International Journal of Modern Physics C,

QUANTUM-STATISTICAL SIMULATIONS FOR QUANTUM CIRCUITS

KURT FISCHER†, HANS-GEORG MATUTTIS*,
NOBUYASU ITO*, MASAMICHI ISHIKAWA†
*Department of Applied Physics, School of Engineering,
The University of Tokyo, Bunkyo-ku, Tokyo 113, Japan

†Mitsubishi Research Institute, Frontier Science Institute,
Otemachi 2-3-6, Chiyoda-Ku , Tokyo 100-8141, Japan

Received (received date)
Revised (revised date)

Keywords: Quantum information, quantum communication, auxiliary field method

Abstract

Using a Hubbard-Stratonovic like decomposition technique, we implemented simulations for the quantum circuits of Simon's algorithm for the detection of the periodicity of a function and Shor's algorithm for the factoring of prime numbers on a classical computer. Our approach has the advantage that the dimension of the problem does not grow exponentially with the number of qubits.

1. Introduction

1.1. *Quantum versus classical computation*

Feynman[1] suggested that because the Hilbert space of a L-particle quantum system increases exponentially with L, a classical computer would need resources growing exponentially in L, in order to simulate such a system. For quantum computers "trace out" all possible paths from the initial to the final state in Hilbert space while classical computers allow only for one path at a time. This is called sometime "massive parallelism" of the quantum circuit. If a quantum algorithm is used to compute a arithmetic function, for example, then the function (or its representation as operator in Hilbert space) can act on all input values simultaneously; but we can read out (= measure) only one value at a time; in addition, we cannot chose the output value. However, any interacting system can be mapped onto a noninteracting system by introducing some additional classical auxiliary variables[2]. Thus we can in principle simulate with arbitrary accuracy any observable with the help of the probabilistic Monte Carlo method, but have no proof that such an algorithm may converge fast enough in order to speed up the calculation sufficiently. It remains unclear up to the present whether a quantum computer is in principle faster then a classical probabilistic computer. Hence a numerical experiment has to decide. In this paper, our goal is to simulate Shor's algorithm with a Monte Carlo method.

1.2. *The logical gates and their representation in Hilbert space*

Every quantum bit $\langle x|$, with $x = 0, 1$ is represented by a two dimensional vector $[1-x \quad x]$. In this notation, the NOT operation is represented as $\langle x| \longmapsto \langle x| \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$. For the product state of two quantum spins, the controlled not (CNOT) gate (symbol in Fig. 1) flips the second bit in the product state if and only if the first bit is one. We can represent this as

$$CNOT = H_2 \left(1 - 2n_1 n_2\right) H_2. \tag{0.1}$$

The Hadamard gate H_2 acts on the second bit and the number operator n_i the $i-$th bit,

$$H_i = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}, n_i = \begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix} S). \tag{0.2}$$

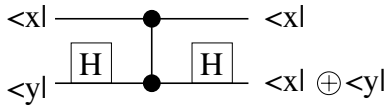

Figure 1: The CNOT-gate: The bullets at the vertices represent the number operator.

The number of gates necessary for representing a given quantum circuit can be reduced by the following consideration[3]: A basic theorem of computer science states that every computation can be reduced to a computation using NAND and the FAN-OUT gates only[4]. Both operations can be emulated in quantum circuits using the Toffoli gate, i.e. a controlled-controlled-NOT gate, see Fig. 2. The Toffoli gate is a three-qubit gate, and it flips the third bit if and only if the first two qubits are both one. The Toffoli gate can then be represented as [5]

$$CNOT = H_3 \left(1 - 2n_1 n_2 n_3\right) H_3, \tag{0.3}$$

with the Hadamard-operation H_3 acting on the third qubit.

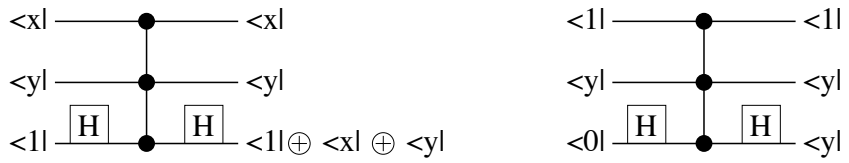

Figure 2: Toffoli gate as NAND and FAN OUT gate

1.3. *The discrete decomposition*

Using the representation of the CNOT and the Toffoli gate, we can split the gates into a sum of products of one-bit gates, where the sum can be performed by Monte Carlo techniques. This method is commonly called auxiliary field technique or Hubbard Stratonovich (HS) transformation in quantum Monte Carlo simulations. In contrast to quantum Monte Carlo simulations, where importance sampling is used on the system's partition function, our decomposition leads onto a simple sampling procedure.

A general consideration in setting up the HS-transformation is to to minimize the space of Monte Carlo configurations. Therefore, we allow only one random variable $\nu = 1, \ldots, n$ per CNOT gate, which leads us to the decomposition

$$1 - 2n_1 n_2 = \sum_{\nu=1}^{n} p_\nu \left(a_\nu + b_\nu n_1\right)\left(c_\nu + d_\nu n_2\right), \tag{0.4}$$

where a configuration ν is weighted with the probability p_ν. One possible decomposition is the discrete, symmetric decomposition analogous to the one introduced by Hirsch for Hubbard-type interactions[2], which for our case looks like

$$1 - 2n_1 n_2 = \frac{1}{2}\left(1 - \sqrt{2}n_1\right)\left(1 + \sqrt{2}n_2\right) + \frac{1}{2}\left(1 + \sqrt{2}n_1\right)\left(1 - \sqrt{2}n_2\right). \tag{0.5}$$

Here, the variables from Eq. (0.4) are $p_1 = p_2 = \frac{1}{2}$, $a_1 = a_2 = c_1 = c_2 = 1$, $b_1 = d_2 = -b_2 = -d_1 = \sqrt{2}$. We have used this decomposition in[6]. In this paper, we will here choose an asymmetric decomposition,

$$1 - 2n_1 n_2 = (1 - n_2) + n_2\left(1 - 2n_1\right), \tag{0.6}$$

with $p_1 = p_2 = 0.5$, $a_1 = 0$, $b_1 = 2$, $c_1 = 1$, $d_1 = -2$ and $a_2 = 2$, $b_2 = -2$, $c_2 = 1$, $d_2 = 0$. This decomposition is particularly suited for the Toffoli gate. In principle, the Toffoli gate could be replaced with six CNOT gates and some one-bit gates, which would result in 2^6 configurations per Toffoli gate. Only 2^2 configurations are need if the decomposition for the CNOT gate in Eq. (0.6) is inserted twice, so that

$$\begin{aligned} 1 - 2n_1 n_2 n_3 &= 1 - n_3 + n_3\left(1 - 2n_1 n_2\right) && (0.7)\\ &= \frac{1}{2}\left(2 - 2n_3\right) + \frac{1}{4}\left(2 - 2n_2\right)2n_3 + \frac{1}{4}\left(1 - 2n_1\right)4n_2 n_3. && (0.8) \end{aligned}$$

Higher order controlled-NOT gates can decomposed recursively in the same way,

$$1 - 2n_1 n_2 \ldots n_\nu = (1 - n_\nu) + n_\nu\left(1 - n_{\nu-1}\right) + \ldots + n_\nu n_{\nu-1} \ldots n_2\left(1 - 2n_1\right) \tag{0.9}$$

Hence, for multiply controlled NOT-gates, only real arithmetic is necessary. As many quantum circuits of interest depend on the usage of the quantum Fourier transform, which intrinsically uses complex phases, nevertheless some complex arithmetic has to be performed.

1.4. *The decomposition with the least variance*

As can be seen from the existence of many free parameters in our Eq. (0.4), we have a great freedom to choose the numerical values for the parameters of the decomposition. For the sake of a minimum variance in the Monte Carlo sampling process, one constraint is to minimize the variance of the probability distribution of the amplitudes, that is, their moments. It turns out that the moments depend on the structure of the logical circuit. However, each circuit consists of a product of the logical gates. Therefore we now describe and assess the contribution to the first and second moment $M_k, k = 1, 2$ resulting from the decomposition of a single CNOT gate. This moments should fulfill $M_k \geq 1$, so that they are a measure of the spreading of the Monte Carlo sampling. In Eq. (0.4), the number operator can assume the values $n_{1,2} = 0, 1$. For each of these four cases, we get the contribution from one of the n possible decompositions as

$$\alpha_\nu = a_\nu c_\nu, \beta_\nu = \left(a_\nu + b_\nu\right)c_\nu, \gamma_\nu = a_\nu\left(c_\nu + d_\nu\right), \text{or } \delta_\nu = \left(a_\nu + b_\nu\right)\left(c_\nu + d_\nu\right). \tag{0.10}$$

The boundary conditions which result from inserting $n_{1,2} = 0, 1$ into Eq. (0.4) are

$$\sum_{\nu=1}^{n} p_\nu \alpha_\nu = \sum_{\nu=1}^{n} p_\nu \beta_\nu = \sum_{\nu=1}^{n} p_\nu \gamma_\nu = 1, \quad \sum_{\nu=1}^{n} p_\nu \delta_\nu = -1, \quad \alpha_\nu \delta_\nu = \beta_\nu \gamma_\nu. \tag{0.11}$$

A natural measure for the moment is the aritmetic mean of the $k-$th powers of the absolute values of the four possible contributions,

$$M_k = \frac{1}{4} \sum_{\nu=1}^{n} p_\nu \left(|\alpha_\nu|^k + |\beta_\nu|^k + |\gamma_\nu|^k + |\delta_\nu|^k \right). \tag{0.12}$$

The triangle inequality

$$\sum_{\nu=1}^{n} p_\nu |\alpha_\nu| \geq \left| \sum_{\nu=1}^{n} p_\nu \alpha_\nu \right| \tag{0.13}$$

etc. yields

$$M_1 \geq 1. \tag{0.14}$$

Hence M_1 is a measure for how much the sampling will spread. We can see that the asymmetric decomposition in Eq. (0.6) fulfills $M_1 = 1$, so it is optimal with respect to the minimization of the fluctuations in the Monte Carlo sampling process. For the second moment M_2, we will prove that

$$M_2 \geq 2. \tag{0.15}$$

This can be seen as follows: Define the real-valued vector $\mathbf{e}$ with $e_\nu^2 = p_\nu$ and e_ν real, and the four vectors $\mathbf{x}, \mathbf{y}, \mathbf{z}, \mathbf{u}$ with

$$x_\nu = e_\nu \alpha_\nu, y_\nu = e_\nu \beta_\nu, z_\nu = e_\nu \gamma_\nu \text{ and } u_\nu = e_\nu \delta_\nu. \tag{0.16}$$

Hence the second moment is

$$M_2 = \frac{1}{4} \left(|\mathbf{x}|^2 + |\mathbf{y}|^2 + |\mathbf{z}|^2 + |\mathbf{y}|^2 \right). \tag{0.17}$$

The boundary conditions from Eq. (0.11) together with $\sum_{\nu=1}^{n} p_\nu = 1$ read now as

$$\mathbf{e}^2 = \mathbf{ex} = \mathbf{ey} = \mathbf{ez} = \mathbf{ez}^* = 1 \text{ and } \mathbf{eu} = \mathbf{eu}^* = -1 \tag{0.18}$$

where $\mathbf{z}^*$ is the complex conjugate of $\mathbf{z}$. ¿From the boundary condition $\alpha_\nu \delta_\nu = \beta_\nu \gamma_\nu$ follows

$$\mathbf{xu} = \mathbf{yz}. \tag{0.19}$$

We have now expressed all boundary conditions and the second moment itself in terms of the four complex vectors $\mathbf{x}, \mathbf{y}, \mathbf{z}, \mathbf{u}$. We apply the Cauchy-Schwartz inequality using Eqs. (0.18), (0.19), so that

$$8 = |(\mathbf{x} - \mathbf{u}^*)\, \mathbf{e}|^2 + |(\mathbf{y} + \mathbf{z}^*)\, \mathbf{e}|^2 \leq |(\mathbf{x} - \mathbf{u}^*)|^2\, \mathbf{e}^2 + |(\mathbf{y} + \mathbf{z}^*)|^2\, \mathbf{e}^2 = 4M_2, \tag{0.20}$$

i.e. $M_2 \geq 2$, what we wanted to show. Again, the decomposition from Eq. (0.6) fulfills $M_2 = 2$. It is straightforward (although tedious) to show that the decomposition in Eq. (0.6) up to symmetries is the only one fulfilling $M_1 = 1$ and $M_2 = 2$, and therefore it is the optimal decomposition.

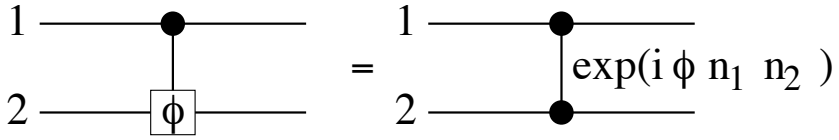

Figure 3: The controlled phase gate

1.5. *The quantum Fourier transform*

In the last section, we represented the elementary logical gates in terms of unitary operators. Just as for classical logical circuits on integrated circuits, every classical computable function can be implemented also on a quantum circuit. To perform the quantum Fourier transform, we need another type of gate, the controlled phase gate. We can write it as in Fig. 3.

The controlled phase gate in Fig. 3 is symmetric in the two qubits. It is a matter of convention to define the controlled phase gate by assigning the $\boxed{\phi}$ to the second qubit as in our case or to the first qubit. The quantum Fourier transform we can now represent as a product of controlled phase gates with the Hadamard gate. Let $\langle x| = \prod_{\nu=0}^{L-1} \langle x_\nu|$ denote the product state representing the binary representation of the L digit number $x = \sum_{\nu=0}^{L-1} x_\nu 2^\nu$. Then its Fourier transformation is

$$\langle x| \longmapsto \sum_{p=0}^{2^L-1} \exp\left(\frac{2\pi \mathrm{i}}{2^L} p x\right) \langle p| = \sum_{p=0}^{2^L-1} \exp\left(\frac{2\pi \mathrm{i}}{2^L} p \sum_{\nu=0}^{L-1} x_\nu 2^\nu\right). \tag{0.21}$$

This we can write as the product

$$\langle x| \longmapsto \prod_{\nu=0}^{L-1} \left(\langle 0| + \langle 1| \exp\left(\frac{2\pi \mathrm{i}}{2^L} p 2^\nu\right)\right). \tag{0.22}$$

Now, because $p = \sum_{w=0}^{L-1} p_w 2^w$, we have

$$\langle x| \longmapsto \prod_{\nu=0}^{L-1} \left(\langle 0| + \langle 1| \prod_{w=0}^{L-1} \exp\left(\frac{2\pi \mathrm{i}}{2^L} p_w 2^{\nu+w}\right)\right). \tag{0.23}$$

This means that we can implement the Fourier transformation using Hadamard gates $\langle 0| \longmapsto \left(\langle 0| + \langle 1|\right)/\sqrt{2}$ and controlled phase gates for pairs of bits ν, w with phases $2\pi 2^{\nu+w-L}$. In practice, the gates with exponentially small phases can be neglected[7], by performing an approximate Fourier transformation to the accuracy of m bits, with just the controlled phase gates with $L \geq \nu + w \geq L - m$. Here the upper limit comes from the fact that for $\nu + w \geq L$, the absolute value of the complex phase is unity and can be neglected.

2. Simon's algorithm

2.1. *The quantum circuit*

We will first concentrate on a simplified version of Simon's algorithm[8,9], which serves as preparatory work for the implementation of Shor's algorithm[10,11]. Simon's

algorithm can detect the period of a function operating in Z_2^L, i.e. the space of numbers which can be represented with L binary digits. Simon's algorithm uses only bit wise addition, without carry, which can be implemented using only the exclusive OR operation "$\oplus$". In Z_2^L, every element fulfills $x \oplus x = 0$. It is important that the periodic function is known to be 2:1; then f with period r, assumes a value y only twice, $y = f(x) = f(x \oplus y)$.

For two L-bit numbers with bits p_i, x_i with scalar $px = p_1x_1 + p_2x_2 + \ldots + p_Lx_L$, the Fourier transform is the Hadamard transformation

$$\langle x| \longmapsto \frac{1}{\sqrt{2}} \sum_{p=0}^{2^L-1} \langle p| \, (-1)^{px} = \prod_{i=1}^{L} \frac{\langle 0|_i + (-1)^{x_i} \langle 1|_i}{\sqrt{2}} \tag{0.24}$$

Simon's algorithm in its quantum version as described above first creates the equally weighted superposition of all states in the first register

$$\langle 0| \otimes \langle 0| \longmapsto \frac{1}{\sqrt{2^L}} \sum_{x=0}^{2^L-1} \langle x| \otimes \langle 0| \tag{0.25}$$

Then we apply the function f

$$\frac{1}{\sqrt{2^L}} \sum_{x=0}^{2^L-1} \langle x| \otimes \langle f(x)| \tag{0.26}$$

After that, we perform a measurement, where we obtain $y_0 = f(x_0) = f(x_0 + r)$ for some value x_0, because f is 2:1. Next we apply the Fourier transform to the first register,

$$2^{-L} \sum_p \left((-1)^{x_0 p} + (-1)^{(x_0+r)p} \right) \langle p| = 2^{1-L} \sum_p (-1)^{x_0 p} \frac{1 + (-1)^{rp}}{2} \langle p|. \tag{0.27}$$

Then we measure first register, and obtain some value p_0. The total transition amplitude A depends on the value of the first and second register

$$A = 2^{1-L} (-1)^{x_0 p_0} \frac{1 + (-1)^{rp_0}}{2} \delta, \tag{0.28}$$

where δ is one if and only if the chosen value of y_0 is a possible value of f, and otherwise zero. The overall sign depends on the unknown x_0, which is one of the two values for x yielding $f(x_0) = y_0$. In a simulation we would be able to predict the sign $(-1)^{x_0 p_0}$ only if we could efficiently invert the function f. Most important, the procedure selects a value of the first register such that

$$rp_0 \equiv 0 \bmod 2. \tag{0.29}$$

Now, sampling $\mathcal{O}(L)$ of such random values suffice to have with high probability a matrix P of maximal rank $L - 1$, so we can then determine the period r of the function, as the unique non-zero solution of

$$rP = 0. \tag{0.30}$$

In the space Z_2^L there are no nontrivial multiples of a nonzero element, so that Eq. (0.30) determines r. Observe further, that this quantum algorithm does not work

for a general function but we need a function which is $2:1$, or more generally a function which is $2m:1$ with a positive integer m.

2.2. *The Monte Carlo simulation*

Simon's algorithm is relatively simple to implement, because
1. It's arithmetic is real valued,
2. the Fourier transform is a one bit operation and
3. the arithmetic uses no carry bits.

2.2.1. *Measurement simulation*

The first problem is the measurement. We cannot measure, i.e. collapse the wave function of the multi-particle problem and measure the partial amplitudes. As we can only measure scalar products, we need a reasonable probability that we get nonzero overlap matrix elements between the initial state and the final state. If the simulation produces a zero scalar product, it could be that either second register was not a possible value of the function, or it could be an "odd" value p_0 in the first register such that

$$p_0 r \equiv 1 \bmod 2. \tag{0.31}$$

We cannot decide in advance the outcome of the measurement, because we have to use the transition amplitude between the given initial and chosen final state, in order to compute the Monte Carlo sum over the auxiliary field configurations. But the 2:1 property helps us here: We know in advance that the probability to find a nonzero value amplitude is 1:4, because in both registers just half of all possible values will occur. Therefore we can just select an arbitrary final state and perform 4 independent Monte Carlo simulations with different initial configurations.

2.3. *Sample functions*

2.3.1. *An elementary linear function*

If we use a linear periodic function with period r,

$$f(x) \equiv f(x+r) \equiv f(x) + f(r) \tag{0.32}$$

then it will be 2:1 if and only if it has rank $L-1$, because then there will be exactly one nonzero r with $f(r) = 0$.

A first, we give a simple example of a function f_1 using only CNOT gates, which puts the first bit to zero and leaves all the other bits unchanged. This means that if $(x_1, x_1, \ldots, x_L)$ and $(y_1, y_1, \ldots, y_L)$ denotes the first and second register, then the unitary representation in Hilbert space of Eq. (0.26) yields

$$x_1 \longmapsto x_1 \qquad y_1 \longmapsto y_1 \tag{0.33}$$

$$x_i \longmapsto x_i \qquad y_i \longmapsto y_i \oplus x_i, i = 2, \ldots L, \tag{0.34}$$

which can be realized using $L-1$ CNOT gates only. The bits $x_i, y_i, \quad i \geq 2$ are processed according to the diagram in Fig. 4.

In this simple example, a Monte Carlo simulation will yield exactly zero if the "wrong" final state is chosen, because there is no mixing between the bits in the initial state. Therefore our simulation will merely test if the nonzero amplitudes are recovered correctly, if we simply ignore the first bit. Using the HS-transformation (Eq. (0.5)) on the circuit given in Fig. 4 for the one-bit 2×2 transition matrices for the $i-$th $x-$bit with value p_i and an auxiliary variable $\tau \in \{0, 1\}$,

$$\frac{1}{2\sqrt{2^2}}(1 \quad 0) \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \begin{pmatrix} 1 & 0 \\ 0 & 1+(-1)^\tau\sqrt{2} \end{pmatrix} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \begin{pmatrix} 1-p_i \\ p_i \end{pmatrix} =$$

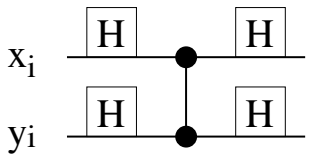

Figure 4: Basic operation for the $i-$th pair of bits of f_1.

$$\frac{1+(-1)_i^p\left(1+(-1)^\tau\sqrt{2}\right)}{4} \tag{0.35}$$

and likewise for the y bit, with $(-1)^\tau$ being replaced by $-(-1)^\tau$. This gives the amplitude

$$A=\prod_{i=2}^{n}\frac{1}{4}\sum_{\tau_i}\left[1+(-1)_i^p\left(1+(-1)^{\tau_i}\sqrt{2}\right)\right]\left[1+(-1)_i^y\left(1-(-1)^{\tau_i}\sqrt{2}\right)\right] \tag{0.36}$$

The above expression we can simplify if we consider the cases $p_i=y_i$ and $p_i\neq y_i$ separately, and sum over the auxiliary variables. This reproduces Eq. (0.28)

$$2^{L-1}A=(-1)^{\sum_i z_i} \tag{0.37}$$

where

$$z_i=\begin{cases}1 & \text{if } p_i=y_i=1,\\ 1 & \text{else.}\end{cases} \tag{0.38}$$

2.3.2. *Linear function with many gates*

To test the feasibility of a Monte Carlo simulation by numerical experiment, we constructed a more complicated linear function

$$\begin{gathered}f_2(x_1,x_2;x_3,\ldots,x_m;x_{m+1},\ldots,x_L)=\\ (x_1+x_2,x_1+x_2;x_1+x_3,\ldots,x_1+x_m;x_{m+1},\ldots,x_L)\end{gathered} \tag{0.39}$$

which corresponds to a stronger connected network. It has the period

$$r=(\underbrace{1,\ldots,1}_{\text{m times}},0,\ldots,0). \tag{0.40}$$

The possible values for $y=(y_1,y_2,\ldots)=f_2(x)$ are the ones which fulfill

$$y_1=y_2 \tag{0.41}$$

and hence $f_2(x_0)=y$ can be inverted (choosing the first bit to be zero) as

$$x_0=(0,y_1;y_3,\ldots,y_L). \tag{0.42}$$

The transition amplitude we can evaluate with Eq. (0.28) . The circuit consists of serially connected CNOT gates, which we can represent as phase gates with phase π like in Fig. 4. Internal Hadamard gates cancel. If we abbreviate

$$g(\tau)=1-(-1)^\tau\sqrt{2}, \tag{0.43}$$

the partial amplitudes are

$$\begin{aligned}
A(p_1) &= 1+(-1)^{p_1} g(-\tau_1)g(-\tau_3)\cdot\ldots\cdot g(-\tau_{2m-1}) \\
A(y_1) &= 1+(-1)^{y_1} g(\tau_1)g(\tau_2), \\
A(p_2) &= 1+(-1)^{p_2} g(-\tau_2)g(-\tau_4), \\
A(y_2) &= 1+(-1)^{y_2} g(\tau_3)g(\tau_4), \\
A(p_i) &= 1+(-1)^{p_i} g(-\tau_{2i}), \\
A(y_i) &= 1+(-1)^{y_i} g(\tau_{2i-1})g(\tau_{2i}), \quad i=3,\ldots,m, \\
A(p_i) &= 1+(-1)^{p_i} g(-\tau_{m+i}), \\
A(y_i) &= 1+(-1)^{y_i} g(\tau_{m+i}), \quad i=m+1,\ldots,L.
\end{aligned} \tag{0.44}$$

The total transition amplitude is then given by their product

$$A = 2^{1-L}(-1)^{x_0 p_0}\frac{1+(-1)^{r p_0}}{2}\delta\prod_{i=1}^{L}\frac{1}{4}\sum_{\tau_i} A(p_i)A(y_i). \tag{0.45}$$

2.3.3. *Asymmetric decomposition*

Quite generally, we need a certain non-commutativity in the quantum circuit in order to obtain significant information about the total sum if we only use a Monte Carlo summation. This we can see if we use for $f_1$the asymmetric decomposition of Eq. (0.6). If the first summand $1-n_2$ corresponds to $\tau_i = 0$, then the amplitude for the $i-$th bit is $(1-p_i)/2$, and for the second summand $p_i(-1)^{y_i}/2$. Hence for a given p, there is only one combination of τ_i such that the product amplitude does not vanish. This single combination fulfills again Eq. (0.37), because the factors $(-1)^{y_i}$ contributes only if $p_i = 1$. In short, all contribution except one are zero, and the Monte Carlo summation is inefficient, because the non-vanishing contribution is found only with probability 2^{1-L} . The reason is that that circuit is not "deep" so that it could contain non-commuting gates.

3. The structure of the Shor algorithm

Let us consider the problem of factoring a large odd number N, which has the two prime factors p and q,

$$N = pq, \tag{0.46}$$

where we do not know neither p or q, but only N. We would like get either p or q with high efficiency. If we choose a number $0 \le a < N$ at random, we can efficiently evaluate their greatest common divisor $\gcd(a,N) \in \{1,p,q\}$ with Euclid's algorithm. If it is either p or q, we are done. Otherwise we will try to get the period of the function

$$f : x \longmapsto a^x \equiv a^{x+r} \bmod N \tag{0.47}$$

i.e. the smallest non-zero integer r for which

$$a^r \equiv 1 \bmod N. \tag{0.48}$$

Then a well-known result of number theory (e.g. Ref.14, theorem A4.13) tells us that with probability of at least 75%, r is even, and that

$$\gcd\left(a^{r/2}-1, N\right) = p \quad or \quad q. \tag{0.49}$$

It remains to find the period r. Because N and r are large, for a number with e.g. 200 binary digits, by trial and error we would need of the order of $2^{200} \approx 10^{60}$

attempts. Up to today, no efficient algorithm is known to solve this problem using resources growing only polynomially in $\log(N)$ on a classical computer[12], but Shor succeeded in proposing a quantum algorithm which we will explain in the following sections and outline how it can be implemented with our Monte Carlo procedure.

With a number of L binary digits, $m_\nu = 0, 1$ the product state $\langle m| = \prod_{\nu=0}^{L-1} \langle m_\nu|$ can be associated. As input state for Shor's algorithm, an equal weight superposition of all possible input states,

$$\frac{1}{\sqrt{2^L}} \sum_{x=0}^{2^L-1} \langle x| = \prod_{\nu=0}^{L-1} \frac{1}{\sqrt{2}} \left(\langle 0| + \langle 1|\right) \tag{0.50}$$

How large L is depending on N we explain in the next section. In the next step, we have to create a circuit for the modular exponentiation

$$\langle x| \otimes \langle 0| \longmapsto \langle x| \otimes \langle a^x \bmod N|. \tag{0.51}$$

This logical logical network can also be used to compute f for the superposition in Eq. (0.50) as well:

$$\frac{1}{\sqrt{2^L}} \sum_{x=0}^{2^L-1} \langle x| \otimes \langle 0| \longmapsto \frac{1}{\sqrt{2^L}} \sum_{x=0}^{2^L-1} \langle x| \otimes \langle a^x \bmod N|. \tag{0.52}$$

Finally, we perform the Fourier transform from Eq. (0.22) , on the first L bits,

$$\frac{1}{\sqrt{2^L}} \sum_{x=0}^{2^L-1} \langle x| \otimes \langle 0| \longmapsto \frac{1}{2^L} \sum_{x,p=0}^{2^L-1} \langle p| \otimes \langle a^x \bmod N| \exp\langle(\frac{2\pi \mathrm{i}}{2^L} px\rangle). \tag{0.53}$$

3.1. *Measurement*

We measure the first register and obtain an observable value p. Using the periodicity r we write $x = qr + s$ with $0 \leq s < r$ and $0 \leq q < q_r \approx 2^L/r$ such that the above sum becomes

$$\frac{1}{2^L} \sum_{p=0}^{2^L-1} \sum_{s=0}^{r-1} \exp\langle(\frac{2\pi \mathrm{i}}{2^L} ps\rangle) \langle p| \otimes \langle a^s \bmod N| \sum_{q=0}^{q_r-1} \exp\langle(\frac{2\pi \mathrm{i}}{2^L} pr\rangle)^q. \tag{0.54}$$

The sum on the right is nearly zero, except when

$$p_d \approx d \frac{2^L}{r} \tag{0.55}$$

for some integer $0 \leq d < r$. This means that the probability of measuring one of the 2^L values of p is nearly zero except for one of the r values p_d, for each of which the probability is of order $1/r$, so that the total probability of observing one of the values p_d is of the order one. The exact figure[10] is near $1/2$, so that the probability of observing another value of p is of the order of $1/(2r^2)$.

3.2. *Extracting the period of the function f*

Eq. (0.55) allows to extract the period of f as follows: We find an $L-$bit approximation to an unknown rational number d/r, from which we only know that its denominator is less than N. However, it is well known from number theory[13] that we can find the rational number with certainty and efficiently with the help of the continued fractions, if $2^L > N^2$.

3.3. *Measurement simulation: Choosing the final state*

In a simulation, we cannot "measure" in the sense that we cannot cause the collapse of the wave function and determine the partial amplitudes. The quantum computer selects by construction with high probability just one of the basis vectors with nonzero, maximal overlap to the final state. However, we have to choose the final state in advance and then to perform the Monte Carlo summation. We have to make sure that nonzero amplitude arises, in order to get a non-vanishing information from the Fourier transform. We cannot be sure whether a vanishing result was due to one of the following causes:

1. One reason may be, that we did not pick a value for p in the first part of the final state, with high amplitude as in Eq. (0.55). If we simply choose one of the 2^L values of p at random, we have an exponentially small probability (of the order of $1/r$) to get one of the values p_d. This we can circumvent as follows: We chose as final state successively the states $(|0\rangle + |1\rangle)^t (|0\rangle)^{L-t}$. As long as $2^t < r$, we will have a transition amplitude $\propto 2^{t/2}/r$, whereas for the value t_0 with $2^{t_0} \geq r > 2^{t_0-1}$ we have the amplitude of the order of $2^{t_0/2}/\sqrt{r} \sim O(1)$ after $\mathcal{O}(\log N)$ attempts.
2. The second possibility is that we did not pick one of the values $a^s \bmod N$ in the second register. There are r values for $a^s \bmod N$, an we chose with probability r/N one of them. The order of r/N is roughly one, so that if we choose a at random, then with sufficient probability we pick a function value.

That means that when we repeat the Monte Carlo sampling with a final state picked according to the procedure outlined above, will get the leading bit of r after $\mathcal{O}(\log N)$ attempts. Then we can repeat the procedure by fixing the second register of the final state, and replace successively the linear superposition $|0\rangle + |1\rangle$ by $|0\rangle$. If the amplitude was negligible, the $i-$th bit must have been one, and if it was finite, the $i-$th bit must be zero. Thus we can infer bit for bit the value of r.

3.4. *Implementation of Shor's algorithm for factoring 15*

Vandersypen et al.[14] have published a quantum circuit for which they succeeded in factoring 15 experimentally (see Fig. 5). We have implemented this circuit using our auxiliary field decomposition method and are also able to perform the factoring.

The function used is

$$f(x) = 7^x \bmod 15. \tag{0.56}$$

The three upper bits denote the first register $p = (p_0, p_1, p_2)$, the four lower bits denote the second register $y = (y_3, y_2, y_1, y_0)$. We monitored the convergence of the Monte Carlo procedure for different right vectors, namely in the binary representation

$$p^{(1)} = 2 = (0,1,0), \qquad y^{(1)} = 13 = (1,1,0,1) \tag{0.57}$$

$$p_2^{(2)} = 2 = (0,1,0), \qquad y^{(2)} = 9 = (1,0,0,1) \tag{0.58}$$

$$p_3^{(3)} = 3 = (1,1,0), \qquad y^{(3)} = 4 = (0,1,0,0) \tag{0.59}$$

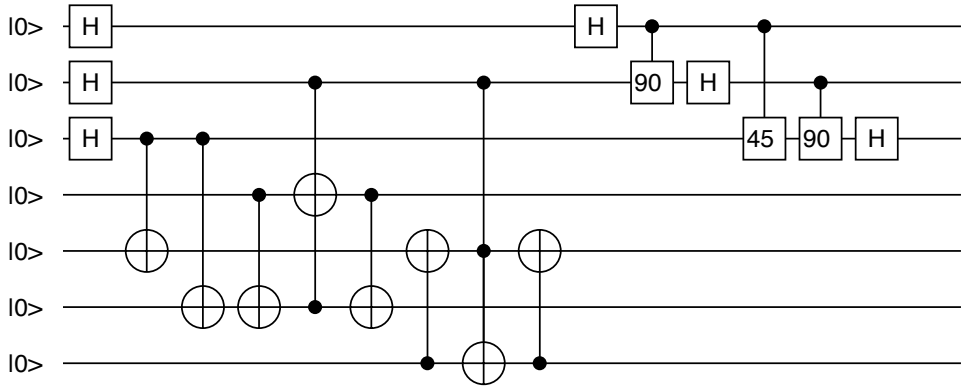

Figure 5: Quantum circuit for factoring 15 after Vandersypen.

The function has period $r = 4$ and can assume the values $1, 7, 4, 13$ as x increases from $0, 1, 2, 3$. The lower 4 bits represent a linear combination of one of the four values $1, 7, 4, 13$. It follows from Eq. (0.55) and $L = 3$ that the upper three bits p are a linear combination of even integers. If we choose p and y a priori, evaluating Eq. (0.54) gives the transition amplitudes (if they are non-zero)

$$A = \frac{1}{4} \mathrm{i}^{\frac{px}{2}}. \tag{0.60}$$

The circuit consists of 6 CNOT and 2 Toffoli gates and 3 controlled phase gates, which in our decomposition give $2^6 4^2 2^3 = 8192$ configurations. These can be easily simulated.

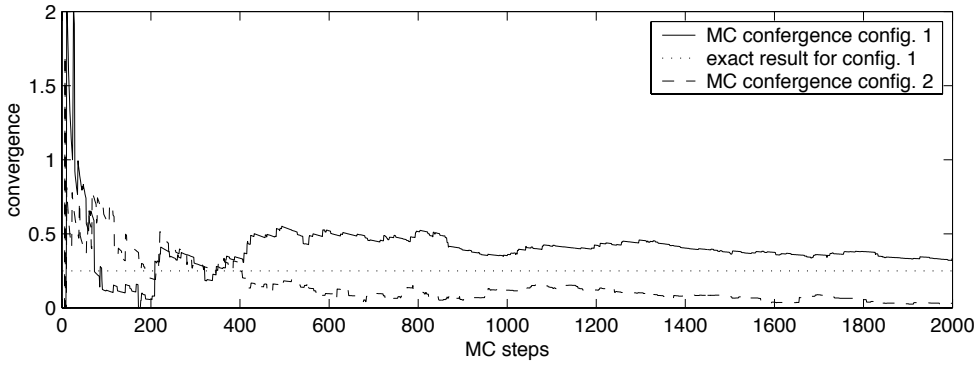

Figure 6: Convergence for the Shor's algorithm for different right vectors.

For $p^{(1)}, y^{(1)}$, the absolute value of the resulting amplitude is 0.25, and the convergence is plotted in Fig. 6 as full line to the exact value (dotted line). For $p^{(2)}, y^{(2)}$, the amplitude goes to 0, because the vector $y^{(2)}$ is not one of the function values $1, 7, 4 = 7^2$ mod15 or $13 = 7^3$ mod15. The convergence to 0 is plotted as broken line to 0, and is also very fast. For the third case $p^{(2)}$ is odd, and the

amplitude is zero. Because in this special case all HS-configurations yield a zero result, we have not shown the convergence in the plot.

4. Conclusion

We outlined in this article a novel strategy to simulate quantum circuits. We showed how we could map the quantum problem onto a classical one, which is accessible for the Monte Carlo sampling method. We found that we have to resort to simple sampling, because there is no obvious cost function. For the near future, we think that our approach using computer simulation will be on a par or superior to experimental implementations.

This work was partially supported by the programme "Research and Development on Quantum-Communication Technology" of the Ministry of Public Management, Home Affairs, Posts and Telecommunications of Japan and by the Inoue Foundation, Japan.

References

1. R.P. Feynman, Int. J. Theor. Phys. 21, 467 (1982).
2. J.E. Hirsch, Phys. Rev. B 28, 4059 (1983).
3. M. A. Nielsen and I.L. Chuang, Quantum Computation and Quantum Information, Cambridge Univ. Press, Cambridge, United Kingdom (2000).
4. Nielsen/Chuang 1.4.1.
5. Nielsen/Chuang, Fig. 4.8, page 182.
6. H.-G. Matuttis, K. Fischer, N. Ito, M. Ishikawa, preceding paper.
7. D. Coppersmith, IBM research report RC 19642 (1994).
8. D. Simon, Proc. 35th Annual Symposium on Foundations of Computer Science, IEEE Press, Los Alamitos, California (1994).
9. P.W. Shor, quant-ph/0005003.
10. P.W. Shor, Proc. 35th Annual Symposium on Foundations of Computer Science, IEEE Press, Los Alamitos, California, 124 (1994).
11. P.W. Shor, SIAM J. Comp. 26, 1484 (1997).
12. A.K. Lenstra and H.W. Lenstra Jr. (Ed), The Development of the Number Field Sieve, Springer-Verlag, New York (1993).
13. Nielsen/Chuang, Appendix 4.
14. L.M.K. Vandersypen, M. Steffen, G. Beyta, C. Yannoni, M. H. Sherwood, I. L. Chuang, *Nature* **414**, p.883 (2001)