5-2020

Why It Is Sufficient to Have Real-Valued Amplitudes in Quantum Computing

Isaac Bautista

Vladik Kreinovich

Olga Kosheleva

Nguyen Hoang Phuong

Comments:
Technical Report: UTEP-CS-20-48

Why It Is Sufficient to Have Real-Valued Amplitudes in Quantum Computing

Isaac Bautista[1], Vladik Kreinovich[1], Olga Kosheleva[4], and
Hoang Phuong Nguyen[2]
[1]University of Texas at El Paso, 500 W. University
El Paso, TX 79968, USA
ibautista@miners.utep.edu, vladik@utep.edu, olgak@utep.edu
[2]Division Informatics, Math-Informatics Faculty
Thang Long University, Nghiem Xuan Yem Road
Hoang Mai District, Hanoi, Vietnam
nhphuong2008@gmail.com

Abstract

In the last decades, a lot of attention has been placed on quantum algorithms – algorithms that will run on future quantum computers. In principle, quantum systems can use any complex-valued amplitudes. However, in practice, quantum algorithms only use real-valued amplitudes. In this paper, we provide a simple explanation for this empirical fact.

1 Formulation of the Problem

Need for quantum computing. For many practical problems, there is still a need for faster computations. For example, current spectacular successes of deep learning (see, e.g., [2]) could be even more spectacular if we could process even more data.

Computers' ability to process information is limited, among other thing, by the fact that all speeds are bounded by the speed of light. Even with a speed of light, sending a signal from one side of a 30 cm-size laptop to another takes 1 nanosecond – the time during which even the cheapest of current computers performs at least 4 operations. So, to make computations faster, it is necessary to make computer components much smaller. Already these components – such as memory cells – consist of a small number of molecules. If we make these cells much smaller, they will consist of only a few molecules. To describe the behavior of such small objects, it is necessary to take into account quantum physics – the physics of the microworld; see, e.g., [1, 4]. Thus, computers need to take into account quantum effects.

Successes of quantum computing. At first, computer engineers viewed quantum effects as a nuisance – since in quantum physics, everything is probabilistic, but we want computers to always produce the same correct result, with probability 1. Because of this probabilistic character of quantum physics, we cannot simply use the same algorithms on the quantum-level computers, we need to come up with new algorithms, algorithms that would provide reliable answers even in the probabilistic environment of quantum physics.

Such algorithms have been invented; see, e.g.., [3]. Interestingly, many of them require even fewer computational steps than the usual non-quantum algorithms. The two most well-known examples are:

- Grover's algorithm that finds an element with the desired property in an unsorted n-element array in time proportional to $\sqrt{n}$ – while non-quantum algorithms require at least n steps, and
- Shor's algorithm that factors large n-digit integers in time bounded by a polynomial of n; this may sound like an academic problem until one realizes that most existing encodings protecting our privacy and security are based on the fact that with non-quantum algorithms, the only known algorithms for such factorization require physically impossible exponential time.

Main idea behind quantum computing. How come quantum computing algorithms can be so much faster? The main explanation is that in quantum physics, with every two states s an s', we can also have superpositions of these states, i.e., states of the type

$$a \cdot s + a' \cdot s',$$

where a and a' are complex numbers (known as *amplitudes*) for which

$$|a|^2 + |a'|^2 = 1.$$

Philosophers and journalists are still arguing among the example – proposed by Nobelist Schroedinger, one of the founding fathers of quantum physics – that we can have a composition of a dead cat and an alive cat, but for particles, such superpositions have been experimentally observed since the early 20th century.

In particular, in a quantum computer, in addition to the usual 0 and 1 states of every bit – which in quantum computing are denoted $|0\rangle$ and $|1\rangle$ – we can also have superpositions of these states, i.e., states of the type

$$c_0|0\rangle + c_1|1\rangle,$$

where c_0 and c_1 are complex numbers for which

$$|c_0|^2 + |c_1|^2 = 1.$$

A quantum system corresponding to a bit is known as an *quantum bit*, or *qubit*, for short.

If we measure the state of the qubit, we will get 0 with probability $|c_0|^2$ and 1 with probability $|c_1|^2$. The fact that these probabilities should add up to 1 explains the above restriction on the coefficients.

Similarly, for 2-bit combinations, in addition to the traditional (non-quantum) states 00, 01, 10, and 11, we can have superpositions

$$c_{00}|00\rangle + c_{01}|00\rangle + c_{10}|00\rangle + c_{11}|11\rangle,$$

where c_{00}, c_{01}, c_{10}, and c_{11} are complex numbers for which

$$|c_{00}|^2 + |c_{01}|^2 + |c_{10}|^2 + |c_{11}|^2 = 1.$$

In general, for an n-qubit system, we can have states

$$c_{0\ldots00}|0\ldots00\rangle + c_{0\ldots01}|0\ldots01\rangle + \ldots + c_{1\ldots11}|1\ldots11\rangle$$

characterized by a complex-valued vector

$$c = (c_{0\ldots00}, c_{0\ldots01}, \ldots, c_{1\ldots11}).$$

How can this help with computations? For example, in the non-quantum search-in-an-array algorithm, the only thing we can do is select an integer i and check whether the i-th element of the array satisfies the desired property. This way, if we make fewer than n checks, we check fewer than n elements and we may miss the desired element – this explains why we need at least n computational steps in the non-quantum case.

In quantum physics, instead of asking for an element number i, we can submit a request which is a superposition of some integers, $c_i|i\rangle + c_{i'}|i'\rangle + \ldots$ This way, in effect, we can check several elements in one step.

This is just an idea, *not* an explanation of Grover's algorithm – on the one hand, we can check several elements, but on the other hand, if we do it naively, the results will be probabilistic – and we want guaranteed bounds. So, to compensate for the probabilistic character of the quantum measurements, we need to use quite some ingenuity. Sometimes, it works – as in Grover's and Shor's cases, sometimes it does not.

Unitary transformations and beyond. When we describe a bit in non-quantum physics, what is important is that we have a system with two states. Which of the two states is associated with 0 and which with 1 does not matter that much. From this viewpoint, all the properties of the bit system are invariant with respect to a swap $0 \leftrightarrow 1$.

In the quantum case, in addition to a swap, we can also have arbitrary unitary transformation $c \to Tc$, where T is a *unitary* matrix, i.e., a matrix for which $TT^\dagger = T^\dagger T = I$, where I is the unit matrix, $T^\dagger_{ij} \stackrel{\text{def}}{=} T^*_{ji}$, and z^8 denotes complex conjugate:

$$(x + y \cdot \mathrm{i})^* \stackrel{\text{def}}{=} x - y \cdot \mathrm{i} \text{ and } \mathrm{i} \stackrel{\text{def}}{=} \sqrt{-1}.$$

In particular, for each qubit, we can have Walsh-Hadamard transformations – actively used in quantum computing – in which

$$T|0\rangle = \frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle$$

and

$$T|1\rangle = \frac{1}{\sqrt{2}}|0\rangle - \frac{1}{\sqrt{2}}|1\rangle.$$

Invariance means, in particular, that for any quantum algorithm that uses states s_1, s_2, etc., and for every unitary transformation T, we can perform the same computations by using instead states Ts_1, Ts_2, etc.

Unitary transformation maps each vector form the original space into a vector from the same space, and preserves the vector's length

$$\|(c_1, c_2, \ldots)\|^2 \stackrel{\text{def}}{=} |c_1|^2 + |c_2|^2 + \ldots$$

One can also consider generalized unitary transformations, when each vector is mapped into a vector from a possibly higher-dimensional space – as long as this transformation preserves the lengths of all vectors. Similarly, for any quantum algorithm that uses states s_1, s_2, etc., and for every generalized unitary transformation T, we can perform the same computations by using instead states Ts_1, Ts_2, etc.

Interesting phenomenon. Many researchers have come up with many creative quantum algorithms for solving important practical problems. And there is a general – and somewhat unexpected – feature of all these algorithms:

- while in general, we can have state with general complex values of the coefficients c_i, but
- in all proposed algorithms, the coefficients are real-valued!

It should be mentioned that this does not mean that we cannot use non-real complex values in these algorithms. For example, one can see that all probabilities remain the same if instead of the original coefficients c_i, we use coefficients $c_i' \stackrel{\text{ef}}{=} \exp(\alpha \cdot \mathrm{i}) \cdot c_i$, where α is a real-valued constant. In particular, if we take

$$\alpha = \frac{\pi}{2},$$

we can replace all real values c_i with purely imaginary values $\mathrm{i} \cdot c_i$.

This possibility also follows from the fact that this transformation can be described as $c \to Tc$, where the diagonal matrix

$$T = \mathrm{diag}(\exp(\alpha \cdot \mathrm{i}), \exp(\alpha \cdot \mathrm{i}), \ldots, \exp(\alpha \cdot \mathrm{i}))$$

is, as one can easily check, unitary.

What the above empirical fact means is that it is *sufficient* to use only real-valued amplitudes – in the sense that whatever we can do with complex-valued amplitudes, we can do with real-valued amplitudes as well.

Important challenge. A natural question is: why? Why real-valued amplitudes are sufficient for quantum computing?

What we do in this paper. In this paper, we provide a simple and natural explanation for this empirical fact – thus showing that this is true not only for all known quantum algorithms: for any future quantum algorithm, it is also sufficient to use real-valued amplitudes.

2 Our Explanation

Main idea: 1-qubit states. Suppose that at some point, a quantum algorithm uses a state

$$c_0|0\rangle + c_1|1\rangle,$$

in which c_0 and c_1 are non-real complex numbers $c_0 = a_0 + b_0 \cdot \mathrm{i}$ and $c_0 = a_1 + b_1 \cdot \mathrm{i}$, i.e., for which the state has the form

$$(a_0 + b_0 \cdot \mathrm{i})|0\rangle + (a_1 + b_1 \cdot \mathrm{i})|0\rangle.$$

Then, we can form a related state of the 2-qubit system, with an additional qubit:

- whose 0 state corresponds to real parts of the amplitudes and
- whose 1 state to the imaginary part:

$$a_0|00\rangle + b_0|01\rangle + a_1|10\rangle + b_1|11\rangle.$$

One can see that this transformation from the original 1-qubit state with complex coefficients to a real-valued 2-qubit state preserves the length of each vector and is, thus, generalized unitary.

2-qubit states. Similarly, we can transform a general 2-qubit state

$$(a_{00} + b_{00} \cdot \mathrm{i})|00\rangle + (a_{01} + b_{01} \cdot \mathrm{i})|01\rangle + (a_{10} + b_{10} \cdot \mathrm{i})|10\rangle + (a_{11} + b_{11} \cdot \mathrm{i})|11\rangle,$$

where a_{ij} and b_{ij} are real numbers, into the following real-valued state of a 3-qubit system with an additional auxiliary qubit:

$$a_{00}|000\rangle + b_{00}|001\rangle + a_{01}|010\rangle + b_{01}|011\rangle + a_{10}|100\rangle + b_{10}|101\rangle + a_{11}|110\rangle + b_{11}|111\rangle.$$

This transformation from the original 2-qubit state with complex coefficients to a real-valued 3-qubit state preserves the length of each vector and is, thus, generalized unitary.

General case. In general, we can transform an arbitrary n-qubit state

$$(a_{0\ldots00} + b_{0\ldots00} \cdot \mathrm{i})|0\ldots00\rangle + (a_{0\ldots01} + b_{0\ldots01} \cdot \mathrm{i})|0\ldots01\rangle + \ldots +$$

$$(a_{1\ldots11} + b_{1\ldots11} \cdot \mathrm{i})|1\ldots11\rangle$$

into the following real-valued state of an $(n+1)$-qubit system with an additional auxiliary qubit:

$$a_{0\ldots00}|0\ldots000\rangle + b_{0\ldots00}|0\ldots001\rangle + a_{0\ldots01}|0\ldots010\rangle + b_{0\ldots01}|0\ldots011\rangle + \ldots +$$

$$a_{1\ldots11}|1\ldots110\rangle + b_{1\ldots11}|1\ldots111\rangle.$$

This transformation from the original n-qubit state with complex coefficients to a real-valued $(n+1)$-qubit state preserves the length of each vector and is, thus, generalized unitary.

Since this transformation is generalized unitary, we can implement any quantum algorithm with the corresponding transformed states Ts_1, Ts_2, ... – i.e., we can indeed implement the original algorithm by using states with real-valued amplitudes only.

Acknowledgments

This work was supported in part by the National Science Foundation grants 1623190 (A Model of Change for Preparing a New Generation for Professional Practice in Computer Science) and HRD-1242122 (Cyber-ShARE Center of Excellence).

References

[1] R. Feynman, R. Leighton, and M. Sands, *The Feynman Lectures on Physics*, Addison Wesley, Boston, Massachusetts, 2005.

[2] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Leaning*, MIT Press, Cambridge, Massachusetts, 2016.

[3] M. A. Nielsen and I. L. Chuang, *Quantum Computation and Quantum Information*, Cambridge University Press, Cambridge, U.K., 2000.

[4] K. S. Thorne and R. D. Blandford, *Modern Classical Physics: Optics, Fluids, Plasmas, Elasticity, Relativity, and Statistical Physics*, Princeton University Press, Princeton, New Jersey, 2017.