FAST AND ACCURATE SIMULATION OF QUANTUM COMPUTING BY MULTI-PRECISION MPS: RECENT DEVELOPMENT

AKIRA SAITOH*

Quantum Information Science Theory Group, National Institute of Informatics, 2-1-2 Hitotsubashi, Chiyoda, Tokyo 101-8430, Japan

The time-dependent matrix-product-state (TDMPS) simulation method has been known as one of fast simulation methods to study time-evolving quantum systems. Here, I report recent development of my open-source C++ library named ZKCM_QC designed for TDMPS simulations of quantum circuits with arbitrary floating-point precision. Simulation performance is reported for well-known quantum algorithms. In addition, it is numerically shown that a trustworthy simulation should be performed in multiprecision and should not involve truncations of nonzero Schmidt coefficients.

Keywords: Time-dependent matrix product states; Quantum computing

1. Introduction

Classical quantum-circuit simulators[1–9] are practical tools to study quantum computing[10] for the time being as it is quite far beneath the stage of production. In this regard, the time-dependent matrix-product-state (TDMPS) method is a useful simulation method, which was introduced by Vidal in 2003,[3] for a fast simulation of quantum computing when it does not involve a large amount of entanglement. It simulates a quantum circuit within the cost of $O(q_{\rm g} m_{\rm max,max}^3)$ where $q_{\rm g}$ is the number of single-qubit and two-qubit operations in the circuit; $m_{\rm max,max} := \max_{s,t} m(s,t)$ with $m(s,t)$ the Schmidt rank for the splitting between the sites s and $s+1$ at time t. Thus, a polynomial time simulation is possible if the Schmidt rank grows only polynomially in the input size.

The TDMPS method has been, however, used mainly for evaluating the time dependence of physical properties of condensed matters[11,12] rather

*Present address: Department of Computer Science and Engineering, Toyohashi University of Technology, 1-1 Hibarigaoka, Tenpaku-cho, Toyohashi, Aichi 441-8580, Japan.

than simulating quantum algorithms in the physics community as far as the author knows. There have been a few works on TDMPS simulations of quantum algorithms: Kawaguchi *et al.*[13] simulated the Grover search for a simple oracle and showed that the simulation cost was polynomial in the number of qubits. This was because of the simple oracle structure. Later I simulated a variant of the Brüschweiler search and showed that all the solutions could be found within polynomial time when the oracle structure was simple enough.[5] Recently, Chamon and Mucciolo[14] theoretically showed that an integer computation based on TDMPS could solve a search problem within a subexponential time (*i.e.* $O(2^{n^c})$ time with $c < 1$, which is smaller than the query complexity of the Grover search) as long as the oracle circuit can be decomposed to less than $O(n^2)$ two-qubit gates. Besides, Bañuls *et al.*[15] used a TDMPS simulation of an adiabatic time evolution for solving an exact-cover SAT problem.[16] They reported that they could simulate a time evolution of a 100-qubit system with the threshold 14 for the Schmidt rank.

Thus, there have been several evidences for the usefulness of TDMPS for fast simulation of quantum algorithms although there have not been many authors working in this direction. It is easily expected that more researchers will have an interest if there are user-friendly free softwares for this purpose. One choice is the well-known ALPS package,[17] which is a general-purpose simulation library for condensed matter physics. It has a routine for TDMPS but it can only be used for simulating an adiabatic time evolution under given initial and final Hamiltonians. The other choice is my C++ library ZKCM_QC,[18] which is an extension library of the ZKCM library[9] developed with an emphasis of an easy-to-use syntax for multiprecision matrix computation. It uses GMP[19] and MPFR[20] as back-end libraries for multiprecision floating-point computation so that the outputs of ZKCM_QC are accurate for more than several tens of qubits for which double-precision computation causes significant rounding errors during simulation.

In this report, we firstly revisit the basics of the TDMPS method in Sec. 2. The demand of multiprecision computation is briefly described in Sec. 3. Actual simulation performance of the ZKCM_QC library is reported in Sec. 4 in which the standard quantum algorithms, namely, the Deutsch-Jozsa algorithm,[21] the Grover search,[22] and the Shor's algorithm[23] are simulated. Simulation results of an in-place addition is also explained separately since this is one of the important components for economical implementation of the Shor's algorithm. Concluding remarks are given in Sec. 5.

2. Basics of the TDMPS method

Here, we begin with a convention of notations. The computational basis is represented as $\{|0\rangle, |1\rangle\}^n$ for n-qubit quantum states with $|0\rangle = \begin{pmatrix}1 & 0\end{pmatrix}^{\mathrm{T}}$ and $|1\rangle = \begin{pmatrix}0 & 1\end{pmatrix}^{\mathrm{T}}$. An n-qubit quantum state is represented as $|\Psi\rangle = \sum_{i_0\cdots i_{n-1}=0\cdots 0}^{1\cdots 1} c_{i_0\cdots i_{n-1}} |i_0 \cdots i_{n-1}\rangle$ with complex amplitudes $c_{i_0\cdots i_{n-1}}$. We employ the Vidal's MPS form[3,5] for our TDMPS simulations:

$$\begin{aligned}|\Psi\rangle = \sum_{i_0\cdots i_{n-1}=0\cdots 0}^{1\cdots 1} \Bigg[& \sum_{v_0=0}^{m_0-1} \sum_{v_1=0}^{m_1-1} \cdots \sum_{v_{n-2}=0}^{m_{n-2}-1} Q_0(i_0, v_0) V_0(v_0) \\ & \times Q_1(i_1, v_0, v_1) \cdots Q_s(i_s, v_{s-1}, v_s) V_s(v_s) Q_{s+1}(i_{s+1}, v_s, v_{s+1}) \cdots \\ & \cdots V_{n-2}(v_{n-2}) Q_{n-1}(i_{n-1}, v_{n-2}) \Bigg] |i_0 \cdots i_{n-1}\rangle,\end{aligned} \tag{1}$$

where we use tensors $\{Q_s\}_{s=0}^{n-1}$ with parameters i_s, v_{s-1}, v_s (v_{-1} and v_{n-1} are excluded) and $\{V_s\}_{s=0}^{n-2}$ with parameter v_s. In addition, tensor $V_s(v_s)$ stores the Schmidt coefficients for the splitting between the sth site and the $(s+1)$th site; m_s is a suitable number of Schmidt coefficients which does not exceed a threshold m_{trunc} of one's choice.

For example, $|0\cdots 0\rangle$ is represented in this form by setting all m_s to 1, and setting all $Q_s(0,0,0)$ and $V_s(0)$ to 1, and $Q_s(1,0,0)$ to 0.

Under the MPS representation, a unitary time evolution can be computed by taking the corresponding space only into account, *i.e.*, we have only to handle the tensors for the space of our concern. For example, consider a unitary operation $U = \sum_{k,k'=0}^{1} U_{kk'} |k\rangle\langle k'|$ acting on the qubit s. Then the resultant state is computed by updating Q_s in the following way. $Q_s(i_s, v_{s-1}, v_s) \overset{U}{\mapsto} \widetilde{Q_s}(i_s, v_{s-1}, v_s)$ with $\widetilde{Q_s}(i_s, v_{s-1}, v_s) = \langle i_s | \sum_{k,k'} U_{k,k'} Q_s(k', v_{s-1}, v_s) | k\rangle$.

Time evolution under a two-qubit unitary operation acting on qubits s and $s+1$, $U = \sum_{k_s k_{s+1}, k'_s k'_{s+1}} U_{(k_s k_{s+1})(k'_s k'_{s+1})} |k_s\rangle |k_{s+1}\rangle \langle k'_s| \langle k'_{s+1}|$, can also be computed in a similar manner with a little complicated process. This process updates the tensors Q_s, V_s, and Q_{s+1}. Let us firstly write the state in the following way.

$$\begin{aligned}|\Psi\rangle = \sum_{v_{s-1}=0}^{m_{s-1}} \sum_{i_s=0}^{1} \sum_{i_{s+1}=0}^{1} \sum_{v_{s+1}=0}^{m_{s+1}} \Big[& V_{s-1}(v_{s-1}) |v_{s-1}\rangle \\ & \otimes W(i_s, i_{s+1}, v_{s-1}, v_{s+1}) |i_s\rangle |i_{s+1}\rangle \otimes V_{s+1}(v_{s+1}) |v_{s+1}\rangle \Big]\end{aligned}$$

with $|v_{s-1}\rangle$ the left Schmidt vectors for the splitting between sites $s-1$ and s, $W(i_s, i_{s+1}, v_{s-1}, v_{s+1}) = \sum_{v_s} Q_s(i_s, v_{s-1}, v_s) V_s(v_s) Q_{s+1}(i_{s+1}, v_s, v_{s+1})$, and $|v_{s+1}\rangle$ the right Schmidt vectors for the splitting between sites $s+1$

and $s+2$. The unitary transformation is applied to the tensor W in the following way. $W(i_s, i_{s+1}, v_{s-1}, v_{s+1}) \overset{U}{\mapsto} \widetilde{W}(i_s, i_{s+1}, v_{s-1}, v_{s+1})$ with

$$\widetilde{W}(i_s, i_{s+1}, v_{s-1}, v_{s+1}) =$$
$$\langle i_s|\langle i_{s+1}| \sum_{k_s k_{s+1}, k'_s k'_{s+1}} U_{(k_s k_{s+1})(k'_s k'_{s+1})} W(k'_s, k'_{s+1}, v_{s-1}, v_{s+1})|k_s\rangle|k_{s+1}\rangle.$$

Now the resultant state is written as

$$|\widetilde{\Psi}\rangle = \sum_a \sum_b R_{ab}|a\rangle|b\rangle$$

with labels $a = (v_{s-1} i_s)$ and $b = (i_{s+1} v_{s+1})$, and matrix R whose (a, b) element is $R_{ab} = V_{s-1}(v_{s-1})\widetilde{W}(i_s, i_{s+1}, v_{s-1}, v_{s+1})V_{s+1}(v_{s+1})$. Then, we perform a singular value decomposition (SVD) of R. This results in $R = ADB^\dagger$ where A is a $2m_{s-1} \times 2m_{s-1}$ unitary matrix, D is a $2m_{s-1} \times 2m_{s+1}$ matrix with only diagonal elements in the upper-left side, and B is a $2m_{s+1} \times 2m_{s+1}$ unitary matrix. This SVD is performed in the way that the singular values are found in the descending order in D. Suppose there are q nonvanishing singular values. This SVD can also be written as $R_{ab} = \sum_{d=0}^{q-1} A_{ad} D_{dd} (B_{bd})^*$. We choose at most $m_{\rm trunc}$ elements among D_{dd}'s (from larger to smaller) and store them into the tensor $\widetilde{V_s}(v_s)$. Hence, $\widetilde{m_s} = \min(q, m_{\rm trunc})$ elements are stored. Then, we have

$$|\widetilde{\Psi}\rangle = \sum_{v_s=0}^{\widetilde{m_s}-1} \widetilde{V_s}(v_s)|l(v_s)\rangle|r(v_s)\rangle,$$

where $|l(v_s)\rangle = \sum_a A_{a v_s}|a\rangle$ and $|r(v_s)\rangle = \sum_b (B_{b v_s})^*|b\rangle$. Note that $|l(v_s)\rangle$ and $|r(v_s)\rangle$ are represented in the basis $|v_{s-1}\rangle|i_s\rangle$ and the basis $|i_{s+1}\rangle|v_{s+1}\rangle$, respectively. By writing the basis labels explicitly, we have $|l(v_s)\rangle = V_{s-1}(v_{s-1})\widetilde{Q_s}(i_s, v_{s-1}, v_s)|v_{s-1}\rangle|i_s\rangle$ and $|r(v_s)\rangle = \widetilde{Q_{s+1}}(i_{s+1}, v_s, v_{s+1})V_{s+1}(v_{s+1})|i_{s+1}\rangle|v_{s+1}\rangle$. In this way, the tensors Q_s, V_s, and Q_{s+1} are updated to $\widetilde{Q_s}$, $\widetilde{V_s}$, and $\widetilde{Q_{s+1}}$.

It is well-known that single-qubit and two-qubit operations are sufficient for performing universal quantum computation. The ZKCM_QC library, however, uses three-qubit operations as basic operations in order to avoid an overhead in a circuit construction. Simulation of a three-qubit quantum gate acting on consecutive qubits requires an update of tensors Q_s, V_s, Q_{s+1}, V_{s+1}, and Q_{s+2}. This is a more complicated process than the above-described one. For the details, see the appendix of Ref. 9.

It is also possible to simulate a single-qubit projective measurement. Consider a projection operation P acting on the sth qubit. First we update the tensor

Q_s to $\widetilde{Q_s}$ with $\widetilde{Q_s}(i_s, v_{s-1}, v_s) = \langle i_s| \sum_{i'_s} Q_s(i'_s, v_{s-1}, v_s) P |i'_s\rangle / \sqrt{\mu}$ with $\mu = \| \sum_{i_s, v_{s-1}, v_s} Q_s(i_s, v_{s-1}, v_s) P |i_s\rangle \|^2$. Then, we need to update the tensors corresponding to the other qubits as long as they are correlated with the sth qubit. This can be done by sequentially using the same process as applying $I \otimes I$ to the consecutive qubits, moving the *cursor* from s to 0 and also from s to $n-1$. We need not to update the tensors beyond the place where the Schmidt rank is one.

2.1. *Operator-space TDMPS*

Although I do not use the operator-space TDMPS for simulations presented in this contribution, it is beneficial to mention about it as it is quite straight-forward to migrate from the standard TDMPS. Here, we follow the formulation by Zwolak.[24] It is used for simulating time evolution of a density operator under trace-preserving completely-positive (TPCP) maps.

The few things we should pay attention to are the computational basis and the definition of the inner product. As for the basis for a qubit, we need to employ an operator basis, typically $\{|0), |1), |2), |3)\}$ where $|0) = I$, $|1) = X$, $|2) = Y$, and $|3) = Z$ are standard Pauli matrices [here, $Z = \mathrm{diag}(1, -1)$]. As for the inner product, its definition should be given as $(A|B) = \mathrm{Tr}(A^\dagger B)/d$ for operators A and B acting on a d-dimensional Hilbert space. Then the operator MPS representation of an n-qubit density matrix is

$$
\begin{aligned}
|\rho) = \sum_{i_0 \cdots i_{n-1} = 0 \cdots 0}^{3 \cdots 3} \Bigg[& \sum_{v_0=0}^{m_0-1} \sum_{v_1=0}^{m_1-1} \cdots \sum_{v_{n-2}=0}^{m_{n-2}-1} Q_0(i_0, v_0) V_0(v_0) \\
& \times Q_1(i_1, v_0, v_1) \cdots Q_s(i_s, v_{s-1}, v_s) V_s(v_s) Q_{s+1}(i_{s+1}, v_s, v_{s+1}) \cdots \\
& \cdots V_{n-2}(v_{n-2}) Q_{n-1}(i_{n-1}, v_{n-2}) \Bigg] |i_0 \cdots i_{n-1}).
\end{aligned} \tag{2}
$$

This is quite similar to the MPS of a pure state we have seen in Eq. (1). For a simple example, $(I/2)^{\otimes n}$ is represented by an operator MPS with tensor data $Q_0(0,0) = 1/2$, $Q_s(0,0,0) = 1/2$ ($s = 1, \ldots, n-2$), $Q_{n-1}(0,0) = 1/2$, and $V_s(0) = 1$ ($s = 0, \ldots, n-2$).

Time evolution of $|\rho)$ caused by a TPCP map Λ can be simulated in the same manner as we simulate a unitary time evolution in the standard TDMPS method. Here, Λ must be represented as a square matrix with the $|k)(k'|$ notation, *i.e.*, $\Lambda = \sum_{kk'} \Lambda_{kk'} |k)(k'|$. For example, time evolution caused by a TPCP map Λ acting on the sth qubit is simulated by updating $Q_s(i_s, v_{s-1}, v_s)$ to $\widetilde{Q_s}(i_s, v_{s-1}, v_s) = (i_s| \sum_{k=0,k'=0}^{3,3} \Lambda_{k,k'} Q_s(k', v_{s-1}, v_s) |k)$. Time evolution caused by a two-qubit TPCP map can also be simulated in

the same manner as we have already seen for the two-qubit unitary gate simulation in the standard TDMPS method.

It should be useful to mention that a unitary operation U acting on a d-dimensional Hilbert space can be easily translated into a map $\mathcal{U}$ acting on a d^2-dimensional operator Hilbert space. Let us denote the operator basis operators as $|k) \equiv |\sigma_k)$ with σ_k the kth basis operator. Then, the (i,j) element of the matrix representation of $\mathcal{U}$ is $\mathcal{U}_{ij} = (i|\mathcal{U}|j) = \mathrm{Tr}(\sigma_i U \sigma_j U^\dagger)/d$.

3. Necessity of multiprecision computation

Multiprecision computation has been utilized in computational physics to avoid the accumulation of rounding errors in sensitive simulations of dynamics,[25] but has not been widely used in the community. For the TDMPS simulation of quantum circuits, I demonstrated that multiprecision computation is requisite in order for avoiding a significant error in the simulation results.[18]

The particular findings in Ref. 18 were as follows. In both cases, a truncation of nonzero Schmidt coefficients was not employed.
(i) A TDMPS simulation of a five-qubit circuit for a three-qubit Grover search was performed. In the Grover routine R_{G}, the oracle part was set to $U_{\mathrm{o}} = 1 - 2|101\rangle\langle 101|$. Thus, R_{G} was set to $(1 - 2|s\rangle\langle s|)U_{\mathrm{o}}$ with $|s\rangle = (1/\sqrt{8})\sum_{x_0x_1x_2=000}^{111} |x_0x_1x_2\rangle$. The exact probability of finding the solution after twenty Grover iterations, $p_{20} = |\langle 101|R_{\mathrm{G}}^{20}|s\rangle|^2$, was calculated exactly by a symbolic computation. The error in the probability $\widetilde{p_{20}}$ computed by the simulation was evaluated as the quantity $E = |\widetilde{p_{20}} - p_{20}|$. This numerical error E was plotted against the floating point precision employed for the simulation. It was found that E was larger than 0.035 and did not change largely until the precision was enhanced beyond the double precision. It suddenly dropped around the 55-bits precision and almost vanished for more than 70-bits precision.
(ii) An n qubit circuit performing $(\mathrm{QFT}^{-1})(\mathrm{QFT})(\mathrm{CNOT}_{0,n-1})H_0|0_0\cdots 0_{n-1}\rangle$ was considered, where QFT is a quantum Fourier transform, $\mathrm{CNOT}_{0,n-1}$ is a controlled NOT gate acting on the 0th and $(n-1)$th qubits, and H_0 is the Hadamard transform acting on the 0th qubit. Since $(\mathrm{QFT}^{-1})(\mathrm{QFT})$ is just an identity map, the resultant state should be $(|0_00_{n-1}\rangle + |1_01_{n-1}\rangle)/\sqrt{2} \otimes |0_1\cdots 0_{n-2}\rangle$. In a TDMPS simulation, however, there is a numerical error to some extent in the computed resultant state. The error was quantified by $E = |\langle 00|\widetilde{\rho'}|11\rangle - 1/2|$ with $\widetilde{\rho'}$ the computed reduced density matrix of the 0th and the $(n-1)$th qubits of the resultant state. For $n = 8, 14$, and 20, E was plotted against

the floating point precision. It was found that, for all of these values of n, a sudden drop of E was observed.
These simulation results suggest that there are cases where a numerical error is significant until we go beyond a certain threshold for the floating point precision. It is thus recommended to see the behavior of computational results as functions of the floating point precision in TDMPS simulations of quantum circuits.

4. Performance

As mentioned, my simulation library ZKCM_QC uses multiprecision floating-point operations provided by GMP[19] and MPFR[20] for basic operations. Thus the basic operations are inevitably slow in comparison to fixed-precision floating-point operations. The way to improve the performance by the author's effort is therefore limited to choosing algorithms for matrix manipulations carefully and making technical elaborations. Here, I report the recent progress in this regard. Firstly, an effort of speeding-up the routine for Hermitian-matrix diagonalization will be reported in subsection 4.1. Then, the simulation performance of my library will be reported for typical quantum algorithms in the remaining subsections. More specifically, results for the Deutsch-Jozsa algorithm, the Grover search, an in-place addition, and the Shor's prime factorization are explained in subsections 4.2, 4.3, 4.4, and 4.5, respectively.

4.1. *Recent speedup in Hermitian-matrix diagonalization*

Speed of Hermitian matrix diagonalization is a large factor of the actual speed of the TDMPS simulation using ZKCM_QC since it uses the routine of Hermitian-matrix diagonalization for the singular value decomposition. As of version 0.3.6 of ZKCM, the speed has been improved significantly and now that it is faster than that of the famous PARI library.[26] Our routine uses the standard Householder-QR method for Hermitian matrices[27] and is named "diag_H".

It should be noted that conventional multiprecision routines for diagonalization have not been useful for our purpose. The rountine "`eigen`" of PARI does not work for degenerate subspaces (see the PARI/GP bug report logs - #1349, August 2012). Thus it cannot be used in TDMPS simulations of quantum circuits, since most of the subspaces we handle are degenerate (this situation is quite common whenever we have Hadamard gates and CNOT gates). PARI also has another routine "`jacobi`" but this

works for real symmetric matrices only. Thus a workaround is needed to use it for our purpose. [For an Hermitian matrix A, the workaround is to use a symmetric matrix $\begin{pmatrix} \mathrm{Re}(A) & -\mathrm{Im}(A) \\ \mathrm{Im}(A) & \mathrm{Re}(A) \end{pmatrix}$ (see Ch. 11.5 of Ref. 28).] In addition, it uses the Jacobi's method so that it is slower than the Householder's method. Furthermore, we need to double the precision of an input matrix when "`jacobi`" is used together with this workaround, in order to support specified precision as far as I tested. Another routine is found in the multiprecision LAPACK.[29] Nevertheless, this library cannot be used for the back-end of TDMPS because it has a serious bug in the matrix diagonalization (see a bug report on 26 July 2012 in the Mplapack-devel mailing list). It fails for computing eigenvalues for some matrices with relatively large corner elements and some vanishing center elements (typically a density matrix of an entangled state); this bug has been unfixed yet.

Here I compare the routine "`diag_H`" of ZKCM versions 0.3.6 and 0.3.2 with the routines "`eigen`" and "`jacobi`" of PARI version 2.5.3. For this comparison, I test the average time consumption to find all the eigenvectors of a random $N \times N$ Hermitian matrix with a unit Frobenius norm for precision `prec` [bits].

First I set N to 100 and tried several values of precision (Table 1). The time consumptions of the routines were on the same order of magnitude for the tested precision between 256 and 1280 [bits]. "`diag_H`" of ZKCM version 0.3.6 was the fastest among the compared routines. Second, I fixed the precision to 768 [bits] and tried several values of N. As shown in Table 2, for the tested values between 25 and 125, the time consumptions were again on the same order of magnitude, and "`diag_H`" of ZKCM version 0.3.6 was the fastest for $N \geq 75$.

4.2. *Simulation of the Deutsch-Jozsa algorithm*

Here we will see a TDMPS simulation of the Deutsch-Jozsa algorithm.[21] In a brief explanation, the problem instance is a function $f : \{0,1\}^l \to \{0,1\}$ that is either balanced or constant. [Note: f is balanced if $\#\{\mathbf{x}|f(\mathbf{x}) = 0\} = \#\{\mathbf{x}|f(\mathbf{x}) = 1\}$ where $\mathbf{x} \in \{0_0 \cdots 0_{l-1}, \ldots, 1_0 \cdots 1_{l-1}\}$; f is constant if $f(\mathbf{x})$ is same for all $\mathbf{x}$.] The question is to decide whether f is balanced or constant. This takes $1 + 2^{l-1}$ queries for the worst case in classical computation. In contrast, it takes only a single query in quantum computation using the Deutsch-Jozsa algorithm. A sketch of the algorithm is as follows. (i) We apply $H^{\otimes l} V_f H^{\otimes l}$ to the l-qubit state $|0_0 \cdots 0_{l-1}\rangle$, where V_f is an operation mapping each $|\mathbf{x}\rangle$ to $(-1)^{f(\mathbf{x})}|\mathbf{x}\rangle$.

Table 1. Comparison of the average real time consumption for the program routines to find all the eigenvectors of a normalized random 100×100 Hermitian matrix. The average was taken over ten different matrices. The standard deviation is shown in parentheses in small fonts. "prec" stands for the precision. [Precision was doubled for"`jacobi`" (see the text)]. The programs were run as a single thread on a machine with an Intel Core i5 M460 2.53GHz CPU, 4GB memory, and the Fedora 15 64-bit OS. Note: For precision 256 [bits], function "`eigen`" of PARI stopped with an error and output no result.

`prec`	ZKCM 0.3.6, `diag_H` [sec]	ZKCM 0.3.2, `diag_H` [sec]	PARI, `eigen` [sec]	PARI, `jacobi` [sec]
256	73.0 (0.525)	175 (0.105)	N/A (N/A)	103 (0.324)
512	109 (0.624)	259 (0.160)	171 (1.27)	265 (0.974)
768	171 (0.259)	413 (0.378)	237 (1.24)	477 (2.10)
1024	276 (1.68)	632 (0.358)	378 (1.58)	726 (2.24)
1280	394 (2.18)	903 (0.315)	503 (1.21)	1020 (5.47)

Table 2. Comparison of the average real time consumption for the program routines to find all the eigenvectors of a normalized random $N \times N$ Hermitian matrix under the fixed precision 768 [bits] [precision was doubled for"`jacobi`" (see the text)]. The average was taken over ten different matrices. The standard deviation is shown in parentheses in small fonts. The programs were run as a single thread on a machine with an Intel Core i5 M460 2.53GHz CPU, 4GB memory, and the Fedora 15 64-bit OS.

N	ZKCM 0.3.6, `diag_H` [sec]	ZKCM 0.3.2, `diag_H` [sec]	PARI, `eigen` [sec]	PARI, `jacobi` [sec]
25	1.66 (0.0142)	3.41 (0.0152)	0.941 (0.00406)	5.99 (0.0699)
50	15.3 (0.0530)	34.5 (0.0624)	14.5 (0.0248)	50.9 (0.364)
75	61.2 (0.170)	146 (0.238)	74.3 (0.636)	182 (0.977)
100	171 (0.259)	413 (0.378)	237 (1.24)	477 (2.10)
125	386 (0.909)	961 (1.10)	596 (1.72)	1070 (7.40)

(ii) We measure the l qubits in the computational basis. The probability of finding the qubits in 0's simultaneously in this measurement vanishes when f is balanced; in contrast, it is exactly unity when f is constant.
More details of the algorithm are found in, *e.g.*, Sec. 3.1.2 of Ref. 10.

Here, let us consider a particular function $f(\mathbf{y}_0 \cdots \mathbf{y}_{N_g-1}) = \bigoplus_{i=0}^{N_g-1} g(\mathbf{y}_i)$ with $g(x_0x_1x_2x_3) = (x_0 \wedge x_1) \vee (x_1 \wedge x_2) \vee (x_2 \wedge x_3)$ where $\mathbf{y}_i \in \{0,1\}^4$ and $x_j \in \{0,1\}$; N_g is a positive integer (symbol $\bigoplus$ stands for the exclusive OR operation). In figure 1, the quantum circuit of the algorithm for this function is depicted. This function is a balanced function for any $N_g \geq 1$. By the structure of the circuit, each of the N_g measurements should results in Prob(0000) = 0 if there is no numerical error during simulation. This fact is easily proved: Assume that we have different values of Prob(0000) for two different bundles of qubits in the output. This contra-

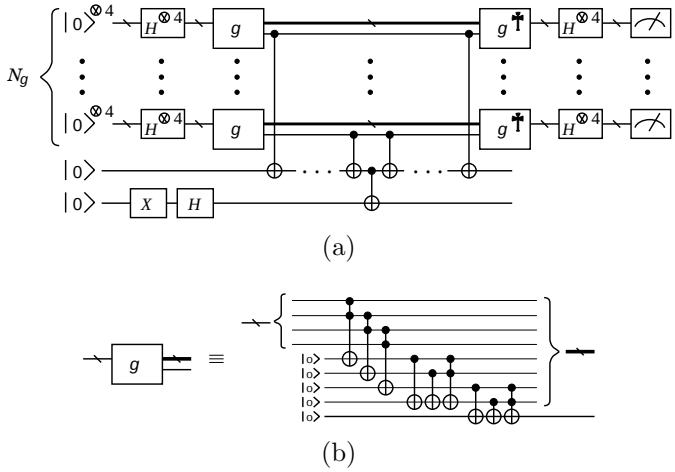

Fig. 1. (a) Quantum circuit of the Deutsch-Jozsa algorithm for the specified function (see the text). (b) Internal structure of gate g.

dicts to the fact that the bundles are equivalent to each other by the circuit structure. Thus the assumption is denied.

In my previous contribution,[9] I employed $N_g = 7$ (namely, 65 qubits in total) and showed that truncation of even a single nonzero Schmidt coefficient caused a significant error in the computed value of Prob(0000). This was because none of nonzero Schmidt coefficients was negligible.

Now I show in this report how the simulation running time and $m_{\max,\max}$ grows as the number n of qubits grows. As shown in Fig. 2, time consumption seems to be on the order of n^3 although further investigation is required. This is probably because the value of $m_{\max,\max}$ became invariant for n larger than a certain value as shown in the figure. This phenomenon appeared probably because the quantum circuit was highly structured.

4.3. *Simulation of the Grover search*

In this subsection, the simulation performance of the ZKCM_QC library is evaluated for an example of the Grover's quantum search.[22] It was shown by Kawaguchi *et al.*[13] that the Grover search can be simulated efficiently with TDMPS if a simple oracle circuit is chosen (see also my TDMPS simulation of a bulk-ensemble database search[5]). Here we consider the Grover search to solve a 3SAT[16] instance. 3SAT is a problem to decide if there is an assignment to the variables $x_0, \cdots, x_{v-1}$ such that a given conjunctive nor-

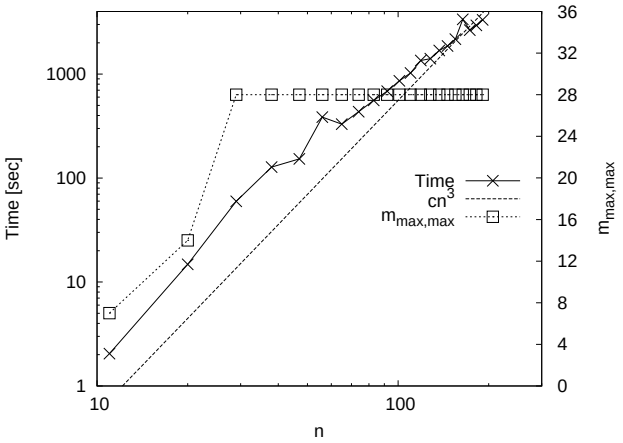

Fig. 2. Plots of running time and $m_{\max,\max}$ as functions of the number of qubits, $n = 9N_g + 2$. The line of cn^3 with $c = 5.58 \times 10^{-4}$ is also shown (using the left vertical axis). Non-averaged raw data were used for the data points. The program was run as a single thread on a machine with four Intel Xeon E7-8837 2.67GHz CPUs, 315GB memory, and the RedHat Enterprise Linux 6 64-bit OS.

mal form (CNF) $\phi(x_0, \cdots, x_{v-1}) = C_0 \wedge \cdots \wedge C_{c-1}$ becomes true (namely, 1), where C_j is a clause $(l_\alpha \vee l_\beta \vee l_\gamma)$ with some three literals. Here, a literal l_τ is a variable x_τ or its logical negation $\neg x_\tau$. A CNF is given without ambiguity, *i.e.*, it is written in terms of $x_0, \cdots, x_{v-1}$ and logical operations.

Unlike the true Grover search for a quantum computer, a TDMPS simulation of the Grover search has only to handle a single query when an instance of a decision problem is given. This is because a slight change occurs in the polarization of the oracle qubit after a single query if and only if the given instance has a truth assignment among the 2^n possible assignments. Figure 3 describes the construction of an oracle quantum circuit for a 3SAT instance.

Consider the following instance: $\phi(x_0, \cdots, x_6) = (\neg x_2 \vee x_4 \vee x_6) \wedge (x_0 \vee x_1 \vee \neg x_3) \wedge (\neg x_1 \vee x_2 \vee \neg x_5) \wedge (x_0 \vee x_1 \vee x_4) \wedge (\neg x_0 \vee \neg x_4 \vee x_6) \wedge (x_2 \vee x_3 \vee \neg x_6) \wedge (\neg x_2 \vee \neg x_5 \vee \neg x_6) \wedge (x_1 \vee \neg x_2 \vee \neg x_3) \wedge (x_1 \vee \neg x_3 \vee x_5) \wedge (x_1 \vee x_4 \vee \neg x_6)$, which is satisfiable. For this instance, a TDMPS simulation of a single query of the Grover search was performed. The program was run as a single thread on the machine with four Intel Xeon E7-8837 2.67GHz CPUs, 315GB memory, and the RedHat Enterprise Linux 6 64-bit OS. It took 132 minutes to perform this simulation when the floating-point precision was 412 [bits]. This is extremely expensive since any classical random seek may solve it

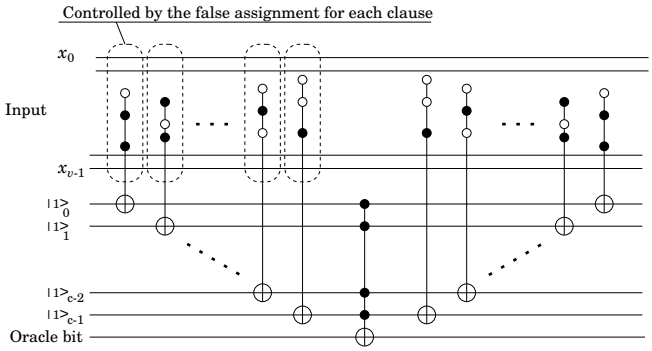

Fig. 3. Quantum oracle circuit for a 3SAT instance with v variables and c clauses.

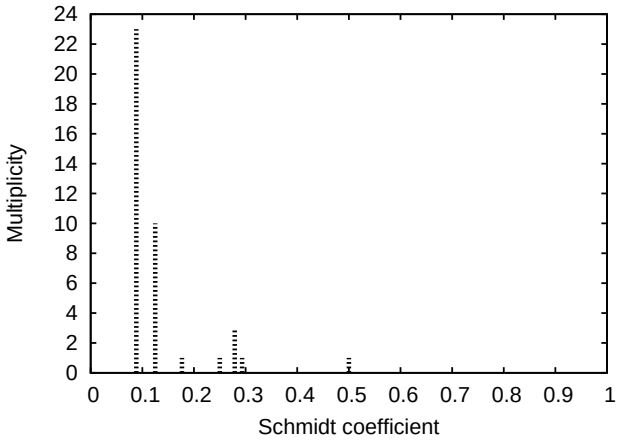

Fig. 4. Distribution of nonzero Schmidt coefficients at the point the Schmidt rank reached 40.

within one millisecond. A possible reason of time consumption is a rather large maximum Schmidt rank. It was 40 and none of the nonzero Schmidt coefficients was negligible (Fig. 4).

4.4. *Simulation of a QFT-based in-place addition*

The in-place arithmetic circuits[30] using quantum Fourier transform (QFT) are quite often used for economical construction of quantum circuits as they do not require ancillary qubits. It is theoretically easily shown that a

TDMPS simulation of a QFT-based addition circuit is fast in the sense that the Schmidt rank during QFT does not exceed the degree of superposition of the input state represented in the computational basis. Here, some theoretical explanation is given and a TDMPS simulation of a quantum circuit involving a simple QFT-based adder is performed.

As is well-known in this community, QFT is defined by

$$\mathrm{QFT} : |a\rangle \mapsto \frac{1}{\sqrt{2^n}} \sum_{l=0}^{2^n-1} e^{i2\pi la/2^n} |l\rangle$$

for an n-bit unsigned integer $a = a_{n-1}a_{n-2}\cdots a_0$. The resultant state is known to be separable:

$$\frac{1}{\sqrt{2^n}} \sum_{l=0}^{2^n-1} e^{i2\pi la/2^n} |l\rangle = |\phi_0(a)\rangle|\phi_1(a)\rangle\cdots|\phi_{n-1}(a)\rangle$$

with

$$|\phi_k(a)\rangle = (|0\rangle + e^{i2\pi(0.a_k a_{k-1}\cdots a_0)}|1\rangle)/\sqrt{2},$$

where $(0.a_k a_{k-1}\cdots a_0) = a_k/2 + a_{k-1}/2^2 + \cdots + a_0/2^{k+1}$.

A quantum circuit implementing QFT usually changes the state of the kth qubit step by step in the following way.[30] Here, $R_k = \mathrm{diag}(1, e^{i2\pi/2^k})$.

$$\begin{aligned} &|a_k\rangle \xrightarrow{H} (|0\rangle + e^{i2\pi(0.a_k)}|1\rangle)/\sqrt{2} \\ &\xrightarrow{R_2 \text{ conditioned on } a_{k-1}} (|0\rangle + e^{i2\pi(0.a_k a_{k-1})}|1\rangle)/\sqrt{2} \\ &\longrightarrow \cdots \xrightarrow{R_{k+1} \text{ conditioned on } a_0} (|0\rangle + e^{i2\pi(0.a_k a_{k-1}\cdots a_0)}|1\rangle)/\sqrt{2}. \end{aligned} \tag{3}$$

Thus, the initial state $|a_k\rangle$ of each qubit evolves to $|\phi_k(a)\rangle$ without introducing entanglement with other qubits. When a superposition $|\psi_{\mathrm{in}}\rangle = \sum_a c_a|a\rangle$ of computational basis states $|a\rangle$ is input to QFT, each qubit of each $|a\rangle$ evolves in the above manner. Therefore, the Schmidt rank for any splitting between two consecutive qubits does not exceed the degree of superposition of the input state, namely, the number of nonzero c_a's, throughout the QFT process.

The circuit structure of QFT employed in the ZKCM_QC library is the one introduced by Fowler *et al.*[31] for the linear nearest neighbor (LNN) architecture as illustrated in Fig. 5. Quantum circuits designed for the LNN architecture are suitable for TDMPS simulations because the MPS data structure for TDMPS is a sort of LNN coupling structures. One can follow how each qubit evolves by tracing the SWAP gates in the figure; it is easily verified to be in the same manner as (3).

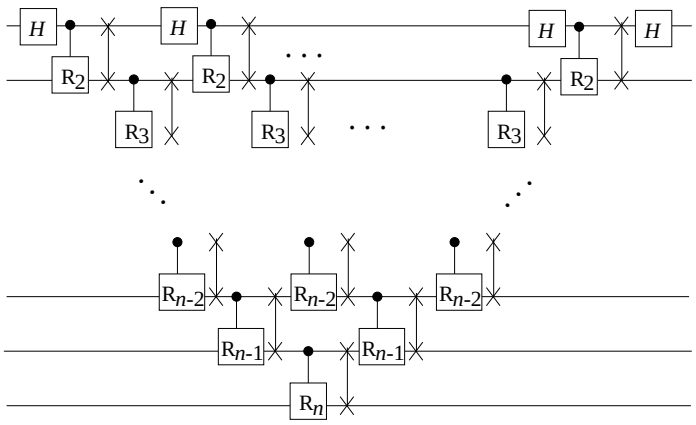

Fig. 5. Illustration of the QFT circuit introduced by Fowler *et al.*[31] for the LNN architecture.

Now we revisit an in-place addition to add an integer $b = b_{n-1}b_{n-2}\cdots b_0$ to $|a\rangle$. Note that, if b is just a classical data, it is not necessary to keep it in a classical or quantum register. The process[30] of in-place addition is as follows.
(i) $|a\rangle \overset{\text{QFT}}{\mapsto} \bigotimes_k |\phi_k(a)\rangle$.
(ii) Each qubit goes through the process:

$$
\begin{aligned}
&|\phi_k(a)\rangle \xrightarrow{b_k-\text{controlled } R_1} (|0\rangle + e^{i2\pi(0.a_k a_{k-1}\cdots a_0 + 0.b_k)}|1\rangle)/\sqrt{2} \\
&\xrightarrow{b_{k-1}-\text{controlled } R_2} (|0\rangle + e^{i2\pi(0.a_k a_{k-1}\cdots a_0 + 0.b_k b_{k-1})}|1\rangle)/\sqrt{2} \\
&\longrightarrow \cdots \xrightarrow{b_0-\text{controlled } R_{k+1}} (|0\rangle + e^{i2\pi(0.a_k a_{k-1}\cdots a_0 + 0.b_k b_{k-1}\cdots b_0)}|1\rangle)/\sqrt{2} \\
&= |\phi_k(a+b \mod 2^n)\rangle
\end{aligned}
$$

(iii) $\bigotimes_k |\phi_k(a+b \mod 2^n)\rangle \overset{\text{QFT}^{-1}}{\mapsto} |a+b \mod 2^n\rangle$.

Step (ii) is a Fourier-domain addition, which alone is often used as a main component for constructing arithmetic quantum circuits.

It is now clear that the Schmidt rank of any nearest-neighbour splitting does not exceed the initial degree of superposition in the computational basis throughout the in-place addition. Now we are going to see a numerical result visualising this property.

Let us consider a simple example where $|0\cdots01\rangle$ is added to the GHZ state $(|0\cdots0\rangle + |1\cdots1\rangle)/\sqrt{2}$ (thus the resultant state is $(|0\cdots01\rangle + |0\cdots00\rangle)/\sqrt{2}$). The quantum circuit is illustrated in Fig. 6. A TDMPS simulation of this circuit is performed for input sizes n up to 100. As shown

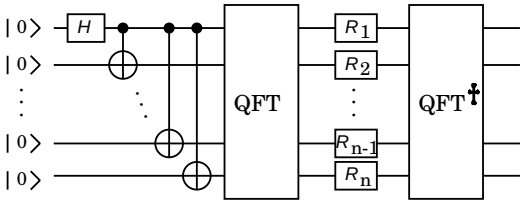

Fig. 6. Illustration of a simple example of QFT-based addition (see the text for more explanation).

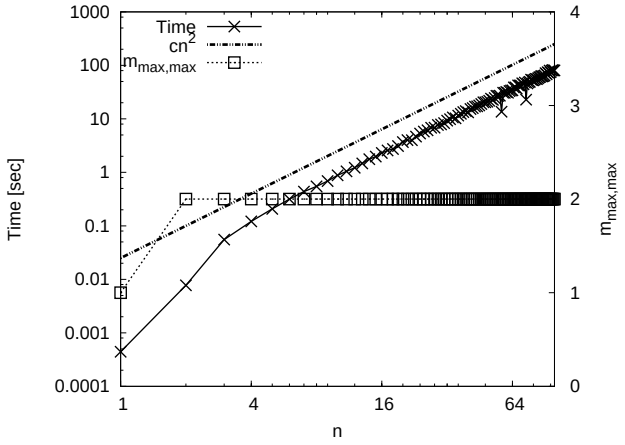

Fig. 7. Plots of running time and $m_{\max,\max}$ in the TDMPS simulation of the quantum circuit illustrated in Fig. 6 as functions of n. The floating-point precision was set to 256. The line of cn^2 is shown with $c = 0.025$ (using the left vertical axis). The program was run as a single thread on the machine with four Intel Xeon E7-8837 2.67GHz CPUs, 315GB memory, and the RedHat Enterprise Linux 6 64-bit OS.

in Fig. 7, the maximum Schmidt rank $m_{\max,\max}$ was 2 for all $n \geq 2$; this is in accordance with the theoretical observation. The figure also shows time consumption as a function of n. It seems to be bounded from above by cn^2 with $c = 0.025$. This is reasonable as the number of quantum gates in the circuit is on the order of n^2 and the maximum Schmidt rank is constant. It should be noted that theoretical analyses of more realistic QFT-based arithmetic circuits are not so easy as the present case; there should be ancillary qubits for conditioning arithmetics and concatenations of Fourier-domain operation units. It is hoped that a theoretical estimation of the maximum Schmidt rank will be made for each QFT-based operation unit. This will

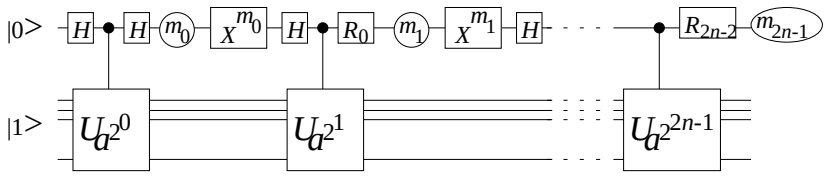

Fig. 8. Semiclassical-QFT-based quantum circuit for Shor's prime factorization introduced by Beauregard.[33]

be useful to analyze the QFT-based variants[31,32] of Shor's quantum prime factorization.

4.5. *Simulation of the Shor's algorithm*

At last, we will see the performance of the ZKCM_QC library for simulating Shor's algorithm.[23] As is well-known, prime factorization by Shor's algorithm is the most significant application of quantum computing. Known classical algorithms take subexponential time while Shor's algorithm takes only polynomial time for factoring a composite number.

Here, I consider Beauregard's circuit construction[33] where a semiclassical QFT is utilized (Fig. 8) to reduce the upper-side resister to a single qubit. As for the modular exponentiation, I used Fowler *et al.*'s construction[31] for the linear nearest neighbor (LNN) architecture, which uses LNN QFT-based Fourier-domain additions internally. In total, the circuit has $2n+4$ qubits and $O(n^4)$ quantum gates[31] for an n-bit-long composite number N. It should be mentioned that several authors[34–36] discussed the simulability of QFT-based variants of Shor's algorithm in relation with the TDMPS and related methods, which is still an open question.

Here, several randomly-generated composite numbers have been tried and solved correctly by the TDMPS simulation of quantum prime factorization. At most a 25-bit number has been tried [thus the circuit width (the number of qubits) has been up to 54]. For this simulation, ZKCM_QC ver.0.1.2 has been used together with ZKCM ver.0.3.6. The running time looks growing only polynomially in n as shown in Fig. 9. Note that this running time is for the entire prime factorization process including the TDMPS simulation of the quantum circuit and the subsequent classical integer computation. By the fitting, it looks that $O(n^4)$ time is enough for simulation. Of course, larger n should be tried to see the tendency for a practical range, say, $n = 128$ or 256. This will be reachable by massive parallel computation in near future.

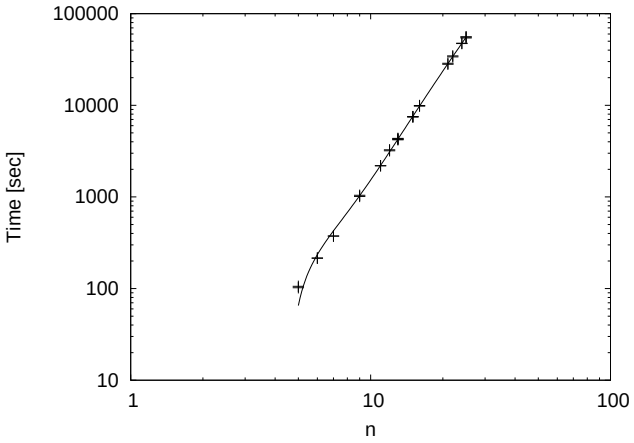

Fig. 9. Running time as a function of the bit length n of a composite number. The curve is the polynomial curve fitting with degree four. The floating-point precision was set to 128. The program was run as a single thread on the machine with four Intel Xeon E7-8837 2.67GHz CPUs, 315GB memory, and the RedHat Enterprise Linux 6 64-bit OS.

5. Concluding remarks

Several simulation results of the TDMPS simulation of quantum circuits have been introduced in this report. It has been often the case that the TDMPS method has been rather economical and able to simulate a relatively large quantum circuit within practical time. It has been indicated that the method is especially economical in simulating the quantum Fourier transform (QFT). It is interesting to further investigate the ability to simulate a QFT-based variant of quantum prime factorization by continuing the simulation work shown in Sec. 4.5. The TDMPS method, however, is not always very fast. The time consumption for a certain small 3SAT instance was significantly large in the TDMPS simulation of a single query of the Grover search. This suggests that the TDMPS simulation is not suitable for solving NP-hard database search problems although one cannot state it definitely with this example alone.

As I showed in Refs. 9 and 18, the TDMPS method is sensitive to rounding errors and truncation errors when it is used for simulating quantum computing. This is why multiprecision computation has been employed in my library. Nonetheless, owing to the poor hardware support for multiprecision floating-point operations in commercial CPUs, the constant factor of computational cost is presently significantly large. It is hoped that an

automated parallelization will be implemented in the library so that it can handle larger quantum circuits feasible for solving practical computational problems.

Acknowledgments

This work was supported by the Grant-in-Aid for Scientific Research from JSPS (Grant No. 25871052).

References

1. J. Niwa, K. Matsumoto and H. Imai, *Phys. Rev. A* **66**, 062317 (2002).
2. G. Viamontes, I. Markov and J. Hayes, *Quantum Inf. Process.* **2**, 347 (2003).
3. G. Vidal, *Phys. Rev. Lett.* **91**, 147902 (2003).
4. S. Aaronson and D. Gottesman, *Phys. Rev. A* **70**, 052328 (2004).
5. A. SaiToh and M. Kitagawa, *Phys. Rev. A* **73**, 062332 (2006).
6. K. D. Raedt, K. Michielsen, H. D. Raedt, B. Trieu, G. Arnold, M. Richter, T. Lippert, H. Watanabe and N. Ito, *Comput. Phys. Comm.* **176**, 121 (2007).
7. F. Tabakin and B. Juliá-Díaz, *Comput. Phys. Comm.* **180**, 948 (2009).
8. E. Gutiérrez, S. Romero, M. Trenas and E. Zapata, *Comput. Phys. Comm.* **181**, 283 (2010).
9. A. SaiToh, *Comput. Phys. Comm.* **184**, 2005 (2013), arXiv:1303.6034.
10. J. Gruska, *Quantum Computing* (McGraw-Hill, Berkshire, UK, 1999).
11. K. A. Hallberg, *Adv. Phys.* **55**, 477 (2006).
12. U. Schollwöck, *Ann. Phys.* **326**, 96 (2011).
13. A. Kawaguchi, K. Shimizu, Y. Tokura and N. Imoto, Classical simulation of quantum algorithms using the tensor product representation *Preprint* arXiv:quant-ph/0411205.
14. C. Chamon and E. R. Mucciolo, *Phys. Rev. Lett.* **109**, 030503 (2012).
15. M. C. Bañuls, R. Orús, J. I. Latorre, A. Pérez and P. Ruiz-Femenía, *Phys. Rev. A* **73**, 022344 (2006).
16. M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness* (W.H. Freeman and Co., New York, 1979).
17. B. Bauer *et al.*, *J. Stat. Mech.* **2011(05)**, P05001; (2011), http://alps.comp-phys.org/ .
18. A. SaiToh, A multiprecision C++ library for matrix-product-state simulation of quantum computing: Evaluation of numerical errors, in *Proceedings of the Conference on Computational Physics 2012, J. Phys.: Conf. Ser.*, (IOP, London, 2013, Kobe, Japan, 14-18 October 2012). arXiv:1211.4086.
19. The GNU Multiple Precision Arithmetic Library http://gmplib.org/.
20. L. Fousse, G. Hanrot, V. Lefèvre, P. Pélissier and P. Zimmermann, *ACM Trans. Math. Software* **33**, 13; (2007), http://www.mpfr.org/.
21. D. Deutsch and R. Jozsa, *Proc. Royal Soc. London A* **439**, 553 (1992).
22. L. K. Grover, A fast quantum mechanical algorithm for database search, in *Proceedings of the 28th Annual ACM Symposium on Theory of Computing*

(STOC 1996), (ACM Press, New York, 1996, Philadelphia, PA, 22-24 May 1996).
23. P. W. Shor, *SIAM J. Comput.* **26**, 1484 (1997).
24. M. P. Zwolak, Dynamics and simulation of open quantum systems PhD Thesis, California Institute of Technology, 2008.
25. D. H. Bailey, R. Barrio and J. M. Borwein, *Appl. Math. Comput.* **218**, 10106 (2012).
26. The PARI Group, PARI/GP http://pari.math.u-bordeaux.fr/.
27. D. Mueller, *Numer. Math.* **8**, 72 (1966).
28. W. Press, S. Teukolsky, W. Vetterling and B. Flannery, *Numerical Recipes: The Art of Scientific Computing (3rd ed.)* (Cambridge University Press, Cambridge, UK, 2007).
29. M. Nakata, The MPACK (MBLAS/MLAPACK); a multiple precision arithmetic version of blas and lapack http://mplapack.sourceforge.net/.
30. T. G. Draper, Addition on a quantum computer *Preprint* arXiv:quant-ph/0008033.
31. A. G. Fowler, S. J. Devitt and L. C. L. Hollenberg, *Quantum Inf. Comput.* **4**, 237 (2004).
32. Y. Takahashi, N. Kunihiro and K. Ohta, *Quantum Inf. Comput.* **7**, 383 (2007).
33. S. Beauregard, *Quantum Inf. Comput.* **3**, 175 (2003).
34. A. Kawaguchi, K. Shimizu, Y. Tokura and N. Imoto, Classical simulation of the modular exponentiation using the tensor product decomposition in: Extended Abstract Booklet of the 11th Quantum Information Technology Symposium, Kyoto, Japan, 6-7 December 2004 (unpublished, in Japanese with English abstract), pp.163-166, technical report No. QIT2004-82.
35. D. E. Browne, *New J. Phys.* **9**, 146 (2007).
36. N. Yoran and A. J. Short, *Phys. Rev. A* **76**, 060302(R) (2007).