Modeling quantum information systems

Paul E. Black[a] and Andrew W. Lane[b]

[a]National Institute of Standards and Technology, 100 Bureau Drive, Gaithersburg, MD, USA;
[b]University of Kentucky, 773 Anderson Hall, Lexington, KY, USA

ABSTRACT

A simulator for quantum information systems cannot be both general, that is, easily used for every possible system, and efficient. Therefore, some systems will have aspects which can only be simulated by cunning modeling. On the other hand, a simulation may conveniently do extra-systemic processing that would be impractical in a real system. We illustrate with examples from our quantum computing simulator, QCSim. We model the [3,1] Hamming code in the presence of random bit flip or generalized amplitude damping noise, and calculate the expected result in one simulation run, as opposed to, say, a Monte Carlo simulation, and keep the original state to compute the chance of successful transmission, too. We also model the BB84 protocol with eavesdropping and random choice of basis and compute the chance of information received faithfully. Finally, we present our simulation of teleportation as an example of the trade-off between complexity of the simulation model and complexity of simulation inputs and as an example of modeling measurements and classical bits.

Keywords: quantum computing, modeling, quantum simulation, BB84, teleportation

1. INTRODUCTION

It is impractical to write a single program that accurately handles all possible noisy quantum information systems. Physicists and engineers ingeniously use the quirks or exotic behaviors of physical systems, and they are unlikely to be constrained by what we can or cannot simulate easily. Therefore we must be similarly ingenious about modeling physical systems for simulation.

On the other hand, since simulators are computer programs, not physical constructs, simulations may include extra-systemic processing. We should be open to use extra qubits, gates, or operations beyond those in the physical system we model. For example, we can include extra qubits to model some aspect of the environment.

In contrast to writing a custom program to compute the solution to a problem, a simulator is a largely fixed piece of software that can be used for many problems*. The advantage is that the simulator is easier to use and the code can be validated to minimize software errors. The disadvantage is that it cannot possibly handle every physical system efficiently, as we said previously. However, in the field of quantum information we can limit ourselves to discrete time simulation, instead of continuous time models, and finite dimensional spaces without too much loss. Other simulator choices are not as clear.

1.1. Simulating With Concrete vs. Symbolic Values

Our first choice is between concrete and symbolic values. Concrete values are numbers: 0, $1/\sqrt{2}$, or $-.4i|10\!>$. Symbolic values are associated with variables. If the inputs to our simulation, for instance, the initial state or the probability of an error during a gate operation, are concrete values, we can simply compute the result.

In contrast, if the inputs can have symbolic values, we have more expressive power. For example we could conceivably make the initial state $\alpha|00\!>+\beta|01\!>+\gamma|10\!>+\delta|11\!>$ to represent any possible two qubit states. If we then, for instance, have the simulator apply a Hadamard operation to the second qubit, the resultant symbolic state is $((\alpha+\beta)|00\!>+(\alpha-\beta)|01\!>+(\gamma+\delta)|10\!>+(\gamma-\delta)|11\!>)/\sqrt{2}$. Symbolic amplitudes or probabilities can be handled in a straight-forward manner with properly programmed symbolic math packages. We could even use

paul.black@nist.gov, Telephone: 1 301 975 4794. awlane0@uky.edu

*Custom programs reuse code and take parameters, and simulators are often extended, so the distinction between the kinds of programs is not absolute.

symbolic values in the state description, for instance, $1/2|ab'> + \sqrt{3}/2|a'b>$. Although a boolean simplification package would handle symbolic states, expressions may grow extremely complex. Applying a Hadamard to the symbolic state $|a>$ yields $(|0> + (-1)^a|1>)/\sqrt{2}$. Even if we simulate with symbolic values, we may have to examine graphs of results or do extensive further analysis to understand the significance of a result.

1.2. Pure Quantum States, Mixed States, or a Combination

If a simulation only carries pure quantum states, e.g. those expressed with Dirac's ket notation, it cannot represent partial or total decoherence. A simulation needs to implicitly or explicitly handle mixed states to faithfully simulate decoherence or noise processes. The most general description of the state of a system with N two-level particles, or qubits, requires a density matrix: a $2^N \times 2^N$ matrix with complex entries. Instead of an explicit representation, we could use some data structure with the equivalent information content, like Quantum Information Decision Diagrams.[1] Since many states have a lot of regularity, they could even be represented as a tensor product of pure quantum states, classical variables and density matrices as needed.

To estimate the information redundancy in typical mixed states, we ran instrumented simulations of a simple error correcting code (ECC) circuit (Sec. 3.1) and of BB84 (Sec. 3.2). The ECC had a maximum of 24 non-zero entries, out of $2^{2\times 7} = 16\ 384$, and BB84 had a maximum of 48 non-zero entries, out of $2^{2\times 8} = 65\ 536$. A simulation of Grover's algorithm for six qubits (64 possible "database" entries) uses seven qubits. Most density matrix entries are non-zero, but there are very narrow spikes in the number of unique entries. Out of a total of 255 simulation steps, only 25 steps have more than 100 unique entries, and only three have more than 1000 unique entries. The maximum number of unique entries is 2592 of 16 384. These results are very strong indications that explicitly representing complete density matrices is wasteful.

An alternative way to maintain complete information about the state is to include extra "environment" qubits. In this case, only pure states of the combined model and environment qubits would be needed. We have not investigated this possibility.

1.3. A Single Complete Run, Randomized Simulation, or Branching Simulation

Rather than carrying the complete state information during the simulation, one might run many deterministic or nondeterministic simulations. A nondeterministic simulation, for instance a Monte Carlo or quantum trajectory approach, randomly chooses certain parameters in each run. Such a randomized simulation is run many times, and the outcomes are averaged to approximate the expected results of a physical system.[2]

A deterministic or branching simulation is limited to modeling systems with discrete possibilities. When the simulation reaches a choice, e.g. when a bit flip error may occur, it proceeds with one choice while saving the state so that it can later return and process the other choice(s). The state records the probability of the choices made so far. The weighted average of all runs is the expected result. We illustrate the possibility of a branching simulator in Sec. 3.2.

2. A QUANTUM INFORMATION SYSTEM SIMULATOR

In this section we describe the simulator we use, QCSim.[3] QCSim is written in C++. It does a discrete-time simulation of a system of 2-level quantum entities (qubits). QCSim reads a file describing the quantum system. The description is written in a very small subset of QHDL.[4] A description begins by declaring the names, and implicitly the number, of qubits. Next comes the initial state of the quantum system, specified in a Dirac or "ket" notation, which is normalized. The notation limits the initial state to a pure state. There is no mechanism to add qubits after the initial declaration. Any number of quantum gate operations or commands follow. To give the reader an idea of such description, Fig. 1 gives the QCSim file for the teleportation circuit graphically depicted in Fig. 4.

All operations and commands are applied to the current state, which is stored as complex numbers in a density matrix. On our machine, QCSim can only allocate memory for 13 qubits. Although QCSim only uses concrete values, the linear combination of tensored states allows us to simulate a base case, for instance $(|0> + |1>)/\sqrt{2}$, from which we can compute particular cases of interest.

```
variable psi, EPRpairAlice, EPRpairBob:  qubit;
= 2|000> + i|100>;                # The EPR pair qubits are both 0.
hadamard(EPRpairAlice);           # Create Bell state
cnot(EPRpairAlice, EPRpairBob);   # in the EPR pair.

cnot(psi, EPRpairAlice);          # Alice entangles her state
hadamard(psi);                    # with the EPR pair,
measure(psi);                     # then measures her qubits.
measure(EPRpairAlice);

cnot(EPRpairAlice, EPRpairBob);   # Bob applies the results
copZ(psi, EPRpairBob);            # to his EPRpair qubit.

traceOver(EPRpairAlice);
traceOver(psi);                   # We ignore Alice's qubits
print_state();                    # to see Bob's state clearly.
```

Figure 1. QCSim code to simulate quantum teleportation. Parameters are cnot(control, target).

Gate operations are also written in C++. They directly manipulate the state density matrix. That is, rather than multiplying the state by an operation and its complex conjugate, the operator performs a single manipulation of the matrix representing the state. Since quantum operations are linear, an N-qubit operation works with at most 2^{2N} entries at a time, inherently eliminating multiplications by 0.

For some operators, the manipulation is simple. The Pauli X operator simply swaps certain rows and certain columns, while the Pauli Z operator just negates certain off-diagonal entries, or *correlations*. We refer to the off-diagonal entries as correlations, as opposed to the state probabilities, which are entries on the diagonal. The particular rows, columns, or correlations effected are derived from the order of the tensored qubits. QCSim has the following single-qubit operations.

1. Pauli X, Y, and Z
2. rotate through an arbitrary angle about the Y axis
3. Hadamard
4. probabilistic X, Y, and Z gate[5]
5. generalized amplitude damping[5]
6. a synthetic noisy Hadamard[6]

It also has the following multi-qubit operations.

1. controlled not, controlled Z, and controlled Hadamard
2. not gates with two (Toffoli) to nine controlling qubits
3. depolarizing controlled Z or phase[7]

Using macros and existing functions, it is straight-forward to implement a new operation given its effect on a single qubit. The macros and functions encode nested loops, optimized for fastest memory access. Control qubits are implemented as a mask to indicate which values to manipulate and which values to leave alone.

In addition to gate operation, QCSim has the following special commands.

1. Print the current state as a density matrix.
2. Trace over a qubit.
3. Measure a qubit in the computational basis.

The density matrix on the left below is a general representation of the quantum state of a single qubit. Measuring the qubit in the computational basis changes the state to 0 with probability α and 1 with probability β. We can represent this mixed state with the density matrix on the right.

$$\begin{pmatrix} \alpha & \beta \\ \gamma & \delta \end{pmatrix} \overset{\text{measure}}{\rightsquigarrow} \begin{pmatrix} \alpha & 0 \\ 0 & \delta \end{pmatrix}$$

Probabilities are unchanged, and correlations are zeroed. The effect on density matrices representing many qubits can be derived from tensoring. The measurement operation in QCSim zeroes the appropriate correlations in the density matrix. The measurement outcome itself does not need to be stored anywhere, say, in a classical variable. Instead, the measured qubit can be used to condition gates, as in the teleportation example in Sect. 3.3.

3. MODELS OF THREE QUANTUM INFORMATION SYSTEMS

We now present our three examples. The first, a [3,1] Hamming Code for error correction, shows how we can use additional qubits and gate operations to compute useful information within the simulation itself. It also shows in passing how a seemingly irrelevant change may significantly improve a design in certain cases, strengthening the case for careful analysis or simulation of designs.

The next example, the BB84 protocol, demonstrates how certain operations, such as random choice or measurement in different bases, can be modeled with convenient primitives. We also point out how different modeling choices lead to different information from a simulation run.

Finally, the teleportation example shows trade-offs between what should be simulated and what should be handled as inputs to or controls for the simulation. It also shows that it may be better to have the information content of the simulation be a superset of the information content of the physical system.

3.1. Modeling [3,1] Hamming Code Error Correction

The simple [3,1] Hamming Code protects a qubit from one bit flip error by replicating it as three qubits. With a bit flip error model, where each qubit has a 1 % chance of being flipped, an error occurs when two or three qubits are flipped, giving an error rate of $.01 \times .01 \times (3 \times .99 + .01) = .000298$. Table 1 shows the result of a simulation with of this model. We trace over all but one qubit, then print the density matrix for the remaining qubit. As explained below, it is processed to give the error rate.

Figure 2 shows a quantum circuit simulation model. All qubits, except the original qubit, labeled "orig", are initialized to $|0>$. First, we make three replicas of the original qubit. An actual design might only add two additional qubits for replicas. In this model we keep the original qubit for later computation. Note that one of the replication qubits, rep2, is flipped before being encoded. For generalized amplitude damping errors, a flipped qubit can decrease the error rate by almost a third. Design-specific optimizations can help.[8]

Table 1. Simulated error rate of .000298 for an ECC using a noisy channel. The design uses a [3,1] Hamming code, and each qubit has a 1 % probability of a bit flip.

		$\|0>$	$\|1>$
no error	$\|0>$	0.999702	0
error	$\|1>$	0	0.000298

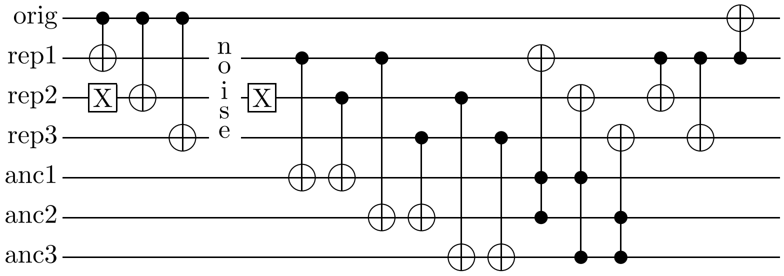

Figure 2. Model of a [3,1] Hamming code: encoding, noise, parity computation, correction and error rate computation.

Next a noise process is applied to all the replicas. The flipped replica is inverted, parities are computed using ancilla qubits, and the parity computations "correct" the replicas, if necessary. At this point, all three replicas would have the same value if there were no more than one bit flip. In a physical system, the replicas would probably be discarded. In the model, we use two extra-systemic cnot operations to disentangle the replicas.

The final gate is not in the physical system, either. It computes the error rate. If there is no error, the gate finishes swapping the original $|0>$ state of "rep1" into the original qubit yielding a 0 error rate. Tracing over all the qubits except the original then printing the resulting density matrix displays the error rate directly.

Using generalized amplitude damping noise with a 1 % chance of damping to $|0>$, an error only occurs when a 1 is sent and the two unflipped qubits (rep1 and rep3 in Fig. 2) are damped. Assuming an equal likelihood of sending 0 or 1, the error rate is $1/2(.01 \times .01) = .00005$. The simulation output is in Table 2.

With this model, we can easily see the result if rep2 is not flipped. An error occurs when a 1 is sent and two or three qubits are damped. In this case, shown in Table 3, the chance of an error is $1/2(3 \times .01 \times .01 \times .99 + .01 \times .01 \times .01) = .000149$, almost three times the error rate if a qubit is flipped.

3.2. Modeling the BB84 Protocol

The BB84 protocol[9] helps two separated parties establish a shared, secret key over an insecure quantum channel. More importantly, an eavesdropper cannot gain significant information about the key without causing detectable errors. Our model, shown in Fig. 3, includes the transmission, interception, final reception and reconciliation phases of the protocol. Privacy amplification is difficult to model since it deals with a series of photons, not a single photon. Since it is a classical computation, we leave it out. We use two extra qubit variables, infos unequal and error, and a few extra operations to compute the error rate as a part of the simulation.

The protocol begins with the transmitter, Alice, randomly choosing a basis. The decision on how to simulate random events, introduced in Sect. 1.3, is important here. A Monte Carlo, or quantum trajectory, approach randomly chooses the basis in each simulation run and averages the outcomes of many runs to approximate the

Table 2. Simulated error rate of .00005 for an ECC using a noisy channel. The design uses a [3,1] Hamming code, flips one qubit, and has generalized amplitude damping noise with 1 % chance of damping.

		$\|0>$	$\|1>$
no error	$\|0>$	0.99995	0
error	$\|1>$	0	5e-05

Table 3. Simulated error rate of .000149 for an ECC without the flipped qubit. Otherwise the same as in Table 2.

		$\|0>$	$\|1>$
no error	$\|0>$	0.999851	0
error	$\|1>$	0	0.000149

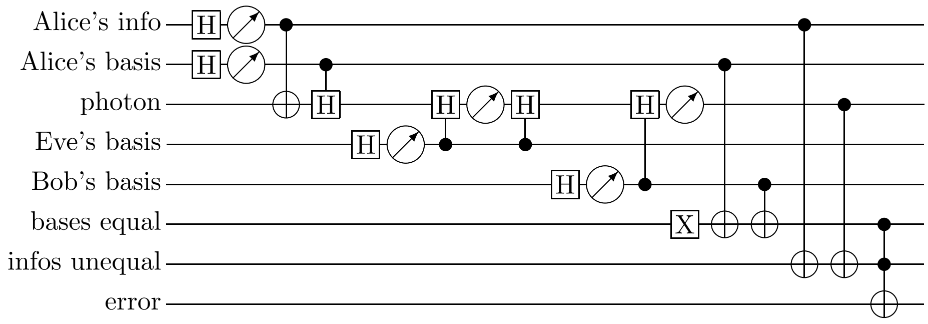

Figure 3. Model of BB84: transmission, interception, arrival, reconciliation and error rate computation.

expected results. A branching or deterministic simulations makes each possible choice in turn and finds the weighted average of all runs for the expected result.

Instead, we model Alice's random choice by a Hadamard operation followed by a measurement (the symbol). Since qubit states are modeled as a density matrix, this puts the qubit in a equally mixed state of 0 and 1—exactly what we want. Thus a single simulation run computes the result if Alice chooses a 0 basis and the result if she chooses a 1 basis.

Alice also sends information, 0's and 1's, on the photons. From a modeling standpoint the information is random: the system is (and should be!) insensitive to Alice's message. Hence we model the information as a random choice, like the basis: a Hadamard and a measurement. The density matrix has enough information content to handle all four possible choices of basis and information in a single computation. Using a cnot gate, Alice encodes the information on the photon and rotates the photon if a 1 basis is chosen. Here we could add a noise or error operation if we want to model the effect of noisy transmission channel.

We model the situation where Eve intercepts every photon. Or, this simulation only applies to photons what Eve intercepts. Having no information, Eve randomly chooses a basis and measures in that basis. We could have written code for a conditional measurement gate. Instead, we model it as a conditional rotation, then a measurement in the computational basis, then a complementary conditional rotation.

Next Bob gets the photon; we model his choice-and-measurement the same as Eve's. Since we no longer need the photon, only the 0 or 1 measurement, we do not need to conditionally rotate the photon back. We use the photon as a classical bit.

The quantum simulation could end here, since the remaining physical system is purely classical. However, as with the preceding example, we use additional qubit variables to model some of the classical system and even do the error rate computation. Alice and Bob reconcile their choices of bases. The variable "bases equal" carries this calculation. We calculate the error rate by comparing Alice's initial information and the state of the photon ("infos unequal"). If the bases are equal and the information is different, an error occurred. Because the simulator maintains probabilities of states, we can read the chance of different bases (50 %), no error (37.5 %) or error (12.5 %) directly from the result, shown in Table 4.

Table 4. Simulated error rate of BB84. Half the time the bases differ. Eavesdropping causes errors in 1/4 of the rest.

		$\vert 00>$	$\vert 01>$	$\vert 10>$	$\vert 11>$
bases differ (ignore errors)	$\vert 00>$	0.5	0	0	0
	$\vert 01>$	0	0	0	0
bases equal, no error	$\vert 10>$	0	0	0.375	0
bases equal, error	$\vert 11>$	0	0	0	0.125

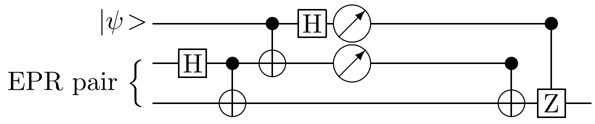

Figure 4. Model of quantum teleportation.

3.3. Modeling Quantum Teleportation

Creating a model to simulate quantum teleportation, shown in Fig. 4, is simpler than the preceding examples, but still has complications. In this example, we particularly show the trade off between having a more complex simulation model with simpler inputs or a simpler model with more elaborate inputs.

The QCSim syntax to express the initial state is limited to a sum of kets with complex amplitudes, which are normalized. We want to initialize $|\psi\rangle$ to the state shown in Table 5. We chose this state because it has a succinct printed representation (no irrational coefficients), yet all the matrix entries are different. With this state we can easily tell if, say, there were an error in our code or in our model that swaps the correlations. This state can also be succinctly specified by the following QCSim initialization statement, which is the second line in Fig. 1.

```
= 2|000> + i|100>;
```

This initializes the the system to $(\sqrt{.8}|0\rangle + \sqrt{.2}\, i|1\rangle)|00\rangle$. Specifically the qubits labeled "EPR pair" are initialized to $|00\rangle$. We can put them in the Bell state with an additional Hadamard and cnot gate. With this choice we can easily change the initial state of $|\psi\rangle$.

The alternative to giving a simple initial state and creating the EPR pair during the simulation is to initially specify the Bell state for the EPR pair and $|\psi\rangle$ with the following QCSim initialization statement.

```
= 2|000> + 2|011> + i|100> + i|111>;
```

The state going into the teleportation circuit proper is the same either way. Clearly QCSim's syntax could be extended to more easily specify this particular case. Regardless, one often finds similar trade-offs between putting more complexity in the simulation model itself, putting complexity into the inputs or simulation controls, or doing more analysis after the simulation.

In another aspect of modeling, it turns out that Alice's measurements are not needed in this simulation at all: we can apply the entire state of Alice's qubit of the EPR pair and ψ to Bob's qubit of the EPR pair and get the right result. However, in a physical teleportation realization we only send classical information, so we use measurement operations to be confident that only classical information is used. Since QCSim does not have classical bit variables per se and the measurement operation just collapses the quantum state into a classical (though possibly mixed) state, we simply continue to use the two qubits that were measured.

4. CONCLUSIONS

To accurately and efficiently simulate a noisy quantum information system consisting of more than about a dozen qubits, a mixed mode simulator is needed. A mixed mode simulator uses classical bits, pure quantum states, and sparse or compressed density matrices as needed to represent the state of a quantum system.

Table 5. Simulated result of teleporting a quantum state. This is identical to the initial state of $|\psi\rangle$.

	$\|0\rangle$	$\|1\rangle$
$\|0\rangle$	0.8	0-0.4i
$\|1\rangle$	0+0.4i	0.2

Even with such a simulator, modeling a physical system may take ingenuity. The ECC example demonstrates how extra-systemic qubits and gates can compute results within the simulation, reducing the need to analyze simulation results later. It also shows in passing how a seemingly insignificant difference in a physical implementation can greatly improve the performance of a quantum information system. The BB84 model illustrates how random choice and classical bits can be simulated with Hadamard operations and measurement. It again shows that classical processing steps in an algorithm may be excluded or included for different reasons. Teleportation exemplifies one possible trade off between model complexity and input complexity.

To write faithful, efficient models of quantum information systems, we must creatively map physical systems into simulation primitives.

ACKNOWLEDGMENTS

We thank Matthew Cass, who wrote the initial version of QCSim in 2002 at NIST. This work was partially supported by the Quantum Information Science and Technology (QuIST) program of the Defense Advanced Research Projects Agency under contract L492.

REFERENCES

1. G. F. Viamontes, I. L. Markov, and J. P. Hayes, "Improving gate-level simulation of quantum circuits," *Quantum Information Processing* **to appear**, 2004. quant-ph/0309060.
2. J. Niwa, K. Matsumoto, and H. Imai, "General-purpose parallel simulator for quantum computing." quant-ph/0201042, Jan. 2002.
3. P. E. Black and A. W. Lane, "QCSim." http://hissa.nist.gov/%7Eblack/Quantum/qcsim.html, Mar. 2004.
4. M. Stillerman, "A design language for quantum computing," Tech. Rep. TR-03-0001, Odyssey Research Associates, Inc., Ithaca, New York, 2003.
5. M. A. Nielsen and I. L. Chuang, *Quantum Computation and Quantum Information*, pp. 376–382, Cambridge University Press, 2000.
6. E. Hsu, "Quantum computing simulation optimizations and operational errors on various 2-qubit multiplier circuits." quant-ph/0208113, Aug. 2002.
7. G. K. Brennen, D. Song, and C. J. Williams, "A quantum computer architecture using nonlocal interactions." quant-ph/0301012, Jan. 2003.
8. H. D. Raedt, A. Hams, K. Michielsen, S. Miyashita, and K. Saito, "On the problem of programming quantum computers." quant-ph/0008015, Sept. 2000.
9. C. H. Bennett and G. Brassard, "Public key distribution and coin tossing," in *Proceedings of IEEE International Conference on Computers, Systems and Signal Processing*, pp. 175–179, IEEE, December 1984. Bangalore, India, 1984.