



OPEN

A quantum circuit simulator and its applications on Sunway TaihuLight supercomputer

Zhimin Wang¹, Zhaoyun Chen^{2,3}, Shengbin Wang¹, Wendong Li¹, Yongjian Gu¹✉, Guoping Guo^{2,3}✉ & Zhiqiang Wei^{1,4}✉

Classical simulation of quantum computation is vital for verifying quantum devices and assessing quantum algorithms. We present a new quantum circuit simulator developed on the Sunway TaihuLight supercomputer. Compared with other simulators, the present one is distinguished in two aspects. First, our simulator is more versatile. The simulator consists of three mutually independent parts to compute the full, partial and single amplitudes of a quantum state with different methods. It has the function of emulating the effect of noise and support more kinds of quantum operations. Second, our simulator is of high efficiency. The simulator is designed in a two-level parallel structure to be implemented efficiently on the distributed many-core Sunway TaihuLight supercomputer. Random quantum circuits can be simulated with 40, 75 and 200 qubits on the full, partial and single amplitude, respectively. As illustrative applications of the simulator, we present a quantum fast Poisson solver and an algorithm for quantum arithmetic of evaluating transcendental functions. Our simulator is expected to have broader applications in developing quantum algorithms in various fields.

In recent years, tremendous technological progress has been made in the construction of quantum computers, especially with superconducting qubits^{1,2}. As these nascent quantum computers become competitive against classical computers in simulating general quantum circuits, an interesting race come to the climax. The quantum beings are eager to accomplish the first demonstration of quantum supremacy^{1,3}, while classical beings try to push back the classical simulation barrier as far as possible⁴⁻⁷.

During the race, many novel methods and programs are developed to simulate quantum circuits efficiently on conventional computers⁸⁻¹¹, including parallel platforms¹²⁻¹⁶, FPGA-based hardware^{17,18}, etc. In fact, classical simulation of quantum computation is vital both for the verification of quantum computers and for the assessment of the correctness and performance of new quantum algorithms. The fundamental task of such simulation is to calculate all or a certain number of amplitudes of quantum states produced by a quantum circuit.

However, it is extremely expensive to simulate quantum computation classically because of the curse of dimensionality, i.e., the memory and time requirements grow exponentially with the number of qubits. For instance, to accurately simulate a quantum system with 50 qubits, one needs a classical computer with slightly more than 16 Petabytes of memory (with double precision). Moreover, increasing the number of qubits by one requires a doubling of the amount of memory space. Performing such a large-scale computation requires one to take advantage of the state-of-the-art high-performance distributed computation.

In the present work, we develop a new quantum circuit simulator on the Sunway TaihuLight supercomputer. Albeit other simulators have been developed on supercomputers including Sunway TaihuLight^{12,14,16}, our simulator is designed to be a powerful tool for quantum algorithm research. The simulator consists of three mutually independent sub-programs to calculate the full, partial and single amplitudes of a quantum state with three completely different methods. Therefore, a wide range of number of qubits and circuit depths can be covered. This could provide choices when people execute quantum algorithms of different fields. In addition, it can emulate the effect of noise and support more quantum operations, such as the controlled and inverse operations on a group of gates, which are very useful in practical applications. On the other hand, the efficiency of the simulator is high. The algorithms of the simulator has a two-level parallel structure to fully take advantage of the Sunway system architecture. We can simulate random quantum circuits with 40, 75 and 200 qubits on the full, partial

¹College of Information Science and Engineering, Ocean University of China, Qingdao 266100, China. ²CAS Key Laboratory of Quantum Information, University of Science and Technology of China, Hefei 230026, China. ³Origin Quantum Computing Company Limited, Hefei 230026, China. ⁴High Performance Computing Center, Pilot National Laboratory for Marine Science and Technology (Qingdao), Qingdao 266100, China. ✉email: yjgu@ouc.edu.cn; gpguo@ustc.edu.cn; weizhiqiang@ouc.edu.cn

and single amplitude, respectively. With this simulator, we further develop quantum algorithms for solving the Poisson equations and for quantum arithmetic of evaluating transcendental functions.

Simulation techniques

The present quantum circuit simulator consists of three mutually independent sub-programs, referred to as three working modes of the simulator, i.e. full amplitude, partial amplitude and single amplitude mode. The fundamental methodologies for the three modes are completely different. They are, respectively, direct evolution of quantum state, circuit partition by decomposing controlled-Z gate¹⁰, and the complex undirected graphical model⁹. In addition, noisy one- and two-qubit gates are defined to emulate the effect of noise. A description of the instruction set of our simulator and an illustrative example of the input and output are given in the supplementary material.

Sunway TaihuLight supercomputer. Before proceeding to the details of the simulation techniques, we first give a brief introduction of the classical hardware. Our simulator is developed based on the Sunway TaihuLight at the National Supercomputer Center in Wuxi, China. The Sunway TaihuLight is so far the most powerful supercomputer in China. It can reach a peak performance of 125 PFlops, and had ranked the first in the TOP500 list for four times in the years of 2016 and 2017.

The supercomputer consists of 40,960 homegrown processors called SW26010. Each SW26010 processor contains four core-groups. Each core-group contains one management processing element (hereafter called as master core) with a memory space of 8 GB, and 64 computing processing elements (hereafter called as slave core) in an 8×8 array¹⁹. Within a core-group, the 64 slave cores can communicate with each other in a few cycles. In the present work, one core-group is set as a unique MPI process. When mentioning a computational node, it refers to one core-group, namely 1 master core plus 64 slave cores. The simulator is written by C++ language.

To take full advantage of the system architecture of Sunway TaihuLight, we implement algorithms of the three working modes in a two-level parallel way. More specifically, the entire simulation are first divided equally to the available nodes, which is the first level of parallel. In each node corresponding to a unique MPI process, the computing task is further assigned to the 64 slave cores equally, while the master core is responsible for the process control and I/O operation. This is the second level of parallel. The specific designs of algorithms are discussed in the subsequent sections.

Full amplitude mode. The full amplitude mode of the simulator is an instance of the so-called Schrodinger simulation. It is based on the direct evolution of quantum state through the product of unitary operations, contrasting to the linear combinations of unitary operations²⁰. All the information of the quantum state is precisely maintained and updated step-by-step throughout the simulation. The Schrodinger approach is straightforward, and it could provide a great speed in simulating low-width circuits. However, when processing many-qubits circuits, it requires a significant amount of RAM to store all amplitudes. In the present work, we use at most 16,384 computational nodes, roughly 10% of the computing resource of Sunway TaihuLight, and can simulate a quantum circuit with up to 40 qubits on this mode.

Now we use the single-qubit and controlled two-qubit operations as examples to illustrate the distributed implementation of Schrodinger simulation. It is well known that an n -qubit quantum state can be represented by Dirac notations and column vectors as follows,

$$\begin{aligned} |\varphi\rangle &= \sum_{i=0}^{2^n-1} \alpha_i |i\rangle = \sum_{i_{n-1} \cdots i_0 = \{0,1\}^n} \alpha_i |i_{n-1} \cdots i_k \cdots i_0\rangle \quad (\text{Dirac}). \\ &= (\alpha_0, \alpha_1, \cdots, \alpha_{2^n-1})^T \quad (\text{column vector}) \end{aligned} \quad (1)$$

where the decimal and binary index are related by $i = i_{n-1} \times 2^{n-1} + \cdots + i_k \times 2^k + \cdots + i_0 \times 2^0$. In practice, we store all the amplitudes α_i in the memory during the simulation and update them according to the action of unitary operations.

Let U^k represents a single-qubit gate acting on the k th qubit, namely i_k in Eq. (1). It can be easily verified that the amplitudes can be updated in the following way,

$$\begin{pmatrix} \alpha'_i \\ \alpha'_{i+2^k} \end{pmatrix} = U^k \begin{pmatrix} \alpha_i \\ \alpha_{i+2^k} \end{pmatrix} = \begin{pmatrix} a & b \\ c & d \end{pmatrix} \begin{pmatrix} \alpha_i \\ \alpha_{i+2^k} \end{pmatrix} = \begin{pmatrix} a\alpha_i + b\alpha_{i+2^k} \\ c\alpha_i + d\alpha_{i+2^k} \end{pmatrix}. \quad (2)$$

for all $(i)_{10} = (i_{n-1} \cdots i_k \cdots i_0)_2$ with $i_k = 0$.

Note that the amplitudes indexed by $i_k = 1$ are calculated when traversing the index of $i + 2^k$. As can be seen from the equation, for one action of U^k , all the 2^n amplitudes are changed. Thus, the one single-qubit operation corresponds to a computation scale of 2^n additions and multiplications. The controlled two-qubit operation can be implemented similarly. Let $CU^{q,k}$ represents a controlled two-qubit gate, where the qubit q (k) is the control (target) bit. That is to say, when i_q is zero, the gate $CU^{q,k}$ will do nothing; when i_q is 1, the gate performs the same transformation as Eq. (2). This can be formalized as

$$\begin{cases} \alpha'_i = \alpha_i, & \text{with } i_q = 0; \\ \begin{pmatrix} \alpha'_i \\ \alpha'_{i+2^k} \end{pmatrix} = \begin{pmatrix} a\alpha_i + b\alpha_{i+2^k} \\ c\alpha_i + d\alpha_{i+2^k} \end{pmatrix}, & \text{with } i_q = 1, i_k = 0 \end{cases} \quad (3)$$

for all $(i)_{10} = (i_{n-1} \cdots i_q \cdots i_k \cdots i_0)_2$ or $(i_{n-1} \cdots i_k \cdots i_q \cdots i_0)_2$

Here we remark that although the above single-qubit and controlled two-qubit operations are enough as they form a universal set for quantum computation²¹, our simulator could support more quantum gates and operations. They are very useful in the practical design of quantum circuits. Particularly, the simulator supports arbitrary single-qubit rotation gates, controlled operation on a group of gates, and inverse operation on a group of gates, etc. (see supplementary material for details).

The above two equations are of great importance because they lend the process of updating amplitudes to parallelization and distribution. That is, they update amplitudes via 2^n computations of $a\alpha_i + b\alpha_j$, as shown in Eq. (3), not by multiplying a full $2^n \times 2^n$ matrix on the column vector. Such equations can be implemented in a way of two-level parallel. Specially, all the amplitudes are divided equally to the nodes and stored in the corresponding master cores. Then the master core in each node calls the slave cores to update the amplitudes in parallel.

In summary, the program of this mode proceeds in the following three steps:

1st: Configure the computation nodes. Then every node parses the script to obtain a linked list recording instructions of the quantum circuit.

2nd: Assign all the amplitudes equally to the nodes. The amplitudes are initialized as zero in the master core of each node.

3rd: The master core traverses every node of the linked list in turn, and prepares the computing parameters, including the matrix coefficients of the gate, the number of amplitudes of each node, starting address of the target amplitude, etc. Then the master core assign the task of updating the amplitudes equally to the 64 slave cores. The slave cores get the required data using the address information according to Eqs. (2) or (3), and compute the new amplitude values, and then sent them back to the same position in the master core.

Partial amplitude mode. The partial amplitude mode use a hybrid algorithm to simulate a quantum circuit with more than 50 qubits but of limited depth. Generally, in this mode the original quantum circuit are divided into several sub-circuits with less qubits, which are then simulated independently using the same method as the full amplitude mode. With 16,384 computational nodes, we can simulate a quantum circuit with up to 75 qubits under this mode. Below is a brief introduction of the partition scheme of the circuit. More information can be found in our previous paper¹⁰.

The controlled-Z gates can be decomposed into the projection and single-qubit Z gates as follows,

$$CZ^{ij} = P_0^i \otimes I^j + P_1^i \otimes Z^j = \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix}^i \otimes I^j + \begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix}^i \otimes Z^j. \quad (4)$$

The superscripts represent that qubit i is the control qubit and qubit j the target qubit. On the left hand side of the equation, qubits i and j are entangled, while on the right hand side they are independent. Therefore, after decomposing the CZ gate, the quantum states of qubits i and j can evolve independently, and then be recombined to get the final state. This turns out to be a very useful method of reducing the memory requirements when simulating a quantum circuit with many qubits.

Now we take a quantum circuit with 8 qubits and 8 layers of depth as an example to illustrate the partition scheme. The circuit is shown in Fig. 1. The circuit is made up of two blocks, that is, the upper block with qubits from 0 to 3, and the lower one with the other qubits. The two blocks are entangled by the CZ gates in 7th and 8th layer. The entanglement between the two blocks can be dismissed by decomposing the two CZ gates in turn, as shown in Fig. 1. After the decomposition, the original circuit results in four circuits, whose upper and lower blocks are untangled. Then each of the four circuits can be divided into two sub-circuits with a half number of qubits, which can be simulated independently. Therefore, the task of simulating the original circuit with 8 qubits is converted to the simulation of 8 independent sub-circuits with 4 qubits. The number of amplitudes stored in the memory is reduced from 2^8 to 2^7 . Since the sub-circuits are simulated in a parallel way, the time span of the simulation is also reduced.

There are also restrictions on the partition scheme. The gates crossing the dividing line should be the controlled two-qubit gate, such as the CNOT and CZ gates, not the gate like SWAP. Furthermore, the number of sub-circuits grows exponentially with the number of decomposed CZ gates. For example, if there is one more CZ gate crossing the dividing line between qubits 3 and 4 in Fig. 1, the partition is not efficient. Therefore, this method is suitable for quantum circuits with low depth and large sampling number (the large sampling number is originate from the fact that all the sub-circuits are simulated on the full amplitude mode).

In summary, the program for the partial amplitude mode proceeds in the following four steps:

1st: Configure the computation nodes. Then every node parses the script to extract the gates. Judge whether the gates crossing the dividing line is the controlled two-qubit gates, and decompose it by doubling the circuit. The dividing line is always set to be in the middle of qubits.

2nd: Cut each of the final circuits into two sub-circuits along the dividing line. There should be 2^{c+1} sub-circuits generated, where c is the number of decomposed gates. Establish a linked list of quantum gates for each sub-circuit.

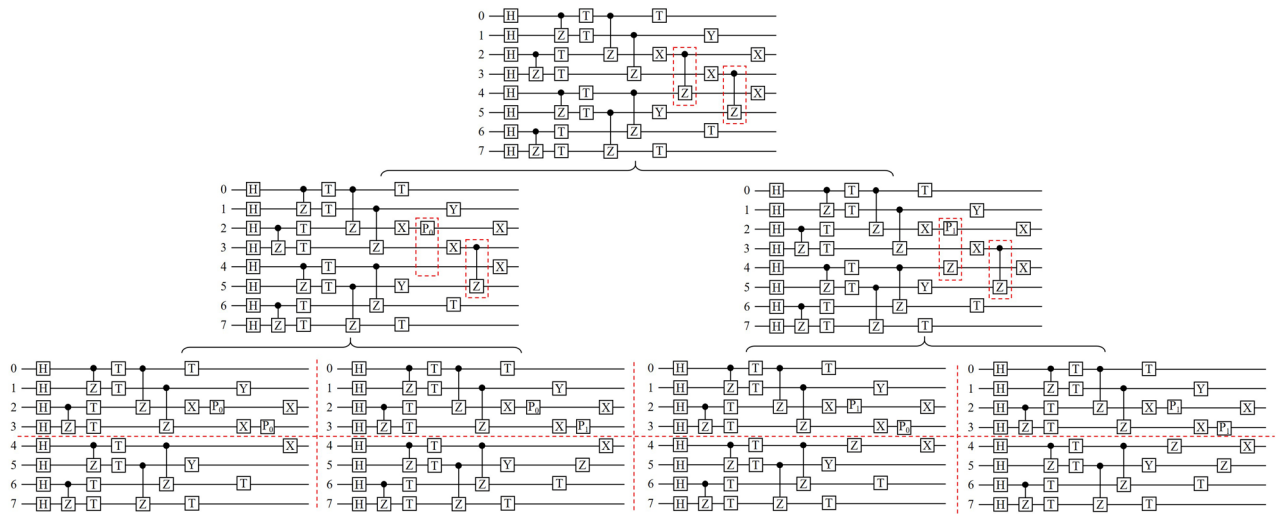


Figure 1. The partition scheme for a quantum circuit with 8 qubits and 8 depths¹⁰. The first line shows the original circuit. In the second line, the CZ gate in 7th depth is decomposed to projection and single-qubit Z gate, then the original circuit converts to two circuits. In the third line, the CZ gate in 8th depth is further decomposed, and the number of circuit is again double. In each of the final 4 circuits, the upper and lower parts is unentangled, and they can be simulated independently.

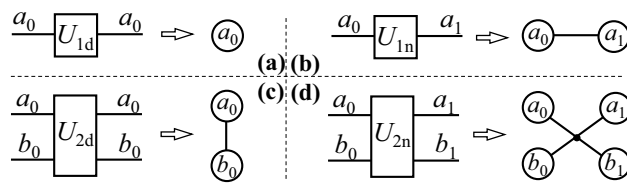


Figure 2. The quantum circuit representations of qubits and gates, and the corresponding undirected graphical model. (a) The diagonal one-qubit gate, (b) the non-diagonal one-qubit gate, (c) the diagonal two-qubit gate, (d) the non-diagonal two-qubit gate. The Boolean variables of a_0, a_1, b_0 and b_1 represent the bit value of the state, which is 0 or 1. The vertices in the graph corresponds to the Boolean variables in the circuit, and the edges corresponds to the gates.

- 3rd: Assign the task of simulating the sub-circuits equally to the nodes. The result of assignment would be that one node simulates one sub-circuit, one node simulates several sub-circuits, or several nodes simulate one sub-circuit. The simulations are implemented in the same way as the full amplitude mode.
- 4th: Combine the state of each sub-circuit to get the final states.

Single amplitude mode. The single amplitude mode makes use of undirected graphical model to be capable of simulating quantum circuit with much more qubits. Broadly, the original quantum circuit is first mapped to an undirected graphical model, then the undirected graph is split into several ones by fixing the value of variables, and then the resulting graphs are processed in parallel by the vertical variable elimination algorithm.

The undirected graph model is a way of interpreting the relation between the change of bit values of qubit state and the quantum gates. Naturally, the bit of state will change with actions of a sequence of quantum gates. We define a sequence of Boolean variables to describe the change. For example, being acted upon by the Pauli- X and H gate in sequence, the state $|0\rangle$ will be first changed to $|1\rangle$, then to $1/\sqrt{2}(|0\rangle - |1\rangle)$. Then the corresponding Boolean variables are $a_0 = 0, a_1 = 1$, and $a_2 = \{0, 1\}$, respectively. The undirected graph is constructed based on the Boolean variables and quantum gates. Specially, each Boolean variable in the circuit corresponds exactly to one vertex in the graph, and one or multiple gates in the circuit result in one edge in the graph.

The rule of mapping a quantum circuit to an undirected graph is simple and easy to follow⁹. It is summarized to four cases as shown in Fig. 2. For the diagonal one- or two-qubit gate, it does not change the Boolean variable, so the vertices corresponding to the same variable merge into one. For example, the CZ gate will transform the state $|11\rangle$ to $-|11\rangle$ without flipping of the bit, so the input and output vertices are merged as shown in Fig. 2c. The cross lines in the graph should be considered as one line, which corresponds to one gate, as shown in Fig. 2d. Figure 3 presents an example to further illustrate the mapping of a circuit to undirected graph.

After getting the undirected graph, tensor techniques are used to process it. One edge in the graph corresponds to a particular tensor, and the number of vertices connecting to the edge is the rank of the tensor. For

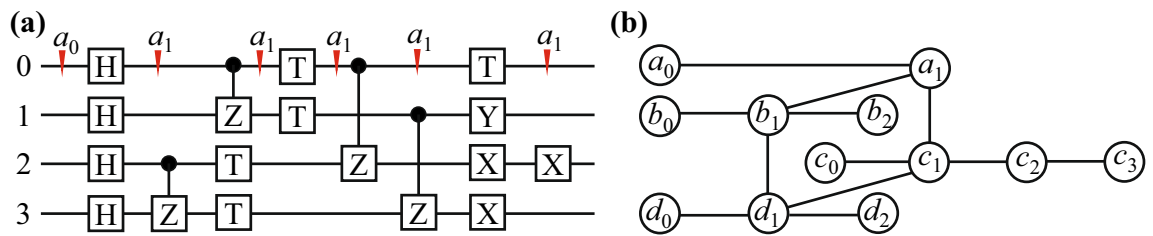


Figure 3. An illustration example of mapping a quantum circuit to the undirected graphical model. **(a)** A quantum circuit adapted from the part of Fig. 1, **(b)** the corresponding undirected graph. The triangle symbols are used to explain the change of Boolean variables along the worldline of qubit 0. Note that since the cross lines are considered as one line in the undirected graph, the vertices are rearranged to avoid the false crossover.

example, the edge in Fig. 2d corresponds to a tensor T of rank 4, with 2^4 elements indexed by $T_{a_0 b_0 a_1 b_1}$. The elements of tensor T are filled using U_{2n} in the lexicographical order of the index, such as that $T_{00,00} = (U_{2n})_{0,0}$, $T_{00,10} = (U_{2n})_{0,2}$, $T_{01,00} = (U_{2n})_{1,0}$, $T_{10,00} = (U_{2n})_{2,0}$ and so on.

There are two kinds of processes performed on the undirected graph, which are edge merging and vertex elimination. Edge merging means that two edges connecting to the same vertex are merged to one. This is actually to merge two tensors with the same subscript into one. For instance, suppose that the edge between vertices b_0 and b_1 in Fig. 3b corresponds to a tensor $A_{b_0 b_1}$, and the edge between vertex b_1 and d_1 corresponds to $B_{b_1 d_1}$, then the two edges merge into one to get a higher-rank tensor as $C_{b_0 b_1 d_1} = A_{b_0 b_1} B_{b_1 d_1}$.

Vertex elimination reduces the number of vertices connecting to a particular edge. This is actually a variant of tensor contraction. We do this using two different methods, of which one is a differential way and the other an integral way. In the differential method, the variable corresponding to a vertex is fixed to be 0 and 1²². For example, the vertex b_1 in Fig. 3b is fixed to 0, then the tensor $B_{b_1 d_1}$ is converted to $B_{0 d_1}$. Thus, the tensor rank is reduced from 2 to 1, and the number of elements from 2^2 to 2. The expense of this method is that it doubles the graph. That is, the graph needs to be computed twice with the target variable being 0 and 1, respectively. In the integral method, all the elements of a tensor corresponding to a specific subscript are summed over to eliminate that index. For instance, the subscript b_1 in tensor $C_{b_0 b_1 d_1}$ is eliminated by $C'_{b_0 d_1} = C_{b_0 0 d_1} + C_{b_0 1 d_1}$, so the vertex b_1 is eliminated from the edge corresponding to tensor $C_{b_0 b_1 d_1}$.

In summary, the program for the single amplitude mode proceeds in the following four steps:

1st: Configure the computation nodes. Then every node parses the script to obtain a linked list recording instructions of the quantum circuit. Map the quantum circuit to the undirected graphical model using the linked list.

2nd: Eliminate the vertices in the first and last depth of the graph according to the specified initial and measurement states using the differential vertex elimination method. Since the initial and measurement states are certain, this step does not double the number of graphs.

3rd: Find the top N vertices with the largest number of connecting edges. Then perform the differential vertex elimination on the N vertices, and this result into 2^N graphs. Assign the task of simulating the 2^N graphs equally to the nodes. (Note that eliminating the top N high-degree vertices would be not the best way of simplifying the graph. The treewidth of the graph really matters, but it is NP-complete to determine^{9,22}. For simplicity, we choose the top N high-degree vertices to remove at this step.)

4th: For each graph, eliminate all the vertices. Specifically, for each vertex, first merge all the connecting edges into one in the order of rank, and then eliminate this vertex using the integral method. Multiply the elements of the tensors corresponding to the left edges, and obtain the amplitude of each graph. Sum over the amplitude of each graph to get the final amplitude of the state to measure.

Simulation of the effect of noise. In practical quantum devices, qubits are performed imperfectly. Various kinds of noise would randomly induce errors on the states of qubits. Particularly, in the coming NISQ era, quantum computers have noisy gates unprotected by quantum error correction²³. Thus, it is important to characterize the effect of noise by classical simulations.

The effect of noise can be described by a series of super operators $\{K_1, K_2, \dots, K_s\}$, which satisfy the relation $\sum_i K_i^\dagger K_i = I$. For the single-qubit gate, we consider the following six kinds of noise,

$$\begin{aligned}
 \text{Bit flip : } & K_1 = \sqrt{p} \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, K_2 = \sqrt{1-p} \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}; \\
 \text{Phase flip : } & K_1 = \sqrt{p} \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, K_2 = \sqrt{1-p} \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}; \\
 \text{Bit - Phase flip : } & K_1 = \sqrt{p} \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, K_2 = \sqrt{1-p} \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix}; \\
 \text{Amplitude Damping : } & K_1 = \begin{bmatrix} 1 & 0 \\ 0 & \sqrt{1-p} \end{bmatrix}, K_2 = \begin{bmatrix} 0 & \sqrt{p} \\ 0 & 0 \end{bmatrix}; \\
 \text{Phase Damping : } & K_1 = \begin{bmatrix} 1 & 0 \\ 0 & \sqrt{1-p} \end{bmatrix}, K_2 = \begin{bmatrix} 0 & 0 \\ 0 & \sqrt{p} \end{bmatrix}; \\
 \text{Depolarizing : } & K_1 = \sqrt{1-3p/4} \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, K_2 = \sqrt{p/2} \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}, K_3 = \sqrt{p/2} \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix}, K_4 = \sqrt{p/2} \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}.
 \end{aligned} \tag{5}$$

The value p in the equation is on $[0, 1]$, which is proportional to the noise intensity. Specifically, for the first three kinds of noise, when p approaches 1, the noise close to zero; for the last three kinds of noise, when p approaches zero, the noise close to zero.

For the two-qubit gate, the noise operators are defined as the Kronecker products of single-qubit gates. For example, suppose the noise operators of single-qubit gates are $\{K_1, K_2\}$ and $\{M_1, M_2\}$, respectively. Then, the noise operators of two-qubit gate are $\{K_1 \otimes M_1, K_1 \otimes M_2, K_2 \otimes M_1, K_2 \otimes M_2\}$.

In the program, the procedure of simulating the noise goes as follows:

1st. Determine the class of quantum gates specified to be noisy and the kind of noise. Let every operator of $\{K_1, K_2, \dots, K_s\}$ act on the present quantum state using the same method as the full amplitude mode. Then calculate the modulus of the states, namely the probabilities of the states.

2nd. Produce a random number between 0 and 1, and compare it with the above sequence of probabilities, then determine which sub-operator K_i to be used. Multiply the matrix K_i with the quantum gate to obtain a new matrix, i.e., the noisy gate.

3rd. Update the state by the new matrix using the same method as the full amplitude mode. Finally, normalize the quantum state (the noisy gate may not be unitary).

To sum up, we have discussed the basic principles of the full, partial and single amplitude modes, as well as the way of defining noisy gate to emulate the effect of noise. Subsequently, we introduce numerical results and applications of the present simulator.

Results and applications

To characterize the performance of the simulator, we first implement the random quantum circuits (RQCs) generated using the prescription of Google²⁴. Then we demonstrate the quantum circuits for solving the Poisson equations and for the quantum arithmetic of evaluating transcendental functions. Here, we remark that the quantum fast Poisson solver and quantum arithmetic algorithms are implemented mainly on the full amplitude mode since these circuits have relatively few qubits and high depth. We leave such applications to future work that the partial and single amplitude modes as well as the function of emulating the effect of noise are exploited.

Implementation of RQCs. The full amplitude mode is the foundation of the other two modes, because the resulting sub-circuits in partial and single modes are finally simulated using the same method as the full amplitude mode. The main factor of limiting the computing speed of full amplitude mode is the data communication between nodes. According to Eqs. (2) and (3), when updating one term of amplitudes α_p , one need another term α_{i+2}^k , which may be stored in another core-group or another SW26010 processors. As shown in Fig. 4, for the one-qubit gate, the speed of computation on a state stored in one core-group (node) is about ten times faster than that in different core-groups. On the other hand, amplitudes being stored in one SW26010 processor or two has almost no influence.

For the partial amplitude mode, we simulate a sequence of RQCs with 4096 nodes. The running time is shown in Fig. 5. In addition to the numbers of qubit and depth, the structure of the lattice of qubits also has a big impact on the running time, as shown by the results of 60 qubits (6×10 and 5×12).

For the single amplitude mode, we simulate RQCs with 49, 110 and 200 qubits using 256 nodes. The running time is shown in Fig. 6. By taking advantage of the distributed computing system, we accomplished the simulation of circuits with up to 200 qubits and 21 depths.

Quantum fast Poisson solver. The Poisson equation is a widely used partial differential equation across many areas of physics and engineering. For instance, when simulating the dynamic process of ocean current, the Navier–Stokes equations²⁵ can be reduced to the Poisson equation under certain conditions²⁶. Solving the Poisson equation, thus, constitutes the most computationally intensive part of the ocean current simulation. We develop a quantum algorithm for solving the multi-dimensional Poisson equation²⁷. It could provide an exponential speedup to some degree over the classical counterparts. Here, we remark that for the one-dimensional Poisson equation, there may exist more efficient quantum algorithms²⁸. It could be implemented on the near-term NISQ devices. We leave this point to future work.

The general idea of our quantum fast Poisson solver is straightforward. First, we discretize the Laplacian operator to a square matrix using the central difference approximation, and then solve the resulting linear system

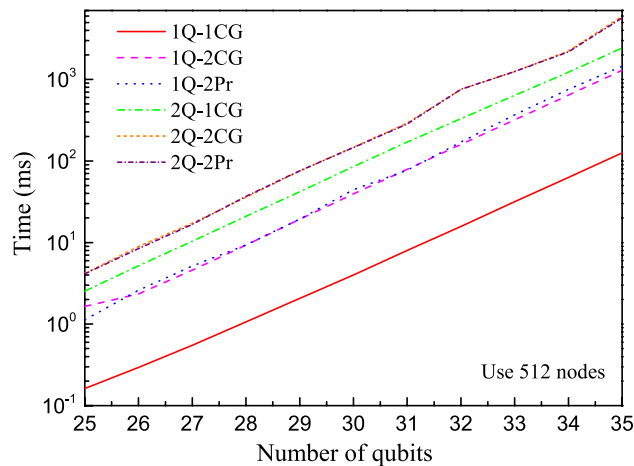


Figure 4. The time span of performing one- and two-qubit gates in different cases on the full amplitude mode. 1Q-1CG means that a one-qubit gate acts on a state stored in one core-group. 1Q-2CG means that a one-qubit gate acts on a state stored in two different core-groups, but in the same SW26010 processor. 1Q-2Pr means that a one-qubit gate acts on a state stored in two SW26010 processors. 2Q represents the two-qubit gate.

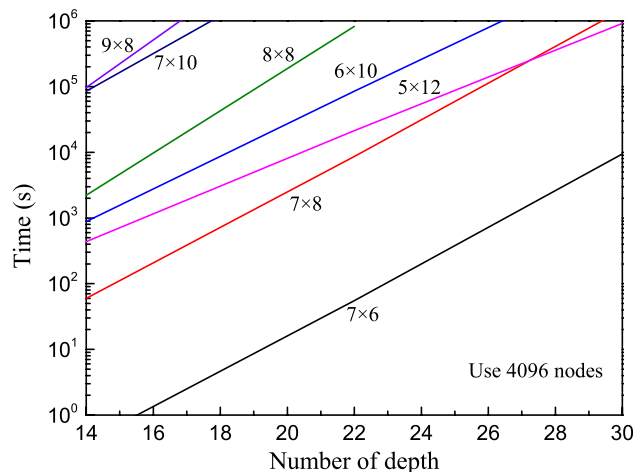


Figure 5. The time span of executing RQCs on the partial amplitude mode. The product of two numbers represents the total number of qubits.

of equations using the Harrow-Hassidim-Lloyd (HHL) algorithm²⁹. Schematically, the algorithm is shown in Fig. 7. It consists of three main stages, i.e., phase estimation, controlled rotation and uncomputation. The complexity of our algorithm is $O(d \log^2(\varepsilon^{-\alpha}))$ in qubits and $O(\kappa d \log^3(\varepsilon^{-\alpha}))$ in quantum operations, where ε is the error of the solution, d the dimension of Poisson equation, $\alpha > 0$ a smoothness constant and κ the condition number of the discretized matrix. On the other hand, any direct or iterative classical algorithms have a cost of at least ε^{-ad} ³⁰. Thus, our quantum Poisson solver could provide an exponential speedup over classical methods in the terms of dimension.

To demonstrate the correctness of the algorithm, we propose a simplified version of the circuit with four discretized points²⁷. The circuit consists of 38 qubits and 800 gates. It is simulated using the full amplitude mode, and the run time is 20 min with 4096 nodes. The input state is $\frac{1}{\sqrt{2}}|01\rangle + \frac{1}{2}|10\rangle + \frac{1}{2}|11\rangle$. This corresponds to a Poisson equation with the solution of (0.9053, 1.1036 0.8018), which turns to (0.553 0.674 0.490) after normalization. The output state is $0.551|01\rangle + 0.675|10\rangle + 0.491|11\rangle$, which is consistent with the real solution with an error less than 0.5%. The running results verify the correctness of our algorithm.

Quantum arithmetic of transcendental functions. Quantum arithmetic in the computational basis constitutes the fundamental component of many circuit-based quantum algorithms. A vast amount of literature provided quantum circuits for solving the algebraic functions, including the addition³¹, multiplication³², reciprocal³³, and square root³⁴ operations, etc. However, studies about the higher-level transcendental functions are scarce^{35,36}. We develop a novel quantum algorithm, the qFBE (quantum Function-value Binary Expansion)

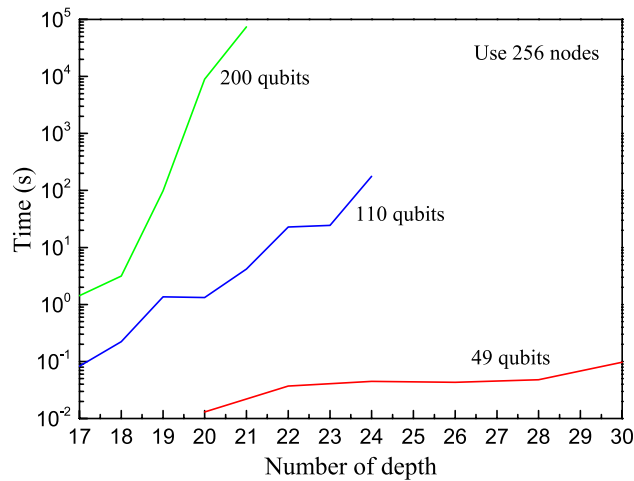


Figure 6. The time span of executing random quantum circuits on the single amplitude mode. The circuit with 200 qubits and 21 depths was simulated.

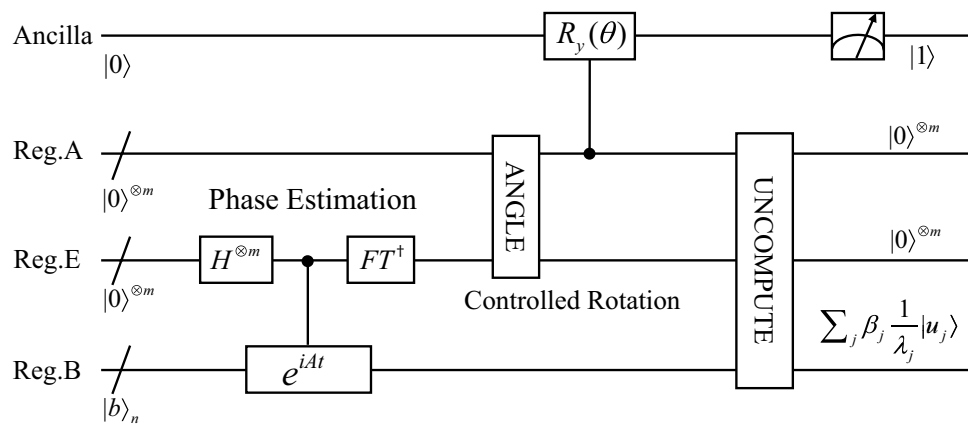


Figure 7. The overall quantum circuit of the quantum fast Poisson solver. It has four main registers, i.e. register B, E, A and Ancilla. Register B is used to encode the coefficients of Poisson equation. It is the input of the circuit. The approximated eigenvalues are stored in register E. Register A is used to store angles for the controlled rotation operation. The Ancilla register will transduce the reciprocals of eigenvalues to the amplitudes.

method, to evaluate the transcendental functions³⁷. The qFBE provides a unified and programmed solution for the evaluation of logarithmic, exponential, trigonometric and inverse trigonometric functions.

Our qFBE method can be used to evaluate two classes of functions: the Class 1 including $\log_2(x)$, $\ln(x)$, $\arccos(x)$, $\arcsin(x)$, $\text{arccot}(x)$ and $\arctan(x)$, and Class 2 including 2^x , e^x , $\cos(x)$, $\sin(x)$, $\cot(x)$, $\tan(x)$. More specifically, suppose the functions of Class 1 are define as $f: I \rightarrow [0,1]$ with $I \subseteq \mathbb{R}$, then the function value can be expanded in a binary form as follows³⁸,

$$f(x) = \sum_{n \geq 0, a_n \in D_1} \frac{1}{2^{n+1}} \tag{6}$$

$$\text{with } a_0 = x, a_{n+1} = \begin{cases} r_0(a_n) & \text{if } a_n \in D_0 \\ r_1(a_n) & \text{if } a_n \in D_1 \end{cases}$$

where D_0 and D_1 are subintervals of I with $D_0 \cup D_1 = I$, $D_0 \cap D_1 = \emptyset$; r_0 and r_1 are functions defined as $r_0: D_0 \rightarrow I$, $r_1: D_1 \rightarrow I$. On the other hand, the functions of Class 2 can be approximated in the following way³⁷,

$$f(x) = a_{n+1} \tag{7}$$

$$\text{with } x = (0.v_{n-1}v_{n-2} \dots v_1v_0), v_i \in \{0, 1\},$$

$$\text{and } a_0 = \text{const}, a_{i+1} = \begin{cases} r_0^{-1}(a_i) & \text{if } v_i = 0 \\ r_1^{-1}(a_i) & \text{if } v_i = 1 \end{cases}$$

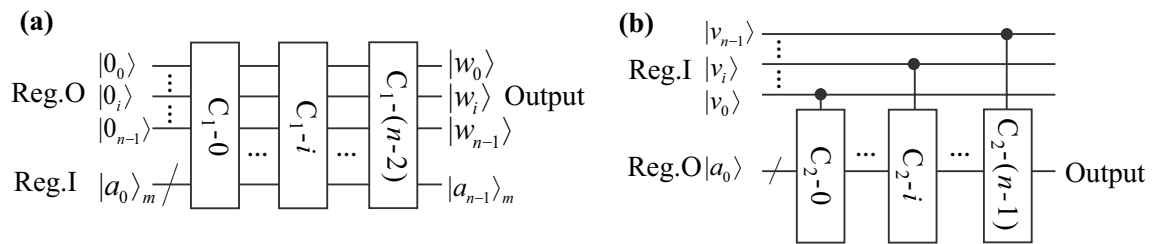


Figure 8. The overall circuits for evaluating functions in Group 1 (a), and Group 2 (b). The Group 1 circuit consists of $n-1$ modules G_{1-i} , which implement the recursions in Eq. (7). Similarly, the Group 2 circuit consists of n modules G_{2-i} , which corresponds to the iterations in Eq. (8). Both circuits mainly include two registers for inputs and outputs.

Apparently, Eq. (6) outputs the function value digit-by-digit in a recursive way, while Eq. (7) approximates the function value step-by-step in an iterative way.

The complexity of evaluating transcendental functions by the qFBE method is $nO(m)$ in qubits and $nO(m^2)$ in quantum gates, where n is the number of qubits to encode input or output and m the number of qubits to store the intermediate values. The cost of our method is comparable with the best known results³⁶ at worst case; while when the input binary has a small number of bits, our method cost much lower. Furthermore, all digits of the binary output can be exact, which makes the control of error propagation easy. The qFBE method provides a unified and programmed solution for most transcendental functions, and the circuits are compact and modular which are easy to be implemented on the virtual or the future real quantum machine.

The quantum circuits for evaluating functions of Class 1 and 2 are shown in Fig. 8 (a) and (b), respectively. For functions of Class 1, the circuit consists of $(n-1)$ modules, which actually implement the recursions in Eq. (6). Each module outputs one bit of the solution. For functions of Class 2, the circuit consists of n modules, which approximate the function value step-by-step according to Eq. (7). The last module outputs the final solution. We present the complete quantum circuits for all the functions in Class 1 and 2 and demonstrate the correctness of these circuits on the simulator, which include $\text{arccot}(x)/\pi$, $\cos(\pi x)$, $\text{arccos}(x)/\pi$, $\cot(\pi x)$, 2^x and $\log_2(x)$ ³⁷. The running results verify the correctness of our algorithm.

Conclusions

We have developed an efficient quantum circuit simulator on the Sunway TaihuLight supercomputer. The simulator possesses three working modes, being capable of calculating the full, partial and single amplitudes of a quantum state. The three modes are built using entirely different methodologies. They are the direct evolution of quantum states, circuit partition by decomposing controlled-Z gate and the complex undirected graphical model. Our simulator has the function of emulating the effects of noise, and it supports many kinds of useful quantum gates and operations. To make full use of the Sunway distributed system, the simulation was implemented in a two-level parallel way. With 16,384 computational nodes, roughly 10% of the computing resource of the Sunway, random quantum circuits with up to 40, 75 and 200 qubits can be simulated on full, partial and single amplitude modes, respectively.

Based on the simulator, we further developed the quantum algorithms for solving the Poisson equations and for quantum arithmetic of evaluating transcendental functions. The present quantum fast Poisson solver takes the HHL algorithm as the framework, and provides an exponential speedup over the classical methods in the terms of dimension. The qFBE method provides a unified and programmed way of evaluating the transcendental functions, including the logarithmic, exponential, arc-cosine, arc-sine, cosine, sine, arc-cotangent, arc-tangent, cotangent and tangent functions.

For future work, we will (1) advance the study of quantum Poisson solver to further reduce the algorithm complexity and quantify the effect of noise, and (2) optimize the qFBE circuits by selecting the proper circuits of evaluating algebraic functions. Furthermore, we will expand the applications of the present simulator to other fields, like variational quantum algorithms and quantum machine learning.

Received: 13 October 2020; Accepted: 9 November 2020

Published online: 11 January 2021

References

- Arute, F. *et al.* Quantum supremacy using a programmable superconducting processor. *Nature* **574**, 505 (2019).
- Kjaergaard, M. *et al.* Superconducting qubits: current state of play. *Annu. Rev. Condens. Matter. Phys.* **11**, 369 (2020).
- Dalzell, A. M., Harrow, A. W., Koh, D. E. & Placa, R. L. L. How many qubits are needed for quantum computational supremacy? *Quantum* **4**, 264 (2020).
- Pednault, E., Gunnels, J. A., Nannicini, G., Horesh, L. & Wisnieff, R. Leveraging secondary storage to simulate deep 54-qubit Sycamore circuits. *arXiv*, 1910.09534v2 (2019).
- Villalonga, B. *et al.* Establishing the quantum supremacy frontier with a 281 Pflops/s simulation. *Quantum Sci. Technol.* **5**, 3 (2020).
- Gray, J. & Kourtis, S. Hyper-optimized tensor network contraction. *arXiv* **2002**, 01935 (2020).
- Huang, C., *et al.* Classical simulation of quantum supremacy circuits. *arXiv*, 2005.06787 (2020).
- Pednault, E., *et al.* Breaking the 49-qubit barrier in the simulation of quantum circuits. *arXiv*, 1710.05867 (2017).

9. Boixo, S., Isakov, S. V., Smelyanskiy, V. N. & Neven, H. Simulation of low-depth quantum circuits as complex undirected graphical models. *arXiv*, 1712.05384v2 (2018).
10. Chen, Z.-Y. *et al.* 64-qubit quantum circuit simulation. *Sci. Bull.* **63**, 964 (2018).
11. Zulehner, A. & Wille, R. Advanced simulation of quantum computations. *IEEE Trans. Comput. Aided Des. Integr. Circuits Syst.* **38**, 5 (2019).
12. Li, R.-L., Wu, B.-J., Ying, M.-S., Sun, X.-M. & Yang, G.-W. Quantum supremacy circuit simulation on Sunway TaihuLight. *IEEE Trans. Parallel. Distrib. Syst.* **31**, 4 (2020).
13. Jones, T., Brown, A., Bush, I. & Benjamin, C. QuEST and high performance simulation of quantum computers. *Sci. Rep.* **9**, 10736 (2019).
14. Raedt, H. D. *et al.* Massively parallel quantum computer simulator, eleven years later. *Comput. Phys. Commun.* **237**, 47 (2019).
15. Guo, C. *et al.* General-purpose quantum circuit simulator with projected entangled-pair states and the quantum supremacy frontier. *Phys. Rev. Lett.* **123**, 190501 (2019).
16. Chen, M.-C. *et al.* Quantum-teleportation-inspired algorithm for sampling large random quantum circuits. *Phys. Rev. Lett.* **124**, 080502 (2020).
17. Pilch, J. & Długopolski, J. An FPGA-based real quantum computer emulator. *J. Comput. Electron.* **18**, 329 (2019).
18. Mahmud, N., El-Araby, E. & Caliga, D. Scaling reconfigurable emulation of quantum algorithms at high precision and high throughput. *Quantum Eng.* **1**, e19 (2019).
19. Fu, H.-H. *et al.* The Sunway TaihuLight supercomputer: system and applications. *Sci. China Inf. Sci.* **59**, 072001 (2016).
20. Long, G.-L. General quantum interference principle and duality computer. *Commun. Theor. Phys.* **45**, 825 (2006).
21. Nielsen, M. A. & Chuang, I. L. *Quantum Computation and Quantum Information* 171–202 (Cambridge University Press, Cambridge, 2010).
22. Chen, J.-X., Zhang, F., Huang, C., Newman, M. & Shi, Y.-Y. Classical simulation of intermediate-size quantum circuits. *arXiv*, 1805.01450v2 (2018).
23. Preskill, J. Quantum Computing in the NISQ era and beyond. *Quantum* **2**, 79 (2018).
24. Boixo, S. *et al.* Characterizing quantum supremacy in near-term devices. *Nat. Phys.* **14**, 595–600 (2018).
25. Lukaszewicz, G. & Kalita, P. *Navier-Stokes Equations: An Introduction with Applications* (Springer International Publishing, Cham, 2016).
26. Steijl, R. & Barakos, G. N. Parallel evaluation of quantum algorithms for computational fluid dynamics. *Comput. Fluids* **173**, 22–28 (2018).
27. Wang, S. B. *et al.* Quantum fast Poisson solver: the algorithm and complete and modular circuit design. *Quantum Inf. Process* **19**, 170 (2020).
28. Wang, S. B., *et al.* A quantum Poisson solver implementable on NISQ devices. *arXiv*, 2005.00256 (2020).
29. Harrow, A. W., Hassidim, A. & Lloyd, S. Quantum algorithm for linear systems of equations. *Phys. Rev. Lett.* **103**, 150502 (2009).
30. Ritter, K. & Wasilkowski, G. W. On the average case complexity of solving Poisson equations. *Lect. Appl. Math.* **32**, 677 (1996).
31. Draper, T. G., Kutin, S. A., Rains, E. M. & Svore, K. M. A logarithmic-depth quantum carry-lookahead adder. *Quantum Inf. Comput.* **6**, 351 (2006).
32. Rines, R. & Chuang, I. High Performance Quantum Modular Multipliers. *arXiv*, 1801.01081 (2018).
33. Thapliyal, H., Munoz-Coreas, E., Varun, T.S.S. & Humble, T.S. Quantum circuit designs of integer division optimizing T-count and T-depth. *arXiv*, 1809.09732 (2018).
34. Munoz-Coreas, E. & Thapliyal, H. T-count and qubit optimized quantum circuit design of the non-restoring square root algorithm. *ACM J. Emerg. Technol. Comput. Syst.* **14**, 3 (2018).
35. Bhaskar, M. K., Hadfield, S., Papageorgiou, A. & Petras, I. Quantum algorithms and circuits for scientific computing. *Quantum Inf. Comput.* **16**, 197 (2016).
36. Häner, T., Roetteler, M. & Svore, K.M. Optimizing quantum circuits for arithmetic. *arXiv*, 1805.12445 (2018).
37. Wang, S. B. *et al.* Quantum circuits design for evaluating transcendental functions based on a function-value binary expansion method. *Quantum Inf. Process* **19**, 347 (2020).
38. Borwein, J. M. & Girgensohn, R. Addition theorems and binary expansions. *Can. J. Math.* **47**, 262 (1995).

Acknowledgements

We are very grateful to the National Supercomputing Center in Wuxi for the great computing resource. The present work is financially supported by the National Natural Science Foundation of China (Grant No. 61575180, 61701464, 11475160) and the Pilot National Laboratory for Marine Science and Technology (Qingdao).

Author contributions

Z.W. designed the quantum algorithms for applications, participated partially in the design of the simulator and prepared the manuscript. Z.C. wrote the simulation programs and tested the simulator. S.W. and W.L. designed the quantum algorithms for applications and tested the simulator. Y.G., G.G. and Z.W. planned, organized and supervised the project. All authors discussed the results and reviewed the manuscript.

Competing interests

The authors declare no competing interests.

Additional information

Supplementary Information The online version contains supplementary material available at <https://doi.org/10.1038/s41598-020-79777-y>.

Correspondence and requests for materials should be addressed to Y.G., G.G. or Z.W.

Reprints and permissions information is available at www.nature.com/reprints.

Publisher's note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.



Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit <http://creativecommons.org/licenses/by/4.0/>.

© The Author(s) 2021