of the photon despite the finite efficiency of photodetectors and the destruction of the read-out photons. This method provides accurate nondemolition measurements of arbitrary sets of states even when the only available read-out operation is noisy, inaccurate, and destructive.

System preparation and measurement are quantum computations in their own right. The preparations and measurements that can practically be performed are those that can be done efficiently, in a relatively small number of steps. Fortunately, in a quantum simulation, many desired quantities can be read out directly. For example, measurement of the correlation function $\langle M(0)M(t)\rangle$ in a simulated spin glass only requries making repeated measurements in which one has prepared the simulator variable corresponding to a spin in states of different initial magnetization $M(0)$, simulated the time evolution of the spins over time t, then measured the final magnetization of the spin $M(t)$. P repetitions are required to build up the correlation function to an accuracy $\sqrt{P}$.

Efficient Simulation of Local Open Systems

For arbitrary open systems as for closed systems, the difficulty of simulation rises exponentially as the size of the system increases. To simulate an arbitrary time evolution of an open system of N spin-½ particles takes $\approx 2^{4N}$ steps. But physical systems do not evolve arbitrarily. As with Hamiltonian systems, quantum simulation shows its power when the open system to be simulated evolves according to local interactions between its variables and between its variables and the environment. Consider as above a system with N variables, each of which interacts with at most k others, with Hamiltonian $H = \Sigma_{i=1}^{\ell} H_i$, where the H_i may be time-dependent as before. Now the system is allowed to interact with its environment with local interactions $H_E = \Sigma_{j=1}^{\ell E} H_j^E$, where each term in the sum acts on at most k_E variables of the system and environment. Because an open system is being simulated by embedding it in a closed system, Eqs. 1 and 2 hold as above for the combined system-environment simulation: Simulation of the time evolution of the joined system-environment to an accuracy ε over time t then requires a number of steps $\approx(\ell m^2 + \ell_E m_E^4)nt^2/\varepsilon$, where m and m_E are the maximum dimensions of the local system-environment Hilbert spaces acted on by H_i, H^{Ej}, as above (*40*). The time over which the simulation takes place is proportional to t, as in the case of the closed system. Just as for the case of closed systems, the quantum simulation of open systems uses a number of variables proportional to the number of system variables and takes time proportional to the time over which the system evolves.

For example, consider the problem of simulating the behavior of a set of spins that interact with each other and with the environment. Suppose that the interaction of each spin with its environment is Bloch-like, so that in the absence of spin-spin interactions, each spin would evolve according to the Bloch equation, with longitudinal relaxation time T_1, transverse relaxational time T_2, and natural frequency ω. In this situation, only one bit is required to simulate the environment because the same bit can be used to simulate first the environment of one spin, then the environment of another, and so on. Although the simulation progresses by small perturbations as in Eq. 1, it can still be used to extract nonperturbative features. For example, quantum computation could be used to simulate an annealing process to find the ground state of the system.

Exploiting the Environment

So far, the quantum simulator has been assumed to be itself a closed system, isolated from its environment. However, no real quantum computer is totally isolated; quantum computers are open systems subject to thermal fluctuations and decoherence. In addition, the tools used to control the quantum dynamics may themselves induce decoherence. For quantum computations such as factoring decoherence is a liability, but for quantum simulations of open systems decoherence can be an asset. Often, the interaction of the variables of the quantum computer with their environment can be used to mimic the interaction of the simulated system variables with their environment.

For example, consider the use of an ion-trap computer (*22*) to perform a quantum simulation of the coupled Bloch spins above. Suppose that each ion also has a Bloch-like interaction with its environment characterized by parameters $\tilde{T}_1$, $\tilde{T}_2$, and $\tilde{\omega}$,. These parameters are determined by characteristics of the computer's environment such as temperature, pressure, density, viscosity, and electric field strength and can be varied by manipulating the environment. By tuning the computer's Bloch parameters so that $\tilde{T}_1 = \alpha T_1$, $\tilde{T}_2 = \alpha \tilde{T}_2$, $\tilde{\omega} = \alpha^{-1}\omega$, the time over which the computer interacts with its environment can be adjusted so that the decoherence and noise induced in the qubits of the computer by its environment simulate the decoherence and noise induced in the spins by their environment.

This method of simulating the system environment by using the computer's environment works as long as the operators that determine the coupling of the computational variables to their environment have the same form as the operators that determine the coupling of the system variables to their environment. The temperature, pressure, and density of the computer's environment can then be tuned to make the effect of both environments the same up to a time scale. In the case of simulating spins with ions, ions at room temperature could be used to simulate spins occurring at microkelvin. The use of environments to simulate environments is more limited than the method for simulating the environment discussed in the previous section: The coupling of the computer to its environment must take the same functional form as the coupling of the system to its environment. Nonetheless, direct environmental simulation has the advantage of exploiting interactions that are present in any case and that would otherwise constitute unwanted noise.

The use of noise and decoherence to simulate noise and decoherence allows systems that are far too noisy and decoherent to factor numbers to function as effective simulators. Semiconductor quantum devices such as quantum dots and wells typically have decoherence time scales that are shorter than or comparable with the time required to flip them from one state to another. As a result, they are not an appropriate technology for performing long, coherent quantum computations. But they could prove suitable for simulating other noisy and decoherent systems.

Conclusion

Feynman was correct that quantum computers could provide efficient simulation of other quantum systems. A quantum computer with a few tens of quantum bits could perform in a few tens of steps simulations that would require Avogadro's number of memory sites and operations on a classical computer. A mere 30 or 40 quantum bits

would suffice to perform quantum simulations of multidimensional fermionic systems such as the Hubbard model that have proved resistant to conventional computational techniques. Hundreds to thousands of bits may be required to simulate accurately systems with continuous variables such as lattice gauge theories or models of quantum gravity. Current quantum logic devices can perform operations on two quantum bits (*23*, *24*, *26*); however, ion-trap quantum computers with a few tens of quantum bits apparently require only minor modifications of current technology (*24*). Although a quantum simulator with three or four quantum bits would be too small to solve classically intractable problems, it would still be large enough to test many of the ideas presented here. As suggested in (*13*), such simulators would also be able to create and test the properties of exotic quantum states such as Greenberger-Horne-Zeilinger states. Another possibility (*13*) is that by modulating the interactions between spins, atoms, or quantum dots in large arrays (*41*), one could perform massively parallel quantum simulations involving many quantum systems at once. The wide variety of atomic, molecular, and semiconductor quantum devices available suggests that quantum simulation may soon be a reality.

REFERENCES AND NOTES

1. P. Benioff, *J. Stat. Phys.* **22**, 563 (1980); *Phys. Rev. Lett.* **48**, 1581 (1982); *J. Stat. Phys.* **29**, 515 (1982); *Ann. N.Y. Acad. Sci.* **480**, 475 (1986).
2. R. P. Feynman, *Opt. News* **11**, 11 (1985); *Found. Phys.* **16**, 507 (1986); *Int. J. Theor. Phys.* **21**, 467 (1982).
3. W. H. Zurek, *Phys. Rev. Lett.* **53**, 391 (1984).
4. A. Peres, *Phys. Rev. A* **32**, 3266 (1985).
5. N. Margolus, *Ann. N.Y. Acad. Sci.* **480**, 487 (1986); in *Complexity, Entropy, and the Physics of Information*, vol. 8 of *Santa Fe Institute Series*, W. H. Zurek, Ed. (Addison-Wesley, Redwood City, CA, 1991), pp. 273–288.
6. D. Deutsch, *Proc. R. Soc. London Ser. A* **400**, 97 (1985).
7. See, however, D. Deutsch and R. Jozsa, *ibid.* **439**, 553 (1992), for an interesting oracle problem that can be performed more quickly by quantum computers.
8. D. Deutsch, *ibid.* **425**, 73 (1989).
9. Y. Yamamoto, M. Kitegawa, K. Igeta, in *Proceedings of the 3rd Asia-Pacific Physics Conference*, Y. W. Chan, A. F. Lenng, C. N. Yang, K. Young, Eds. (World Scientific, Singapore, 1988), pp. 779–799.
10. G. J. Milburn, *Phys. Rev. Lett.* **62**, 2124 (1989).
11. R. Landauer, *Int. J. Theor. Phys.* **21**, 283 (1982); *Found. Phys.* **16**, 551 (1986); *Nature* **335**, 779 (1988); in *Nanostructure Physics and Fabrication*, M. A. Reed and W. P. Kirk, Eds. (Academic Press, Boston, 1989), pp. 17–29; *Phys. Today* **42**, 119 (October 1989); in *Proceedings of the 3rd International Symposium on Foundations of Quantum Mechanics*, Tokyo, 28 to 31 August 1989 (Physical Society of Japan, Tokyo, 1990), p. 407; *Physica A* **168**, 75 (1990); *Phys. Today* **44**, 23 (May 1991); in *Proceedings of the Workshop on Physics of Computation II*, Dallas, 17 to 20 November 1994, D. Matzke, Ed. (IEEE Computer Society, Los Alamitos, CA, 1994), pp. 54–58.
12. K. Obermayer, W. G. Teich, G. Mahler, *Phys. Rev. B* **37**, 8096 (1988); W. G. Teich, K. Obermayer, G. Mahler, *ibid.*, p. 8111; W. G. Teich and G. Mahler, *Phys. Rev. A* **45**, 3300 (1992).
13. S. Lloyd, *Science* **261**, 1569 (1993); *ibid.* **263**, 695 (1994).
14. P. Shor, in *Proceedings of the 35th Annual Symposium on Foundations of Computer Science*, Santa Fe, NM, 20 to 22 November 1994, S. Goldwasser, Ed. (IEEE Computer Society, Los Alamitos, CA, 1994), p. 124.
15. D. P. DiVincenzo, *Science* **270**, 255 (1995).
16. ———, *Phys. Rev. A* **50**, 1015 (1995).
17. A. Barenco, D. Deutsch, A. Ekert, R. Jozsa, *Phys. Rev. Lett.* **74**, 4083 (1995).
18. T. Sleator and H. Weinfurter, *ibid.*, p. 4087.
19. G. M. Palma, K.-A. Suominem, A. Ekert, *Proc. R. Soc. London Ser. A* **452**, 567 (1996).
20. S. Lloyd, *Phys. Rev. Lett.* **75**, 346 (1995).
21. D. Deutsch, A. Barenco, A. Ekert, *Proc. R. Soc. London Ser. A* **449**, 669 (1995).
22. J. I. Cirac and P. Zoller, *Phys. Rev. Lett.* **74**, 4091 (1995).
23. Q. A. Turchette, C. J. Hood, W. Lange, H. Mabuchi, H. J. Kimble, *ibid.* **75**, 4710 (1995).
24. C. Monroe, D. M. Meekhof, B. E. King, W. M. Itano, D. J. Wineland, *ibid.*, p. 4714.
25. W. G. Unruh, *Phys. Rev. A* **51**, 992 (1995).
26. M. Brune *et al.*, *Phys. Rev. Lett.* **72**, 3339 (1994); P. Domokos, J. M. Raimond, M. Brune, S. Haroche, *Phys. Rev. A* **52**, 3554 (1995).
27. R. Landauer, *Philos. Trans. R. Soc. London Ser. A* **353**, 367 (1995); in *Proceedings of the Drexel-4 Symposium on Quantum Nonintegrability: Quantum Classical Correspondence*, D. H. Feng and B. L. Hu, Eds. (International Press, Boston, 1996).
28. I. L. Chuang, R. Laflamme, P. W. Shor, W. H. Zurek, *Science* **270**, 1633 (1995).
29. P. W. Shor, *Phys. Rev. A* **52**, 2493 (1995); A. R. Calderbank and P. W. Shor, *Phys. Rev. A*, in press; A. Steane, in preparation.
30. M. H. Kalos, Ed., *Monte Carlo Methods in Quantum Problems* (Reidel, Dordrecht, Netherlands, 1984).
31. K. Binder, Ed., *The Monte Carlo Method in Condensed Matter Physics* (Springer-Verlag, New York, 1995).
32. C. Rebbi, Ed., *Lattice Gauge Theories and Monte Carlo Simulations* (World Scientific, Singapore, 1983).
33. W. S. Warren, D. P. Weitekamp, A. Pines, *J. Chem. Phys.* **73**, 2084 (1980).
34. V. Ramakrishna, M. V. Salapaka, M. Dahleh, H. Rabitz, A. Peirce, *Phys. Rev. A* **51**, 960 (1995).
35. A. Yao, in *Proceedings of the 34th Annual Symposium on Foundations of Computer Science*, S. Goldwasser, Ed. (IEEE Computer Society, Los Alamitos, CA, 1995), p. 352.
36. H. Mabuchi and H. J. Kimble, *Opt. Lett.* **19**, 749 (1993); A. B. Matsko, S. P. Vyatchanin, H. Mabuchi, H. J. Kimble, *Phys. Lett. A* **192**, 175 (1994); H. J. Kimble *et al.*, in *Proceedings of the International Conference on Atomic Physics*; A. S. Parkins, P. Marte, P. Zoller, H. J. Kimble, *Phys. Rev. Lett.* **71**, 3095 (1993).
37. For the simulated trajectory always to remain within ε of the desired trajectory, the experimenter must be able to enact not only $e^{iH_j t}$, but $e^{-iH_j t}$ as well. See, for example, R. W. Brockett, R. S. Millman, H. J. Sussman, Eds., *Differential Geometric Control Theory* (Birkhauser, Boston, 1983), and Z. Li and J. F. Canney, Eds., *Nonholonomic Motion Planning* (Kluwer Academic, Boston, 1993).
38. The extra factor of N is required to conform with the normal rules for phases of fermions on a lattice in second quantized form. See, for example, A. L. Fetter and J. D. Walecka, *Quantum Theory of Many-Particle Systems* (McGraw-Hill, New York, 1971), and J. W. Negele and H. Ormand, *Quantum Many-Particle Systems* (Addison-Wesley, Redwood City, CA, 1988).
39. K. Kraus, *States, Effects, and Operations: Fundamental Notions of Quantum Theory* (Springer-Verlag, Berlin, 1983).
40. There is an extra subtlety in the open system case. For the quantum analog simulation of open systems to be efficient, the time evolution of the open system must be a finite-order quantum Markov process, that is, the environment must have finite time and space correlation lengths.
41. C. B. Murray, C. R. Kagan, M. G. Bawendi, *Science* **270**, 1335 (1995).

42. Supported in part by grant N00014-95-1-0975 from the Office of Naval Research. This work is part of the Quantum Information and Computation project (QUIC).

8 February 1996; accepted 10 June 1996

A Crosslinked Cofactor in Lysyl Oxidase: Redox Function for Amino Acid Side Chains

Sophie Xuefei Wang, Minae Mure,* Katalin F. Medzihradszky, Alma L. Burlingame, Doreen E. Brown, David M. Dooley, Alan J. Smith, Herbert M. Kagan, Judith P. Klinman†

A previously unknown redox cofactor has been identified in the active site of lysyl oxidase from the bovine aorta. Edman sequencing, mass spectrometry, ultraviolet-visible spectra, and resonance Raman studies showed that this cofactor is a quinone. Its structure is derived from the crosslinking of the ε-amino group of a peptidyl lysine with the modified side chain of a tyrosyl residue, and it has been designated lysine tyrosylquinone. This quinone appears to be the only example of a mammalian cofactor formed from the crosslinking of two amino acid side chains. This discovery expands the range of known quino-cofactor structures and has implications for the mechanism of their biogenesis.

Lysyl oxidase (LO, E.C. 1.4.3.13) is an extracellular, matrix-embedded protein. It has a central role in the biogenesis of connective tissue by way of posttranslational oxidative modification of the ε-amino group of lysine side chains in elastin and collagen to form inter- and intrachain crosslinks (*1*, *2*). The physiologic importance of LO is well established. Decreased LO activity is observed in diseases of im-