Workshop on Quantum Physics and Communication QPC 2007
October 15-19, 2007 Dubna, Russia

A *Mathematica* Program for Constructing Quantum Circuits and Computing Their Unitary Matrices

Vladimir P. Gerdt
Joint Institute for Nuclear Research
E-mail: gerdt@jinr.ru

Robert Kragler
University of Applied Sciences
E-mail: kragler@hs-weingarten.de

Alexander N. Prokopenya
Brest State Technical University, Belarus
E-mail: prokopenya@brest.by

Contents

Why are quantum computations important?

One reason for this is the potential ability of a quantum computer to do certain computational tasks much more efficiently than they can be done by any classical computer

Two the most famous examples of such calculations are Shor's algorithm for efficient factorization of large integers and Grover's algorithm of element search in an unsorted list.

Although real quantum computers have not yet been built, it is nevertheless worthwhile to simulate quantum computation on a classical computer. Among two equivalent models of quantum computation – *quantum Turing machine* and the *circuit model* – the last one is more convenient both for simulation and application.

What is the circuit model of computation?

The circuit model of computation was introduced first for a classical computer that can be considered as an electrical circuit made up of wires and logical gates. The wires are used to carry information within the circuit, whereas the logic gates perform manipulations of the information, converting it from one form to another. Note that inside a classical computer any information is encoded into a sequence of bits which are the elementary units of information. And any complex logic operation can be represented as an ordered sequence of some elementary logic gates which act on single *bits* or pairs of bits. Using special notation for these gates, one can easily visualize a circuit and clearly show the structure of computations. Thus, the circuit model turns out to be very convenient and realistic for many applications and is widely used in computer science.

A quantum circuit is constructed in a similar way and may be represented as a sequence of quantum logic gates acting on the *qubits* which are the elementary units of quantum information.

Qubit is an elementary unit of quantum information

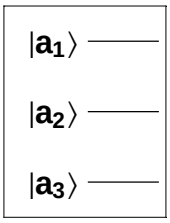
Quantum information is represented as a sequence of quantum bits or *qubits*. A qubit is a two-level quantum system that can be prepared, manipulated and measured in a controlled way. The state of a qubit is denoted as $|a\rangle$ corresponding to the standard Dirac notation for quantum mechanical states. Two possible states for a qubit are usually denoted by $|0\rangle$ and $|1\rangle$ which correspond to the states 0 and 1 for classical bits. But in contrast to classical bits, qubit as a quantum system may exist not only in one of the states $|0\rangle$ or $|1\rangle$ but also in the state $|a\rangle$ which is a *superposition* of these states

$$|a\rangle = \alpha\,|0\rangle + \beta\,|1\rangle,$$

where α and β are complex numbers constrained by the normalization condition $|\alpha|^2 + |\beta|^2 = 1$. Thus, the state of a qubit is represented by the vector $|a\rangle$ in the two-dimensional complex vector space, where the special states $|0\rangle$ and $|1\rangle$ form an orthonormal basis and are known as computational basis states.

6 of 45

A set of n qubits forms a quantum memory register, where the input data and any intermediate results of computations are held. It is shown on diagrams as a column of states of the form $|a_j\rangle\ (j = 1, 2, \ldots, n)$ from which quantum wires start.

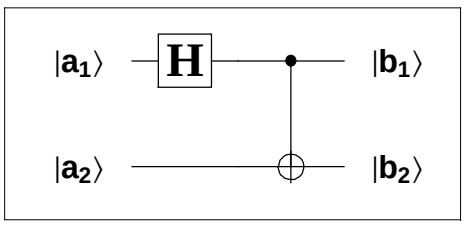


Although a quantum circuit does not contain any wires as such, the term "wires" is merely used to show evolution of qubits acted on by various quantum gates.

7 of 45

General structure of a quantum circuit

Generally, the structure of any quantum circuit can be readily understood from the following very simple quantum circuit containing two qubits and two quantum gates.



The circuit is to be read from left-to-right. It means that a column of two qubits $|a_1\rangle$ and $|a_2\rangle$ in the left-hand side of the diagram corresponds to the initial state of the quantum register. Then it is successively acted on by two quantum gates and its final state is shown on the right-hand side of the diagram as the column of qubits $|b_1\rangle$ and $|b_2\rangle$. Note that a quantum circuit containing more qubits and quantum gates can be built in a similar way.

The single-qubit quantum gates

As in the case of classical computation, there are two groups of elementary quantum gates which perform manipulation of quantum information. The first group consists of the single-qubit gates. Such gates have only one input and one output wires and are depicted by some capital letter placed into a square. Following Nielsen M. and Chuang I., we will use here only six non-trivial single-qubit quantum gates which are shown in the following table together with their matrix representation with respect to the computational basis states.

Hadamard gate	Pauli – X	Pauli – Y	Pauli – Z	Phase	$\pi/8$ – gate	Identity
H	X	Y	Z	S	T	
$\frac{1}{\sqrt{2}}\begin{pmatrix}1 & 1\\ 1 & -1\end{pmatrix}$	$\begin{pmatrix}0 & 1\\ 1 & 0\end{pmatrix}$	$\begin{pmatrix}0 & -i\\ i & 0\end{pmatrix}$	$\begin{pmatrix}1 & 0\\ 0 & -1\end{pmatrix}$	$\begin{pmatrix}1 & 0\\ 0 & i\end{pmatrix}$	$\begin{pmatrix}1 & 0\\ 0 & e^{i\frac{\pi}{4}}\end{pmatrix}$	$\begin{pmatrix}1 & 0\\ 0 & 1\end{pmatrix}$

Using this set of single-qubit matrices one can construct any operation on a single qubit. Note that the matrix of any single-qubit gate shows how the gate acts on the states of the computational basis, for example

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

$$H\,|1\rangle = \frac{|0\rangle - |1\rangle}{\sqrt{2}}$$

Two-qubit quantum gates

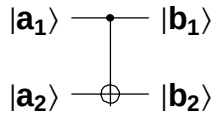
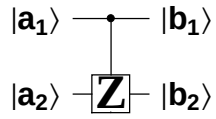
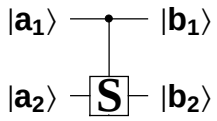
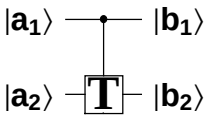
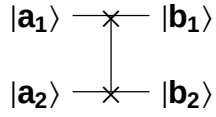
A two-qubit gate has two input and two output qubits. As the state of one qubit is represented by the vector in the two-dimensional complex vector space with basis vectors $|0\rangle$ and $|1\rangle$, the state of the system of two independent qubits may be determined as a *direct* (tensor) *product* of the two-dimensional vector spaces associated with each qubit. Such a vector space has four basis vectors

$|0\rangle_1 \otimes |0\rangle_2 \equiv |00\rangle;\ \ |0\rangle_1 \otimes |1\rangle_2 \equiv |01\rangle;\ \ |1\rangle_1 \otimes |0\rangle_2 \equiv |10\rangle;\ \ |1\rangle_1 \otimes |1\rangle_2 \equiv |11\rangle;$

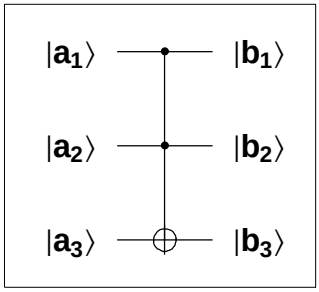
where the symbol $\otimes$ denotes a direct product of basis states associated with respect to the first $|0\rangle_1$, $|1\rangle_1$ and the second $|0\rangle_2$, $|1\rangle_2$ qubits. Thus, the system of two qubits is a four-dimensional space of states with computational basis states $|00\rangle$, $|01\rangle$, $|10\rangle$, $|11\rangle$.

Hence, any two-qubit quantum gate can be represented as a 4×4 matrix in terms of the computational basis states $|00\rangle$, $|01\rangle$, $|10\rangle$, $|11\rangle$.

Some examples of two-qubit gates together with their graphical and matrix representation are shown in the following table.

CNOT gate or Controlled – X		$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}$
Controlled – Z		$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 \end{pmatrix}$
Controlled – S		$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & i \end{pmatrix}$
Controlled – T		$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & e^{i\frac{\pi}{4}} \end{pmatrix}$
SWAP gate		$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$

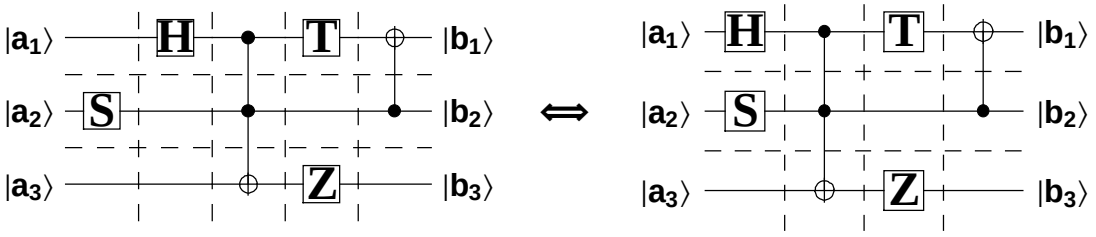
The set of six single-qubit gates shown in the table above together with the **CNOT**-gate is a universal set of gates. It means that any quantum computation can be decomposed in terms of this set of gates. In practice, however, there may be more specialized quantum gates which would enable us to construct more compact circuits for specific computations. We will use also the so-called **Toffoli gate** or *controlled-controlled-gate*. It can be considered as a generalization of the CNOT-gate and has three qubits for input and output : two of them are the control qubits and one is the target qubit. The state of the target qubit will be flipped only if both control qubits are in state $|1\rangle$.



13 of 45

Table representation of quantum circuits

Before describing an algorithm for generating quantum circuits let us note that any quantum circuit may be represented as a rectangular table. The number of rows in the table is equal to the number of qubits in the circuit whereas the number of columns depends on the number of quantum gates and their arrangement. Consider, for example, a quantum circuit containing three qubits and six quantum gates. Obviously, we can draw this circuit in different ways.



14 of 45

Drawing dashed lines, we can separate neighboring rows and columns in the table in such a way that each of its cell will contain some elementary gate (wires without any gates can be considered as identity quantum gates, i.e. gates making identical transformation of the corresponding qubits).

Obviously, we can shift the S-gate to the second column in the table without any disturbance of the circuit. Then the first column will contain only three identity gates and thus can be removed from the table. Hence, the two diagrams in the figure above represent the same circuit but the table on the right-hand side is smaller and simpler. In general, drawing the circuit, we will adhere the following convention:

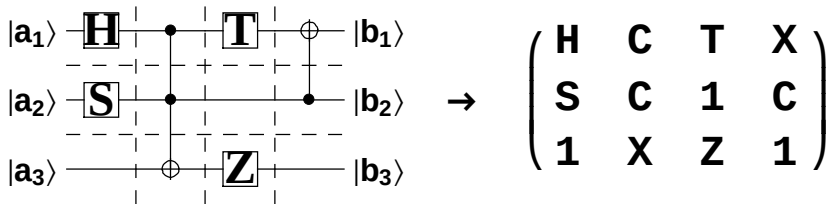
each column in the table can contain either one multi-qubit gate or only single-qubit gates, there are no neighboring columns which contain only single-qubit gates acting on different qubits.

Note, that according to this convention, we cannot shift the Pauli-Z gate to the last column because it would turn out to be in the same column together with multi-qubit CNOT gate.

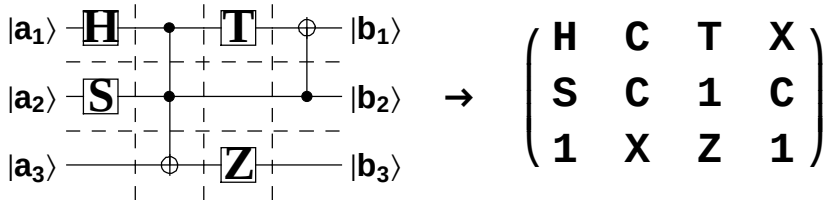
15 of 45

Matrix representation of the circuit

Thinking of any quantum circuit as a table of elementary quantum gates, we can define a matrix whose elements comprise some symbols, denoting the gates, and this matrix will contain all information on the circuit. An example of such a matrix is shown in the figure below, where ***C*** and ***X*** correspond to the control and target qubits in the ***Toffoli*** and ***CN***-gate, respectively (the symbols corresponding to the single-qubit gates are obvious). Obviously, this matrix completely determines the structure of this quantum circuit under consideration.



16 of 45



In our *Mathematica* package we just use such a matrix representation for quantum circuits and define the function circuit[mat_?MatrixQ] which generates the quantum circuit, corresponding to any given matrix **mat**. This is quite cumbersome and we will not describe it here. Of course, before evaluating this function, we should choose a set of symbols, denoting different quantum gates, and define the functions which determine the corresponding graphical objects.

17 of 45

Generating the quantum circuits

We define a function matrixGenerating which generates a cell, containing two *Mathematica* expressions: the matrix *mat* with given number of rows and columns (all of its elements are preset to 1) and the function circuit[mat]

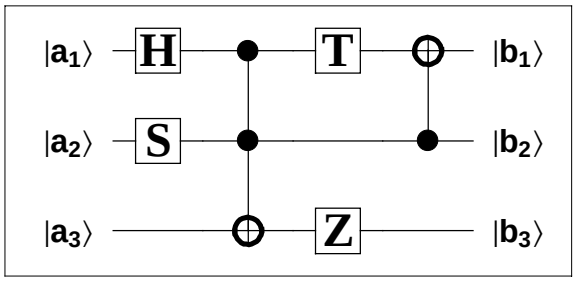
```
matrixGenerating
```

```
mat = {{1, 1, 1, 1}, {1, 1, 1, 1}, {1, 1, 1, 1}}; circuit[mat]
```

18 of 45

Now, in order to generate a quantum circuit, it is sufficient to replace "1" in the matrix *mat* by symbols "C", "H", "S","X", "Y", "Z", "T" corresponding to the constituent quantum gates, and evaluate the cell, for example,

```
mat = {{H, C, T, X}, {S, C, 1, C}, {1, X, Z, 1}}; circuit[mat]
```



It should be mentioned that using the functions *matrixGenerating* and *circuit* we can generate any quantum circuit.

19 of 45

Computing the circuit matrix

A system of n qubits has 2^n basis states. They are obtained as direct product of the basis states $|0\rangle$ and $|1\rangle$ associated with all $\boldsymbol{n}$ qubits. For $n = 3$, for example, they are

```
Table[StringReplacePart["|j⟩", ToString[j], {2, 2}] →
   IntegerDigits[j, 2, 3], {j, 0, 7}] // ColumnForm
```

```
|0⟩ → {0, 0, 0}
|1⟩ → {0, 0, 1}
|2⟩ → {0, 1, 0}
|3⟩ → {0, 1, 1}
|4⟩ → {1, 0, 0}
|5⟩ → {1, 0, 1}
|6⟩ → {1, 1, 0}
|7⟩ → {1, 1, 1}
```

20 of 45

Hence, the unitary matrix $\boldsymbol{U}$ defined by a quantum circuit with $\boldsymbol{n}$ qubits may be represented as a $2^n \times 2^n$ matrix with respect to these basis states.

As the circuit is read from left-to-right and we use the matrix ***mat*** to represent the circuit, then the matrix ***U*** can be written as the following product

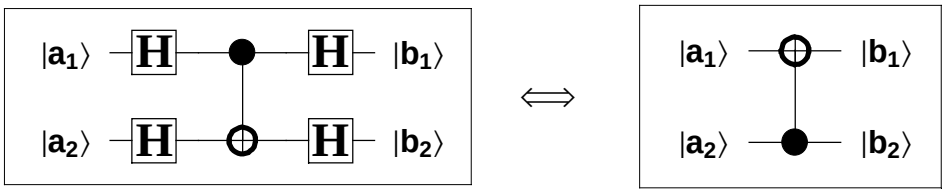
$$U = U_m . U_{m-1} \ldots U_1$$

where U_j **(*j* = 1, 2, ..., *m*)** is the 2×2 matrix defined by the corresponding quantum gate in the *j* th column of the matrix ***mat*** ; ***m*** denotes the number of columns.

The unitary matrix ***U*** is computed by the function matrixU[mat]

Example 1.

Using the functions *circuit* and *matrixU*, we can easily generate and analyze different quantum circuits. For example, one can show that the following two quantum circuits turn out to be equivalent.



Actually, we define matrices corresponding to these quantum circuits and compute their unitary matrices.

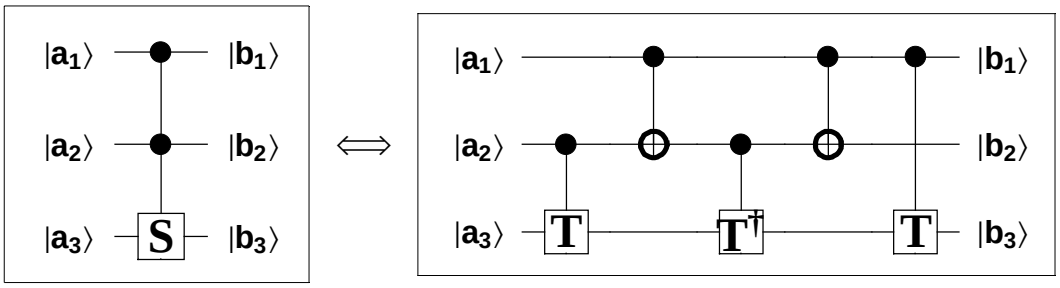
```
mat1 = {{H, C, H}, {H, X, H}}; mat2 = {{X}, {C}};
Map[MatrixForm, {matrixU[mat1], matrixU[mat2]}]
```

$$\left\{\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix}, \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix}\right\}$$

One can easily see that these two quantum circuits determine the same unitary matrices. Hence, they are equivalent.

Example 2.

Let us now show the equivalence of the following two quantum circuits.



24 of 45

We define their matrices and compute the corresponding unitary matrices of the circuits.

```
mat1 = {{C}, {C}, {S}}; mat2 = {{1, C, 1, C, C}, {C, X, C, X, 1}, {T, 1, T†, 1, T}};
Map[MatrixForm, {matrixU[mat1], matrixU[mat2]}]
```

$$\left\{\begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & i \end{pmatrix}, \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & i \end{pmatrix}\right\}$$

Now their equivalence is obvious.

25 of 45

It should be noted that similar representation may be used for any two unitary operators U_1 and U_2, satisfying the condition $U_2^2 = U_1$. In the case of $U_1 = Z$, $U_2 = S$, for example, we obtain

```
mat1 = {{C}, {C}, {Z}}; mat2 = {{1, C, 1, C, C}, {C, X, C, X, 1}, {S, 1, S†, 1, S}};
Map[MatrixForm, {matrixU[mat1], matrixU[mat2]}]
```

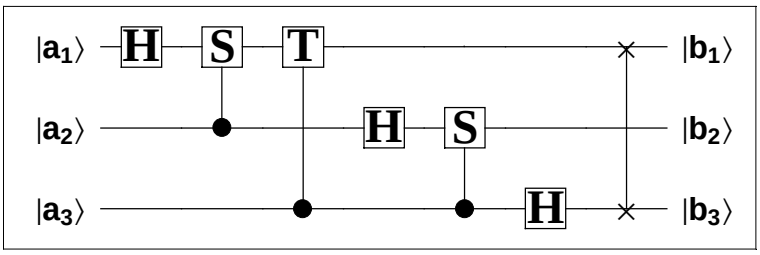
$$\left\{\begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 \end{pmatrix}, \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 \end{pmatrix}\right\}$$

As matrices of the circuits coincide, the quantum circuits themselves are equivalent.

26 of 45

Example 3.

Let us consider now a quantum circuit corresponding to the algorithm of quantum Fourier transformation (see: Nielsen M. and Chuang I. Quantum Computation and Quantum Information.)



First of all we generate a skeleton 3×7 matrix with the function ***matrixGenerating***

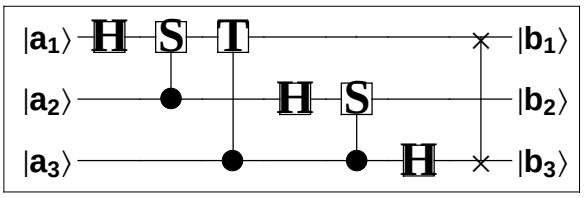
27 of 45

```
matrixGenerating
```

```
mat = ( 1 1 1 1 1 1 1
        1 1 1 1 1 1 1
        1 1 1 1 1 1 1 ) ; circuit[mat]
```

Then we replace some "1" in the matrix ***mat*** according to the diagram above and evaluate the function circuit[mat]. As a result, the corresponding quantum circuit is obtained as output.

```
mat = ( H S T 1 1 1 SW
        1 C 1 H S 1 1
        1 1 C 1 C H SW ) ; circuit[mat]
```



28 of 45

Finally, we evaluate the function matrixU[mat] and obtain the unitary matrix U defined by this circuit.

```
matrixU[mat] // MatrixForm
```

$$\begin{pmatrix}
\frac{1}{2\sqrt{2}} & \frac{1}{2\sqrt{2}} & \frac{1}{2\sqrt{2}} & \frac{1}{2\sqrt{2}} & \frac{1}{2\sqrt{2}} & \frac{1}{2\sqrt{2}} & \frac{1}{2\sqrt{2}} & \frac{1}{2\sqrt{2}} \\
\frac{1}{2\sqrt{2}} & \frac{e^{\frac{i\pi}{4}}}{2\sqrt{2}} & \frac{i}{2\sqrt{2}} & \frac{i e^{\frac{i\pi}{4}}}{2\sqrt{2}} & -\frac{1}{2\sqrt{2}} & -\frac{e^{\frac{i\pi}{4}}}{2\sqrt{2}} & -\frac{i}{2\sqrt{2}} & -\frac{i e^{\frac{i\pi}{4}}}{2\sqrt{2}} \\
\frac{1}{2\sqrt{2}} & \frac{i}{2\sqrt{2}} & -\frac{1}{2\sqrt{2}} & -\frac{i}{2\sqrt{2}} & \frac{1}{2\sqrt{2}} & \frac{i}{2\sqrt{2}} & -\frac{1}{2\sqrt{2}} & -\frac{i}{2\sqrt{2}} \\
\frac{1}{2\sqrt{2}} & \frac{i e^{\frac{i\pi}{4}}}{2\sqrt{2}} & -\frac{i}{2\sqrt{2}} & \frac{e^{\frac{i\pi}{4}}}{2\sqrt{2}} & -\frac{1}{2\sqrt{2}} & -\frac{i e^{\frac{i\pi}{4}}}{2\sqrt{2}} & \frac{i}{2\sqrt{2}} & -\frac{e^{\frac{i\pi}{4}}}{2\sqrt{2}} \\
\frac{1}{2\sqrt{2}} & -\frac{1}{2\sqrt{2}} & \frac{1}{2\sqrt{2}} & -\frac{1}{2\sqrt{2}} & \frac{1}{2\sqrt{2}} & -\frac{1}{2\sqrt{2}} & \frac{1}{2\sqrt{2}} & -\frac{1}{2\sqrt{2}} \\
\frac{1}{2\sqrt{2}} & -\frac{e^{\frac{i\pi}{4}}}{2\sqrt{2}} & \frac{i}{2\sqrt{2}} & -\frac{i e^{\frac{i\pi}{4}}}{2\sqrt{2}} & -\frac{1}{2\sqrt{2}} & \frac{e^{\frac{i\pi}{4}}}{2\sqrt{2}} & -\frac{i}{2\sqrt{2}} & \frac{i e^{\frac{i\pi}{4}}}{2\sqrt{2}} \\
\frac{1}{2\sqrt{2}} & -\frac{i}{2\sqrt{2}} & -\frac{1}{2\sqrt{2}} & \frac{i}{2\sqrt{2}} & \frac{1}{2\sqrt{2}} & -\frac{i}{2\sqrt{2}} & -\frac{1}{2\sqrt{2}} & \frac{i}{2\sqrt{2}} \\
\frac{1}{2\sqrt{2}} & -\frac{i e^{\frac{i\pi}{4}}}{2\sqrt{2}} & -\frac{i}{2\sqrt{2}} & -\frac{e^{\frac{i\pi}{4}}}{2\sqrt{2}} & -\frac{1}{2\sqrt{2}} & \frac{i e^{\frac{i\pi}{4}}}{2\sqrt{2}} & \frac{i}{2\sqrt{2}} & \frac{e^{\frac{i\pi}{4}}}{2\sqrt{2}}
\end{pmatrix}$$

29 of 45

Multiplying the matrix obtained by the corresponding adjoint matrix we prove that it is a unitary one.

```
%.Transpose[Conjugate[%]] // MatrixForm
```

$$\begin{pmatrix}
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1
\end{pmatrix}$$

30 of 45

Alternative method based on the polynomial equations

For quantum circuits containing only the Toffoli and Hadamard gates there exists an alternative method for constructing the circuit matrix. The method exploits the *algebra of multivariate polynomials* associated with the circuit.

To construct the system of multivariate polynomials one can apply *Feynman*'s quantum-mechanical sum-over-paths approach to a quantum circuit. This means that every path any quantum gate in the circuit under consideration acts as its classical counterpart. In doing so, the classical gate for the quantum Hadamard gate outputs the path variable $x \in \mathbb{F}_2$ irrespective of the input. Its value determines one of the two possible paths of computation. Thereby, the classical Hadamard gate acts as

$$a_i \mapsto x, \quad a_i, x \in \mathbb{F}_2$$

whereas the classical Toffoli gate acts as

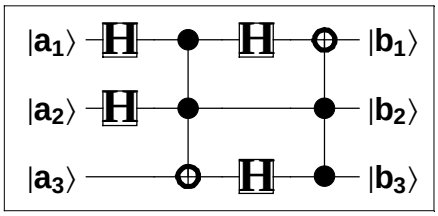
$$(a_1, a_2, a_3) \mapsto (a_1, a_2, a_3 \oplus a_1 a_2)$$

with $\oplus$ denoting addition modulo 2.

Example

Let us now consider example of a 3 qubit circuit built of Hadamard and Toffoli gates. Again we generate, first, the 3 × 4 matrix corresponding to the example by means of the functions *matrixGenerating* and *circuit*.

```
mat = {{H, C, H, X}, {H, C, 1, C}, {1, X, H, C}}; circuit[mat]
```



Its circuit matrix computed by the function matrixU[mat] is given by

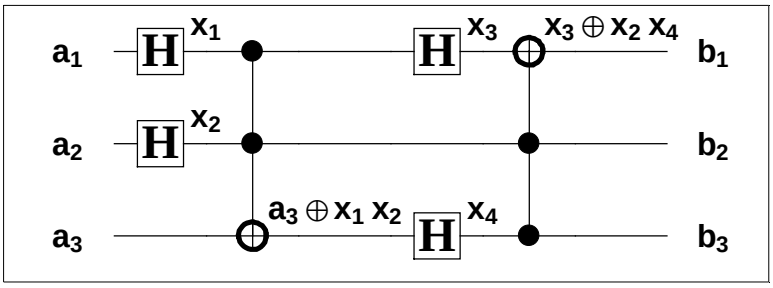
```
matrixU[mat] // MatrixForm
```

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

Turning back to *Feynman*'s sum-over-paths approach, a classical path is defined by a sequence of classical bit strings

$$a, a_1, a_2, \ldots, a_m = b$$

produced from action of the classical gates. Each set of values of the path variables x_i gives a sequence of classical bit strings which is called an admissible classical path. All path variables and, thus, all admissible classical paths for the circuit above are explicitly shown below.



34 of 45

Each admissible classical path is provided with a phase which is determined by the Hadamard gates applied. The phase is changed only when the input and output of the Hadamard gate are simultaneously equal to 1. Thereby, this gives the formula

$$\varphi(\mathbf{x}) = \sum_{\text{Hadamard gates}} input \cdot output \qquad (1)$$

with the sum evaluated in $\mathbb{F}_2$. As to Toffoli gates, they do not change the phase. For the circuit above the phase of the path $\boldsymbol{x}$ reads

$$\varphi(\boldsymbol{x}) = a_1 x_1 \oplus a_2 x_2 \oplus x_1 x_3 \oplus a_3 x_4 \oplus x_1 x_2 x_4$$

35 of 45

According to the Feynman's sum-over-paths method the matrix element of a quantum circuit is the sum over all the allowed paths from the classical state **a** to **b**

$$\langle \boldsymbol{b} \mid U_f \mid \boldsymbol{a} \rangle = \frac{1}{\sqrt{2^h}} \sum_{\boldsymbol{x}:\boldsymbol{b}(\boldsymbol{x})=\boldsymbol{b}} (-1)^{\varphi(\boldsymbol{x})}$$

where h is the number of Hadamard gates. Apparently, the terms in the sum have the same absolute value but may vary in sign.

Let $\boldsymbol{N_0}$ be the number of *positive* terms in the sum and $\boldsymbol{N_1}$ be the number of *negative* terms:

$$N_0 = |\{\mathbf{x} \mid \mathbf{b}(\mathbf{x}) = \mathbf{b} \text{ and } \varphi(\mathbf{x}) = 0\}|, \qquad (2)$$

$$N_1 = |\{\mathbf{x} \mid \mathbf{b}(\mathbf{x}) = \mathbf{b} \text{ and } \varphi(\mathbf{x}) = 1\}|. \qquad (3)$$

Hence, $\boldsymbol{N_0}$ and $\boldsymbol{N_1}$ count the number of solutions for the indicated systems of $n+1$ polynomials in h variables over $\mathbb{F}_2$. Then the matrix element may be written as the difference

$$\langle \mathbf{b} \mid U_f \mid \mathbf{a} \rangle = \frac{1}{\sqrt{2^h}} (N_0 - N_1). \qquad (4)$$

36 of 45

In our *Mathematica* package there is function polynomials[mat_?MatrixQ] which constructs and outputs the set of polynomials over $\mathbb{F}_2$. This function outputs the polynomials in the form $\boldsymbol{b}(\boldsymbol{x}) + \boldsymbol{b} = 0$ and adds the phase polynomial $\varphi(\boldsymbol{x})$ to the system. For the circuit above function polynomials[mat] gives

```
polynomials[mat] /. (0 ⊕ a_) → a /.
                x_ (y_ ⊕ z_) → x y ⊕ x z /.
              x_ ⊕ (a_ ⊕ b_) → x ⊕ a ⊕ b /.
              (x_ ⊕ a_) ⊕ b_ → x ⊕ a ⊕ b /.
          (x_ ⊕ a_ ⊕ c_) ⊕ b__ → x ⊕ a ⊕ c ⊕ b // ColumnForm
```

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

The upper three polynomial here are those generated by the output bit string relating the input and output qubit values for admissible paths coded in terms of the variables $\{x_1, x_2, x_3, x_4\}$. The bottom polynomial is the phase polynomial.

Solving circuit polynomial system

To count the number of solutions in $\mathbb{F}_2$ for the polynomial systems (2) and (3) in order to apply formula (4) we rewrite them into the form

$$F_0 = \{b(x) + b,\ \phi(x)\}, \tag{5}$$

$$F_1 = \{b(x) + b,\ \phi(x) + 1\}. \tag{6}$$

Here F_0 denotes the output of the function polynomials[mat] in our *Mathematica* package. It is convenient to transform the system into the canonical Gröbner basis form. The Gröbner basis method is the most universal algorithmic tool for investigation and solving multivariate polynomial systems.

To compute N_0 and N_1 one can convert F_0 and F_1 into an appropriate triangular form providing elimination of the path variables $x_1, \ldots, x_h$. One of such triangular forms is the pure lexicographical Gröbner basis that can be computed by means of *Mathematica* with built-in module for computing polynomial Gröbner bases.

For the system of polynomials F_0 in (5) shown above the lexicographical Gröbner basis for the ordering on the variables $x_1 > x_2 > x_3 > x_4$ is given by

$$G_0 : \begin{cases} g_1 = a_1\, x_1 \oplus b_1\, x_1 \oplus a_2\, b_2 \oplus a_3\, b_3, \\ g_2 = x_2 \oplus b_2, \\ g_3 = x_3 \oplus b_1 \oplus b_2\, b_3, \\ g_4 = x_4 \oplus b_3. \end{cases} \tag{7}$$

The Gröbner basis (7) can easily be obtained with *Mathematica*. To do this it is sufficient to define the polynomial set (5) as *Mathematica* polynomial list by the command

```
F0 = {x3 + x2 * x4 + b1,  x2 + b2,  x4 + b3,
      a1 * x1 + a2 * x2 + x1 * x3 + a3 * x4 + x1 * x2 * x4};
```

39 of 45

and invoke the *Mathematica* function *GroebnerBasis* with the arguments specified as follows

```
GB0 = GroebnerBasis[F0, {x1, x2, x3, x4},
                    MonomialOrder → Lexicographic, Modulus → 2]
```

```
{b3 + x4, b1 + b2 b3 + x3, b2 + x2, a2 b2 + a3 b3 + a1 x1 + b1 x1}
```

40 of 45

Similarly, for the system of polynomials $\boldsymbol{F_1}$ in (6) the Gröbner basis is

$$G_1 : \begin{cases} g_1 = a_1 x_1 \oplus b_1 x_1 \oplus a_2 b_2 \oplus a_3 b_3 \oplus 1, \\ g_2 = x_2 \oplus b_2, \\ g_3 = x_3 \oplus b_1 \oplus b_2 b_3, \\ g_4 = x_4 \oplus b_3. \end{cases} \tag{8}$$

The lexicographical Gröbner bases (7) and (8) immediately yield the following conditions on the parameters providing existence of 2 solutions in $\mathbb{F}_2$:

$$G_0 : a_1 \oplus b_1 = a_2 b_2 \oplus a_3 b_3 = 0, \tag{9}$$

$$G_1 : a_1 \oplus b_1 = 0,\ a_2 b_2 \oplus a_3 b_3 = 1. \tag{10}$$

If conditions (9) are satisfied then, respectively, the polynomial system $\boldsymbol{G_1}$ (resp. $\boldsymbol{F_1}$) has no common roots, and, vise-versa, if conditions (10) are satisfied then $\boldsymbol{G_0}$ has no roots and $\boldsymbol{G_1}$ has two roots. In all other cases there is one root of $\boldsymbol{G_0}$ and one root of $\boldsymbol{G_1}$.

In that way, the 8×8 matrix for the circuit above is easily determined by formulae (4) where numbers $\boldsymbol{N_0}$ and $\boldsymbol{N_1}$ are defined from systems (7) and (8). As a result, the matrix given by function matrixU[mat] is obtained.

41 of 45

A n–qubit circuit with h –Hadamard gates the polynomial systems (5) and (6) contains $n + 1$ polynomials in h-variables $\boldsymbol{x} = \{x_1, x_2, \ldots, x_h\}$ and $2n$-parameters $\boldsymbol{a} = \{a_1, a_2, \ldots, a_n\}$, $\boldsymbol{b} = \{b_1, b_2, \ldots, b_n\}$. 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 $\mathbb{F}_2$. By this reason, generally, to increase efficiency of computation with the use of the *Mathematica* function *GroebnerBasis* one should add to each of the systems (5) and (6) the binomials of the form

$$x_i^2 + x_1 \quad (i = 1, \ldots, h). \tag{11}$$

and also take into account the restrictions

$$a_j^2 + a_j = 0,\ b_j^2 + b_j = 0 \quad (j = 1, \ldots, n). \tag{12}$$

Due to the last restrictions all the intermediate polynomials arising at the Gröbner basis construction by Buchberger's algorithm admit substantial simplification.

42 of 45

It turns out that if one uses another algorithmic approach for the construction of Gröbner bases called *involutive* (Gerdt, Blinkov'98), then it is possible to avoid handling extra polynomials (11). In doing so, one can work with variables directly as with elements in $\mathbb{F}_2$.

The first implementation in C++ of an involutive algorithm for computation of Gröbner basis over $\mathbb{F}_2$ with polynomial variables from $\mathbb{F}_2$ is described in another talk presented here. After proper optimization it is planned to incorporate that C++ code into the open source software GINV which is a C++ module of Python and oriented to compute Gröbner bases for polynomial ideals and modules by involutive methods.

43 of 45

It should be noted that solving systems of multivariate polynomial equations variables over $\mathbb{F}_2$ whose variables take values in $\mathbb{F}_2$ is also of interest in ***cryptoanalysis***. One of the attacks of a HFE (Hidden Fields Equations) public key cryptosystem is based on the construction of a Gröbner basis for multivariate polynomial system over finite fields. In particular, quadratic n polynomials in n with $n \geq 80$, variables over field $\mathbb{F}_2$ was recommended as a public key, and $n = 80$ was suggested as the first challenge.

In the paper of

Faugere J.C. & Joux A. ("*Algebraic cryptanalysis of Hidden Field Equations (HFE) Using Gröbner Bases*". LNCS 2729, Springer-Verlag, 2003, pp. 44–60)

this challenge was broken by the Gröbner basis computation by means of the C program implementing the author's algorithm. This remarkable computational result gives hope that construction of the circuit matrices by means of polynomial systems may be computationally superior to the linear algebra based method (sect.4) for circuits with $n >> h$ where n and h are, as above, the numbers of qubits and Hadamard gates.

44 of 45

Conclusion

In the current talk we present the first version of our *Mathematica* package for simulation of quantum circuits. It provides a user-friendly graphical interface to specify a quantum circuit, to draw it, and to construct the corresponding unitary matrix for quantum computation defined by the circuit. The matrix is computed by means of the linear algebra tools built into *Mathematica*.

Besides, it is possible to construct the system of multivariate polynomials associated with the circuit containing only the Toffoli and Hadamard gates.

To provide a higher computational efficiency in construction of Groebner bases we developed a C++ program to be presented in the next talk.

45 of 45

Acknowledgements

The contribution of one of the authors (V.P.G.) was supported by the JINR first priority topic No. 05-6-1060-2005/2007 extended for the period 2008/2010 and 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.