Письма в ЭЧАЯ. 2009. Т. 6, № 7(156). С. 32–38

A MATHEMATICA PROGRAM FOR CONSTRUCTING QUANTUM CIRCUITS AND COMPUTING THEIR UNITARY MATRICES[1]

V. P. Gerdt [a, 2], *R. Kragler* [b, 3], *A. N. Prokopenya* [c, 4]

[a] Joint Institute for Nuclear Research, Dubna
[b] University of Applied Sciences, Weingarten, Germany
[c] Brest State Technical University, Brest, Belarus

In this paper we briefly describe a *Mathematica* program for simulation of quantum circuits and illustrate some of its facilities by simple examples. Unlike other *Mathematica*-based quantum simulators, our program provides a user-friendly graphical interface for generating quantum circuits and computing the circuit unitary matrices. In addition to standard linear algebra-based tools, our program implements special computer-algebra technique for constructing the multivariate polynomial system that, for a circuit composed from the Toffoli and Hadamard gates, uniquely defines the circuit matrix.

В статье дается краткое описание программы на языке системы *Mathematica* для моделирования квантовых схем. Работа программы демонстрируется на простых примерах. В отличие от других подобных программ на языке системы *Mathematica* предлагаемая программа имеет дружественный графический пользовательский интерфейс для построения квантовых схем и вычисления соответствующих им унитарных матриц. Кроме того, на основе методов компьютерной алгебры в программе реализована процедура построения полиномиальных систем уравнений со многими неизвестными, которая полностью определяет унитарную матрицу для схем, составленных из вентилей Тоффоли и Адамара.

PACS: 03.67.Ac, 01.30.Cc, 03.67.-a

INTRODUCTION

Of two models of quantum computation — the quantum Turing machine and the circuit model — whose equivalence was rigorously shown in [1] for the QP problems, the latter one turned out to be more useful for practical purposes such as the design and analysis of quantum computers, quantum information processing and quantum algorithms [2].

A quantum circuit can be understood as a device consisting of logical quantum gates that are arranged in the device according to steps in which the gates process qubits in time.

[1]The contribution of one of the authors (V. P. G.) was partially supported by grant 07-01-00660 from the Russian Foundation for Basic Research and by grant 5362.2006.2 from the Ministry of Education and Science of the Russian Federation.

[2]E-mail: gerdt@jinr.ru

[3]E-mail: kragler@hs-weingarten.de

[4]E-mail: prokopenya@brest.by

It is usually assumed that the time runs from the left to right-hand side of the diagram depicting the quantum circuit. The number of gates in a circuit characterizes its size as well as computational complexity of the quantum computational process that is implemented in the circuit.

Since, in spite of some exciting rumors that are spread by the Canadian company D-Wave (see the Web page `http://www.dwavesys.com/`), realistic quantum computers have not yet been built, it is worthwhile to simulate quantum computation on a classical computer, and there is quite a number of such simulators (see, for example, [3,4]).

We developed the first version of a *Mathematica* program [5] that provides, unlike other *Mathematica*-based simulators, such as [4], a user-friendly interface for assembling quantum circuits and for computing their unitary matrices. In doing so, our program, in addition to the straightforward computation of the circuit matrix by means of the *Mathematica* built-in linear algebra facilities, provides users with the special routine to generate a system of multivariate polynomials for a circuit constructed from the Toffoli and Hadamard gates. This system, whose coefficients are elements in the finite field F_2 (i.e., with values 0 or 1 only), is such that its number of common roots in F_2 defines the matrix elements of circuit matrix [6].

Since similar multivariate polynomial systems over F_2 that are of interest in cryptoanalysis [7] were recently effectively solved in [8] by means of Gröbner bases [9] for the case of 80 variables, it gives a real hope that this computer algebra method may be competitive with or even superior to the direct linear algebra methods. They always suffer from the exponential blow-up in size of the matrices required to simulate quantum circuits. In paper [10] some related algorithmic and implementation aspects of constructing Gröbner bases for the polynomial systems over F_2 are suggested.

Below we describe briefly our *Mathematica* program and illustrate its facilities by an example from [5] with some elucidation of the method used in the program for generation of the circuit polynomial equations.

1. INPUT OF A CIRCUIT AND ITS GRAPHICAL OUTPUT

A circuit may contain any fixed number of qubits. Our program draws them as a column of states in the form $|a_j\rangle$ $(j = 1, 2, \ldots)$, and the output states of qubits in the form $|b_j\rangle$ $(j = 1, 2, \ldots)$. It contains the built-in data base of gates which includes the following gates [2]:

- **one-qubit gates:** Hadamard, Pauli X, Pauli Y and Pauli Z, Phase S and the $\pi/8$ or T;
- **two-qubit gates:** Controlled-X (CNOT), Controlled-Z, Controlled-S, Controlled-T and Swap gate;
- **three-qubit gates:** Toffoli (CCNOT).

This set of basis gates can be easily extended by the user. The set of six one-qubit gates indicated above, together with the CNOT-gate, forms a universal set of gates [2]. Another universal set is formed by the Hadamard and Toffoli gates [11]. It means that any quantum computation can be decomposed in terms of the gates which are contained in any of the two universal sets.

A quantum circuit is represented in our program as a rectangular table whose rows correspond to qubits, whereas columns contain gates, and, thus, the number of columns depends on the number of quantum gates and their arrangement. In drawing the circuit, we

```
      ⎛1  1  1  1  1  1⎞
mat = ⎜1  1  1  1  1  1⎟ ; circuit[mat]
      ⎝1  1  1  1  1  1⎠
```

Fig. 1. Cell produced for matrix 3×6

```
      ⎛C  H  X  H  C  1⎞
mat = ⎜X  H  C  1  C  H⎟ ; circuit[mat]
      ⎝C  1  C  H  X  1⎠
```

Fig. 2. Specification of entries in the circuit matrix of Fig. 1

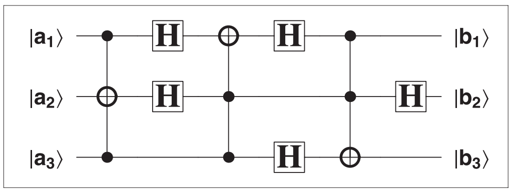

Fig. 3. The circuit generated

follow the rule: each column in the table can contain either one multi-qubit gate or only one-qubit gates and there are not any neighboring columns containing only one-qubit gates acting on different qubits.

In this section we consider an example of a three-qubit circuit that contains five Hadamard and three Toffoli gates. To input it into the program, one uses the command `matrixGenerating` which opens an interactive window and asks the user to enter the number of rows (qubits) and the number of columns for the circuit. When these numbers are 3 and 6, respectively, the program outputs the skeleton table shown in Fig. 1. The unit entries of the skeleton matrix can now be interactively upgraded to specify the circuit. In the specification shown in Fig. 2 symbols C and X set the control and target qubits, respectively, for the Toffoli gate, whereas symbol H sets the Hadamard gate. Then command `circuit[mat]` of Fig. 2 outputs the circuit with five Hadamard gates and three Toffoli gates as shown in Fig. 3.

2. COMPUTATION OF THE CIRCUIT MATRIX

The unitary 8×8 matrix (*circuit matrix*) determined by the circuit of Fig. 3 is computed by calling function `matrixU[mat]` with the same argument as used in Figs. 1 and 2. The output of function `matrixU[mat]` is shown in Fig. 4.

In the given case the matrix is computed by means of the linear algebra routines built in *Mathematica*. Because of exponential size $2^n \times 2^n$ of the matrix for n input qubits, this straightforward linear algebra method cannot be directly applied on a typical modern computer to circuits whose number of qubits exceeds 15. For this reason we have written the special command `polynomials` for generating a system of multivariate (Boolean) polynomials over F_2 associated with the circuit and such that their number of common roots in F_2 uniquely defines the elements of circuit matrix. This association of polynomials with the quantum circuits built from the Hadamard and Toffoli gates was established in paper [6] by means of the quantum mechanical Feynman's sum-over-paths method specified to quantum circuits. For all that, polynomial Boolean variables in the system are *path variables* which are in one-to-one correspondence with the Hadamard gates entering in the circuit. The coefficients in polynomials are Boolean polynomials themselves in the parameters a_i, b_i $(i = 1, 2, \ldots, n)$ defining (classical) states of the input and output qubits.

$$\frac{1}{\sqrt{2}}\begin{pmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & -1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & -1 & 1 & 0 \end{pmatrix}$$

Fig. 4. The unitary matrix for circuit of Fig. 3

More preciously, the classical gate for the quantum Hadamard gate outputs the path variable $x \in \mathbb{F}_2$ [6] irrespective of the input. Its value determines one of the two possible paths of computation. Thereby, the classical Hadamard gate acts at qubit a as

$$a \mapsto x, \qquad a, x \in \mathbb{F}_2.$$

Classically, the Toffoli gate acts on the triple of qubits with control qubits a_1, a_2 and target qubit a_3 in the following way:

$$(a_1, a_2, a_3) \mapsto (a_1, a_2, a_3 \oplus\, a_1 a_2)\,,$$

where $\oplus$ denotes (Boolean) addition modulo 2.

In Feynman's sum-over-paths approach, action of a quantum circuit is given as a sum over all possible classical paths. A classical path is defined by a sequence of classical bit strings of the form $\mathbf{a} = \mathbf{s_1}, \mathbf{s_2}, \ldots, \mathbf{s_m} = \mathbf{b}$ obtained from action of the classical gates. Each set of values of the path variables x_i gives a sequence of classical bit strings which form an *admissible classical path.*

For the circuit of Fig. 3 the path variables and, thus, all admissible classical paths can be explicitly shown by invoking the function `circuitPol`. As shown in Fig. 5, this function depicts the circuit together with the path variables and the related classical bit strings.

The sequence of classical bit strings for this circuit is given by $\mathbf{a} = \{a_1, a_2, a_3\} = \mathbf{s_1}$, $\mathbf{s_2} = \{a_1, a_2 \oplus a_1 a_3, a_3\}$, $\mathbf{s_3} = \{x_1, x_2, a_3\}$, $\mathbf{s_4} = \{x_1 \oplus x_2 a_3, x_2, a_3\}$, $\mathbf{s_5} = \{x_3, x_2, x_4\}$, $\mathbf{s_6} = \{x_3, x_2, x_4 \oplus x_2 x_3\}$, $\mathbf{s_7} = \{x_3, x_5, x_4 \oplus x_2 x_3\} = \mathbf{b}$.

Each admissible classical path has a phase factor. The phase is determined in terms of the Hadamard gates applied [6] and is changed only if both input and output of the Hadamard gate are equal to 1. It yields the formula

$$\varphi(\mathbf{x}) = \sum_{\text{Hadamard gates}} \text{input} \bullet \text{output}, \tag{1}$$

where summation is done in $\mathbb{F}_2$. For all that Toffoli gates do not affect the phase.

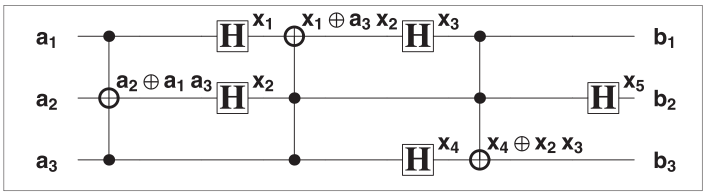

Fig. 5. Path variables for circuit of Fig. 3

In the example of Fig. 3 the phase of the path $\mathbf{x}$ is given by the expression (cf. Fig. 6)

$$\varphi(\mathbf{x}) = a_1x_1 \oplus a_2x_2 \oplus a_1a_3x_2 \oplus x_1x_3 \oplus a_3x_2x_3 \oplus a_3x_4 \oplus x_2x_5.$$

Feynman's sum-over-paths method derives the following representation for matrix elements of a circuit matrix U as sums over all allowed paths from the initial classical state $\mathbf{a}$ to the final classical state $\mathbf{b}$ [6]:

$$\langle \mathbf{b}|\, U\, |\mathbf{a}\rangle = \frac{1}{\sqrt{2^h}} \sum_{\mathbf{x}:\mathbf{b}(\mathbf{x})=\mathbf{b}} (-1)^{\varphi(\mathbf{x})}.$$

The sum is evaluated over h Hadamard gates which are contained in the circuit.

Let N_0 be the number of positive terms in the sum and N_1 the number of negative terms:

$$N_0 = |\ \{\mathbf{x} \mid \mathbf{b}(\mathbf{x}) = \mathbf{b} \wedge \varphi(\mathbf{x}) = 0\}\ |, \tag{2}$$

$$N_1 = |\ \{\mathbf{x} \mid \mathbf{b}(\mathbf{x}) = \mathbf{b} \wedge \varphi(\mathbf{x}) = 1\}\ |\,. \tag{3}$$

Thus, N_0 and N_1 count, respectively, the number of solutions in $\mathbb{F}_2^h$ for systems (2) and (3) of $n+1$ polynomials in h variables over $\mathbb{F}_2$. Thereby the matrix element of the circuit unitary matrix U may be written as the difference:

$$\langle \mathbf{b}|\, U\, |\mathbf{a}\rangle = \frac{1}{\sqrt{2^h}} \left(N_0 - N_1\right). \tag{4}$$

Our *Mathematica* package contains the function `polynomials` which constructs the set of polynomials over $\mathbb{F}_2$. This set follows from the bit string of the form $\mathbf{b}(\mathbf{x}) = \mathbf{b}$ that relates the output classical qubit states $\mathbf{b}$ with the path variables. Here $\mathbf{b}(\mathbf{x})$ denotes the last bit string $\mathbf{s_m}$ in the admissible path set which depends polynomially on the path variables $\mathbf{x} = \{x_1, \ldots, x_h\}$. The circuit unitary matrix is determined by the number of solutions for polynomial systems (2) and (3) in $\mathbb{F}_2^h$ with the input and output bit variables a_i, b_i taking values in $\mathbb{F}_2$. For this purpose the function `polynomials` of our *Mathematica* program provides output of polynomials in the form $\mathbf{b}(\mathbf{x}) \oplus \mathbf{b} = 0$ and adds the phase polynomial (1) to the system.

For the circuit of Fig. 3 the command `polynomials[mat]` outputs the system shown in Fig. 6. The first three polynomials in Fig. 6 are those generated by the output bit string relating the input and output qubit values for admissible paths (see Fig. 5) coded in terms of the variables $\{x_1, x_2, x_3, x_4, x_5\}$. The last polynomial is the phase polynomial defined by formula (1).

To count the total number of solutions for the polynomial systems as given in (2) and (3), when the variables take their values in $\mathbb{F}_2$ one can use the method of Gröbner or involutive

$$x_3 \oplus b_1$$
$$x_5 \oplus b_2$$
$$x_4 \oplus x_2\, x_3 \oplus b_3$$
$$a_1\, x_1 \oplus a_2\, x_2 \oplus a_1\, a_3\, x_2 \oplus x_1\, x_3 \oplus a_3\, x_2\, x_3 \oplus a_3\, x_4 \oplus x_2\, x_5$$

Fig. 6. Polynomial system for the circuit of Fig. 3

bases (see [10]). For the system of polynomials shown in Fig. 6, the lexicographical Gröbner basis for the ordering on the variables $x_5 \succ x_4 \succ x_3 \succ x_2 \succ x_1$ is given by

$$G: \begin{cases} g_1 = a_1x_1 \oplus b_1x_1 \oplus a_2x_2 \oplus a_1a_3x_2 \oplus b_2x_2 \oplus a_3b_3, \\ g_2 = x_3 \oplus b_1, \\ g_3 = b_1x_2 \oplus x_4 \oplus b_3, \\ g_3 = x_5 \oplus b_2. \end{cases} \tag{5}$$

In this case the Gröbner basis (5) can easily be computed with *Mathematica* [5]. Having the lexicographical Gröbner basis (5) computed, it is easy [5] to construct the 8×8 matrix for the circuit of Fig. 3 by formula (4). As a result, the matrix in Fig. 4 is obtained.

Generally, for an n-qubit circuit with h Hadamard gates (the number of Toffoli gates is not of importance here) the polynomial system associated with the circuit contains $n+1$ polynomials in h variables $\mathbf{x} = \{x_1, x_2, \ldots, x_h\}$ and $2n$ parameters $\mathbf{a} = \{a_1, a_2, \ldots, a_n\}$, $\mathbf{b} = \{b_1, b_2, \ldots, b_n\}$. These parameters determine the values of the input and output qubits, respectively. To apply formula (4) for computing the circuit matrix by the Gröbner bases method, one needs to take into account that both variables and parameters are elements in the finite field $\mathbb{F}_2$. For this reason, to convert the polynomial system into the Gröbner basis form using the *Mathematica* function `GroebnerBasis`, one should add to the polynomial systems binomials of the form $x_i^2 + x_1$ $(i = 1, \ldots, h)$. These extra polynomials may substantially increase the volume of computation needed for construction of Gröbner bases since one has to process a large number of S-polynomials [9].

In contrast, the algorithms and software described in [10] are oriented to compute the required Gröbner basis without explicit use of the binomials indicated. This is one of the reasons of higher computational efficiency of this software in comparison with *Mathematica* (see benchmarking in [10]).

REFERENCES

1. *Yao A.* Quantum Circuit Complexity // Proc. of the 34th IEEE Symp. on Foundations of Computer Science. Los Alamitos, CA, 1993. P. 352–360.

2. *Nielsen M., Chuang I.* Quantum Computation and Quantum Information. Cambridge Univ. Press, 2000.

3. *De Raedt H., Michielsen K.* Computational Methods for Simulating Quantum Computers. Handbook of Theor. and Comp. Nanotechnology / Eds.: M. Rieth and W. Schommers. Forschungszentrum Karlsruhe, 2006. V. 3; quant-ph/0406210.

4. *Julia-Diaz B., Burdis J. M., Tabakin F.* QDENSITY — A Mathematica Quantum Computer Simulation. Elsevier Sci., 2005; http://www.pitt.edu/∼tabakin/QDENSITY

5. *Gerdt V. P., Kragler R., Prokopenya A. N.* A Mathematica Package for Construction of Circuit Matrices in Quantum Computation // Comp. Algebra and Differential Equations. Acta Acad. Aboensis B. 2007. V. 67, No. 2. P. 28–38.

6. *Dawson C. M. et al.* Quantum Computing and Polynomial Equations over the Finite Field Z_2. quant-ph/0408129.

7. *Patarin J.* Hidden Fields Equations (HFE) and Isomorphisms of Polynomials (IP): Two New Families of Asymmetric Algorithms // EUROCRYPT'96. V. 1070 of LNCS 1070. Springer-Verlag, 1996. P. 33–48.

8. *Faugère J. C., Joux A.* Algebraic Cryptanalysis of Hidden Field Equations (HFE) Using Gröbner Bases. LNCS 2729. Springer-Verlag, 2003. P. 44–60.

9. Gröbner Bases and Applications / Eds. Buchberger B. and Winkler F. Cambridge Univ. Press, 1998.

10. *Gerdt V. P., Zinin M. V.* An Algorithmic Approach to Solving Polynomial Equations Associated with Quantum Circuits // This volume.

11. *Aharonov D.* A Simple Proof that Toffoli and Hadamard Gates Are Quantum Universal. quant-ph/0301040.