Vol.2021-QS-2 No.15
2021/3/29

Quantum Speedup of Monte Carlo Integration with respect to the Number of Dimensions and its Application to Finance

Kazuya Kaneko,[1] Koichi Miyamoto,[2,1,*] Naoyuki Takeda,[1] and Kazuyoshi Yoshino[1]

[1]*Mizuho-DL Financial Technology Co., Ltd.*
2-4-1 Kojimachi, Chiyoda-ku, Tokyo, 102-0083, Japan
[2]*Center for Quantum Information and Quantum Biology,*
Institute for Open and Transdisciplinary Research Initiatives, Osaka University
1-3 Machikaneyama, Toyonaka, Osaka, 560-8531, Japan

(Dated: February 22, 2021)

Monte Carlo integration (MC) using quantum computers has been widely investigated, including applications to concrete problems. It is known that quantum algorithms based on quantum amplitude estimation (QAE) can compute an integral with a smaller number of iterative calls of the quantum circuit which calculates the integrand, than classical methods call the integrand subroutine. However, the issues about the iterative operations *in* the integrand circuit have not been discussed so much. That is, in the high-dimensional integration, many random numbers are used for calculation of the integrand and in some cases similar calculations are repeated to obtain one sample value of the integrand. In this paper, we point out that we can reduce the number of such repeated operations by a combination of the nested QAE and the use of pseudorandom numbers (PRNs), if the integrand has the separable form with respect to contributions from distinct random numbers. The use of PRNs, which the authors originally proposed in the context of the quantum algorithm for MC, is the key factor also in this paper, since it enables parallel computation of the separable terms in the integrand. Furthermore, we pick up one use case of this method in finance, the credit portfolio risk measurement, and estimate to what extent the complexity is reduced.

I. INTRODUCTION

Monte Carlo integration (MC) is one of the important examples of computational tasks which quantum computers can speed up[2, 3]. One of the reasons for its importance is the fact that it is widely used in industries, especially finance. Financial firms are performing enormous MC calculations for various purposes, so quantum speedup of such tasks may provide large impacts for them[1]. Some papers have already investigated how to apply the quantum algorithm for MC to concrete problems in finance: for example, portfolio risk measurement[6–8] and pricing of financial derivatives[9–14][2].

The quantum algorithm for MC is based on quantum amplitude estimation (QAE), which was originally investigated in [18] and also studied in the recent papers[3, 19–22]. It is often said that the quantum methods provide quadratic speedup compared with the classical method. The meaning is as follows. Both the quantum and classical MC methods call the *oracle*, that is, the quantum circuit and the subroutine respectively, for calculation of the integrand. In the former and the latter, the estimation error of the integral behaves as $O(N^{-1})$ and $O(N^{-1/2})$, respectively, where N is the *oracle call number*. Equivalently, for the given tolerance δ, the quantum and classical methods require the $O(\delta^{-1})$ and $O(\delta^{-2})$ oracle call, respectively. Therefore, the quantum method can save the number of repeated oracle call tremendously.

On the other hand, in MC, we often perform another type of repeated calculations, which has not been paid close attention to so far. Specifically, when the dimension D of the integration is very high, similar calculations can be repeated so many times in a call of the oracle, that is, in the flow for calculation of one sample value of the integrand. Let us see a concrete example of this: credit portfolio risk measurement. A credit portfolio is a collection of loans that a bank holds. Each bank is monitoring some metrics which represent risks originating from defaults of obligors. Major metrics include value at risk (VaR), the percentile point of the loss caused by defaults, and conditional VaR (CVaR), the expectation value of the loss under the condition that it is larger than VaR. In calculating them using MC, the values of the loss are randomly generated many times. The flow of calculating a sample value of the loss is roughly as follows: (i) generate a random number (RN) x for an obligor, (ii) determine whether he defaults or not according to x, (iii) if he defaults, add the exposure on him[3] to the loss, then (iv) repeat steps (i)-(iii) for all obligors. As this example shows, in the high-dimensional MC where many RNs are necessary, we sometimes run many iterations of similar calculations, each of which uses a different RN.

In this paper, we propose a method based on QAE which speeds up such a type of repeated calculation. In this new method, there are two key points to make QAE applicable. First, it is necessary that the integrand is *separable*. Although we will strictly state the meaning in Section III, the separable form roughly means that the contributions from different RNs to the integrand are separated into different terms. This

* koichi.miyamoto@qiqb.otri.osaka-u.ac.jp

[1] See [4] as a textbook of financial engineering and see [5] as a reference which focuses on MC used in finance

[2] See [15–17] as reviews for application of quantum computing to finance, including MC and other aspects.

[3] This means the loss which arises if he defaults. In general, it is estimated by the product of the loan amount and the loss given default, the ratio of the amount which the bank fails to recover.

is necessary for computing the integrand separately for each dimension. Second, this method uses pseudorandom numbers (PRNs). PRN sequences are seemingly random but deterministic sequences generated by some recursion formulas. In many cases, we can also use simple formulas to *jump* to the arbitrary position in the PRN sequence, that is, we can get the value of the i-th element not by repeatedly using its recursion formula i times. The authors originally proposed to use PRNs in the quantum algorithm of MC[8, 14]. In the case of the separable integrand, the use of a PRN sequence is crucial to achieve quantum parallel computation of separated terms. Note that, when we use elements in a PRN sequence for a separable integrand, each element is used as a sample value of one of its arguments and thus determines a value of one of separated terms.This implies that, we can construct a quantum circuit which receives an index specifying a term in the integrand as its input and gives a sample value of the term corresponding to the index. Inputting a superposition of all indexes to this circuit, we can compute the terms in quantum parallelism. Therefore, we can replace the naive iterative calculation with the QAE-based calculation, that is, a combination of quantum parallel computation of separated terms and summing them up by QAE. The number of calling the circuit to a separated term changes from $O(D)$ to $O(\delta^{-1})$, which means that we can accomplish the reduction by a factor $O(\delta^{-1}/D)$.

However, we can not immediately conclude that the new method necessarily reduces computational time. That is, time for calculating *one* term in the new method can be larger than that in the previous method. This is because the new method replaces the recursive formula in the previous method with the jump formula and the latter is typically costly than the former. If we write the times for calculating a term in the new and previous methods as $T_{\text{one,new}}$ and $T_{\text{one,prev}}$ respectively, computational time reduction by the new method is $T_{\text{one,new}}\delta^{-1}/T_{\text{one,prev}}D$.

Despite this point, we can find a concrete example where the new method actually reduces the total computational time. We will take credit portfolio risk measurement as a concrete problem and the permutated congruential generator (PCG)[23], which we originally proposed to use in the quantum algorithm for MC in [8], as a concrete PRN generator. We will see that, in a typical setting, we can reduce the *T-count*, a popular metric of computational time cost defined later, by several tens of percent.

The rest of this paper is organized as follows. In Section II, we briefly review the quantum algorithm for MC and use of PRN in it. In Section III, we present the outline of the new method we propose. In Section IV, we consider application of the new method to credit portfolio risk measurement with PCG and estimate the expected speedup. Section V summarizes this paper.

This paper is the short version of the full paper[1]. For the full detail, see [1].

II. THE REVIEW OF THE QUANTUM ALGORITHM FOR MC

A. The quantum algorithm for MC

Let us start with reviewing the quantum algorithm for MC[2]. We here present the flow of calculating the expectation value $E[F(\vec{x})]$ of the function F depending on $\vec{x} = (x_1, ..., x_N)$, the vector of the N stochastic variables. It can be divided into the following four steps. First, we create a superposition of possible values of $\vec{x}$ on a quantum register R_{RN} based on its probability distribution. That is, we create $\sum_i \sqrt{p_i}\,|\vec{x}_i\rangle$, where $\vec{x}_i = (x_1^{(i)}, ..., x_N^{(i)}), i = 1, 2, ...$ is the i-th possible value of $\vec{x}$, p_i is the probability that $\vec{x} = \vec{x}_i$ and $|\vec{x}_i\rangle = |x_1^{(i)}\rangle ... |x_N^{(i)}\rangle$ is the tensor product of states representing the values of the elements of $\vec{x}_i$. Note that x_j must be approximated in some discretized way if it is continuous. Second, we calculate the integrand into another register R_{int} using R_{RN}. Note that the results for many patterns of $\vec{x}$ are simultaneously calculated in quantum parallelism. Third, by controlled rotation, the integrand value is encoded into the amplitude of the ancilla qubit R_{ph}. Finally, using QAE [3, 18–22], we estimate the probability that R_{ph} takes $|1\rangle$, which is equal to the expectation value we want.

From the first to the third steps, the quantum state is transformed as follows:

$$\begin{aligned}
&|0\rangle|0\rangle|0\rangle \\
&\to \left(\sum_i \sqrt{p_i}\,|\vec{x}_i\rangle\right)|0\rangle|0\rangle \\
&\to \left(\sum_i \sqrt{p_i}\,|\vec{x}_i\rangle|F(\vec{x}_i)\rangle\right)|0\rangle \\
&\to \sum_i \sqrt{p_i}\,|\vec{x}_i\rangle|F(\vec{x}_i)\rangle\left(\sqrt{1-F(\vec{x}_i)}\,|0\rangle + \sqrt{F(\vec{x}_i)}\,|1\rangle\right) =: |\Psi\rangle\,.
\end{aligned} \tag{1}$$

Here, the first, second and third kets correspond to $R_{\text{RN}}, R_{\text{int}}$ and R_{ph}, respectively.

We then explain the final step, QAE, based on [18]. At first, we define some symbols. We define θ as

$$|\Psi\rangle = \cos(\theta\pi)\,|\Psi_0\rangle + \sin(\theta\pi)\,|\Psi_1\rangle\,, 0 < \theta < \frac{1}{2}, \tag{2}$$

where $|\Psi_0\rangle$ and $|\Psi_1\rangle$ are the states where R_{ph} is $|0\rangle$ and $|1\rangle$ respectively. Note that

$$\sin^2(\theta\pi) = E_F := \sum_i p_i F(\vec{x}_i) \tag{3}$$

is the expectation value we want. Besides, we write the operation corresponding to the whole of (1) as A and define the operation Q on the system consisting of $R_{\text{RN}}, R_{\text{int}}$ and R_{ph} as

$$Q := -AS_0A^{-1}S_1, \tag{4}$$

where S_0 multiply the state by -1 if all qubits are $|0\rangle$ or do nothing otherwise and S_1 multiply the state by -1 if R_{ph} is

$|1\rangle$ or do nothing otherwise. Then, preparing another register R_θ with m qubits and using an algorithm containing $M-1$ iterations of calling Q, where $M=2^m$ and each Q is controlled by R_θ, we can create the state

$$|\Phi_M(\theta)\rangle := \frac{1}{\sqrt{2}}\left(e^{i\theta\pi}|\Psi_+\rangle|\phi_M(\theta)\rangle - e^{-i\theta\pi}|\Psi_-\rangle|\phi_M(1-\theta)\rangle\right). \tag{5}$$

Here, the second kets $|\phi_M(\theta)\rangle$ and $|\phi_M(1-\theta)\rangle$ correspond to R_θ and $|\Psi_\pm\rangle := \frac{1}{\sqrt{2}}(|\Psi_1\rangle \pm i|\Psi_0\rangle)$. Besides, $|\phi_M(\theta)\rangle$ is defined as

$$|\phi_M(\theta)\rangle := U_M^{-1}|S_M(\theta)\rangle, \tag{6}$$

where $|S_M(y)\rangle$ is the state defined for the real number $y \in (0,1)$ as

$$|S_M(y)\rangle := \frac{1}{\sqrt{M}}\sum_{x=0}^{M-1} e^{2\pi i x y}|x\rangle, \tag{7}$$

and U_M^{-1} is the inverse of quantum Fourier transformation U_M on R_θ, that is,

$$U_M : |x\rangle \mapsto \frac{1}{\sqrt{M}}\sum_{y=0}^{M-1} e^{2\pi i x y/M}|y\rangle, x = 0,1,...,M-1. \tag{8}$$

We then measure R_θ in $|\Phi_M(\theta)\rangle$ and interpret the measurement outcome $\tilde{\theta}$ as a number in $[0,1)$ with m fractional bits. If $\tilde{\theta} > 1/2$, we replace $\tilde{\theta}$ with $1-\tilde{\theta}$. Then, this $\tilde{\theta}$ is close to θ with high probability:

$$\begin{aligned}\Pr\left(\tilde{\theta}=\tilde{\theta}'\right) &= \frac{1}{2}\left[\left|\langle\tilde{\theta}'|\phi_M(\theta)\rangle\right|^2 + \left|\langle 1-\tilde{\theta}'|\phi_M(1-\theta)\rangle\right|^2\right] \\ &= \frac{\sin^2(M(\tilde{\theta}'-\theta)\pi)}{M^2\sin^2((\tilde{\theta}'-\theta)\pi)} \\ &=: G(\tilde{\theta}';\theta,M),\end{aligned} \tag{9}$$

and this leads to

$$\Pr\left(|\tilde{\theta}-\theta| < \frac{1}{M}\right) = \frac{\sin^2(M\delta\pi)}{M^2\sin^2(\delta\pi)} + \frac{\sin^2\left(M\left(\frac{1}{M}-\delta\right)\pi\right)}{M^2\sin^2\left(\left(\frac{1}{M}-\delta\right)\pi\right)} \geq \frac{8}{\pi^2}, \tag{10}$$

where $\delta = |\theta - \lfloor M\theta\rfloor/M|$. Inequality (10) means that we can estimate θ, or equivalently, E_F with the worst-case error proportional to M^{-1} by $O(M)$ calls of the integrand circuit A. This is called the "quadratic speedup" compared with classical MC, where the error is proportional to the inverse square root of the number of calls to the integrand subroutine.

We here make some comments. Firstly, it is sufficient to make only S_0 and S_1 controlled among the operations in Q in order to make Q controlled. Two A's do not have to be controlled. We can easily see this as Q becomes the identical transformation I except an overall constant factor if S_0 and S_1 are replaced with I's. Therefore, if the integrand calculation included in A makes the dominant contribution to complexity, making Q be controlled increases complexity only slightly. Secondly, in the QAE, the total number of integrand calculation and its inverse is nearly equal to $2M$, since the dominant contribution to the number comes from the about M operations of controlled Q, and each of them contains one integrand calculation and one inverse.

B. Use of pseudorandom number in the quantum algorithm for MC

We here briefly review the quantum method for MC using PRNs[4], which is originally proposed in [8]. When we apply the quantum algorithm for MC to an extremely high-dimensional integration, it is necessary to generate as many RNs as the number of dimensions, in order to compute the integrand. If we naively assign a register to each RN and create a superposition of possible values, the required qubit numbers increases in proportion to the number of dimensions. In order to avoid this, we can adopt the following way. First, as preparation, we choose a PRN sequence and set two registers, $R_{\rm samp}$ and $R_{\rm PRN}$. Then, we create a superposition of integers, which specify the start points of the PRN sequence, on $R_{\rm samp}$. For example, if we need $N_{\rm RN}$ RNs to compute the integrand, we can set the start points to the 1st, $(N_{\rm RN}+1)$-th, $(2N_{\rm RN}+1)$-th, ... elements in the sequence[5]. With each start point, we sequentially generate PRNs on $R_{\rm PRN}$. This is possible because a PRN sequence is a deterministic sequence whose recursion equation is explicitly given, and in [8] we gave the implementation of one specific PRN generator, PCG, on quantum circuits. Using the PRNs, we compute the integrand step by step. Finally, the expectation value of the integrand is calculated by QAE. In this way, since we need only $R_{\rm samp}$ and $R_{\rm PRN}$ to generate PRNs, the required qubit number is now independent of the number of dimensions and much smaller than the naive way. The drawback is the increase of the circuit depth.

In this paper, we propose another way for MC using PRNs, where we generate them not sequentially but in a quantum superposition, as explained in section III.

III. THE NEW METHOD FOR MC WITH SPEEDUP WITH RESPECT TO THE NUMBER OF DIMENSION

A. The outline

1. *The problem*

In this section, we present a method to speed up the iterative calculation in computing the integrand in the quantum algorithm for MC, which we call the *new method*. First of all, let us clearly state the problem to which the new method

[4] Of course, regardless of whether it is done in a classical or quantum way, MC based on PRNs can induce additional errors, since PRNs are not truly random but deterministic. Every PRN generator does not have perfect statistical properties, for example, numbers in a PRN sequence inevitably have correlations to some extent. As far as the authors know, for PRN sequences which are widely used today, no established way to estimate errors in MC due to statistical poorness is known. In many practical cases, we check randomness of a given PRN sequence through some statistical tests and use it neglecting errors if it passes the tests. The inventor of PCG claims that it passes TestU01[24], a widely-used test suite for PRN.

[5] Note that $N_{\rm RN}$ should be sufficiently smaller than the period of the PRN sequence. Conversely, we should choose a PRN sequence whose period is long enough.

TABLE I: The quantum registers used in the new method we propose.

Symbol	Usage
R_{samp}	The register where we create the superposition of the indexes j which specify one sample set of the stochastic variables $(\epsilon^{\mathrm{PR}}_{\mathrm{com},j}, \epsilon^{\mathrm{PR}}_{1,j}, ..., \epsilon^{\mathrm{PR}}_{D,j})$.
R_{dim}	The register where we create the superposition of the indexes i which specify one individual stochastic variable $\epsilon^{\mathrm{PR}}_{i,j}$.
R_{com}	The register where we output $\epsilon^{\mathrm{PR}}_{\mathrm{com},j}$.
R_{ind}	The register where we output $\epsilon^{\mathrm{PR}}_{i,j}$.
$R_{\vec{c}}$	The register where we load $\vec{c}_i$.
R_f	The register where we output $f(\epsilon^{\mathrm{PR}}_{\mathrm{com},j}, \epsilon^{\mathrm{PR}}_{i,j}; \vec{c}_i)$.
$R_{\mathrm{ph},f}$	The single-qubit register where we encode $f(\epsilon^{\mathrm{PR}}_{\mathrm{com},j}, \epsilon^{\mathrm{PR}}_{i,j}; \vec{c}_i)$ as the amplitude of $\|1\rangle$.
R_{ctr1}	The register which works as control bits in the inner QAE. After the inner QAE, the sum of $f(\epsilon^{\mathrm{PR}}_{\mathrm{com},j}, \epsilon^{\mathrm{PR}}_{i,j}; \vec{c}_i)$'s over i is encoded here.
R_g	The register where we output $g\left(\sum_{i=1}^{D} f(\epsilon^{\mathrm{PR}}_{\mathrm{com},j}, \epsilon^{\mathrm{PR}}_{i,j}; \vec{c}_i)\right)$.
$R_{\mathrm{ph},g}$	The single-qubit register where we encode $g\left(\sum_{i=1}^{D} f(\epsilon^{\mathrm{PR}}_{\mathrm{com},j}, \epsilon^{\mathrm{PR}}_{i,j}; \vec{c}_i)\right)$ as the amplitude of $\|1\rangle$.
R_{ctr2}	The register which works as control bits in the outer QAE. After the outer QAE, E_{samp} is encoded here.

can be applied. Here and hereafter, we consider the MC to calculate the expectation value $E[F]$ of the function F, which depends on some stochastic variables and takes the *separable* form given by

$$F(\epsilon_{\mathrm{com}}, \{\epsilon_i\}_{i=1,...,D}; \{\vec{c}_i\}_{i=1,...,D}) = g\left(\sum_{i=1}^{D} f(\epsilon_{\mathrm{com}}, \epsilon_i; \vec{c}_i)\right). \quad (11)$$

That is, we can calculate F by summing up the values of one common function f with different inputs and operating the overall function g to the sum. Here, the meanings of the symbols are as follows. D is a natural number which satisfies $D \gg 1$. ϵ_{com} and $\epsilon_1, ..., \epsilon_D$ are mutually independent stochastic variables. The former is the *common stochastic variable*, which is used in all elements in the sum. The latter are *individual stochastic variables*. They are independent and identically distributed and each of them is used in only one term in the sum. Although we hereafter consider ϵ_{com} as a single stochastic variable for simplicity, it is straightforward to generalize the discussion to the case where ϵ_{com} is a vector of multiple stochastic variables. Totally, the number of the stochastic variables is $D+1$ and so is the dimension of the MC. $\vec{c}_1, ..., \vec{c}_D$ are sets of constant parameters.

2. The new method

Then, let us consider how to calculate the expectation value $E[F]$ for the function F in the form of (11).

The new method which we propose here is based on the PRN-approach of MC on quantum computers, which we have explained in Section II B. In this approach, we sample many sets of the values of the stochastic variables using a PRN generator. That is, in the current problem, we obtain the values of ϵ_{com} and $\epsilon_1, ..., \epsilon_D$ in the j-th sample set by sequentially applying the elements in a given PRN sequence $\{x_i\}_{i=1,2,...}$:

$$\epsilon^{\mathrm{PR}}_{\mathrm{com},j} := f_{\epsilon_{\mathrm{com}}}(x_{(j-1)(D+1)+1}), \epsilon^{\mathrm{PR}}_{i,j} := f_{\epsilon}(x_{(j-1)(D+1)+i+1}). \quad (12)$$

Here, $f_{\epsilon_{\mathrm{com}}}$ and f_{ϵ} are the functions to transform the PRNs, which obey the uniform distribution in many cases, so that their distributions match that of ϵ_{com} and $\epsilon_1, ..., \epsilon_D$, respectively. We will consider how to perform such transformations in section III B. Then, $E[F]$ is estimated as

$$E_{\mathrm{samp}} := \frac{1}{N_{\mathrm{samp}}} \sum_{j=1}^{N_{\mathrm{samp}}} g\left(\sum_{i=1}^{D} f(\epsilon^{\mathrm{PR}}_{\mathrm{com},j}, \epsilon^{\mathrm{PR}}_{i,j}; \vec{c}_i)\right), \quad (13)$$

where N_{samp} is the number of the samples. The statistical error of E_{samp}, that is, the confidence interval scales as $O(1/\sqrt{N_{\mathrm{samp}}})$.

The important point is that we can see $f(\epsilon^{\mathrm{PR}}_{\mathrm{com},j}, \epsilon^{\mathrm{PR}}_{i,j}; \vec{c}_i)$ as a function of i and j. That is, if we can implement the following circuits

- U_f
  This calculates $f(\epsilon_{\mathrm{com}}, \epsilon_i; \vec{c}_i)$ for the given ϵ_{com}, ϵ_i and $\vec{c}_i$.
- $U_{\epsilon_{\mathrm{com}}}$
  This calculates $f_{\epsilon_{\mathrm{com}}}(x)$ for the given x.
- U_{ϵ}
  This calculates $f_{\epsilon}(x)$ for the given x.
- U_J
  This makes the PRN sequence $\{x_i\}_{i=1,2,...}$ jump to the given position. Here, we define *making* $\{x_i\}_{i=1,2,...}$ *jump* as the following operation: for a given integer $j \geq 1$, calculating x_j.
- $U_{\vec{c}}$
  This loads $\vec{c}_i$ into a register for the given i.

we can implement the circuit to calculate $\epsilon^{\mathrm{PR}}_{\mathrm{com},j}, \epsilon^{\mathrm{PR}}_{i,j}$ in (12) and then $f(\epsilon^{\mathrm{PR}}_{\mathrm{com},j}, \epsilon^{\mathrm{PR}}_{i,j}; \vec{c}_i)$ for the given i and j. Especially, availability of a formula for jump to a specified position is a beneficial feature of some kinds of PRNs, including PCG considered later, and it enables us to implement U_J easily. Including this point, we will explain how to implement these circuits in section III B.

If we can calculate the above function on a quantum computer, we can take the following way to calculate E_{samp} in (13). We call this a *nested QAE*, since it performs the summation over the sample index j by QAE, which we call the

outer QAE, and in each iteration in the outer QAE, another QAE, which we call the *inner* QAE, runs for the summation over i, the index of the terms. The outline is as follows. First, we make the superposition of states which correspond to the various sets of (i, j). Second, we calculate $f(\epsilon^{\mathrm{PR}}_{\mathrm{com},j}, \epsilon^{\mathrm{PR}}_{i,j}; \vec{c}_i)$ for the various pairs of (i, j) in quantum parallelism. We then use the inner QAE: we sum up these values of f over i for the each value of j without sequential calculation and addition of f. After operating g on the sum, we use the outer QAE to get the sum over j, that is, E_{samp}, avoiding sequential calculation again.

Note that the key factor is the map $(i, j) \mapsto f(\epsilon^{\mathrm{PR}}_{\mathrm{com},j}, \epsilon^{\mathrm{PR}}_{i,j}; \vec{c}_i)$. Thanks to it, we can compute f for various inputs in quantum parallelism and create the superposition of states corresponding to the various values of f, then finally apply the inner QAE to the superposition to estimate the sum of f's with smaller complexity than sequential computation. We again emphasize that using PRN enables us to implement this map.

The detailed steps of the new method are as follows. Preparing the registers shown in Table I, each of which is initialized to $|0\rangle$, we perform the followings:

1. Create $\frac{1}{\sqrt{N_{\mathrm{samp}}}} \sum_{j=1}^{N_{samp}} |j\rangle$ on R_{samp}.
2. With the input j on R_{samp}, calculate $\epsilon^{\mathrm{PR}}_{\mathrm{com},j}$ in (12) on R_{com} using U_J and $U_{\epsilon_{\mathrm{com}}}$.
3. Create $\frac{1}{\sqrt{D}} \sum_{i=1}^{D} |i\rangle$ on R_{dim}.
4. With the inputs i on R_{dim} and j on R_{samp}, calculate $\epsilon^{\mathrm{PR}}_{i,j}$ in (12) on R_{ind} using U_J and U_ϵ.
5. With the input i on R_{dim}, load $\vec{c}_i$ on $R_{\vec{c}}$ using $U_{\vec{c}}$.
6. With the inputs on $R_{\mathrm{com}}, R_\epsilon$ and $R_{\vec{c}}$, calculate $f(\epsilon^{\mathrm{PR}}_{\mathrm{com},j}, \epsilon^{\mathrm{PR}}_{i,j}; \vec{c}_i)$ on R_f.
7. Using the rotation controlled by R_f, transform $R_{\mathrm{ph},f}$ to $\sqrt{1 - f(\epsilon^{\mathrm{PR}}_{\mathrm{com},j}, \epsilon^{\mathrm{PR}}_{i,j}; \vec{c}_i)}\,|0\rangle + \sqrt{f(\epsilon^{\mathrm{PR}}_{\mathrm{com},j}, \epsilon^{\mathrm{PR}}_{i,j}; \vec{c}_i)}\,|1\rangle$. Then, the probability that $R_{\mathrm{ph},f}$ is 1 under the condition that R_{samp} is j is

$$S_j := \frac{1}{D} \sum_{i=1}^{D} f(\epsilon^{\mathrm{PR}}_{\mathrm{com},j}, \epsilon^{\mathrm{PR}}_{i,j}; \vec{c}_i). \tag{14}$$

8. Using the inner QAE, output S_j on R_{ctr1}. Strictly speaking, this step creates the state where the distribution of the value on R_{ctr1} is sharply peaked around θ_j, which is defined through $\sin^2(\theta_j \pi) := S_j$ (see (15) for the detail).
9. With the input $\tilde{\theta}$ on R_{ctr1}, calculate $\tilde{g}(\tilde{\theta}) := g(D \sin^2(\tilde{\theta}\pi))$, which is close to $g\left(\sum_{i=1}^{D} f(\epsilon^{\mathrm{PR}}_{\mathrm{com},j}, \epsilon^{\mathrm{PR}}_{i,j}; \vec{c}_i)\right)$ for $\tilde{\theta} \approx \theta_j$, on R_g.
10. Using the rotation controlled by R_g, transform $R_{\mathrm{ph},g}$ to $\sqrt{1 - \tilde{g}(\tilde{\theta})}\,|0\rangle + \sqrt{\tilde{g}(\tilde{\theta})}\,|1\rangle$.
11. Using the outer QAE, estimate the probability of observing 1 on $R_{\mathrm{ph},g}$, which is nearly equal to E_{samp} (see (17)).

The state is transformed through the above steps of 1-10 as follows. Here, the first to tenth kets correspond to $R_{\mathrm{samp}}, R_{\mathrm{com}}, R_{\mathrm{dim}}, R_{\mathrm{ind}}, R_{\vec{c}}, R_f, R_{\mathrm{ph},f}, R_{\mathrm{ctr1}}, R_g$ and $R_{\mathrm{ph},g}$, respectively.

$$|0\rangle|0\rangle|0\rangle|0\rangle|0\rangle|0\rangle|0\rangle|0\rangle|0\rangle|0\rangle$$

$$\xrightarrow{1} \frac{1}{\sqrt{N_{\mathrm{samp}}}} \sum_{j=1}^{N_{samp}} |j\rangle|0\rangle|0\rangle|0\rangle|0\rangle|0\rangle|0\rangle|0\rangle|0\rangle|0\rangle$$

$$\xrightarrow{2} \frac{1}{\sqrt{N_{\mathrm{samp}}}} \sum_{j=1}^{N_{samp}} |j\rangle|\epsilon^{\mathrm{PR}}_{\mathrm{com},j}\rangle|0\rangle|0\rangle|0\rangle|0\rangle|0\rangle|0\rangle|0\rangle|0\rangle$$

$$\xrightarrow{3} \frac{1}{\sqrt{N_{\mathrm{samp}} D}} \sum_{j=1}^{N_{samp}} \sum_{i=1}^{D} |j\rangle|\epsilon^{\mathrm{PR}}_{\mathrm{com},j}\rangle|i\rangle|0\rangle|0\rangle|0\rangle|0\rangle|0\rangle|0\rangle|0\rangle$$

$$\xrightarrow{4,5} \frac{1}{\sqrt{N_{\mathrm{samp}} D}} \sum_{j=1}^{N_{samp}} \sum_{i=1}^{D} |j\rangle|\epsilon^{\mathrm{PR}}_{\mathrm{com},j}\rangle|i\rangle|\epsilon^{\mathrm{PR}}_{i,j}\rangle|\vec{c}_i\rangle|0\rangle|0\rangle|0\rangle|0\rangle|0\rangle$$

$$\xrightarrow{6} \frac{1}{\sqrt{N_{\mathrm{samp}} D}} \sum_{j=1}^{N_{samp}} \sum_{i=1}^{D} |j\rangle|\epsilon^{\mathrm{PR}}_{\mathrm{com},j}\rangle|i\rangle|\epsilon^{\mathrm{PR}}_{i,j}\rangle|\vec{c}_i\rangle|f(\epsilon^{\mathrm{PR}}_{\mathrm{com},j}, \epsilon^{\mathrm{PR}}_{i,j}; \vec{c}_i)\rangle|0\rangle|0\rangle|0\rangle|0\rangle$$

$$\xrightarrow{7} \frac{1}{\sqrt{N_{\mathrm{samp}} D}} \sum_{j=1}^{N_{samp}} \sum_{i=1}^{D} |j\rangle|\epsilon^{\mathrm{PR}}_{\mathrm{com},j}\rangle|i\rangle|\epsilon^{\mathrm{PR}}_{i,j}\rangle|\vec{c}_i\rangle|f(\epsilon^{\mathrm{PR}}_{\mathrm{com},j}, \epsilon^{\mathrm{PR}}_{i,j}; \vec{c}_i)\rangle\left(\sqrt{1 - f(\epsilon^{\mathrm{PR}}_{\mathrm{com},j}, \epsilon^{\mathrm{PR}}_{i,j}; \vec{c}_i)}\,|0\rangle + \sqrt{f(\epsilon^{\mathrm{PR}}_{\mathrm{com},j}, \epsilon^{\mathrm{PR}}_{i,j}; \vec{c}_i)}\,|1\rangle\right)|0\rangle|0\rangle|0\rangle$$

$$
\begin{aligned}
&=: \frac{1}{\sqrt{N_{\mathrm{samp}}}} \sum_{j=1}^{N_{samp}} |j\rangle |\epsilon^{\mathrm{PR}}_{\mathrm{com},j}\rangle \left(\sqrt{1-S_j}\,|\Psi_0^{(j)}\rangle + \sqrt{S_j}\,|\Psi_1^{(j)}\rangle \right) |0\rangle |0\rangle |0\rangle \\
&\xrightarrow{8} \frac{1}{\sqrt{2N_{\mathrm{samp}}}} \sum_{j=1}^{N_{samp}} |j\rangle |\epsilon^{\mathrm{PR}}_{\mathrm{com},j}\rangle \left(|\Psi_+^{(j)}\rangle |\phi_M(\theta_j)\rangle + |\Psi_-^{(j)}\rangle |\phi_M(1-\theta_j)\rangle \right) |0\rangle |0\rangle \\
&= \frac{1}{\sqrt{2N_{\mathrm{samp}}}} \sum_{j=1}^{N_{samp}} \sum_{\tilde{\theta}\in I_M} |j\rangle |\epsilon^{\mathrm{PR}}_{\mathrm{com},j}\rangle \left(\langle\tilde{\theta}|\phi_M(\theta_j)\rangle |\Psi_+^{(j)}\rangle |\tilde{\theta}\rangle + \langle 1-\tilde{\theta}|\phi_M(1-\theta_j)\rangle |\Psi_-^{(j)}\rangle |1-\tilde{\theta}\rangle \right) |0\rangle |0\rangle \\
&\xrightarrow{9} \frac{1}{\sqrt{2N_{\mathrm{samp}}}} \sum_{j=1}^{N_{samp}} \sum_{\tilde{\theta}\in I_M} |j\rangle |\epsilon^{\mathrm{PR}}_{\mathrm{com},j}\rangle \left(\langle\tilde{\theta}|\phi_M(\theta_j)\rangle |\Psi_+^{(j)}\rangle |\tilde{\theta}\rangle + \langle 1-\tilde{\theta}|\phi_M(1-\theta_j)\rangle |\Psi_-^{(j)}\rangle |1-\tilde{\theta}\rangle \right) |g(D\sin^2(\tilde{\theta}\pi))\rangle |0\rangle \\
&\xrightarrow{10} \frac{1}{\sqrt{2N_{\mathrm{samp}}}} \sum_{j=1}^{N_{samp}} \sum_{\tilde{\theta}\in I_M} |j\rangle |\epsilon^{\mathrm{PR}}_{\mathrm{com},j}\rangle \left(\langle\tilde{\theta}|\phi_M(\theta_j)\rangle |\Psi_+^{(j)}\rangle |\tilde{\theta}\rangle + \langle 1-\tilde{\theta}|\phi_M(1-\theta_j)\rangle |\Psi_-^{(j)}\rangle |1-\tilde{\theta}\rangle \right) |g(D\sin^2(\tilde{\theta}\pi))\rangle \left(\sqrt{1-\tilde{g}(\tilde{\theta})}\,|0\rangle + \sqrt{\tilde{g}(\tilde{\theta})}\,|1\rangle \right),
\end{aligned}
\tag{15}
$$

where $I_M := \{0/M, 1/M, ..., (M-1)/M\}$, $M = 2^{n_M}$, n_M is the qubit number of R_{ctr1} and

$$
\begin{aligned}
|\Psi_0^{(j)}\rangle &:= \frac{1}{\sqrt{D}\sqrt{1-S_j}} \sum_{i=1}^{D} \sqrt{1-f(\epsilon^{\mathrm{PR}}_{\mathrm{com},j}, \epsilon^{\mathrm{PR}}_{i,j}; \vec{c}_i)}\,|i\rangle |\epsilon^{\mathrm{PR}}_{i,j}\rangle |\vec{c}_i\rangle |f(\epsilon^{\mathrm{PR}}_{\mathrm{com},j}, \epsilon^{\mathrm{PR}}_{i,j}; \vec{c}_i)\rangle |0\rangle, \\
|\Psi_1^{(j)}\rangle &:= \frac{1}{\sqrt{D}\sqrt{S_j}} \sum_{i=1}^{D} \sqrt{f(\epsilon^{\mathrm{PR}}_{\mathrm{com},j}, \epsilon^{\mathrm{PR}}_{i,j}; \vec{c}_i)}\,|i\rangle |\epsilon^{\mathrm{PR}}_{i,j}\rangle |\vec{c}_i\rangle |f(\epsilon^{\mathrm{PR}}_{\mathrm{com},j}, \epsilon^{\mathrm{PR}}_{i,j}; \vec{c}_i)\rangle |1\rangle, \\
|\Psi_\pm^{(j)}\rangle &:= \frac{1}{\sqrt{2}} \left(|\Psi_1^{(j)}\rangle \pm i\,|\Psi_0^{(j)}\rangle \right),
\end{aligned}
\tag{16}
$$

are the states in the tensor product space of $R_{\mathrm{dim}}, R_{\mathrm{ind}}, R_{\vec{c}}, R_f$ and $R_{\mathrm{ph},f}$. In (15), we omit R_{ctr2} since it is used only in the step 11. In the final state in (15), the probability of observing 1 on $R_{\mathrm{ph},g}$ is

$$
p_1 = \frac{1}{N_{\mathrm{samp}}} \sum_{j=1}^{N_{samp}} \sum_{\tilde{\theta}\in I_M} G(\tilde{\theta}; \theta_j, M)\tilde{g}(\tilde{\theta}), \tag{17}
$$

where G is defined as (9). Since $G(\tilde{\theta}; \theta_j, M)$ has a sharp peak around $\tilde{\theta} = \theta_j$, p_1 is nearly equal to E_{samp}. We will discuss the error in section III C.

3. *Comparison with the previous method*

For completeness, in [1], we outline the calculation procedure in the previous method, in which we use not the inner QAE but the simple iteration for the repeated calculation in the integrand, although we omit it in this version. For the detail, see [1].

B. The parts of the circuit

We here consider how to implement the component circuits listed in section III A.

- U_f

  This depends on the problems, so we here simply assume that it is implementable. In section IV, we consider its implementation for a concrete problem, that is, credit portfolio risk measurement.

- $U_{\epsilon_{\mathrm{com}}}, U_\epsilon$

  We here assume that PRNs obey the uniform distribution in [0, 1], as usual. There are various ways to transform a uniform random number x to a random number y which obeys the desired distribution. One is the inverse sampling method. That is, we can transform x as $y = \Phi^{-1}(x)$, where Φ^{-1} is the inverse of the cumulative distribution function (CDF) for the desired distribution. In [14], the quantum circuit to calculate Φ^{-1}_{SN}, the inverse CDF for the standard normal distribution, is presented. It is based on the piecewise polynomial approximation of Φ^{-1}_{SN} presented in [25]. We expect that the inverse CDFs for other distributions are also implemented in the similar way.

- U_P, U_J

  Every PRN sequence has an explicit recursion formula. Besides, for many widely-used PRN sequences, the simple formula to make the sequence jump to the desired position is explicitly given. We can construct

quantum circuits corresponding these formulae. Especially, in Section IV, we will discuss how to construct, taking a concrete PRN generator, PCG, as an example.

- $U_{\vec{c}}$

  If we can use a quantum random access memory (qRAM)[26], we can implement $U_{\vec{c}}$ trivially. Here, a qRAM is a quantum realization of associative data structure. It refers to an index i on a register and creates the state $|d_i\rangle$ which corresponds to the data d_i associated with i on another register. That is, it performs the following operation: $|i\rangle|0\rangle \mapsto |i\rangle|d_i\rangle$. Hereafter, we simply assume its availability.

C. Error and complexity

Sometimes, it is roughly said that in the QAE-based MC method the number of repeated calculations of the integrand sufficient for the tolerance error δ is $\sim \delta^{-1}$. On the basis of such a rough estimation, let us clarify the situation where the new method we propose is more advantageous than previous one.

In the current problem, calculation of $f(\epsilon^{\mathrm{PR}}_{\mathrm{com},j}, \epsilon^{\mathrm{PR}}_{i,j}; \vec{c}_i)$ is the most frequent procedure, so we focus on the number N_f of this calculation and its relation to the error. Here and hereafter, the word *calculation of f* means the repeated block in calculation of the sum of f's and therefore includes some operations in addition to calculating f itself. More specifically, in the new method, calculation of f corresponds to Q in the inner QAE, or, in other words, the steps 3 to 7 in the calculation flow presented in Section III A 2. On the other hand, in the previous method, calculation of f consists of (i) a progress of PRN, (ii) a conversion of RN from uniform to standard normal (iii) a calculation of f, (iv) an addition of f and (v) uncomputation of (ii) and (iii).

In the previous method, because we calculate the sum of f's in (13) by sequential calculations and additions of f and use QAE only for the sum over the sample index j, it is necessary to take

$$N_{f,\mathrm{prev}} \sim D\delta^{-1} \tag{18}$$

for the tolerance error δ. Here, the subscript 'prev' means that the expression is for the previous method. Note that the sequential evaluation of the sum of f's causes no error. On the other hand, in the new method where the nested QAE is used, requiring that the error is at most δ in each QAE leads to

$$N_{f,\mathrm{new}} \sim \delta^{-2}, \tag{19}$$

where the subscript 'new' means that the expression is for the new method. Therefore, comparing $N_{f,\mathrm{prev}}$ and $N_{f,\mathrm{new}}$, we see that the new method reduces N_f if the inverse of the tolerance is smaller than the dimension of the integration, that is,

$$\delta^{-1} \lesssim D. \tag{20}$$

The above estimation is illustrative but not strict since the result of the inner QAE is output as the superposition of the states, which correspond to the values distributing around the true value of the sum of f's. Let us evaluate the error by considering this distribution. Here, we assume that g is smooth, since in practical uses of MC the integrand is at least piecewise smooth and a finite number of points where g is non-smooth do not affect the integral in most cases. Considering the fact that $G(\tilde{\theta}; \theta_j, M)$ has a sharp peak around θ_j, we approximate $\tilde{g}(\tilde{\theta}) = g(D\sin^2(\tilde{\theta}\pi))$ as the first degree Taylor expansion around $\tilde{\theta} = \theta_j$:

$$\tilde{g}(\tilde{\theta}) \simeq g(DS_j) + Dg'(DS_j)(\sin^2(\tilde{\theta}\pi) - S_j), \tag{21}$$

where we used $\sin^2(\theta_j\pi) = S_j$. Using this and $\sum_{\tilde{\theta}\in I_M} G(\tilde{\theta}; \theta_j, M) = 1$, p_1 becomes

$$p_1 \simeq \frac{1}{N_{\mathrm{samp}}} \sum_{j=1}^{N_{\mathrm{samp}}} g(DS_j)\Big(1 + \Delta(D, S_j, M)\Big), \tag{22}$$

where the error term $\Delta(D, S_j, M)$ is defined as

$$\Delta(D, S_j, M) := \frac{Dg'(DS_j)}{g(DS_j)} \sum_{\tilde{\theta}\in I_M} G(\tilde{\theta}; \theta_j, M)\Big(\sin^2(\tilde{\theta}\pi) - \sin^2(\theta_j\pi)\Big). \tag{23}$$

As shown in the appendix of [1],

$$|\Delta(D, S_j, M)| < \frac{DS_j|g'(DS_j)|}{g(DS_j)} \frac{1/M}{S_j} + O\left(\frac{1}{M^2}\right). \tag{24}$$

(24) reasonably means the following. In the usual situation where $DS_j|g'(DS_j)|/g(DS_j) \sim 1$, which means that the change of the argument of g by $O(1)$ factor leads to the change of g by $O(1)$ factor, the deviation of p_1 from E_{samp} due to the inner QAE is negligible if $1/M$ is small compared with S_j. $1/M \ll S_j$ can be rephrased that $R_{\mathrm{ctr},1}$, the output register for the inner QAE, has the large number of qubits enough to precisely estimate θ_j, or equivalently, S_j. In summary, it is required that

$$M > (l\delta_{\mathrm{rel}})^{-1}, \tag{25}$$

where l is the typical scale of S_j, and δ_{rel} is the tolerance relative error on $g(DS_j)$, and so the number $N_{f,\mathrm{QAE1}}$ of calculations of f in the inner QAE, which is related to M as $N_{f,\mathrm{QAE1}} \simeq M$, is at least $(l\delta_{\mathrm{rel}})^{-1}$. Therefore, if

$$(l\delta_{\mathrm{rel}})^{-1} < D, \tag{26}$$

the new method reduces the number of calculations of f by a factor

$$\frac{(l\delta_{\mathrm{rel}})^{-1}}{D}. \tag{27}$$

We here make an important comment. Although the new method can reduce the number of calculations of f, the total calculation time might not necessarily decrease. This is because the steps in calculating f are different between the previous and new methods. In the sequential calculation of f in the previous method, we progress the PRN sequence step

by step. On the other hand, in the new method, we make the PRN sequence jump to the specified position to get a RN input to f. Usually, the jump takes a much larger computational cost than the progress. If we write the times for *one* calculation of f in the previous and new methods as $T_{\text{one,prev}}$ and $T_{\text{one,new}}$ respectively, the ratio of the total computational time in the new method to that in the previous method is

$$\frac{T_{\text{one,new}}}{T_{\text{one,prev}}} \frac{(l\delta_{\text{rel}})^{-1}}{D}. \tag{28}$$

In section IV, taking a concrete problem, credit portfolio risk measurement, and a concrete PRN generator, PCG, we will discuss the above point more rigorously and estimate the extent of computational time reduction by the new method.

IV. EXAMPLE: CREDIT PORTFOLIO RISK MEASUREMENT WITH PCG

In this section, we consider credit portfolio risk measurement as an example problem where the new method can be applied, taking PCG[23] as a concrete PRN generator. First, we briefly explain the outlines of credit portfolio risk measurement and PCG, and then estimate the extent of complexity reduction.

A. Credit portfolio risk measurement

One of the representative problems to which MC is often applied in finance is credit portfolio risk measurement. Each bank has a credit portfolio, that is, a collection of many loans or debts, which is exposed to risks of defaults of obligors. Banks evaluate such credit risks by some *risk measures*, which correspond to some kinds of estimation of the loss by defaults. The major ones are the value-at-risk (VaR), the percentile point (say, 99%) of loss distribution, and the conditional VaR (CVaR), the expectation value of loss under the condition that it exceeds the VaR. Such quantities are usually calculated by some mathematical model, for example the Merton model[27], in combination with MC. The calculation in the Merton model with MC on a quantum computer has already been considered in [7, 8]. For the details of the model and its implementation to a quantum computer, we here only refer to these papers. The point we should note here is that this problem is actually in the scope of this paper. That is, the integrand can be written as $g(L)$, where L is the random loss and the function g is set according to the type of the risk measure. L is calculated as

$$\begin{aligned} L &= \sum_{i=1}^{N_{\text{obl}}} f(\epsilon_{\text{com}}, \epsilon_i; E_i, \alpha_i, z_i) \\ f(\epsilon_{\text{com}}, \epsilon_i; E_i, \alpha_i, z_i) &= E_i \Theta(Z_i, z_i) \\ Z_i &= \alpha_i \epsilon_{\text{com}} + \sqrt{1-\alpha_i^2}\epsilon_i. \end{aligned} \tag{29}$$

Here, the meanings of the symbols are as follows. $\Theta(x, y)$ is the indicator function, which takes 1 if $x < y$ or 0 otherwise. N_{obl} is the number of the obligors. E_i is the exposure of the i th obligor. Note that it must be normalized so that $E_i \leq 1$. For example, we may divide exposures by the largest one. α_i, z_i are the model parameters for the i th obligor; see [8] for the detail. In addition to a common RN ϵ_{com}, we generate one RN ϵ_i for the i-th obligor to determine whether he defaults or not, which means the total number of RNs required to get one sample value of the loss is $N_{\text{obl}} + 1$. For VaR, g is taken as

$$g(L) = \Theta(L_\alpha, L), \tag{30}$$

that is, we can search (e.g. binary search) L_α satisfying $E[g(L)] = \alpha$, which means L_α is the $(1-\alpha)$-percentile point of the loss. For CVaR, we take

$$g(L) = CL\Theta(L_\alpha, L), \tag{31}$$

where the VaR L_α is predetermined and C is a normalization factor to make $g \leq 1$. As a whole, we can see that the integrand form matches (11).

B. PCG

Reference [8] picked up PCG[23] as a PRN generator which can be implemented in a quantum circuit. PCG is the combination of linear congruential generator (LCG) and permutation of bit string. The n-th element of a PCG sequence x_n is recursively defined as follows:

$$\begin{cases} \tilde{x}_{i+1} = (a\tilde{x}_i + c) \mod m \\ x_i = f^{\text{perm}}(\tilde{x}_i), \end{cases} \tag{32}$$

where $\tilde{x}_i$ is the background LCG sequence, $a, c, m, \tilde{x}_0$ are integer parameters satisfying $a > 0, c \geq 0, m > 0, 0 \leq \tilde{x}_0 < m$ and f^{perm} is permutation of a bit string, for which [23] presented some patterns. Note that we can make LCG, and therefore PCG too, jump to the specified position by the following formula

$$\tilde{x}_i = \left(a^i \tilde{x}_0 + \frac{c(a^i - 1)}{a - 1} \right) \mod m. \tag{33}$$

For the further details of PCG, consult [23].

Thanks to the above jump formula, we can implement the jump operator U_J for PCG. In fact, we have already presented the circuit U_J for such a jump in [8], along with U_P for the recursion formula (32)[6].

C. Reduction of complexity

As discussed in section III C, the new method reduces the number of calculations of f if (26) is satisfied. However, the

[6] Note that, in [8], U_P and U_J are represented by different symbols, P_{PRN} and J_{PRN}, respectively.

total complexity might not necessarily decrease, since the new method replaces the progress of the PRN sequence in the previous method with the jump, a more costly operation.

Considering this, let us estimate the extent of computational time reduction by the new method in credit portfolio risk measurement with PCG. At first, we estimate complexity of one calculation of f, which is repeated most, in the two methods. We here take T-count as a measure of computational time cost. T-count is the number of T-gates used in a given quantum circuit. Since the T-gate is expected to be most time-consuming in the *Clifford+T* gate set[28], a widely-considered universal gate set, T-count is a widely-used metric of computational time cost. Besides, we make the following assumptions on numbers of digits for various numbers:

- We use the n_{PRN}-bit PCG, and therefore so is the background LCG. We use only top n_{dig} digits for calculation, since lower bits have poorer statistical properties[23].
- For numerical numbers, we use n_{dig}-bit fixed-point numbers.

Then, we can estimate as follows.

(1) the previous method

Among operations constituting calculation of f in the previous method, it is sufficient to consider the following ones, which are more costly than others:

- a progress of PCG

  As shown in (32), this consists of a modular multiplication, a modular addition and a permutation. As discussed in [14], the dominant contribution to complexity comes from a modular multiplication. If we perform this in the self-updating way in order to avoid adding qubits at every multiplication, we have to combine two non-self-updating modular multiplications into self-updating one. As a result, the T-count is $140n_{\mathrm{PRN}}^2$ as estimated in [14][7].

- a RN conversion and uncomputation

  As mentioned above, we use the inverse sampling method combined with the piecewise polynomial approximation of Φ_{SN}^{-1}[25]. According to the estimation in [14], one conversion costs T-count of $105n_{\mathrm{dig}}^2 + 28n_{\mathrm{dig}}n_{\mathrm{ICDF}}$ and the total T-count is the double of it. Here, n_{ICDF} is the number of the intervals in the piecewise approximation.

Calculation of f makes only subdominant contributions to complexity, since it contains non-modular additions and multiplications, which is less costly than modular ones. Similarly, increment of R_{count} and adding f are also subdominant. Besides, we assume that cost of loading/unloading $\vec{c}_i$ is subdominant. Actually, a qRAM is designed so that only $O(n)$ quantum logic gates are activated while a record is loaded from a qRAM storing 2^n records[26]. In the current case, loading parameters for an obligor requires activation of $O(n_{\mathrm{obl}})$ gates.

In total, T-count for a calculation of f is

$$T_{\mathrm{one,prev}} \simeq 140n_{\mathrm{PRN}}^2 + 210n_{\mathrm{dig}}^2 + 56n_{\mathrm{dig}}n_{\mathrm{ICDF}}. \quad (34)$$

(2) the new method

In this case, calculation of f is equivalent to Q in the inner QAE. Among the operations in it, the dominant contributors to complexity are the following:

- two jumps of PCG

  Here, "two" is because Q contains A and its inverse. As shown in (33), a jump contains a modular exponentiation and this makes the dominant contribution to complexity. A modular exponentiation can be constructed as $2n_{\mathrm{exp}}$ modular multiplications, where n_{exp} is the number of digit of the exponent[33]. From (12), we see that the exponent is now $(j-1)(N_{\mathrm{obl}}+1)+i+1$, since the dimension D is now N_{obl}. Here, $0 \le i \le N_{\mathrm{obl}}, 0 \le j \le N_{\mathrm{samp}}$. Therefore, the exponent can be expressed by $n_{\mathrm{samp}} + n_{\mathrm{obl}}$ bits, where for simplicity we assume that N_{samp} and N_{obl} are now powers of two: $N_{\mathrm{samp}} = 2^{n_{\mathrm{samp}}}, N_{\mathrm{obl}} = 2^{n_{\mathrm{obl}}}$. As a result, T-count for a jump is that for a modular multiplication times $2(n_{\mathrm{samp}}+n_{\mathrm{obl}})$, that is, $140(n_{\mathrm{samp}}+n_{\mathrm{obl}})n_{\mathrm{PRN}}^2$. Two jumps cost doubly.

- two conversions of RN from uniform to standard normal

  Same as in the previous method.

Other operations are subdominant for complexity:

- Controlled S_0

  This is equivalent to a multiply-controlled Toffoli gate. It has T-count linear with respect to the number of the control qubits[29, 30].

- Controlled S_1

  This is equivalent to just a controlled Z gate.

- f and loading/unloading $\vec{c}_i$

  Same as in the previous method.

- controlled rotation

  This has T-count linear with respect to the logarithm of the required accuracy[7, 31, 32].

In total, T-count for a calculation of f is

$$T_{\mathrm{one,new}} \simeq 280(n_{\mathrm{samp}}+n_{\mathrm{obl}})n_{\mathrm{PRN}}^2 + 210n_{\mathrm{dig}}^2 + 56n_{\mathrm{dig}}n_{\mathrm{ICDF}}. \quad (35)$$

Then, let us take a typical setting in practical use and compare (34) and (35) in the setting. As typical values, we here set $n_{\mathrm{dig}} = 16$, $n_{\mathrm{PRN}} = 64$ [23], $n_{\mathrm{ICDF}} = 109$ [25] and $n_{\mathrm{samp}} =$

[7] In this paper, we take only the leading term for T-count, as in [14]

$n_{\rm obl} = 20$, which correspond to $N_{\rm samp} = N_{\rm obl} = 2^{20} \approx 10^6$. For these values, (34) and (35) become

$$T_{\rm one,prev} \simeq 7.2 \times 10^5, T_{\rm one,new} \simeq 4.6 \times 10^7, \tag{36}$$

respectively, and the ratio is

$$\frac{T_{\rm one,new}}{T_{\rm one,prev}} = 64. \tag{37}$$

Finally, we can compare the total T-counts in the whole processes of the previous and new methods. Combining (28) and (37), we obtain the ratio of the total T-counts as

$$64\frac{(l\delta_{\rm rel})^{-1}}{D}. \tag{38}$$

This means that, if $(l\delta_{\rm rel})^{-1}/D$, the reduction ratio of the number of queries to calculations of f by the new method is smaller than $1/64$, it is more beneficial than the previous one. Then, let us consider a typical setting in credit portfolio risk measurement. We assume that $l = 10^{-2}$, which roughly corresponds to the situation where the total loss is 1% of the total exposure, and $\delta_{\rm rel} = 10^{-2}$. Besides, we are now taking $D = N_{\rm obl} = 2^{20}$. These lead to the query number reduction ratio $(l\delta_{\rm rel})^{-1}/D \simeq 10^{-2}$, and finally the total T-count reduction ratio is about 0.64. That is, we can reduce the total computational time by several tens of percent.

V. SUMMARY

In this paper, we present a version of the quantum method for MC using PRNs. The use of PRNs was originally proposed in [8] for the sake of reduction of qubits in extremely high-dimensional integrations such as credit portfolio risk measurement. As an extension of this, the method proposed in this paper can reduce more complexity. That is, in the case where the integrand has the separable form like (11), the method can reduce the number of repeated calculations over the separated terms compared with the previous method. The key point is that if we use PRN, we can calculate f, a component of the integrand, as a function of the indices i and j, which specify the RN and the sample respectively. This makes it possible to compute f's in quantum parallelism, not sequentially as in the previous method. Combined with QAE, this leads to the reduction of the number of calculations of f, if the dimension (or equivalently the number of RNs) and the tolerance are large enough. We should note that the new method can increase the time for one calculation of f since it replaces the progress of the PRN sequence with the jump, which is more costly. Therefore, the new method might not reduce the total computational time even if the query complexity decreases. Nonetheless, taking T-count as a metric of computational time cost, we saw that the new method actually reduces the total T-count in a typical case of credit portfolio risk measurement with PCG, as shown in Sec. IV

In the original proposal [8], sequential computability of PRNs was the key feature to avoid generation of RNs on different registers and reduce qubits. In this paper, another feature of PRN has been focused. That is, since it is a deterministic sequence whose element can be calculated as a function of the index, we can compute PRNs in quantum parallelism and create a superposition of them. In future works, we will explore the possibility to utilize such a feature in other ways and make quantum algorithm for MC more efficient.

[1] K. Kaneko et al., arXiv:2011.02165
[2] A. Montanaro, Proc. Roy. Soc. Ser. A, 471, 2181 (2015)
[3] Y. Suzuki et. al., Quantum Information Processing, 19, 75 (2020)
[4] J. C. Hull, "Options, Futures, and Other Derivatives", Prentice Hall (2012)
[5] P. Glasserman, "Monte Carlo Methods in Financial Engineering", Springer (2003)
[6] S. Woerner and D. J. Egger, npj Quantum Information, 5(1):1–8 (2019)
[7] D. J. Egger et al., arXiv:1907.03044
[8] K. Miyamoto and K. Shiohara, Phys. Rev. A 102, 022424 (2020)
[9] P. Rebentrost et al., Phys. Rev. A, 98(2), 022321 (2018)
[10] N. Stamatopoulos et al., Quantum 4, 291 (2020)
[11] A. Martin et al., arXiv:1904.05803
[12] S. Ramos-Calderer et al., arXiv:1912.01618
[13] A. C. Vazquez and S. Woerner, arXiv:2005.07711
[14] K. Kaneko et al. arXiv:2007.01467
[15] R. Orus et al., Reviews in Physics 4, 100028 (2019)
[16] D. J. Egger et al., IEEE Transactions on Quantum Engineering, 1, 3101724 (2020)
[17] A. Bouland et al., arXiv:2011.06492
[18] G. Brassard et. al., Contemporary Mathematics, 305, 53 (2002)
[19] S. Aaronson and P. Rall, Symposium on Simplicity in Algorithms, 24–32, SIAM (2020)
[20] D. Grinko et al., arXiv:1912.05559
[21] K. Nakaji, arXiv:2003.02417
[22] T. Tanaka, et al., arXiv:2006.16223
[23] M. E. O'Neill, Harvey Mudd College Computer Science Department Tachnical Report (2014); http://www.pcg-random.org/
[24] P. L'Ecuyer and R. Simard, ACM Transactions on Mathemathical Software 33, 4, 22 (2007)
[25] W. Hörmann and J. Leydold, ACM Transactions on Modeling and Computer Simulation 13(4):347, (2003)
[26] V. Giovannetti et al., Phys. Rev. A78, 052310 (2008)
[27] R. C. Merton, J. Finance, 29, 449 (1974)
[28] X. Zhou et al., Phys. Rev. A62, 052316 (2000)
[29] P. Selinger, Phys. Rev. A 87, 042302 (2013)
[30] D. Maslov, Phys. Rev. A 93, 022311 (2016)
[31] M. Amy, D. Maslov, and M. Mosca, IEEE Trans. CAD 33(10), 1476 (2014)
[32] V. Kliuchnikov et al., IEEE Transactions on Computers, 65, 1, 161 (2016)
[33] V. Vedral et al., Phys. Rev. A 54, 147 (1996)