

## Software emulator of quantum algorithms for sophisticated simulation on a conventional computer

Sergey V. Ulyanov <sup>1,2</sup>✉, Viktor S. Ulyanov <sup>3</sup>

<sup>1</sup> Dubna State University – Institute of System Analysis and Management, Dubna, 141980, Russian Federation

<sup>2</sup> Joint Institute for Nuclear Research – Meshcheryakov Laboratory of Information Technologies, Dubna, 141980, Russian Federation

<sup>3</sup> Moscow State University of Geodesy and Cartography (MIIGAiK), Moscow, 105064, Russian Federation

### For citation

Ulyanov, S.V., Ulyanov, V.S. (2024) ‘Software emulator of quantum algorithms for sophisticated simulation on a conventional computer’, *Software & Systems*, 37(1), pp. 5–17 (in Russ.). doi: 10.15827/0236-235X.142.005-017

### Article info

Received: 14.09.2023

After revision: 21.09.2023

Accepted: 05.10.2023

**Abstract.** A quantum software engineering platform includes quantum computing methods, a quantum algorithm theory and quantum programming. These areas develop according to a technological structure of nanotechnology development for hardware design of various configurations. In about 10 to 30 years we expect the appearing of an industrial quantum computer for real software engineering; this fact is due to overcoming a number of technological difficulties in implementing hardware, as well as the fundamental difficulty of eliminating decoherence physical phenomenon and correcting errors in quantum computers in near future. A key question in quantum computing is searching for quantum algorithms that potentially have a significant advantage and supremacy over classical algorithms for problems of practical interest. Therefore, currently, an approach is being developed to create quantum algorithm structures for quantum simulators with the possibility of effective implementation on classical architecture computers. This paper proposes an effective modelling method with information analysis of quantum search and decision-making algorithm structures in order to eliminate redundancy in practical implementation of a simulator on a classical structure computer. As an example, we demonstrate the method of modeling Grover's quantum search algorithm with stopping the search for a good solution based on the Shannon information entropy minimum principle. There are modeling examples to demonstrate the effectiveness of the developed approach in quantum software engineering and intelligent control robotics.

**Keywords:** quantum algorithm, quantum software engineering, quantum computing, quantum simulator, minimum of Shannon information entropy, termination criteria

**Introduction.** The history of quantum computing starts around the 1980s when during the First Conference on the Physics of Computation Richard Feynman showed that it is not effective to simulate a quantum system evolution on a classical computer. An effective simulation of quantum system has a run-time in polynomial size, i.e. the computational time is smaller than a polynomial function of the problem size. Therefore, relevant simulations of quantum computers will always be larger in size than polynomial time. This leads to super-polynomial time simulations of quantum algorithms; these kinds of simulations have a long runtime for large problems. By separating the problems in smaller parts, we can avoid long runtime. For example, simulating Shor's factoring algorithm on a classical computer takes super-polynomial time. The simulation of quantum algorithms is still constructive for parts of a larger problem and it gives us a basis for comparing experimental and theoretical results. The results from Shor's algorithm might be verified by multiple factors from an algorithm outcome and hence it is

simple to check the results from Shor's factoring algorithm implemented on a quantum computer. It might be more complicated to check the outputs from future algorithms.

However, it is possible to show that Shor's algorithm gives mathematically correct results. But how can we verify that implementing Shor's algorithm on a quantum computer coincides with its mathematical model? A simulation of a quantum algorithm on a classical computer allows comparing a quantum computer outcome with an output from a physically more stable classical computer. When developing quantum algorithms, it is interesting to check new algorithms on a classical computer. This study examines quantum algorithm simulation on a classical computer. The program code implemented on a classical computer will be a straight connection between the mathematical formulation of quantum mechanics and computational methods. A computational language includes such terms as a quantum state, a superposition and other quantum operators.

### Quantum algorithm general structure

The problem solved by a quantum algorithm (QA) can be stated in the symbolic form:

Input *Function*  $f: \{0, 1\}^n \rightarrow \{0, 1\}^m$ .

Problem *Find a certain property of function*  $f$ .

A given function  $f$  is a map of one logical state into another, QA estimates qualitative properties of function  $f$ . Fig. 1 demonstrates a general circuit description of QA.

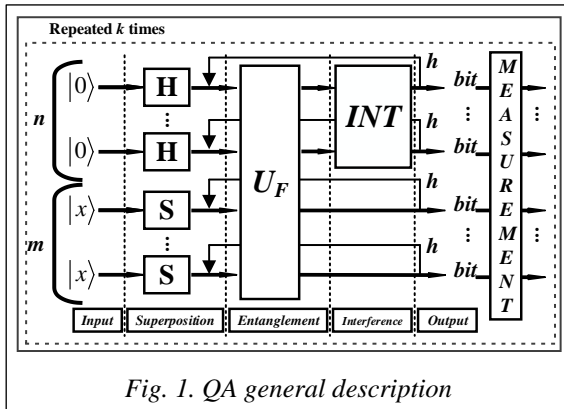


Fig. 1. QA general description

Three main quantum operators (as the superposition, the entanglement (quantum oracle) and the interference) are a background of QA structure design for implementing quantum massive parallel computing. Therefore, they include the matrix design form of three quantum operators: superposition (*Sup*), entanglement ( $U_F$ ) and interference (*Int*) (see, below Fig. 2).

The structure of a quantum algorithmic gate (QAG) in Fig. 1 in a general form can be defined as follows:

$$QAG = \left[ (Int \otimes^n I) \cdot U_F \right]^{h+1} \left[ {}^n H \otimes {}^m S \right], \quad (1)$$

where  $I$  is an identity operator; symbol  $\otimes$  denotes a tensor product;  $S$  is equal to  $I$  or  $H$  and depends on a problem description. The type of operator  $U_F$  physically describes the qualitative properties of function  $f$ .

Figure 2 shows QA steps including described qualitative peculiarities of function  $f$  and physical interpretation of applied quantum operators.

The quantum circuit (Fig. 2) is a high-level description of a method for composing smaller matrices using tensor and dot products in order to generate a finite QAG.

For example, Fig. 3 represents a general approach to Grover's QAG design [1].

The presented HW performs all functional steps of a Grover's QSA. A termination condition criterion is a minimum entropy-based method that is implemented in a digital part together with display output [2].

There are fast algorithms to simulate most of known QAs on classical computers [1] and in computational intelligence toolkit: 1) *Matrix-based approach*; 2) *Model representations* of quantum operators in fast QAs; 3) *Algorithmic-based approach* when matrix elements are calculated on demand; 4) *Problem-oriented approach*, where we succeeded to run Grover's algorithm with up to 64 and more qubits with Shannon entropy calculation (up to 1024 without a termination condition); 5) *Quantum algorithms with a reduced number of operators* (entanglement-free QA, and so on).

In this article we briefly describe main blocks in Fig. 3: a) unified operators; b) problem-oriented operators; c) benchmarks of QA simulation on classical computers; d) quantum control algorithms based on quantum fuzzy inference (QFI)

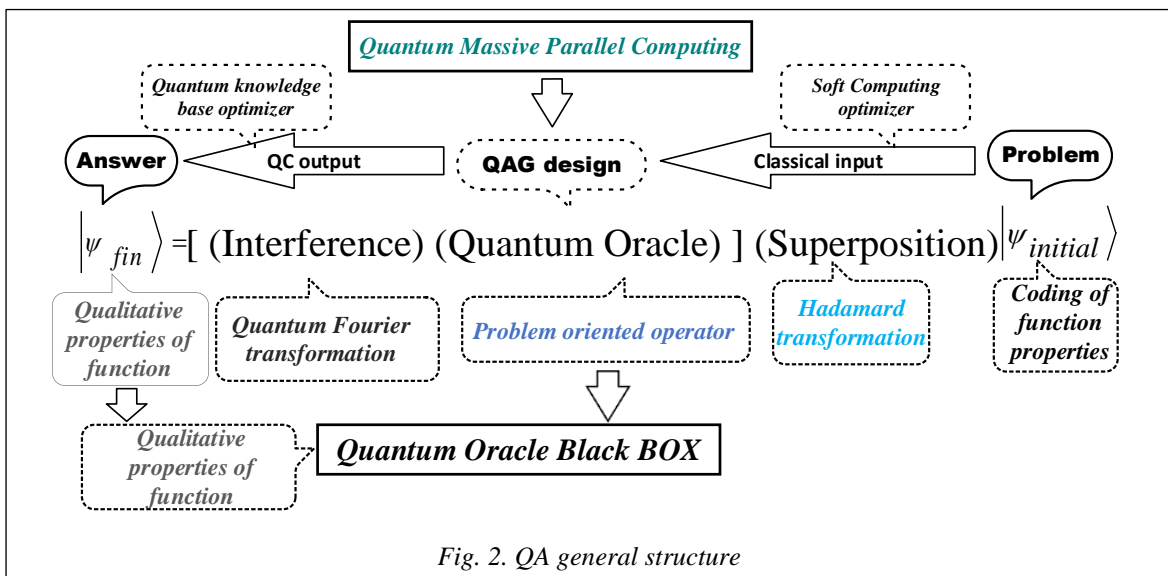
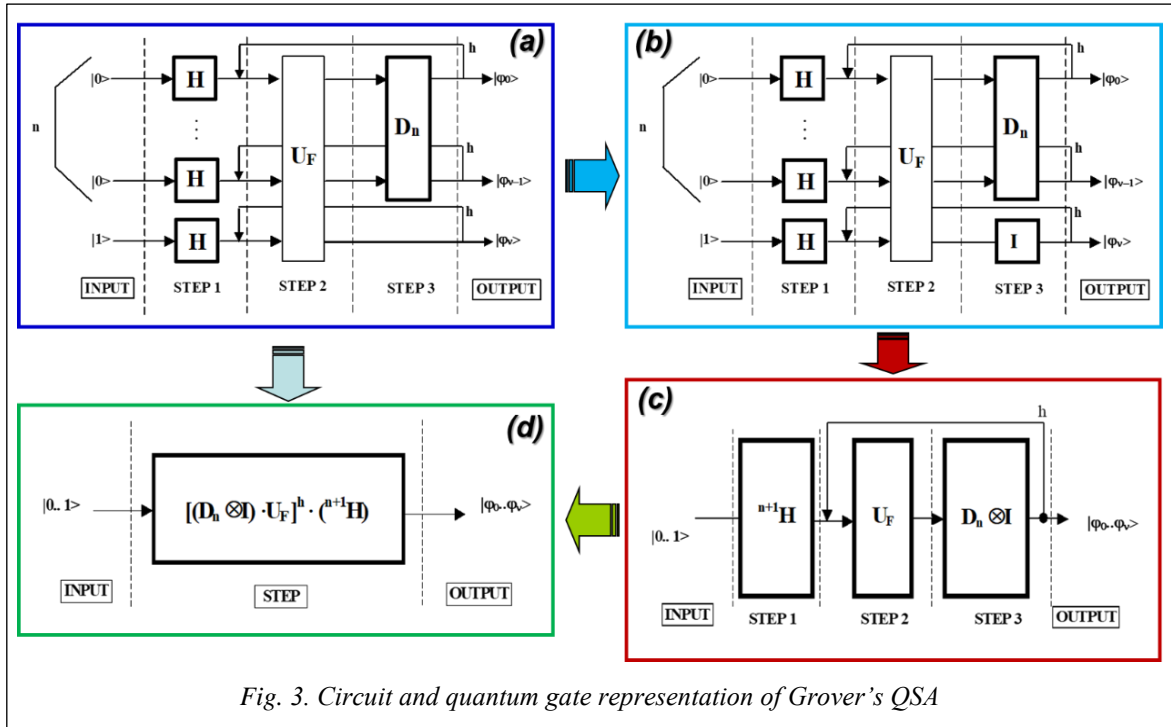


Fig. 2. QA general structure



and quantum genetic algorithm (QGA) as new QSA types.

**Description of quantum operators:  
SW smart toolkit support**

In terms of simulation, we consider the structure of quantum operators as a superposition, entanglement and interference. In this case a superposition and an interference have a more complicated structure and differ from an algorithm to an algorithm [3–5]. We also focus on considering entanglement operators, since they have a similar structure for all QAs and differ only by an analyzed function [6–8].

**QA superposition operators.** In general form, a superposition operator is a combination of tensor products of Hadamard  $H$  operators with identity operator  $I$ :

$$H = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}, I = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}.$$

The superposition operator of most QAs (see Fig. 1) can be expressed as:

$$Sp = \left( \bigotimes_{i=1}^n H \right) \otimes \left( \bigotimes_{i=1}^m S \right),$$

where  $n$  and  $m$  are the numbers of inputs and outputs respectively. Operator  $S$  may be Hadamard  $H$  operator or identity operator  $I$  depending on the algorithm. Table 1 presents the number of outputs  $m$ , as well as the structures of corresponding superposition and interference operators for different QAs.

Table 1

**Parameters of superposition and interference operators of main quantum algorithms**

Algorithm	Superposition	$m$	Interference
Deutsch's	$H \otimes I$	1	$H \otimes H$
Deutsch-Jozsa's	${}^n H \otimes H$	1	${}^n H \otimes I$
Grover's	${}^n H \otimes H$	1	$D_n \otimes I$
Simon's	${}^n H \otimes {}^n I$	$n$	${}^n H \otimes {}^n I$
Shor's	${}^n H \otimes {}^n I$	$n$	$QFT_n \otimes {}^n I$

Elements of the Walsh-Hadamard operator could be obtained as following:

$$[{}^n H]_{i,j} = \frac{(-1)^{ij}}{2^{n/2}} = \frac{1}{2^{n/2}} \begin{cases} 1, & \text{if } i * j \text{ is even,} \\ -1, & \text{if } i * j \text{ is odd,} \end{cases} \quad (2)$$

where  $i = 0, 1, \dots, 2^n$ ,  $j = 0, 1, \dots, 2^n$ . Its elements could be obtained by the simple replication according to the rule presented in Eq. (2). Thus, this approach greatly improves the performance of classical simulation of the Walsh-Hadamard operators.

**Interference operators of main QAs.** Interference operators must be selected individually for each algorithm according to the parameters presented in Table 1; for Grover's algorithm they can be written as a block matrix:

$$[Int^{Grover}]_{i,j} = D_n \otimes I = \left( \frac{1}{2^{n/2}} - {}^n I \right) \otimes I = \left( -1 + \frac{1}{2^{n/2}} \right) \otimes I \Big|_{i=j}, \left( \frac{1}{2^{n/2}} \right) \otimes I \Big|_{i \neq j} =$$

$$= \frac{1}{2^{n/2}} \begin{cases} -I, i = j, \\ I, i \neq j, \end{cases} \quad (3)$$

where  $i = 0, \dots, 2^n - 1, j = 0, \dots, 2^n - 1, D_n$  refers to a diffusion operator:  $[D_n]_{i,j} = \frac{(-1)^{I \wedge (i=j)}}{2^{n/2}}$  [9–12].

Note that with the increasing number of qubits, the gain coefficient becomes smaller.

The matrix dimension increases according to  $2^n$ , but each element can be extracted using Eq. (3) without allocating the entire operator matrix. In a certain form, the operator  $D_n$  (diffusion – inversion about average) in Grover’s algorithm is decomposed as follows:

$$D_n = \frac{1}{\sqrt{2^n}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}^{\otimes n} \cdot \begin{pmatrix} -1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & \ddots & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}^{\otimes n} \cdot \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}^{\otimes n}$$

and can be implemented with  $O(n) = O(\log(N))$  quantum gates. In terms of efficient computation, it means that the form in Eq. (3) is more preferable.

**Entanglement operators of main QAs.** Entanglement operators in a general form are a part of QA; the information about the function (being analyzed) is coded as an input-output relation. In the general approach to coding binary functions into corresponding entanglement gates, an arbitrary binary function is considered as:  $f : \{0,1\}^n \rightarrow \{0,1\}^m$ , such that  $f(x_0, \dots, x_{n-1}) = (y_0, \dots, y_{m-1})$ . First, irreversible function  $f$  transforms into reversible function  $F$  as following:  $F : \{0,1\}^{m+n} \rightarrow \{0,1\}^{m+n}$ , and

$$F(x_0, \dots, x_{n-1}, y_0, \dots, y_{m-1}) = (x_0, \dots, x_{n-1}, f(x_0, \dots, x_{n-1}) \oplus (y_0, \dots, y_{m-1})),$$

where  $\oplus$  denotes addition by module 2. This transformation creates a unitary quantum operator and performs a similar transformation. It is possible design an entanglement operator matrix using reversible function  $F$  according to the following rule:

$$[U_F]_{i^B, j^B} = 1 \text{ iff } F(j^B) = i^B, \\ i, j \in \begin{bmatrix} 0, \dots, 0; 1, \dots, 1; \\ n+m & n+m \end{bmatrix}.$$

$B$  denotes binary coding.

A diagonal block matrix of the form:

$$U_F = \begin{pmatrix} M_0 & & 0 \\ & \ddots & \\ 0 & & M_{2^n-1} \end{pmatrix} \text{ is actually a resulted entan-}$$

glement operator.

Each block  $M_i, i = 0, \dots, 2^n - 1$  can be obtained as follows:

$$M_i = \bigotimes_{k=0}^{m-1} \begin{cases} I, & \text{iff } F(i, k) = 0, \\ C, & \text{iff } F(i, k) = 1, \end{cases} \quad (4)$$

and consists of  $m$  tensor products of  $I$  or  $C$  operators, where  $C$  stays for NOT operator.

Note that an entanglement operator is a sparse matrix and according to this property (4) it is possible to accelerate entanglement operation simulation.

### Structure of QA simulation system in MatLab

Figure 4 shows a software system structure for QA simulation.

The software system is divided into two general sections. The first section involves common functions. The second section involves algorithm-specific functions for implementing particular algorithms.

**Common functions.** The common functions include:

- superposition building blocks,
- interference building blocks,
- bra-ket functions,
- measurement operators,
- entropy calculation operators,
- visualization functions,
- state visualization functions,
- operator visualization functions.

**Algorithm specific functions.** Algorithmic specific functions include:

- entanglement encoders,
- problem transformers,
- result interpreters,
- algorithm execution scripts,
- Deutsch algorithm execution script,
- Deutsch Jozsa algorithm execution script,
- Grover’s algorithm execution script,
- Shor’s algorithm execution script,
- quantum control algorithms as scripts.

**Visualization functions.** Visualization functions are functions that provide visualization display of quantum state vector amplitudes and quantum operator structure.

**Algorithmically-specific functions.** Algorithmically-specific functions provide a set of scripts for QA execution in a command line and tools for QA simulation including quantum control algorithms. The functions of section 2 prepare the appropriate operators of each algorithm and use common functions as operands.

**QA simulation by a command line.** The example of the Grover’s algorithm script is presented at the link <http://swsys.ru/uploaded/image/2024-1/UI-yanov.html>.

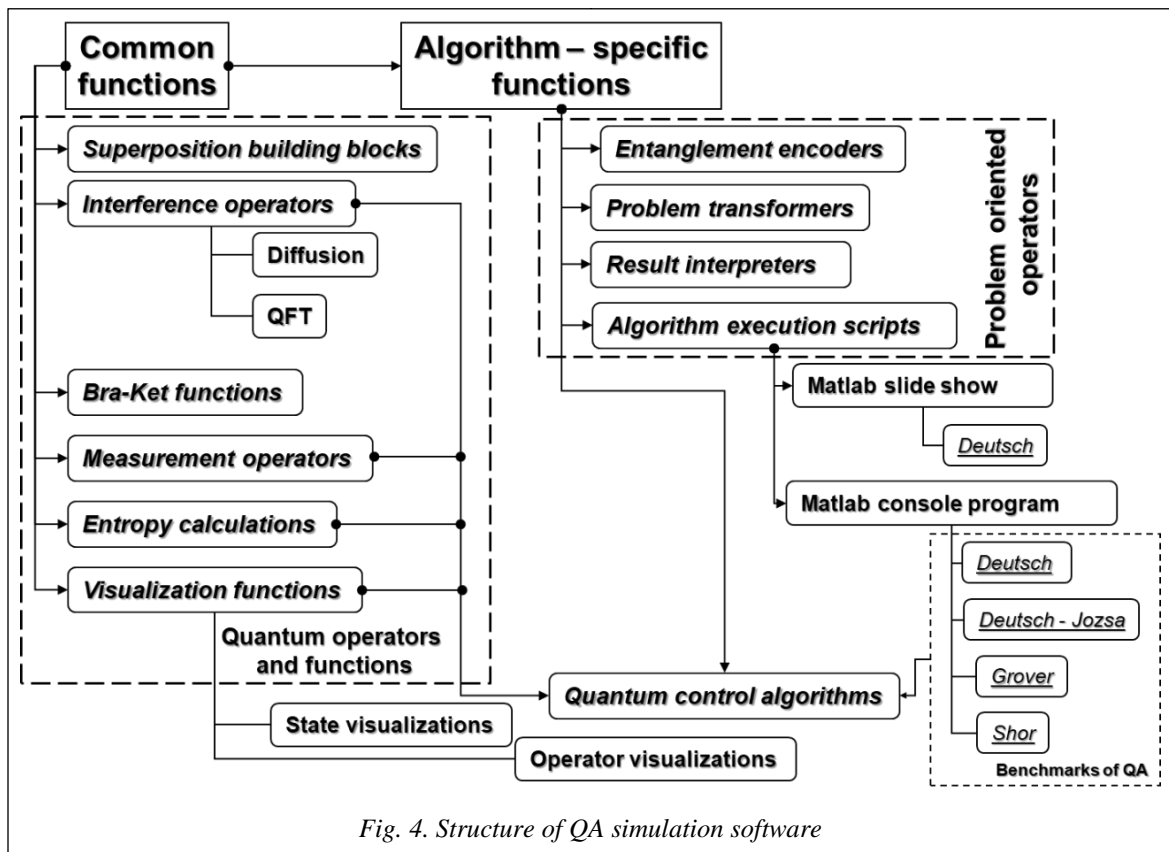


Fig. 4. Structure of QA simulation software

Other known QA can be formulated and executed by similar scripts, and using the corresponding equations presented earlier [9–11].

*Simulating QA as a dynamic system.* In order to simulate dynamic system behavior with quantum effects, it is possible to represent QA as a dynamic system in the form of a block diagram and then simulate its behavior in time. Figure 5 shows an example of a quantum circuit Simulink diagram to calculate the fidelity  $\langle a|a \rangle$  of the quantum state and to calculate the density matrix  $|a\rangle\langle a|$  of the quantum state. *Bra* and *ket* functions are from the common library. This example demonstrates using common functions to simulate QA dynamics.

In Fig. 6, the input is *ket* function. The *ket* function output is provided to the first input of the matrix multiplier and as the second input of the matrix multiplier. The input is also provided to the *bra* function. The *bra* function output is provided to the second input of the matrix multiplier and as the first input of the matrix multiplier. The multiplier output is an input state density matrix. The multiplier output is the input state fidelity.

Figure 7 shows Simulink structure of an arbitrary QA.

We can use such structure to simulate a number of quantum algorithms in Matlab/Simulink environment.

### Dedicated QA emulator

Developments in QA algorithmic representation are also applicable for designing QA software emulators. A key point is reducing multiple matrix operations to vector operations and the following replacement of multiplication operations. This may increase emulation performance, especially for algorithms that do not require complex number operations, and when a quantum state vector has a relatively simple structure (like Grover’s QSA).

The developed software can simulate 4 basic quantum algorithms, e.g. Deutsch-Jozsa, Shor’s, Simon’s and Grover’s. The system uses a unified user-friendly interface for all algorithms providing 3D visualization of state vector dynamics and quantum operators.

On the QA emulator launch window, you can choose to create a new QA model or continue modeling the existing one (<http://www.swsys.ru/uploaded/image/2024-1/13.jpg>).

If we choose creating a new model, then algorithm selection dialog starts. Here we can choose QA and its dimensions.

In fact, the system may operate with up to 50 qubits and more, but, it is better to limit the number of qubits to 10–11 due to visualization problems.

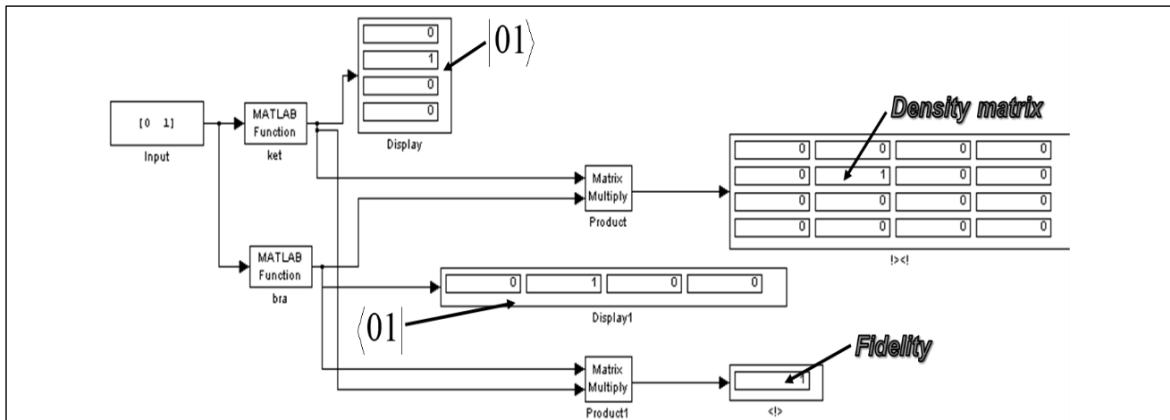


Fig. 5. Simulink diagram for simulating the arbitrary quantum algorithm

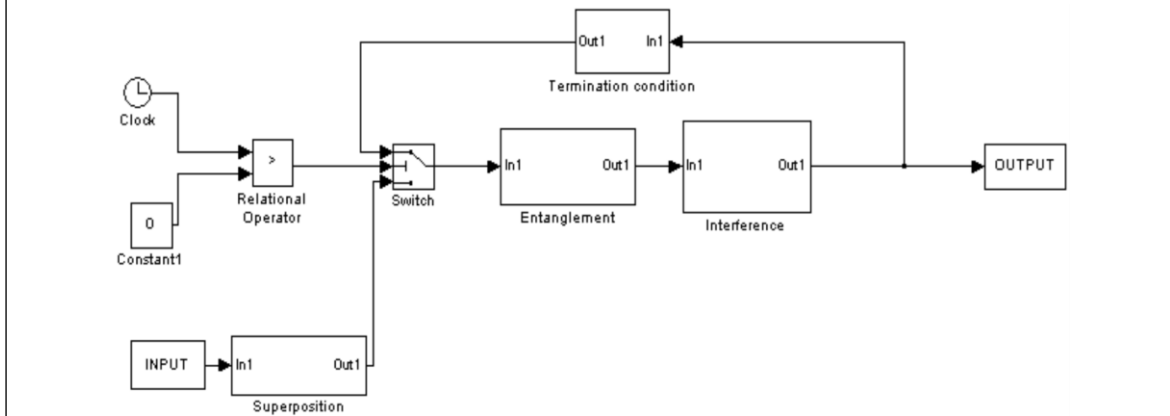


Fig. 6. Simulink diagram for simulating an arbitrary quantum algorithm

Once algorithm initial parameters are set, the system draws an initial state vector and selects an algorithm structure in the system’s main window (Fig. 7).

The main window (Fig. 7) contains all information of the emulated quantum algorithm and enables basic operations and analysis. The form menu has an access to involved quantum operators (Fig. 8), and it is possible to modify input functions.

QAs have reversible nature; therefore, it is possible to make forward and backward algorithm steps by clicking on arrows; currently applied algorithm step will be highlighted on the algorithm diagram.

The emulator menu consists of four components:

1. Item *File* provides basic operations, such as project save/load, and new model creation interface access.
2. Item *Model* provides an access to the input function editor.
3. Item *View* provides an access to operator matrix visualizers, including Superposition, En-

tanglement and Interference operators. It is also possible to get 3D preview of algorithm state dynamics (Fig. 9).

4. From *Help* menu there is an access to the program documentation.

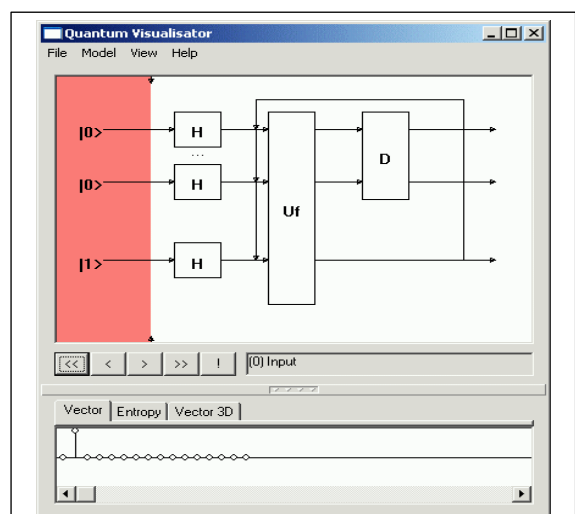


Fig. 7. Main window of QA emulator software (3 qubit Grover QSA)

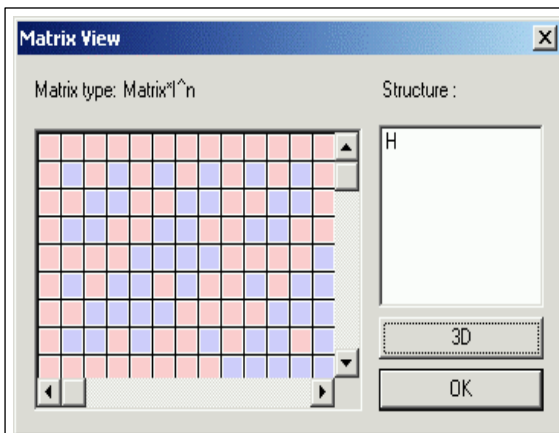


Fig. 8. Plain representation of superposition operator

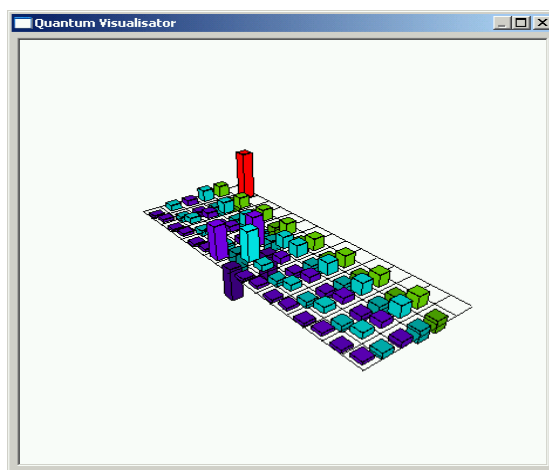


Fig. 9. 3D View of 3 qubit Grover's QSA state vector after two algorithm iterations

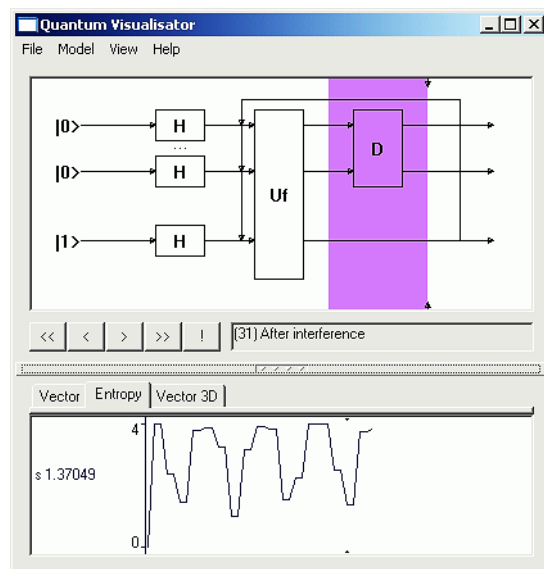


Fig. 10. Shannon entropy dynamics after 31 steps of Grover's QSA

Tabbed interface in the lower part of the window provides an access to Shannon entropy chart and to 3D representation of the state vector dynamics, as well as to usual plain representation of the QA state. The tabbed area size can be modified by dragging a divider. A click on the divider middle point hides the tabbed area from the screen.

Buttons in the middle part of the main window allow making steps by the currently parameterized QA. As it was mentioned above, the system can make forward and backward steps.

If there are enough algorithm steps, then a click on “!” button extracts an answer from the current state vector.

Depending on QA, an appropriate result interpretation routine appears.

The quantum operator visualizer displays the structure of involved quantum operator matrices in plain and in 3D representations.

If an operator consists of a tensor product of smaller operators, it is possible to access sub-blocks of tensor products. The 3D visualizer enables zoom and rotation of the charts.

The input function editor permits to automate the entanglement operator coding process as it was described earlier. For Grover's QSA it is possible to code functions that have more than one positive output.

Figure 10 shows the results of Grover QSA simulation with entropy criteria termination. Figures 11 and 12 demonstrate initial and final states of the developed software emulator with Deutsch-Jozsa, Simon's and Shor's QAs.

The QAG simulation result for the case of defining a constant or balanced function is shown in Figs 13, 14, correspondingly. The QAG entropy evaluation for both cases is also shown there. These results are used for QA stopping criteria below.

The coding sample of input functions and the corresponding 3D representation of entanglement operators of Deutsch-Jozsa, Simon and of Shor's algorithms are presented at the link <http://www.swsys.ru/uploaded/image/2024-1/14.jpg>.

Figure 15 demonstrates Shannon entropy behavior of simulated quantum algorithms after several algorithm iterations. It is clear that its minimum is reached on minimum uncertainty states, regardless a simulated algorithm.

QA simulation results of simulated QA are presented at the link <http://www.swsys.ru/uploaded/image/2024-1/15.jpg>, after a result interpreter.

We presented a design method and hardware implementation of a modular system for implementing Grover's QSA. We also developed hardware design of main quantum operators for

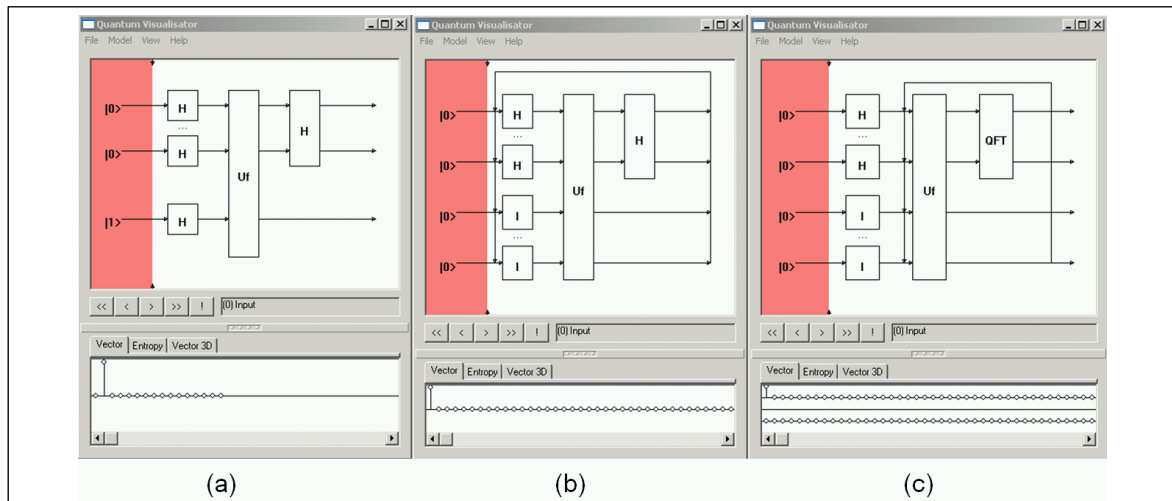


Fig. 11. Initial state of QA emulator for simulating: (a) Deutsch-Jozsa QA, (b) Simon's QA, (c) Shor's QA

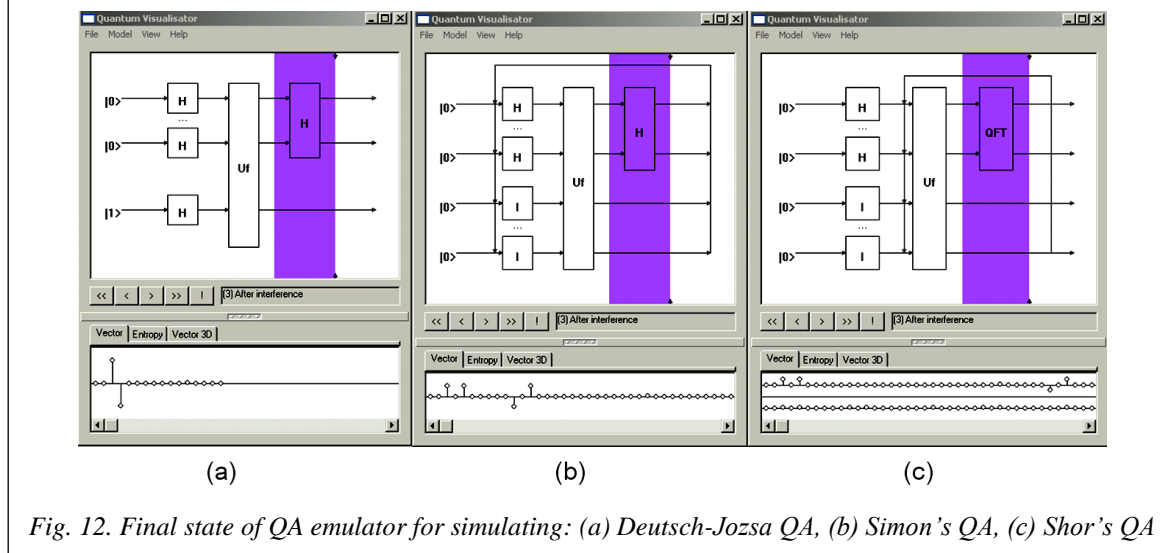


Fig. 12. Final state of QA emulator for simulating: (a) Deutsch-Jozsa QA, (b) Simon's QA, (c) Shor's QA

QA gates simulation on a classical computer. There is a demonstration of hardware implementation of an information criteria as minimum Shannon entropy for quantum algorithm termination.

These results are the background for efficient simulating quantum soft computing algorithms, robust fuzzy control based on quantum genetic (evolutionary) algorithms and quantum fuzzy neural networks (that can be implemented as modified Grover's QSA), AI-problems as quantum game's gate simulation approaches and quantum learning, quantum associative memory, quantum optimization, etc., on a classical computer [12–14].

### Comparing different QA simulation approaches

Tables 2–4 shows a comparison of the developed approaches to QA simulation. In case of

Grover's QSA, Table 2 shows the results from four simulation methods. It is clear that simulation results of each method are the same, however temporal complexity and the data base size may vary depending on an approach. The direct matrix-based approach is simpler, but the qubit number is limited to 12 qubits since operator matrices are allocated in PC memory. The second approach with algorithmic replacement of quantum gates allows increasing the analyzed function degree (number of qubits) up to 20 or more. The problem-oriented approach permits quantum gate applications operating directly with the state vector. This exponentially decreases the number of multiplications, and therefore allows running Grover's algorithm on a PC.

If we use this approach, it is possible to allocate a state vector in PC memory containing 25–26 qubits. An extreme version of Grover's QSA is an approach when the state vector is allocated as a sparse matrix, taking in consideration that with no



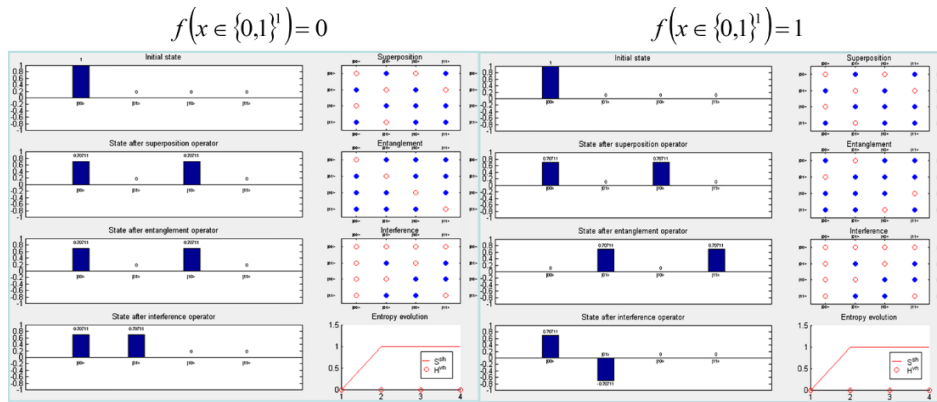


Fig. 13. Deutsch quantum algorithm simulation: 2d algorithm dynamics. Constant functions

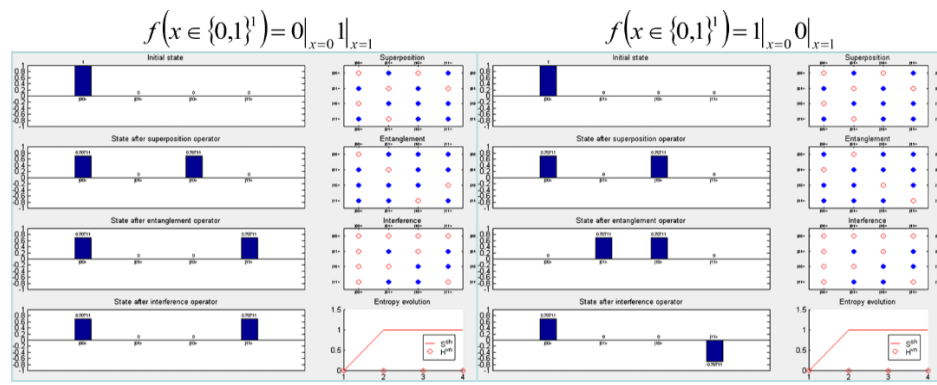


Fig. 14. Deutsch quantum algorithm simulation: 2d algorithm dynamics. Balanced functions

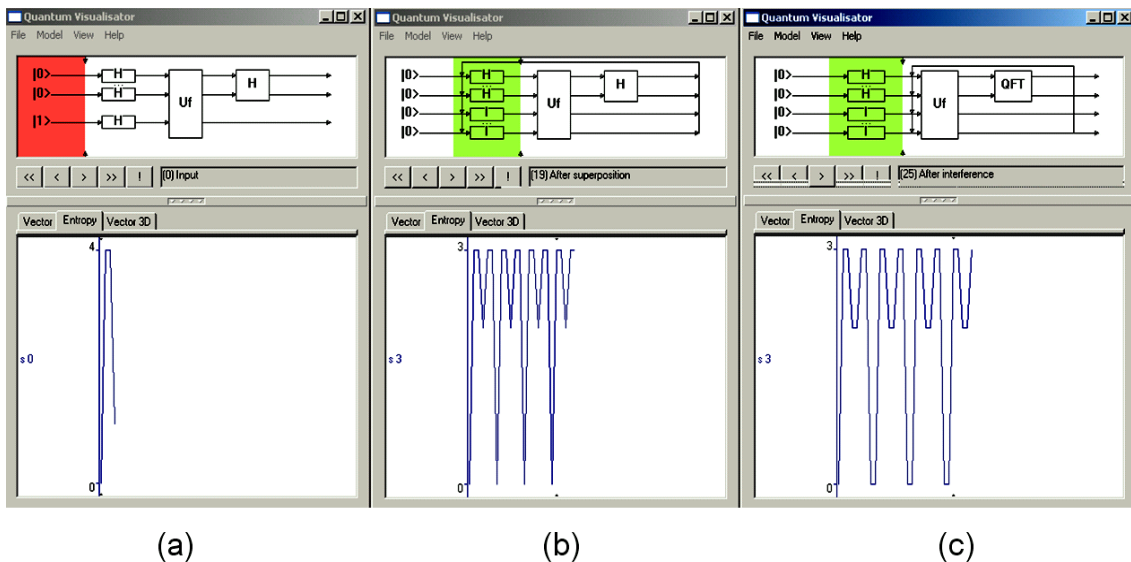


Fig. 15. Shannon entropy dynamics: (a) Deutsch-Jozsa QA, (b) Simon QA, (c) Shor QA

decoherence, most of the values of the probability amplitudes are equal, and thus there is no need to store all of the state vector but only the different parts, which are equal to the number of searched elements +1. Thus, excluding memory limitations,

we can simulate up to 1024 qubits or more with only limitation caused by a floating point number representation (with larger number of qubits, probability amplitudes after superposition approach to machine zero).

Table 2

Results from different approaches for Grover’s QSA simulation

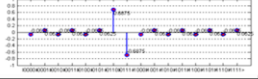
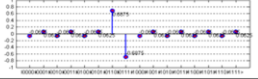
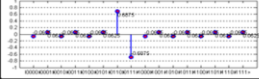
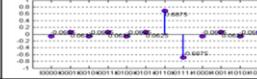
Matrix based approach	Algorithmic approach	Problem oriented approach (emulation, superposition is changed with replication, entanglement is realized as element perturbation, interference as a difference equation)	
		Entire state vector allocation	Sparse state vector allocation
1	2	3	4
<b>Simulation results (4 qubits, state with minimum of Shannon entropy, <math>x_0=0111</math>)</b>			
			
<b>Maximum order reached (number of qubits, simulation on PC with single CPU)</b>			
11+1 (Limited by spatial complexity, matrix approach requires allocation of quantum gate matrix in memory. Gate matrix has a size is $2^n \times 2^n$ )	19+1 (Limited by temporal complexity, we need $6 \times 10^6$ seconds for one iteration with 20 qubits, and $12 \times 10^6$ seconds for 21 qubits).	25 qubit (Limited by spatial complexity required for state vector allocation)	1023+1 without Shannon entropy calculation (Limited by floating point number representation see Figure 3.XX) 64+1 with Shannon entropy calculation (Limited by temporal complexity of calculations)
<b>Time required for one iteration with maximum function order (qubit number) on PIII 1GHz (sec)</b>			
$10^3$	$3 \times 10^5$	10	$\sim 0$ (q-bit number independent)
<b>Remarks</b>			
<ul style="list-style-type: none"> <li>•Possible introduction of excitation;</li> <li>•Generalized for all QA;</li> <li>•High spatial and temporal complexity.</li> </ul>	<ul style="list-style-type: none"> <li>•Relatively high function order;</li> <li>•Required specific R&amp;D for each algorithm;</li> <li>•Applicable for hardware realization;</li> <li>•No floating point operations (preparation and entropy calculation only);</li> <li>•Additional excitations may double temporal complexity.</li> </ul>	<ul style="list-style-type: none"> <li>•Can be used for state values estimation for high order functions;</li> <li>•Impossible introduction of excitations (excitations will cause exponential complexity and algorithm will vanish to 1<sup>st</sup> approach complexity) ;</li> <li>•Applicable only to few algorithms;</li> <li>•Simplest hardware realization.</li> </ul>	

Table 3

Results from different approaches for simulation of Deutsch-Jozsa’s QA

Matrix based approach	Algorithmic approach	Problem oriented approach (emulation, superposition is changed with replication, entanglement is realized as element perturbation, interference as a difference equation)	
		Sparse state vector allocation	
1	2	3	
<b>Maximum order reached (number of qubits, simulation on PC with single CPU)</b>			
11+1 (Limited by spatial complexity, matrix approach requires allocation of quantum gate matrix in memory. Gate matrix has a size is $2^n \times 2^n$ )	19+1 (Limited by temporal complexity)	> 1000 (Limited by floating point number representation)	
<b>Time required for one iteration with maximum function order (qubit number) on PIII 1GHz (sec)</b>			
$10^3$	$10^5$	$\sim 0$ (q-bit number independent)	
<b>Remarks</b>			
<ul style="list-style-type: none"> <li>•Possible introduction of excitation;</li> <li>•Generalized for all QA;</li> <li>•High spatial and temporal complexity.</li> </ul>	<ul style="list-style-type: none"> <li>•Relatively high function order;</li> <li>•Required specific R&amp;D for each algorithm;</li> <li>•Applicable for hardware realization;</li> <li>•No floating point operations (preparation and entropy calculation only);</li> <li>•Additional excitations may double temporal complexity.</li> </ul>	<ul style="list-style-type: none"> <li>•Can be used for state values estimation for high order functions;</li> <li>•Impossible introduction of excitations (excitations will cause exponential complexity and algorithm will vanish to 1<sup>st</sup> approach complexity) ;</li> <li>•Applicable only to few algorithms;</li> <li>•Simplest hardware realization.</li> </ul>	

In the case of Deutsch-Jozsa algorithm simulation, Table 3 shows three simulation approaches.

In this case, the direct matrix-based approach has the same limitations as Grover’s algorithm, and

Table 4

Results from different approaches for simulation of Simon and Shor’s QA

Matrix based approach	Algorithmic approach	Problem oriented approach (emulation, superposition is changed with replication, entanglement is realized as element perturbation, interference as a difference equation)	
		Entire state vector allocation	Sparse state vector allocation
1	2	3	4
<b>Maximum order reached (number of qubits, simulation on PC with single CPU)</b>			
5+1 (Limited by spatial complexity, matrix approach requires allocation of quantum gate matrix in memory. Gate matrix has a size is $2^{n+m}$ )	10+1 (Limited by temporal complexity)	Requires more R&D	Requires more R&D
<b>Time required for one iteration with maximum function order (qubit number) on PIII 1GHz (sec)</b>			
$10^2$	$10^5$	-	-
<b>Remarks</b>			
<ul style="list-style-type: none"> <li>•Possible introduction of excitation;</li> <li>•Generalized for all QA;</li> <li>•High spatial and temporal complexity.</li> </ul>	<ul style="list-style-type: none"> <li>•Relatively high function order;</li> <li>•Required specific R&amp;D for each algorithm;</li> <li>•Applicable for hardware realization;</li> <li>•No floating point operations (preparation and entropy calculation only);</li> <li>•Additional excitations may double temporal complexity.</li> </ul>	<ul style="list-style-type: none"> <li>•Practical application is impossible;</li> <li>•Can be used for state values estimation for high order functions;</li> <li>•Impossible introduction of excitations (excitations will cause exponential complexity and algorithm will vanish to 1<sup>st</sup> approach complexity) ;</li> <li>•Applicable only to few algorithms;</li> <li>•Simplest hardware realization.</li> </ul>	

a PC allows an order up to 11 qubits. The algorithmic approach allows up to 20 qubits or more qubits. The problem-oriented approach with compression gives the same result as Grover’s algorithm.

In case of Simon’s and Shor’s quantum algorithms, Table 4 shows a different algorithm structure. The matrix-based approach allows simulating up to 10 qubits, the algorithmic approach allows simulating up to 20 qubits or more.

Tables 2–4 summarizes the above approaches to QA simulation. The high-level structure of quantum algorithms can be represented as a combination of different superposition entanglement and interference operators. Then depending on algorithm, we can choose a corresponding model and an algorithm structure for simulation. Depending on a current problem, we can choose (if available) one of the simulation approaches and simulate quantum systems of different orders.

The analysis of quantum algorithm dynamics in terms of Shannon information entropy is presented at the link <http://swwsys.ru/uploaded/image/2024-1/Ulyanov1.html>.

An assessment of the of Shannon entropy behavior results for different qubit numbers (1–8) in Shor’s QA is shown in Fig. 16.

**Conclusions**

Efficient simulation of QAs on a classical computer with a large number of inputs is a difficult problem. For example, to directly operate 50

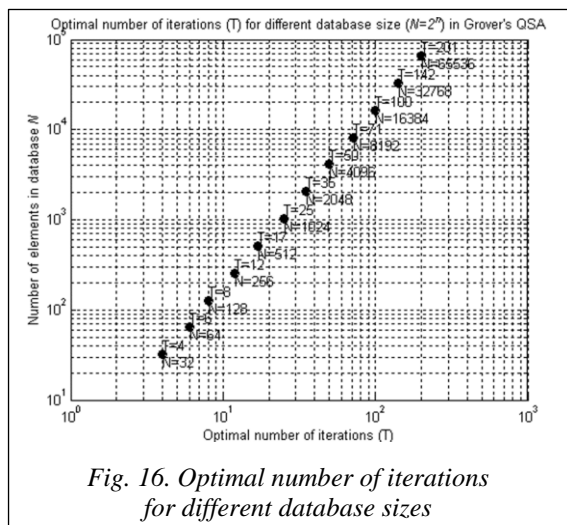


Fig. 16. Optimal number of iterations for different database sizes

qubits with a state vector only, it is necessary to have at least 128 TB of memory. This paper considers such important example as Grover’s QSA and demonstrates the possibility to override spatiotemporal complexity to perform QA efficient simulation on classical computers. We propose an effective modelling method with information analysis of the quantum search and decision-making algorithm structures in order to eliminate redundancy for practical implementation of the simulator on a classical structure computer. As an example, we show the method of modeling Grover’s quantum search algorithm with stopping the search for a good solution based on the Shannon information entropy minimum principle.

## References

1. Ivancova, O.V., Korenkov, V.V., Ulyanov, S.V. (2020) *Quantum Software Engineering Quantum Supremacy Modelling. Pt II: Quantum Search Algorithms Simulator – Computational Intelligence Toolkit*. Moscow, 344 p.
2. Ulyanov, S.V., Ulyanov, V.S. ‘Quantum algorithmic gate-based computing: Grover quantum search algorithm design in quantum software engineering’, *Software & Systems*, 2023, 36(4), pp. 523–538 (in Russ.). doi: 10.15827/0236-235X.142.523-538.
3. Nyman, P. (2022) ‘Simulation of quantum algorithms with a symbolic programming language’, *ArXiv*, art. 0705.3333v2, available at: <https://arxiv.org/abs/0705.3333> (accessed August 24, 2023).
4. Juliá-Díaz, B., Burdis, J.M., Tabakin, F. (2009) ‘QDENSITY – A Mathematica quantum computer simulation’, *CPC*, 180(3), art. 474. doi: 10.1016/j.cpc.2008.10.006.
5. Serrano, M.A., Perez-Castillo, R., Piattini, M. (2022) *Quantum Software Engineering*. Springer Verlag Publ., 321 p.
6. Martyn, J.M., Rossi, Z.M., Tan, A.K., Chuang, I.L. (2021) ‘A grand unification of quantum algorithms’, *ArXiv*, art. 2105.02859v5, available at: <https://arxiv.org/abs/2105.02859> (accessed September 20, 2023).
7. Bharti, K., Cervera-Lierta, A., Kyaw, T.H., Haug, T. et al. (2021) ‘Noisy intermediate-scale quantum (NISQ) algorithms’, *ArXiv*, art. 2101.08448v2, available at: <https://arxiv.org/abs/2101.08448> (accessed September 20, 2023).
8. Georgopoulos, K., Emary, C., Zuliani, P. (2021) ‘Quantum computer benchmarking via quantum algorithms’, *ArXiv*, art. 2112.09457v1, available at: <https://arxiv.org/pdf/2112.09457.pdf> (accessed September 20, 2023).
9. Galindo, A., Martin-Delgado, M.A. (2002) ‘Information and computation: Classical and quantum aspects’, *Rev. Mod. Phys.*, 74(2), pp. 347–377.
10. Abhijith, J., Adedoyin, A., Ambrosiano, J. et al. (2022) ‘Quantum algorithm implementations for beginners’, *ArXiv*, art. 1804.03719v3, available at: <https://arxiv.org/abs/1804.03719v3> (accessed September 20, 2023).
11. Childs, A.M., Coudron, M., Gilani, A.Sh. (2022) ‘Quantum algorithms and the power of forgetting’, *ArXiv*, art. 2211.12447v2, available at: <https://arxiv.org/pdf/2211.12447.pdf> (accessed August 24, 2023).
12. Voichick, F., Li, L., Rand, R., Hicks, M. (2022) ‘Qunity: A unified language for quantum and classical computing’, *ArXiv*, art. 2204.12384v1, available at: <https://arxiv.org/abs/2204.12384> (accessed August 24, 2023).
13. Xu, X., Benjamin, S., Sun, J., Yuan, X., Zhang, P. (2023) ‘A Herculean task: Classical simulation of quantum computers’, *ArXiv*, art. 2302.08880v1, available at: <https://arxiv.org/abs/2302.08880> (accessed August 24, 2023).
14. Cumming, R., Thomas, T. (2022) ‘Using a quantum computer to solve a real-world problem – what can be achieved today?’, *ArXiv*, art. 2211.13080v1, available at: <https://arxiv.org/abs/2211.13080v1> (accessed August 24, 2023).

УДК 512.6, 517.9, 519.6

doi: 10.15827/0236-235X.142.005-017

2024. Т. 37. № 1. С. 5–17

### Программный эмулятор квантовых алгоритмов для эффективного моделирования на персональном компьютере

С.В. Ульянов<sup>1,2</sup>✉, В.С. Ульянов<sup>3</sup>

<sup>1</sup> Государственный университет «Дубна» –  
Институт системного анализа и управления, г. Дубна, 141980, Россия

<sup>2</sup> Объединенный институт ядерных исследований –  
Лаборатория информационных технологий –  
им. М.Г. Мещерякова, г. Дубна, 141980, Россия

<sup>3</sup> Московский государственный университет геодезии  
и картографии (МИИГАиК), г. Москва, 105064, Россия

#### Ссылка для цитирования

Ульянов С.В., Ульянов В.С. Программный эмулятор квантовых алгоритмов для эффективного моделирования на персональном компьютере // Программные продукты и системы. 2024. Т. 37. № 1. С. 5–17. doi: 10.15827/0236-235X.142.005-017

#### Информация о статье

Группа специальностей ВАК: 1.2.1

Поступила в редакцию: 14.09.2023

После доработки: 21.09.2023

Принята к публикации: 05.10.2023

**Аннотация.** Платформой квантовой программной инженерии являются методы квантовых вычислений, теория квантовых алгоритмов и квантовое программирование. Развитие этих направлений зависит от технологической структуры разработки нанотехнологий для аппаратного оформления различных конфигураций. Промышленный квантовый компьютер для реальной программной инженерии ожидается примерно через 10–30 лет, и это связано с преодолением ряда технологических трудностей при реализации аппаратных средств, а также с фундаментальной трудностью устранения физического явления декогеренции и коррекции ошибок в квантовых компьютерах ближайшего будущего. Открытым ключевым вопросом в квантовых вычислениях является поиск квантовых алгоритмов, потенциально обладающих значительным преимуществом и превосходством над классическими алгоритмами для задач, представляющих практический интерес. Поэтому на современном этапе разрабатывается подход

к созданию структур квантовых алгоритмов для квантовых симуляторов с возможностью эффективной реализации на компьютерах с классической архитектурой. В данной статье предложен эффективный метод моделирования с информационным анализом структур квантовых алгоритмов поиска и принятия решений с целью устранения избыточности для практической реализации симулятора на компьютере с классической структурой. В качестве примера продемонстрирован метод моделирования алгоритма квантового поиска Гровера с остановкой поиска хорошего решения на основе принципа минимума информационной энтропии Шеннона. Приведены примеры моделирования принятия решений, демонстрирующие эффективность разработанного подхода в квантовой программной инженерии и интеллектуальной управляющей робототехнике.

**Ключевые слова:** программный эмулятор, квантовый алгоритм, квантовая программная инженерия, квантовые вычисления, квантовый симулятор, минимум информационной энтропии Шеннона, критерий останова

### Список литературы

1. Иванцова О.В., Коренков В.В., Ульянов С.В. Квантовая программная инженерия. Преимущества квантового моделирования. Ч. II: Программно-алгоритмическая поддержка квантового симулятора поисковых алгоритмов. М., 2020. 344 с. (англ.).
2. Ульянов С.В., Ульянов В.С. Квантовые вычисления на основе алгоритмических вентилях: проектирование алгоритма квантового поиска Гровера в квантовой программной инженерии // Программные продукты и системы. 2023. Т. 36. № 4. С. 523–538. doi: 10.15827/0236-235X.142.523-538.
3. Nyman P. Simulation of quantum algorithms with a symbolic programming language. ArXiv, 2022, art. 0705.3333v2. URL: <https://arxiv.org/abs/0705.3333> (дата обращения: 24.08.2023).
4. Juliá-Díaz B., Burdis J.M., Tabakin F. QDENSITY – A Mathematica quantum computer simulation. CPC, 2009, vol. 180, no. 3, art. 474. doi: 10.1016/j.cpc.2008.10.006.
5. Serrano M.A., Perez-Castillo R., Piattini M. Quantum Software Engineering. Springer Verlag Publ., 2022, 321 p.
6. Martyn J.M., Rossi Z.M., Tan A.K., Chuang I.L. A grand unification of quantum algorithms. ArXiv, 2021, art. 2105.02859v5. URL: <https://arxiv.org/abs/2105.02859> (дата обращения: 20.09.2023).
7. Bharti K., Cervera-Lierta A., Kyaw T.H., Haug T. et al. Noisy intermediate-scale quantum (NISQ) algorithms. ArXiv, art. 2101.08448v2. URL: <https://arxiv.org/abs/2101.08448> (дата обращения: 20.09.2023).
8. Georgopoulos K., Emary C., Zuliani P. Quantum computer benchmarking via quantum algorithms. ArXiv, 2021, art. 2112.09457v1. URL: <https://arxiv.org/pdf/2112.09457.pdf> (дата обращения: 20.09.2023).
9. Galindo A., Martin-Delgado M.A. Information and computation: Classical and quantum aspects. Rev. Mod. Phys., 2002, vol. 74, no. 2, pp. 347–377.
10. Abhijith J., Adedoyin A., Ambrosiano J. et al. Quantum algorithm implementations for beginners. ArXiv, 2022, art. 1804.03719v3. URL: <https://arxiv.org/abs/1804.03719v3> (дата обращения: 20.09.2023).
11. Childs A.M., Coudron M., Gilani A.Sh. Quantum algorithms and the power of forgetting. ArXiv, 2022, art. 2211.12447v2. URL: <https://arxiv.org/pdf/2211.12447.pdf> (дата обращения: 24.08.2023).
12. Voichick F., Li L., Rand R., Hicks M. Qunity: A unified language for quantum and classical computing. ArXiv, 2022, art. 2204.12384v1. URL: <https://arxiv.org/abs/2204.12384> (дата обращения: 24.08.2023).
13. Xu X., Benjamin S., Sun J., Yuan X., Zhang P. A Herculean task: Classical simulation of quantum computers. ArXiv, 2023, art. 2302.08880v1. URL: <https://arxiv.org/abs/2302.08880> (дата обращения: 24.08.2023).
14. Cumming R., Thomas T. Using a quantum computer to solve a real-world problem – what can be achieved today? ArXiv, 2022, art. 2211.13080v1. URL: <https://arxiv.org/abs/2211.13080v1> (дата обращения: 24.08.2023).

### Авторы

**Ульянов Сергей Викторович**<sup>1,2</sup>,

д.ф.-м.н., профессор,  
ulyanovsv46\_46@mail.ru

**Ульянов Виктор Сергеевич**<sup>3</sup>,

к.т.н., доцент,  
ulyanovik@mail.ru

### Authors

**Sergey V. Ulyanov**<sup>1,2</sup>,

Dr.Sc. (Physics and Mathematics), Professor,  
ulyanovsv46\_46@mail.ru

**Viktor S. Ulyanov**<sup>3</sup>,

Ph.D. (Robotics and Mechatronics),  
Associate Professor, ulyanovik@mail.ru

<sup>1</sup> Государственный университет «Дубна» –  
Институт системного анализа и управления,  
г. Дубна, 141980, Россия

<sup>2</sup> Объединенный институт ядерных исследований –  
Лаборатория информационных технологий  
им. М.Г. Мещерякова, г. Дубна, 141980, Россия

<sup>3</sup> Московский государственный университет  
геодезии и картографии (МИИГАиК),  
г. Москва, 105064, Россия

<sup>1</sup> Dubna State University –  
Institute of System Analysis and Management,  
Dubna, 141980, Russian Federation

<sup>2</sup> Joint Institute for Nuclear Research –  
Meshcheryakov Laboratory of Information Technologies,  
Dubna, 141980, Russian Federation

<sup>3</sup> Moscow State University of Geodesy  
and Cartography (MIIGAiK),  
Moscow, 105064, Russian Federation