QUANTUM SIMULATION USING A RECURSION

J. KEITH KENEMER

Abstract. Quantum computation can be viewed as a recursion if the amplitudes associated with each eigenstate are written as a function of the bitfields of the eigenstate. This recursion provides some insight into the boundary between classical and quantum computation and suggests several scenarios for performing efficient quantum simulation.

1. A Universal Set of Quantum Logic Gates (Review)

The Hadamard gate, phase gate, controlled-NOT gate, and $\frac{\pi}{8}$-gate are universal for quantum computation. The operation of these quantum logic gates will be briefly reviewed and used to construct a recursive interpretation of quantum computation. The Hadamard gate is defined by the following single-qubit operation which transforms the qubit into a superposition with a phase that depends on the state of the original qubit:

$$H = \frac{1}{\sqrt{2}}\begin{bmatrix}1 & 1\\ 1 & -1\end{bmatrix} \qquad |\, b\,\rangle \rightarrow \frac{1}{\sqrt{2}}(|\, 0\,\rangle + (-1)^b \,|\, 1\,\rangle) \tag{1}$$

The Phase gate is defined by the following single-qubit operation which generates a phase factor dependent on the original qubit state:

$$S = \begin{bmatrix}1 & 0\\ 0 & i\end{bmatrix} \qquad |\, b\,\rangle \rightarrow e^{\frac{i\pi}{2}b} \,|\, b\,\rangle \tag{2}$$

Similarly, the $\frac{\pi}{8}$-gate is defined by the single-qubit operation which also generates a phase factor dependent on the original qubit state:

$$T = \begin{bmatrix}1 & 0\\ 0 & e^{\frac{i\pi}{4}}\end{bmatrix} \qquad |\, b\,\rangle \rightarrow e^{\frac{i\pi}{4}b} \,|\, b\,\rangle \tag{3}$$

The CNOT (controlled-not) gate performs a conditional negation of the target qubit based on the value of the control qubit. The operation preserves the state of the control qubit and replaces the target qubit with the XOR of the control and target qubits.

$$CNOT = \begin{bmatrix}1 & 0 & 0 & 0\\ 0 & 1 & 0 & 0\\ 0 & 0 & 0 & 1\\ 0 & 0 & 1 & 0\end{bmatrix} \qquad |\, ...b_c...b_t...\rangle \rightarrow |\, ...b_c...b_c \oplus b_t...\rangle \tag{4}$$

where the bitfields $b \,\epsilon\, \{0, 1\}$ and Dirac notation is used for the basis states:

$$|0\rangle = \begin{bmatrix}1\\ 0\end{bmatrix} \quad |1\rangle = \begin{bmatrix}0\\ 1\end{bmatrix} \tag{5}$$

2. Quantum Computation as a Recursion

The state of a quantum computation at any time is given by the wavefunction, which is represented by a linear combination of eigenstates, each weighted by the associated amplitude. Each eigenstate represents a possible output value of the computation. The state is a number, decomposable into bitfields, which means that the amplitude associated with each eigenstate is a multivariate function of the eigenstate bitfield values.

$$\psi = \sum_n \alpha_n |\phi_n\rangle = \sum_n \alpha(b_N ... b_k ... b_0) \mid b_N ... b_k ... b_0\rangle \tag{6}$$

$$\alpha = \alpha(b_N, ..., b_k, ...b_0) \tag{7}$$

where each index has the binary representation: $n = \sum_{i=0}^{N-1} b_i 2^i$. If we perform a Hadamard operation on the k-th qubit, the amplitude function will be transformed. Prior to this operation, we note that the wavefunction sum can be partitioned into two components, one where the k-th qubit happens to be 0 and another where it happens to be 1. Therefore the wavefunction can equivalently be written as:

$$\psi = \sum_n \alpha(b_{N-1}...0...b_0) \mid b_{N-1}...b_{k+1}\rangle \mid 0\,\rangle \mid b_{k-1}...b_0\rangle \tag{8}$$

$$+ \sum_n \alpha(b_{N-1}...1...b_0) \mid b_{N-1}...b_{k+1}\rangle \mid 1\,\rangle \mid b_{k-1}...b_0\rangle$$

After performing the Hadamard operation on the kth qubit, the state will be transformed to the following:

$$\psi_H = \sum_n \alpha(b_{N-1}...0...b_0) \mid b_{N-1}...b_{k+1}\rangle \frac{\mid 0\,\rangle + \mid 1\,\rangle}{\sqrt{2}}\rangle \mid b_{k-1}...b_0\rangle \tag{9}$$

$$+ \sum_n \alpha(b_{N-1}...1...b_0) \mid b_{N-1}...b_{k+1}\rangle \frac{\mid 0\,\rangle - \mid 1\,\rangle}{\sqrt{2}}\rangle \mid b_{k-1}...b_0\rangle$$

Combining terms where the k-th qubit is 0 together and terms where the kth-qubit is 1 together, we have:

$$\psi_H = \sum_n \frac{[\alpha(b_{N-1}...0...b_0) + \alpha(b_{N-1}...1...b_0)]}{\sqrt{2}} \mid b_{N-1}...b_{k+1}\rangle \mid 0\,\rangle \mid b_{k-1}...b_0\rangle \tag{10}$$

$$+ \sum_n \frac{[\alpha(b_{N-1}...0...b_0) - \alpha(b_{N-1}...1...b_0)]}{\sqrt{2}} \mid b_{N-1}...b_{k+1}\rangle \mid 1\,\rangle \mid b_{k-1}...b_0\rangle$$

If we view a quantum logic operation as an iteration step, then the amplitudes at the next step (after the Hadamard gate) can be written in terms of the prior amplitudes. By comparing Equations (8) and (10), we see that the amplitudes have been transformed according to the following:

$$\alpha_{n+1}(b_{N-1}...b_k...b_0) = \frac{\alpha_n(b_{N-1}...0...b_0) + (-1)^{b_k}\alpha_n(b_{N-1}...1...b_0)}{\sqrt{2}} \tag{11}$$

Using Equation (2) and (8), we see that the Phase gate operation on the k-th qubit would product the state:

$$\psi_S = \sum_n \alpha(b_{N-1}...0...b_0) \mid b_{N-1}...b_{k+1}\rangle \mid 0\,\rangle \mid b_{k-1}...b_0\rangle \tag{12}$$
$$+\ e^{\frac{i\pi}{2}} \sum_n \alpha(b_{N-1}...1...b_0) \mid b_{N-1}...b_{k+1}\rangle \mid 1\,\rangle \mid b_{k-1}...b_0\rangle$$

and so the associated amplitude transformation for the Phase gate operating on the k-th qubit is:

$$\alpha_{n+1}(b_{N-1}...b_k...b_0) = e^{\frac{i\pi}{2}b_k}\alpha_n(b_{N-1}...b_k...b_0) \tag{13}$$

The $\frac{\pi}{8}$-gate is very similar to the Phase gate, differing only in the phase factor, which results in the following amplitude transformation:

$$\alpha_{n+1}(b_{N-1}...b_k...b_0) = e^{\frac{i\pi}{4}b_k}\alpha_n(b_{N-1}...b_k...b_0) \tag{14}$$

From Equation (4), we see that the CNOT gate performs a 'basis-swapping' operation. Any basis vector where the (control, target) qubits = (1,0) gets replaced with the basis vector where these qubits have the values (1,1). Conversely, any basis vector where the (control, target) qubits = (1,1) get replaced with a basis vector where these qubits are (1,0). Since basis-vector swapping is equivalent to swapping the associated amplitudes, we see that the amplitude transformation for the CNOT gate operating on the b_c and b_t (control and target) qubits is:

$$\alpha_{n+1}(b_{N-1}...b_c...b_t...b_0) = \alpha_n(b_{N-1}...b_c...b_c \oplus b_t...b_0) \tag{15}$$

The quantum logic operations and associated amplitude transformations (iterations) are summarized in the following table:

Hadamard Gate	$\alpha_{n+1}(b_{N-1}...b_k...b_0) = (1/\sqrt{2})[\alpha_n(b_{N-1}...0...b_0) + (-1)^{b_k}\alpha_n(b_{N-1}...1...b_0)]$
Phase Gate	$\alpha_{n+1}(b_{N-1}...b_k...b_0) = e^{\frac{i\pi}{2}b_k}\alpha_n(b_{N-1}...b_k...b_0)$
$\frac{\pi}{8}$ Gate	$\alpha_{n+1}(b_{N-1}...b_k...b_0) = e^{\frac{i\pi}{4}b_k}\alpha_n(b_{N-1}...b_k...b_0)$
CNOT Gate	$\alpha_{n+1}(b_{N-1}...b_c...b_t...b_0) = \alpha_n(b_{N-1}...b_c...b_c \oplus b_t...b_0)$

TABLE 1. Quantum Logic Operations and Associated Amplitude Recursion

In order to perform quantum simulation using the recursion method, we would ideally like to represent the state of the system, which in this case will be the amplitude function, with a polynomial number of classical memory resources and then find the maxima of the magnitude-squared of the amplitude function to find the the highest-probability eigenstates corresponding to the desired solution.

The form of the recursion provides some insight into the subtle nature of quantum computation and suggests several scenarios on how we could use this representation to perform quantum simulation. From the Table 1 summary we can make the following observations:

(1) The amplitude for a specific eigenstate (with fixed bitfield values) can be determined with a polynomial number of classical memory resources (when expressions are not expanded) and for some cases in polynomial time, depending on the topology of the quantum circuit. Small quantum circuits can be simulated by computing this amplitude for all possible eigenstates and using this to compute probabilities.

(2) The amplitude function for any network of Phase, $\frac{\pi}{8}$, and CNOT gates can be represented symbolically using a polynomial number of classical memory resources. A symbolic representation could be any encoding which captures the structure of the recursion in Table 1. Even if the amplitude function is 'expanded' so that all terms for lower levels in the recursion are written out fully at the current level, the memory requirements remain polynomial for this sub-class of quantum circuits. If we assume the maxima of the magnitude-squared of the amplitude function can be found in polynomial time, then the overall simulation could be done in polynomial time. From Table 1, we see that the description of the amplitude function grows linearly with the addition of any of these gates. For the Phase and $\frac{\pi}{8}$ gates, only a small number of bits are required to represent the addition of the phase factors to the amplitude function. For the CNOT gates, the target qubit gets replaced with an XOR of the control and target qubits that this adds only a small number of bits to the description of the amplitude function.

(3) The Hadamard gate presents a challenge, as the symbolic description of the amplitude function approximately doubles with each iteration step in the general case since there are two terms in the Hadamard operation which must be represented/computed from the lower levels of recursion. If the expressions are not expanded the computation can be described with polynomial memory resources but is not polynomial in time due to having two calls in each branch of the recursion. This will not be true in all cases, however, and optimization of the recursion will be considered in the next section. One special case where the Hadamard gate does not present a problem is the generation of the initial superposition, i.e. when an initial state of $|0\rangle$ for all qubits is transformed to a superposition of all basis states with amplitude equal to $\frac{1}{\sqrt{2}^N}$ where N is the number of qubits. This actually corresponds to the simplest possible state in the recursion method, where the amplitude function is a constant.

(4) A trade-off between memory resources and generality of the input states can be made at any step in the iteration. In other words, if we find that memory resources become limited at a particular step of the iteration, we can start to fix bitfields or functions of bitfields, e.g. XORs of bitfields in order to collapse the representation of the amplitude function into a smaller memory space. This is an interesting feature of the recursive method.

3. Optimizing the Recursion

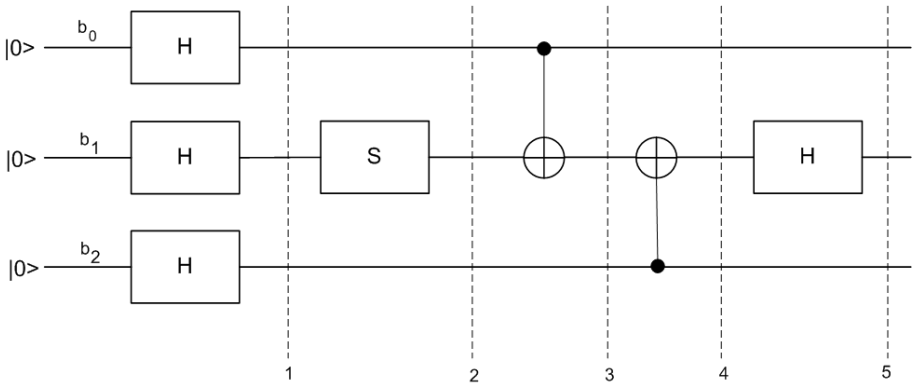
In the previous section we saw that the Hadamard gate creates a requirement for exponential memory resources in the general case when the recursion terms are written out fully and brought up to the current iteration level, i.e when the recursion is 'expanded'. When the computation is not expanded (written in recursive form), it requires only polynomial memory resources but then results in exponential time complexity due to the multiple calls to lower levels to implement the two terms present in the Hadamard operation in each branch of the recursion. There are some special cases where the Hadamard operation can be optimized. An important role of the quantum simulation compiler would be to identify these (and other) special cases that allow for optimization. The following summarizes some of the straightforward optimizations:

(1) The Hadamard operation can be simplified if the amplitude function is quasi-separable, i.e. can be written as a product where at least one of the factors is independent of the bit-field the Hadamard is operating on. When this is the case, the independent portion can be factored out and so the Hadamard gate only operates on the bit-field dependent factor, resulting in a shorter description required for the resulting amplitude function. For example, consider a Hadamard operation on the k-th qubit where the first α_1 factor is independent of the k-th bit-field:

$$\alpha(b_N...b_k...b_0) = \alpha_1(b_N...b_m)\alpha_2(...b_k...) \tag{16}$$

$$H_k[\alpha] = \alpha_1(b_N...b_m)[\alpha_2(...0...) + (-1)^{b_k}\alpha_2(...1...)] \tag{17}$$

(2) The overall simulation can be optimized for memory resources by fixing bit-fields at the expense of reducing generality, as noted in Observation(3). This allows classes of input states to be explored while simplifying the amplitude function. The following simple example illustrates this type of optimization:



The initial Hadamard gates generate all possible 3-bit basis states with equal probability and amplitude $\alpha_1 = \frac{1}{2\sqrt{2}}$. The subsequent gates transform this constant

state according to Table 1. Using the recursion method, the amplitude at each stage in the quantum circuit is calculated as follows:

$$\alpha_2 = \alpha_1 e^{i\frac{\pi}{2}b_1} \tag{18}$$

$$\alpha_3 = \alpha_1 e^{i\frac{\pi}{2}b_0 \oplus b_1} \tag{19}$$

$$\alpha_4 = \alpha_1 e^{i\frac{\pi}{2}b_0 \oplus b_1 \oplus b_2} \tag{20}$$

$$\alpha_5 = \alpha_1 [e^{i\frac{\pi}{2}b_0 \oplus b_2} + (-1)^{b_1} e^{i\frac{\pi}{2}\overline{b_0 \oplus b_2}}] \tag{21}$$

If we impose the constraint $b_0 \oplus b_2 = 0$ then we have imposed the parity constraint that both bits are equal. For this class of input states, the amplitude function would reduce to:

$$\alpha'_5 = \alpha_1 [1 + (-1)^{b_1} e^{i\frac{\pi}{2}}] \tag{22}$$

and so the generality of the input space has been traded for a more compact amplitude representation. It is worth noting that long XOR sequences provide a good optimization opportunity since fixing the value of a long XOR sequence imposes only a mild contraint (parity over many bits) while simultaneously simplifying the amplitude function significantly.

(3) A time-space tradeoff can be made by leaving some terms in the recursion in their unexpanded form. For the case of the Hadamard operation, this can save memory resources at the expense of more computation. For example, consider α_5 rewritten such that it refers to level 4 of the recursion:

$$\alpha''_5 = \alpha_4(b_0, 0, b_2) + (-1)^{b_1} \alpha_4(b_0, 1, b_2) \tag{23}$$

This representation will in general be more compact than the expanded form (Equation (21)), but would require two evaluations of α_4 in any type of calculation involving the amplitude function.

(4) Bitfield constraints can be generalized to include continuous values. One method for quantum simulation would be to utilize a continuous-valued amplitude function and to perform some type of gradient search or genetic algorithm search by fixing specific bit-fields and moving continuously through other bit-fields to search for maxima of the magnitude-squared of the amplitude function. The boolean-valued bitfields would simply be replaced with continuous variables and the XOR generalized to accept continuous values, i.e. $XOR(x, y) = x(1-y) + y(1-x)$. It would be interesting to see how well this type of algorithm would scale up when the number of qubits is large.

4. Summary

The proposed method views quantum computation as a recursion and provides a simple framework for quantum simulation as an alternative to a matrix-based approach. This approach could be useful in exploring the boundary between quantum and classical computation since it provides a great deal of flexibility in trading off computational resources and generality of the input states through the use of constraints. This framework also identifies the action of the Hadamard gate as being an important component in classical non-computability of the quantum computation in the general case. Quantum circuit topologies where the description of the amplitude function is compressible will be amenable to classical simulation.