УДК 512.6, 517.9, 519.6

EXAMPLES OF QUANTUM COMPUTING TOOLKIT: KRONECKER PRODUCT AND QUANTUM FOURIER TRANSFORMATIONS IN QUANTUM ALGORITHMS

Reshetnikov Andrey[1], Tyatyushkina Olga[2], Ulyanov Sergey[3], Tanaka Takayuki[4], Yamafuji Kazuo[5]

[1]*PhD in informatics, associate professor;*
Dubna State University,
Institute of system analysis and management;
141980, Dubna, Moscow reg., Universitetskaya str., 19;
e-mail: agreshetnikov@gmail.com.

[2]*PhD, associate professor;*
Dubna State University,
Institute of system analysis and management;
141980, Dubna, Moscow reg., Universitetskaya str., 19;
e-mail: tyatushkina@mail.ru.

[3]*Doctor of Science in Physics and Mathematics, professor;*
Dubna State University,
Institute of system analysis and management;
141980, Dubna, Moscow reg., Universitetskaya str., 19;
e-mail: ulyanovsv@mail.ru.

[4]*PhD, professor;*
The Graduate School of Information Science and Technology, Hokkaido University;
N14, W9, Sapporo-shi, Hokkaido, Japan;
e-mail: ttanaka@ssc.ssi.ist.hokudai.ac.jp.

[5]*PhD, professor;*
Dept. of Mechanical and Control Eng., University of Electro-Communications;
1-5-1 Chofu, Chofugaoka ,182 Tokyo, Japan;
e-mail: yamafuji@yama.mce.uec.ac.jp.

Quantum mechanics requires the operations of quantum computing to be unitary, and makes it important to have general techniques for developing fast quantum algorithms for computing unitary transformations. A quantum routine for computing a generalized Kronecker product is given. Applications for computing the Walsh-Hadamard and quantum Fourier transform is include also re-development of the according network.

Keyword: quantum computing, generalized Kronecker product, quantum Fourier transform, quantum algorithm

ПРИМЕРЫ ИНСТРУМЕНТАРИЯ КВАНТОВЫХ ВЫЧИСЛЕНИЙ: ПРОИЗВЕДЕНИЕ КРОНЕКЕРА И КВАНТОВОЕ ПРЕОБРАЗОВАНИЕ ФУРЬЕ В КВАНТОВЫХ АЛГОРИТМАХ

Решетников Андрей Геннадьевич[1], Тятюшкина Ольга Юрьевна[2], Ульянов Сергей Викторович[3], Танака Такаюки[4], Ямафуджи Кацуо[5]

[1]*Доктор информатики (PhD in Informatics), к.т.н., доцент;*
ГБОУ ВО МО «Университет «Дубна»,
Институт системного анализа и управления;
141980, Московская обл., г. Дубна, ул. Университетская, 19;
e-mail: agreshetnikov@gmail.com.

[2]*Кандидат технических наук, доцент;*

ГБОУ ВО МО «Университет «Дубна»,
Институт системного анализа и управления;
141980, Московская обл., г. Дубна, ул. Университетская, 19;
e-mail: tyatushkina@mail.ru.

[3]*Доктор физико-математических наук, профессор;*
ГБОУ ВО МО «Университет «Дубна»,
Институт системного анализа и управления;
141980, Московская обл., г. Дубна, ул. Университетская, 19;
e-mail: ulyanovsv@mail.ru.

[4]*Доктор наук (PhD in Informatics),*
Высшая школа информатики и технологии,
Университет Хоккайдо;
N14, W9, Саппоро-Ши, Хоккайдо, Япония;
e-mail: ttanaka@ssc.ssi.ist.hokudai.ac.jp.

[5]*Доктор наук, профессор,*
Факультет механики и технической кибернетики (интеллектуальные системы),
Университет передачи информации;
1-5-1, Япония, Токио, Chofu, Chofugaoka ,182;
e-mail: yamafuji@yama.mce.uec.ac.jp.

Квантовая механика требует, чтобы операции квантовых вычислений были унитарными, и делает важным иметь общие методы для разработки быстрых квантовых алгоритмов вычисления унитарных преобразований. В работе представлена квантовая подпрограмма для вычисления обобщённого произведения Кронекера. Приложения для вычисления Уолша-Адамара и квантового преобразования Фурье включают в себя также проектирование соответствующей сети.

Ключевые слова: квантовые вычисления, обобщённое произведение Кронекера, квантовое преобразование Фурье, квантовые алгоритмы

Introduction

The fundamental object is the unitary transform, which then can be considered a quantum or a classical (reversible) algorithm. With point of view, the problem of finding an efficient algorithm implementing a given unitary transform *U* into small number of "sparse" unitary transforms such that those sparse transforms should be known to be efficiently implementable. As an example of this, the quantum networks implementing the quantum versions of the discrete Fourier transforms can be easily derived from the mathematical descriptions of their classical counterparts [1-18].

Remark. For simplicity, we will use Kronecker product. The other two names: direct product and tensor product are also similar [19, 20].

Remark. More generally, unitary transform can be expressed as a certain generalized Kronecker product (defined below) then, given efficient quantum network implementing each factor in this expression, we also have an efficient quantum network implementing *U*. The expressive power of the generalized Kronecker product includes several new transforms.

Generalized Kronecker products

All matrices through this Section are finite. Recall that a square matrix is unitary if it is invertible and its inverse is the complex conjugate of its transpose. The complex conjugate of a number *c* is denoted by $\overline{c}$.

Let us introduce any definitions.

Definition 1. Let **c** be a $(p \times q)$ matrix and **C** a $(k \times l)$ matrix. The *left* and *right Kronecker product* of $(p \times q)$ and **C** are $(pk \times ql)$ matrices

$$\begin{pmatrix} Ac_{00} & Ac_{01} & \dots & Ac_{0,l-1} \\ Ac_{10} & Ac_{11} & \dots & Ac_{1,l-1} \\ \vdots & \vdots & \vdots & \vdots \\ Ac_{k-1,0} & Ac_{k-1,1} & \dots & Ac_{k-1,l-1} \end{pmatrix} \text{ and } \begin{pmatrix} a_{00}C & a_{01}C & \dots & a_{0,q-1}C \\ a_{10}C & a_{11}C & \dots & a_{1,q-1}C \\ \vdots & \vdots & \vdots & \vdots \\ a_{p-1,0}C & a_{p-1,1}C & \dots & a_{p-1,q-1}C \end{pmatrix},$$

respectively.

Definition 2. Given two tuples of matrices, a $k-$tuple $\text{A} = \left(A^i\right)_{i=0}^{k-1}$ of $\left(p \times q\right)$ matrices and $q-$ tuple $\text{C} = \left(C^i\right)_{i=0}^{q-1}$ of $\left(k \times l\right)$ matrices, the *generalized right Kronecker product* is the $\left(pk \times ql\right)$ matrix **D** = $\text{A} \otimes_R \text{C}$ where

$$d_{ij} = d_{uk+v,xl+y} = a_{ux}^{v} c_{vy}^{x}$$

with $0 \le u \le p, 0 \le v \le k, 0 \le x \le q,$ and $0 \le y \le l$.

Definition 3. The *generalized left Kroneker product* is the $\left(pk \times ql\right)$ matrix **D** = $\text{A} \otimes_L \text{C}$ where $\left(i, j\right)-$th entry holds the value

$$d_{ij} = d_{up+v,xq+y} = a_{vy}^{u} c_{ux}^{y}$$

with $0 \le u \le k, 0 \le v \le p, 0 \le x \le l,$ and $0 \le y \le q$.

Remark. The left Kronecker product denoted by $\mathbf{A} \otimes_L \mathbf{C}$ and the right Kronecker product denoted by $\mathbf{A} \otimes_R \mathbf{C}$**.** when some property holds for both definitions, we use $\mathbf{A} \otimes \mathbf{C}$**.** The Kronecker product is a binary matrix operator as opposed to the tensor product which is binary operator defined for algebraic structures like modules. The Kronecker product can be generalized in different ways. We use the generalized Kronecker product defined in *Definition 1*.

Remark. The generalized right Kronecker product can be found from the standard right Kronecker product by, for each sub-matrix $a_{ux}\mathbf{C}$ in *Definition 1* substituting it with the following sub-matrix

$$\begin{pmatrix} a_{ux}^{0} c_{00}^{x} & a_{ux}^{0} c_{01}^{x} & \dots & a_{ux}^{0} c_{0,l-1}^{x} \\ a_{ux}^{1} c_{10}^{x} & a_{ux}^{1} c_{11}^{x} & \dots & a_{ux}^{1} c_{1,l-1}^{x} \\ \vdots & \vdots & \vdots & \vdots \\ a_{ux}^{k-1} c_{k-1,0}^{x} & a_{ux}^{k-1} c_{k-1,1}^{x} & \dots & a_{ux}^{k-1} c_{k-1,l-1}^{x} \end{pmatrix}.$$

Remark. As for standard Kronecker products, we let $\text{A} \otimes \text{C}$ denote either of the two definitions. If the matrices $A^i = \mathbf{A}$ are all identical, and also $C^i = \mathbf{C}$, the generalized Kronecker product $\text{A} \otimes \text{C}$ reduced to the standard Kronecker product $\mathbf{A} \otimes \mathbf{C}$. Denote $\text{A} \otimes \mathbf{C}$ the generalized Kronecker product of a $k-$tuple A **of** $\left(p \times q\right)$ matrices, and $q-$ tuple C of $\left(k \times l\right)$ identical matrices **C**. Denote $\mathbf{A} \otimes \text{C}$ similarly.

Example. Let us consider the differences between left and right Kronecker products.

$$\boxed{A_R}\rightarrow \Downarrow = \left(\begin{bmatrix}1 & 1\\ 1 & -1\end{bmatrix},\begin{bmatrix}1 & 0\\ 0 & 1\end{bmatrix}\right)\otimes_R \begin{bmatrix}1 & 1\\ 1 & -1\end{bmatrix}\Rightarrow$$

$$\left[\begin{array}{cc|cc} (1\cdot 1) & (1\cdot 1) & (1\cdot 1) & (1\cdot 1)\\ (1\cdot 1) & (1\cdot(-1)) & (0\cdot 1) & (0\cdot(-1))\\ \hline (1\cdot 1) & (1\cdot 1) & ((-1)\cdot 1) & ((-1)\cdot 1)\\ (0\cdot 1) & (0\cdot(-1)) & (1\cdot 1) & (1\cdot(-1)) \end{array}\right] = \begin{bmatrix}1 & 1 & 1 & 1\\ 1 & -1 & 0 & 0\\ 1 & 1 & -1 & -1\\ 0 & 0 & 1 & -1\end{bmatrix}$$

$$\boxed{A_L}\rightarrow \Downarrow = \left(\begin{bmatrix}1 & 1\\ 1 & -1\end{bmatrix},\begin{bmatrix}1 & 0\\ 0 & 1\end{bmatrix}\right)\otimes_L \begin{bmatrix}1 & 1\\ 1 & -1\end{bmatrix}\Rightarrow$$

$$\left[\begin{array}{cc|cc} (1\cdot 1) & (1\cdot 1) & (1\cdot 1) & (1\cdot 1)\\ (1\cdot 1) & ((-1)\cdot 1) & (1\cdot 1) & ((-1)\cdot 1)\\ \hline (1\cdot 1) & (0\cdot 1) & (1\cdot(-1)) & (0\cdot(-1))\\ (0\cdot 1) & (1\cdot 1) & (0\cdot(-1)) & (1\cdot(-1)) \end{array}\right] = \begin{bmatrix}1 & 1 & 1 & 1\\ 1 & -1 & 1 & -1\\ 1 & 0 & -1 & 0\\ 0 & 1 & 0 & 1\end{bmatrix}$$

Thus, as result $A_R \neq A_L$, i.e., left and right Kronecker products are non-commutative relation.

To analyze generalized Kronecker products we need the *shuffle permutation matrix* of dimension $(mn\times mn)$, denoted Π_{mn} (as shorthand for $\Pi_{(m,n)}$), defined by

$$\pi_{rs} = \pi_{dn+l,d'm+l'} = \delta_{dl'}\delta_{d'l},$$

where $0\le d,l'<m; 0\le d',l<n$, and δ_{xy} denotes the Kronecker delta function which is zero if $x\neq y$, and one otherwise. It is unitary and satisfies $\Pi_{mn}^{-1}=\Pi_{mn}^{T}=\Pi_{mn}$. Given two tuples of matrices, *k*-tuple $\text{'A}=\left(A^i\right)_{i=0}^{k-1}$ of $(p\times r)$ matrices and $k-$ tuple $\acute{\text{C}}=\left(C^i\right)_{i=0}^{q-1}$ of $(r\times q)$ matrices, let $\text{'A}\cdot\acute{\text{C}}$ denotes the $k-$ tuple where the *i*-th entry is the $(p\times q)$ matrix A^iC^i, $0\le i\le k$. For any $k-$ tuple 'A of matrices, let $diag(\text{'A})$ denote the direct sum $\bigoplus_{i=0}^{k-1}A^i$ of matrices $A^0,\ldots,A^{k-1}$. The generalized Kronecker products satisfy the following important diagonalization theorem.

Theorem (*Diagonalization theorem*): Let $\text{'A}=\left(A^i\right)_{i=0}^{k-1}$ be a $k-$tuple $(p\times q)$ matrices and $\acute{\text{C}}=\left(C^i\right)_{i=0}^{q-1}$ a $q-$tuple of $(k\times l)$ matrices. Then

$$\begin{array}{|l|l|l|} \hline \text{'A}\otimes_R\acute{\text{C}} & = & \left(\Pi_{pk}\,diag(\text{'A})\,\Pi_{kq}\right)\times\boxed{diag\left(\acute{\text{C}}\right)}\\ \hline \text{'A}\otimes_L\acute{\text{C}} & = & \boxed{diag(\text{'A})}\times\left(\Pi_{kq}\,diag\left(\acute{\text{C}}\right)\Pi_{ql}\right)\\ \hline \end{array}. \qquad (1)$$

<u>*Corollar1*</u>: Let $\text{'A}=\left(A^i\right)_{i=0}^{k-1}$ be a $k-$tuple $(p\times q)$ matrices and $\acute{\text{C}}=\left(C^i\right)_{i=0}^{q-1}$ a $q-$tuple of $(k\times l)$ matrices. Then

$$\begin{array}{|l|l|l|} \hline \underbrace{\text{'A}\otimes_R\acute{\text{C}}}_{\Downarrow} & = & \underbrace{\Pi_{pk}\boxed{\left(\text{'A}\otimes_L\acute{\text{C}}\right)}\Pi_{lq}}_{\Downarrow}\\ \hline \text{'A}\otimes_L\acute{\text{C}} & = & \Pi_{kp}\boxed{\left(\text{'A}\otimes_R\acute{\text{C}}\right)}\Pi_{ql}\\ \hline \end{array} \qquad (2)$$

Remark. Until now, we have not assumed anything about the dimension of the involved matrices. In the text theorem, we assume that the matrices involved are square matrices. The theorem is easily proven from *Diagonalization theorem*. For any $k-$tuple $\text{A} = \left(A^i\right)_{i=0}^{k-1}$ of invertible matrices, let A^{-1} denote the $k-$tuple where *i*-th entry equals the inverse of A^i, $0 \le i \le k$.

Corollary: Let A, C be *m*-tuples of $(n \times n)$ matrices, and D, E be *n*-tuples of $(m \times m)$ matrices. Then

$$\boxed{\text{AC} \otimes \text{DE} \;=\; (\text{A} \otimes \text{I}_m) \times (\text{C} \otimes \text{D}) \times (\text{I}_n \otimes \text{E})} \quad . \tag{3}$$

Furthermore, if the matrices in the tuples A and C are invertible, then

$$\begin{array}{|c|c|c|c|c|} \hline \left(\text{A} \otimes_R \text{C}\right)^{-1} & = & \Pi_{nm}\left(\text{C}^{-1} \otimes_R \text{A}^{-1}\right)\Pi_{mn} & = & \text{C}^{-1} \otimes_{\boxed{L}} \text{A}^{-1} \\ \Downarrow & & & & \Downarrow \\ \hline \left(\text{A} \otimes_L \text{C}\right)^{-1} & = & \Pi_{mn}\left(\text{C}^{-1} \otimes_L \text{A}^{-1}\right)\Pi_{nm} & = & \text{C}^{-1} \otimes_{\boxed{R}} \text{A}^{-1} \\ \hline \end{array} \tag{4}$$

If A and C are unitary then so is $\text{A} \otimes \text{C}$.

Quantum routines (A method for constructing a quantum network for computing any given generalized Kronecker product.) A primary application of this method is a tool to find a quantum network of a given unitary matrix.

As our quantum computing model, we adopt the now widely used quantum gate arrays. Let $\tau : |u, v\rangle \to |u, v \oplus u\rangle$ denote the two-bit exclusive-OR operation, and $\mathfrak{U}$ the set of all one-bit unitary operations. By a basic operation we mean either a $\mathfrak{U}$ operation or the τ operation. The collection of basic operations is universal for quantum networks in the sense that any finite quantum network can be approximated with arbitrary precision by a quantum network *Q* consisting only of gates such operations. Define the one-bit unitary operations

$$\begin{array}{|c|c|c|c|c|c|c|c|c|c|c|c|} \hline X & = & \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} & Z & = & \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} & Y & = & \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix} & H & = & \frac{1}{\sqrt{2}}\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \\ \hline \end{array}$$

Given a unitary matrix $\mathcal{C}$, let $\Lambda\left((j,x),(k,\mathcal{C})\right)$ denote the transform where we apply $\mathcal{C}$ on the *k*-th register iff the *j*-th register equals *x*. Given an *n*-tuple $\mathcal{C} = \left\{\text{C}^i\right\}_{i=0}^{n-1}$ of unitary matrices, let $\Lambda\left((j,i),\left(k,C^i\right)\right)_i$ denote $\Lambda\left((j,n-1),\left(k,C^{n-1}\right)\right)\cdots\Lambda\left((j,0),\left(k,C^0\right)\right)$. Given a *k*-th root of unity, say ω, let $\Phi(\omega)$ denote the unitary transform given by $|u\rangle|v\rangle \to \omega^{uv}|u\rangle|v\rangle$. If the first register holds a value from $\mathbb{Z}_n$, and the second holds a value from $\mathbb{Z}_m$, then $\Phi(\omega) = \Phi_{(n,m)}(\omega)$ can be implemented in $\Theta\left(\lceil \log n \rceil \lceil \log m \rceil\right)$ basic operations.

<u>*Example*</u>: *Typical matrix operations*. Let us consider the implementation of the following operations.

***1.* Quantum shuffle transform**
Let the operation DM_m perform the unitary transform $\|k\rangle\|0\rangle \to \|k\, div m\rangle\|k \bmod m\rangle$, for every *m* > 1. Let *SWAP* denote the unitary transform $\|u\rangle\|v\rangle \xrightarrow{SWAP} \|v\rangle\|u\rangle$. Then Π_{mn} can be implemented on a quantum computer by one application of DM_m, one *SWAP* operation, and one application of DM_m^{-1}, i.e., $\Pi_{mn} \equiv DM_m^{-1}\ SWAP\ DM_m$.

2. Quantum direct sum

Let $\mathcal{C}$ be an *n*-tuple of $(m \times m)$ unitary matrices. It is true that $Diag(\mathcal{C})$ can be implemented as follows,

$$Diag(\mathcal{C}) \equiv DM_n^{-1} \Lambda\left((1,i),\left(2,C^i\right)\right)_i DM_m .$$

In general, the time to compute the direct sum is proportional to the sum of the computation times of each of the conditional C^i transforms. However, if parts of these can be applied in quantum parallel, this improves the running time.

3. Quantum Kronecker product

Let $\mathcal{A}$ be an *m*-tuple of $(n \times n)$ unitary matrices and $\mathcal{C}$ be an *n*-tuple of $(m \times m)$ unitary matrices. By the *Diagonalization* theorem, the generalized Kronecker product can be applied by applying two direct sums and two shuffle transforms. Removing canceling terms we get

$$\mathcal{A} \otimes_R \mathcal{C} \equiv DM_m^{-1} \Lambda\left((2,i),\left(1,A^i\right)\right)_i \Lambda\left((1,i),\left(2,C^i\right)\right)_i DM_m$$

$$\mathcal{A} \otimes_L \mathcal{C} \equiv DM_n^{-1} \Lambda\left((1,i),\left(2,A^i\right)\right)_i \Lambda\left((2,i),\left(1,C^i\right)\right)_i DM_n .$$

Thus, an applications of a generalized right Kronecker product can be divided up into the following four steps: in the first step, we apply DM_m. In the second step, we apply the controlled C^i transform on the second register, and in the third step, the controlled A^i transforms on the first register. Finally, in the last step, we apply DM_m^{-1} to the result.

Example. Let $\mathcal{A}$ be a 4-tuple of (2×2) unitary matrices, and $\mathcal{C}$ a 2-tuple of (4×4) unitary matrices. The generalized Kronecker product $\mathcal{A} \otimes_R \mathcal{C}$ can be implemented by a quantum network. The dots and the circles represents control-bit: if the values with the dots are one, and if the values with the circles are zero, the transform is applied, otherwise it is the identity map. The least (most) significant bit is denoted by LSB (MSB). Note that the most of the transforms are orthogonal, and thus the gates commute.

Remark. Following the ideas of *Griffiths et all*, a semi-classical generalized Kronecker product transform can be defined.

Properties of the Kronecker product in quantum information theory

We will discuss the properties and applications of Kronecker product in quantum theory that is studied thoroughly. The use of Kronecker product in quantum information theory to get the exact spin Hamiltonian and the proof of non-commutativity of matrices, when Kronecker product is used between them is also given. The non-commutativity matrices after Kronecker product are similar or they are similar matrices.

Remark. The use of Kronecker product in quantum information theory is used extensively. But the rules, properties and applications of Kronecker product are not discussed in any quantum textbooks. Even books on mathematical aspects of quantum theory are discussed the properties and applications of Kronecker product in very short without any explanations of its rules. That is, why we applying Kronecker product left/right to any spin operator, why not left or why not right only. They are applying Kronecker product without any explanation. Mathematicians were also not able to give any concrete answer to these questions in more generalized way. We can find textbooks on algebra, where the mathematical aspects are considered in more rigorous and detail. But many books on algebra do not consider the physical aspects of Kronecker product, which are very important in quantum theory. So, we will discuss answers to all of the abovementioned questions, which are now very important in quantum information theory to write exact spin Hamiltonian.

Mathematical aspects of Kronecker product

Let $\hat{A}$ and $\hat{B}$ are two linear operators defined in the finite dimensional *L* and *M* vector spaces on field *F*. $\hat{A} \otimes \hat{B}$ is the Kronecker product of two operators in the space $L \otimes M$. Kronecker product of two matrices are given by the rule: $(A)_{m1,n1} \otimes (B)_{m2,n2} = (C)_{(m1,m2),(n1,n2)}$. After the Kronecker product the dimensions of the finite space becomes $N \cdot M$, where *N* and *M* are dimensions of finite spaces on which operators *A* and *B* are defined.

Properties of the Kronecker products. Let the operators $\hat{A}1, \hat{A}2, \hat{B}1, \hat{B}2$ and $\hat{E}$ (unity matrix) are defined in finite dimensional vector spaces *L1, L2, M1, M2* and $\hat{E}$ is defined in *L1, L2, M1, M2* on the field *F*. The properties of the Kronecker product can be written as following (see Table 1).

Table 1. The properties of the Kronecker product

N	*Kronecker product property*
1	$(A1)_{m1,n1} \otimes 0 = 0 \otimes (B1)_{m1,n1} = 0 \ (0 \ is \ zero \ matrix)$
2	$(E)_{m1,n1} \otimes (E)_{m2,n2} = (E)_{(m1,n1),(m2,n2)}$
3	$\left((A1)_{m1,n1} + (A2)_{m2,n2}\right) \otimes (B1)_{m1,n1} = (A1)_{m1,n1} \otimes (B1)_{m1,n1} + (A2)_{m2,n2} \otimes (B1)_{m1,n1}$
4	$(A1)_{m1,n1} \otimes \left((B1)_{m1,n1} + (B2)_{m2,n2}\right) = (A1)_{m1,n1} \otimes (B1)_{m1,n1} + (A1)_{m1,n1} \otimes (B2)_{m2,n2}$
5	$s \cdot (A1)_{m1,n1} \otimes t \cdot (B1)_{m1,n1} = s \cdot t \cdot (A1)_{m1,n1} \otimes (B1)_{m1,n1} \ (s,t \ are \ constants)$
6	$\left((A1)_{m1,n1} \otimes (B1)_{m1,n1}\right)^{-1} = (B1)^{-1}_{m1,n1} \otimes (A1)^{-1}_{m1,n1}$
7	$\left((A1)_{m1,n1} \cdot (B1)_{m1,n1}\right) \otimes \left((A2)_{m2,n2} \cdot (B2)_{m2,n2}\right) = \left((A1)_{m1,n1} \otimes (A2)_{m2,n2}\right) \cdot \left((B1)_{m1,n1} \otimes (B2)_{m2,n2}\right)$
8	$\left((A1)_{m1,n1} \otimes (B1)_{m1,n1}\right) \neq \left((B1)_{m1,n1} \otimes (A1)_{m1,n1}\right)$

Example. We will prove only one property (8), which is used in quantum information theory. The others can be proved easily by analyzing the proof of property (8).

Theorem: The Kronecker product of two matrices are non-commutative, i.e.

$$\left((A1)_{m1,n1} \otimes (B1)_{m1,n1}\right) \neq \left((B1)_{m1,n1} \otimes (A1)_{m1,n1}\right).$$

Proof. Let two linear operators $\hat{A}$ and $\hat{B}$ with bases *a* and *b* are defined in finite vector spaces *L* and *M* vector spaces on field *F*. The operators can be written into the form of matrices *(A)* and *(B)*. First we will write the L.H.S part of the Kronecker product, i.e., $(A) \otimes (B)$ and then R.H.S $(B) \otimes (A)$. Then, we will compare all the elements of both the L.H.S and R.H.S matrices. If even one of the element with same indices of both the matrices differ then these matrices are not equal or non-commutative.

$$(A)_{m\times n} \otimes (B)_{m\times n} = \begin{pmatrix} A_{1,1}(B)_{m\times n} & A_{1,2}(B)_{m\times n} & \cdots & A_{1,n}(B)_{m\times n} \\ A_{2,1}(B)_{m\times n} & A_{2,2}(B)_{m\times n} & \cdots & A_{2,n}(B)_{m\times n} \\ \cdots & \cdots & \cdots & \cdots \\ A_{m,1}(B)_{m\times n} & A_{m,2}(B)_{m\times n} & \cdots & A_{m,n}(B)_{m\times n} \end{pmatrix} \quad (5)$$

$$(B)_{m\times n} \otimes (A)_{m\times n} = \begin{pmatrix} B_{1,1}(A)_{m\times n} & B_{1,2}(A)_{m\times n} & \cdots & B_{1,n}(A)_{m\times n} \\ B_{2,1}(A)_{m\times n} & B_{2,2}(A)_{m\times n} & \cdots & B_{2,n}(A)_{m\times n} \\ \cdots & \cdots & \cdots & \cdots \\ B_{m,1}(A)_{m\times n} & B_{m,2}(A)_{m\times n} & \cdots & B_{m,n}(A)_{m\times n} \end{pmatrix} \quad (6)$$

The elements of both matrices are not equal. It means that these matrices are non-commutative.

Remark. Only the Kronecker product of two unity matrices are equal, i.e., they are commutative.

Similar operators (matrices)

Let two linear operators $\hat{A}$ and $\hat{B}$ with bases a and b are defined in finite vector spaces L and M vector spaces on field F. The question arises, when operators $\hat{A}$ and $\hat{B}$ are considered similar. Since, we are interested in the similarity of these operators, so we will study the action of these operators on different bases in different vector spaces.

Definition: The operators $\hat{A}: L \to L$ and $\hat{B}: M \to M$ are called similar operators, if they are defined on field F, $\dim \hat{A} = \dim \hat{B}$ and exist isomorphism $\hat{f}: L \to M$, i.e., $\hat{B}(b) = \hat{f}\ \hat{A}\ \hat{f}^{-1}(b)$.

For this case we have the following theorem.

Theorem: *If operators $\hat{A}$ and $\hat{B}$ with bases (a) and (b) are defined in vector spaces L and M on field F then the matrices of operators $\hat{A}$ and $\hat{B}$ in their corresponding bases a and b are similar, i.e.,* $(A)^a = (B)^b$.

Proof. Let the bases $\hat{a}: a_1, \ldots, a_n$ and $b: \hat{f}(L \to M)(a) = a_1', \ldots, a_n'$ of linear operators $\hat{A}$ and $\hat{B}$ are defined in vector spaces *L* and *M* on field *F* then

$\hat{A}(a_j)$	=	$\sum_i A_{i,j}(a_i)$
$\hat{B}(b_j)$	=	$\hat{B}\ \hat{f}(a_j) = \hat{f}\ \hat{A}(_j)$
$= \hat{f}\sum_i A_{i,j}(a_i)$	=	$\sum_i A_{i,j}\hat{f}(a_i) = \sum_i A_{i,j}(a_i')$

.

Hence, A and are similar matrices

<u>*Example*</u>: *More simplified proof of theorem*. Let the linear operator $\hat{C}$ defined in vector space *L* and *M* changes the bases *a* into *b*, i.e.,

b_j	=	$\sum_i C_{i,j}a_i$ (7)	$\hat{A}a_j$	=	$\sum_i A_{i,j}a_i$ (8)

The operator $\hat{A}$ acts on basis *b* gives the matrix $D_{i,j}$ and basis vector b_j

$\hat{B}b_j$	=	$\sum_i B_{i,j}b_i$ (9)	$\hat{A}b_j$	=	$\sum_i D_{i,j}b_i$ (10)

By putting (7) into L.H.S. of (10)

$$\hat{A}\sum_k C_{k,j}a_k = \sum_k C_{k,j}\hat{A}\,a_k = \sum_k\sum_i A_{i,k}C_{k,j}a_i \qquad (11)$$

By putting (7) into R.H.S. of (10)

$$\sum_k D_{k,j}b_k = \sum_k D_{k,j}\sum_i C_{i,k}a_i = \sum_k\sum_i C_{i,k}D_{k,j}a_i\ . \qquad (12)$$

The R.H.S. of (11) and (12) are equal

$$C_{i,k}D_{k,j} = A_{i,k}C_{k,j}$$

or

$$(D) = (C)^{-1}\cdot(A)\cdot(C).$$

The theorem is proved.

Physical aspects of Kronecker product and its applications in quantum information theory

The Kronecker product in group theory is widely used, especially with Wigner D-function. The main purpose of its use in physics is to get the higher dimensional vector space. For example, in atomic physics, when we want to calculate the eigenvalues and eigenvectors of a system of spins ½ or spin Hamiltonian.

We analytically or with help of PC diagonalize spin Hamiltonian and find eigenvectors and eigenvalues with two methods:

1	We should numerate each operator (matrix) of corresponding spin without multiplying (ordinary matrix multiplication) them with each other. By doing this, we can label each matrix of corresponding spin and each operator acts on their corresponding eigenvector.
2	By applying the Kronecker product different spins matrices, e.g., two matrices (dimensions 2×2) of spins ½, we will get the matrix of dimensions (4×4). This method is very compact, which means we can use the computer to get the eigenvectors and eigenvalues of matrix after applying Kronecker product for higher number of spins, e.g., for the system of spins ½.

Remark. But there are some mathematical and physical problems during the process of Kronecker product. These problems are can be described as following.

(*a*)	As it seen from the non-commutative nature of Kronecker product that we do not have right to take Kronecker product for two different spins freely (because they are non-commutative). Then how Kronecker product can be applied in quantum theory.
(*b*)	The non-commutative matrices $\left[(A)\otimes(B)\neq(B)\otimes(A)\right]$ after the Kronecker product are called similar matrices. It means, the eigenvalues of matrix $(AB)=(A)\otimes(B)$ *and* $(BA)=(B)\otimes(A)$ are similar. But eigenvectors of some eigenvalues are misplaced with their neighbourhood eigenvectors. This misplacement can be removed by applying the similar matrix method, which is proved earlier. The similar matrix method becomes more complicated as the dimensions of the vector space increases (number of spins increases).

We will be discussed all of these problems below.

Applications in quantum theory

At the moment the Kronecker product is extensively used in quantum information theory. So, we concentrate on the applications of Kronecker product in information theory. All the applications of Kronecker product in quantum information theory can be easily applied to any other branch of quantum theory where it requires. Examples of Kronecker product are discussed below.

<u>*Example A2.1*</u>: *Hamiltonian of n spins ½ in Nuclear Magnetic Resonance (NMR)*.
Let $\hat{\sigma}_1,\ldots,\hat{\sigma}_n$ are linear spin operators defined in the finite dimensional $S_1,\ldots,S_n$ vector spaces on field *F*. $\hat{\sigma}_1\otimes\hat{\sigma}_2\ldots\otimes\hat{\sigma}_n$ are defined in linear space $S_1\otimes S_2\ldots\otimes S_n$. All the matrices of spin operators $\hat{\sigma}_1,\ldots,\hat{\sigma}_n$ are 2×2 dimensions. For simplicity, we are taking $\hbar=1$.
1. *Case n = 2 spins ½.*
Hamiltonian of two spins $\hat{\sigma}_1$ and $\hat{\sigma}_2$ defined in linear space S_1, S_2. $\hat{\sigma}_{1z}$ and $\hat{\sigma}_{2z}$ coupling with hyperline interaction J_{12} are placed parallel to applied constant magnetic field $B_0\parallel-$ axis: $H_2=-\mu B_0\cdot\left(\hat{\sigma}_{z1}\otimes\hat{E}_2\right)-\mu B_0\cdot\left(\hat{E}_1\otimes\hat{\sigma}_{z2}\right)+J_{12}\left(\hat{\sigma}_{x1}\otimes\hat{\sigma}_{x2}\right)+J_{12}\left(\hat{\sigma}_{y1}\otimes\hat{\sigma}_{y2}\right)+J_{12}\left(\hat{\sigma}_{z1}\otimes\hat{\sigma}_{z2}\right).$
2. C*ase n = 3 spins ½.*
Hamiltonian of three spins

$$H_3 = -\mu B_0 \cdot \left(\hat{\sigma}_{z1} \otimes \hat{E}_2 \otimes \hat{E}_3\right) - \mu B_0 \cdot \left(\hat{E}_1 \otimes \hat{\sigma}_{z2} \otimes \hat{E}_3\right) - \mu B_0 \cdot \left(\hat{E}_1 \otimes \hat{E}_2 \otimes \hat{\sigma}_{z3}\right)$$
$$+ J_{12}\left(\hat{\sigma}_{x1} \otimes \hat{\sigma}_{x2} \otimes \hat{E}_3\right) + J_{23}\left(\hat{E}_1 \otimes \hat{\sigma}_2 \otimes \hat{\sigma}_3\right) + J_{31}\left(\hat{\sigma}_3 \otimes \hat{\sigma}_1 \otimes \hat{E}_2\right)$$

Hamiltonians of higher spins ½ can be written in the same way as for 2 and 3 spins ½.

Kronecker product in quantum information theory to get the spin Hamiltonians

To write the spin Hamiltonian, first of all we should write $\hat{S}_x, \hat{S}_y, \hat{S}_z, \hat{S}^2$ and then add them with their corresponding factors, we will get the spin Hamiltonians (e.g., $\hat{H}_2$ and $\hat{H}_3$). We will proof this later.

Example. Let we want to write the spin Hamiltonian of *n* spins in *NMR*:

1

Total projection of n spins ½ on z-axis is conserved:

$$\hat{S}_z = \frac{1}{2}\left(\hat{\sigma}_{1z} \begin{cases} \bigotimes_{i=2}^{n} \hat{E}_i, & if\ n \geq 2 \\ 1 & if\ n = 1 \end{cases} + \hat{E}_1 \otimes \hat{\sigma}_{2z} \begin{cases} \bigotimes_{i=3}^{n} \hat{E}_i, & if\ n \geq 3 \\ 1 & if\ n = 2 \\ 0 & if\ n < 2 \end{cases} \right.$$
$$\left. + \hat{E}_1 \otimes \hat{E}_2 \otimes \hat{\sigma}_{3z} \begin{cases} \bigotimes_{i=4}^{n} \hat{E}_i, & if\ n > 3 \\ 1 & if\ n = 3 \\ 0 & if\ n < 3 \end{cases} + \cdots \right) \quad (13)$$

The first term in Eq.(13) contains first spin $\hat{\sigma}_{1z}$ with Kronecker product of $\hat{E}_i$ unit matrices of other spins, i.e., second, third and so on. The second term contains first factor unit matrix $\hat{E}_i$ with Kronecker product of second spin $\hat{\sigma}_{2z}$ and unit matrix of other spins. The third and forthcoming terms are written analytically by analyzing preceding. $\hat{S}_x$ and $\hat{S}_y$ can be written by putting $\hat{\sigma}_x$ and $\hat{\sigma}_y$ in place of $\hat{\sigma}_z$.

2

The square of the total spin $\hat{S}^2 = \hat{S}\left(\hat{S}+1\right)$ is conserved and

$$\hat{S} = i\hat{S}_x + j\hat{S}_y + k\hat{S}_z \quad (14) \qquad \hat{S}^2 = \hat{S}_x^2 + \hat{S}_y^2 + \hat{S}_z^2 \quad (15)$$

3

Equations (14) and (15) are constant of motion. That is $\left[\hat{S}_z, \hat{S}^2\right] = 0$. It means that the eigenvalues and eigenvectors of (13) and (15) are identical.

4

The Hamiltonians $\hat{H}_2$ and $\hat{H}_3$ are consist of two parts (13) and (15) with their corresponding factors.

Example: *Proof of* $\hat{H}_2$. To proof $\hat{H}_2$, that it consists of (13) and (15), we should take two spins $\hat{S}_1$ and $\hat{S}_2$ with constants a_i, $i \in x, y, z$. To write $\hat{S}_x, \hat{S}_y$ and $\hat{S}_z$, we use (13), i.e.,

$$\hat{S}_z = a_z\left(\hat{\sigma}_{1z} \otimes \hat{E}_2 + \hat{E}_1 \otimes \hat{\sigma}_{2z}\right), \quad (16)$$
$$\hat{S}_1^2 = \hat{S}_x^2 + \hat{S}_y^2 + \hat{S}_z^2 \quad (17)$$

where

$$\begin{aligned}\hat{S}_x^2 &= \left(\hat{\sigma}_{1z}\otimes\hat{E}_2\right)\left(\hat{E}_1\otimes\hat{\sigma}_{2z}\right)+\left(\hat{\sigma}_{1z}\otimes\hat{E}_2\right)\left(\hat{E}_1\otimes\hat{\sigma}_{2z}\right)\\ &+\left(\hat{\sigma}_{1x}\otimes\hat{E}_2\right)\left(\hat{\sigma}_{1x}\otimes\hat{E}_2\right)+\left(\hat{\sigma}_{1z}\otimes\hat{E}_2\right)\left(\hat{E}_1\otimes\hat{\sigma}_{2x}\right).\\ &+\left(\hat{E}_1\otimes\hat{\sigma}_{2x}\right)\left(\hat{\sigma}_{1z}\otimes\hat{E}_2\right)+\left(\hat{E}_1\otimes\hat{\sigma}_{2x}\right)\left(\hat{E}_1\otimes\hat{\sigma}_{2x}\right)\end{aligned} \quad (18)$$

By using the property (A2.8), we can simplified Eq. (18) to

$$\hat{S}_x^2 = 2a_x^2\left[\left(\hat{E}_1\otimes\hat{E}_2\right)+\left(\hat{\sigma}_{1x}\otimes\hat{\sigma}_{2x}\right)\right]. \quad (19)$$

Similarly, we can calculate $\hat{S}_y^2$ and $\hat{S}_z^2$

$$\hat{S}_x^2 = 2a_y^2\left[\left(\hat{E}_1\otimes\hat{E}_2\right)+\left(\hat{\sigma}_{1y}\otimes\hat{\sigma}_{2y}\right)\right], \quad (20)$$

$$\hat{S}_z^2 = 2a_z^2\left[\left(\hat{E}_1\otimes\hat{E}_2\right)+\left(\hat{\sigma}_{1z}\otimes\hat{\sigma}_{2z}\right)\right]. \quad (21)$$

Now we will add (16), (19), (20), and (21) as following

$$\begin{aligned}\hat{S}_z+\hat{S}_x^2+\hat{S}_x^2+\hat{S}_x^2 &= a_z\left(\hat{\sigma}_{1z}\otimes\hat{E}_2+\hat{E}_1\otimes\hat{\sigma}_{2z}\right)+\sigma_x^2\left[\left(\hat{E}_1\otimes\hat{E}_2\right)+\left(\hat{\sigma}_{1x}\otimes\hat{\sigma}_{2x}\right)\right]\\ &+2a_y^2\left[\left(\hat{E}_1\otimes\hat{E}_2\right)+\left(\hat{\sigma}_{1y}\otimes\hat{\sigma}_{2y}\right)\right]+2a_z^2\left[\left(\hat{E}_1\otimes\hat{E}_2\right)+\left(\hat{\sigma}_{1z}\otimes\hat{\sigma}_{2z}\right)\right]\end{aligned} \quad (22)$$

By analyzing (22) and $\hat{H}_2$, we can see that they are same, only we have to choose corresponding a_i. The term $2a_i^2\left(\hat{E}_1\otimes\hat{E}_2\right)$ does not have meaning, since it is unit matrix. We have proved that $\hat{H}_2$ can be correctly chosen with help of $\hat{S}_z$ and $\hat{S}^2$ using the properties

$\hat{\sigma}_{1x}\otimes\hat{\sigma}_{2x}$	=	$\hat{\sigma}_{2x}\otimes\hat{\sigma}_{1x}$
$\hat{\sigma}_{1y}\otimes\hat{\sigma}_{2y}$	=	$\hat{\sigma}_{2y}\otimes\hat{\sigma}_{1y}$
$\hat{\sigma}_{1z}\otimes\hat{\sigma}_{2z}$	=	$\hat{\sigma}_{2z}\otimes\hat{\sigma}_{1z}$

.

The proof of $\hat{H}_3$ can be written similarly by applying property (8).

Similar matrices in quantum computation

Tensor product, which is very widely, used in quantum theory plays an important role not only for obtaining the higher dimensional Hilbert space but also to optimize the experimental technique. Since, the tensor product is non-commutative in nature, i.e., it cannot be applied freely to any operator without any knowledge of the constituents of the physical system. One very important property of tensor product is similar matrices, which have two optimization properties: reduces the number of pulses to realize some operator in physical experiment; and provides different choices of pulse propagations along different directions of axes. These two properties will be discussed with concrete physical realization of CNOT operator and how we can optimize its realization. We present theoretical result, which is based on the linear algebra (similar operators). The obtained theoretical results optimize the experimental technique to construct quantum computation e.g., reduces the number of steps to perform the logical CNOT (XOR) operation. The present theoretical technique can also be generalized to the other operators in quantum computing and information theory.

CNOT operator gates and CNOT's similar matrices in quantum computing.

CNOT gates or similar matrices play an important role in the construction of quantum computation. All the complex quantum algorithms are based on the computation of NOT and CNOT logical gates or matrices

in quantum computation. Here, we will not discuss the NOT logical gate or matrix, which is very simple negation operation. We will go in deep the mathematical and physical nature of CNOT logical gates. The other complex gates can be constructed on the basis of CNOT logical gates. Moreover, the construction of CNOT matrices plays an important role in quantum computation. Here, we will show that CNOT-matrices can be optimized by using the similar matrices, i.e., we can find different CNOT matrices with the same mathematical properties but some different experimental or physical realization. For better insight to understand the physical nature of similar matrices, we will discuss in the whole item the system of two spins $\hat{\sigma}_1 = \frac{1}{2}$ and $\hat{\sigma}_2 = \frac{1}{2}$ with slightly different resonance frequencies ω_1 and ω_2, and having scalar coupling ω_{12}. The Hamiltonian of the two spins aligned along the *z*-axis with the constant magnetic field

$$\hat{H} = \hbar\omega_1\hat{\sigma}_{1z} \otimes \hat{e}_2 + \hbar\omega_2\hat{e}_1 \otimes \hat{\sigma}_{2z} + \hbar\omega_{12}\hat{\sigma}_{1z} \otimes \hat{\sigma}_{2z} \ , \tag{23}$$

where $\hat{e}_i$, $i \in \{1,2\}$ is identity matrix with 2×2 -dimensions .

Physical differences between the CNOT matrices.

Let us consider the *CNOT* matrix that contains additional spin $\hat{\sigma}_1$ rotation around the z-axis, which is unnecessary and which complicate the experimental realization of the *CNOT* as compare to the *CNOT* matrix used in *NMR*-quantum computation (*Gershenfeld and Cory*, 1997 – 2002; *Jones*, 1998 - 2002). In the mathematical viewpoint, *CNOT* matrices have in this case different mathematical properties, i.e., *CNOT* matrices are not similar matrices due to the additional rotation of spin $\hat{\sigma}_{z1}$ around *z*-axis.

Remark. Moreover, additional rotation takes more time to fulfill the CNOT operation, which slows down the computation of quantum algorithms. The *CNOT* matrix for Hamiltonian (23) obtained by *Gershenfeld* is

$$C_G = \sqrt{-i}\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}. \tag{24}$$

Physically, the C_G matrix was obtained by the following pulse sequences

$$\begin{aligned} C_G &= R_{y2}\left(-\frac{\pi}{4}\right)R_{z1}\left(-\frac{\pi}{4}\right)R_{z2}\left(-\frac{\pi}{4}\right)R_{z12}\left(\frac{\pi}{4}\right)R_{y2}\left(\frac{\pi}{4}\right) \\ &= \exp\left\{-i\frac{\pi}{4}\hat{e}_1 \otimes \hat{\sigma}_{y2}\right\}\exp\left\{-i\frac{\pi}{4}\hat{\sigma}_{z1} \otimes \hat{e}_2\right\}\exp\left\{-i\frac{\pi}{4}\hat{e}_1 \otimes \hat{\sigma}_{z2}\right\}, \\ &\times \exp\left\{i\frac{\pi}{4}\hat{\sigma}_{z1} \otimes \hat{\sigma}_{z2}\right\}\exp\left\{i\frac{\pi}{4}\hat{e}_1 \otimes \hat{\sigma}_{z2}\right\} \end{aligned} \tag{25}$$

where *R* is rotation matrix around the different axes along with the different angles around different axes. The physical interpretation of the matrix R is just the rotation of our physical reference system or quantum tomography.

Remark. The two CNOT matrices for Hamiltonian (23) obtained by *Cory* are

$$C_{C1} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & -1 & 0 \end{pmatrix} \quad C_{C2} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & -1 & 0 & 0 \end{pmatrix} \tag{26}$$

In the experiment of *NMR*, the CNOT matrices (26) are called *Pound-Overhauser* operators, i.e., transformation of spin polarization from one spin to other spin.

The C_{C1} and C_{C2} matrices were obtained with the following pulse sequences

C_{C1}	=	$R_{x2}\left(-\frac{\pi}{4}\right)R_{z2}\left(-\frac{\pi}{4}\right)R_{z12}\left(-\frac{\pi}{4}\right)R_{z12}\left(\frac{\pi}{4}\right)R_{x2}\left(\frac{\pi}{4}\right)$
	=	$\exp\left\{-i\frac{\pi}{4}\hat{e}_1\otimes\hat{\sigma}_{x2}\right\}\exp\left\{-i\frac{\pi}{4}\hat{e}_1\otimes\hat{\sigma}_{z2}\right\}\exp\left\{i\frac{\pi}{4}\hat{\sigma}_{z1}\otimes\hat{\sigma}_{z2}\right\}\exp\left\{i\frac{\pi}{4}\hat{e}_1\otimes\hat{\sigma}_{z2}\right\}$
C_{C2}	=	$R_{x1}\left(-\frac{\pi}{4}\right)R_{z1}\left(-\frac{\pi}{4}\right)R_{z12}\left(-\frac{\pi}{4}\right)R_{z12}\left(\frac{\pi}{4}\right)R_{x1}\left(\frac{\pi}{4}\right)$
	=	$\exp\left\{-i\frac{\pi}{4}\hat{\sigma}_{x1}\otimes\hat{e}_2\right\}\exp\left\{-i\frac{\pi}{4}\hat{\sigma}_{z1}\otimes\hat{e}_2\right\}\exp\left\{i\frac{\pi}{4}\hat{\sigma}_{z1}\otimes\hat{\sigma}_{z2}\right\}\exp\left\{i\frac{\pi}{4}\hat{\sigma}_{x1}\otimes\hat{e}_2\right\}$

Mathematical differences between the CNOT matrices (24) *and* (26). Since, the Hamiltonian of the CNOT matrices (24) and (26) are the same but they are not similar matrices. To check the mathematical nature of (24) and (26), first of all we will write six mathematical properties of similar square matrices *A* and *B* of dimensions 4×4 on the Hilbert space $\mathcal{H}$.

1	Determinant of *A* is equal to determinant of *B*
2	Trace of *A* is equal to trace of *B*
3	If *A* and *B* are nonsingular than A^{-1} and B^{-1} are also similar matrices
4	*A* and *B* are similar matrices, if there exist nonsingular matrix *P* such that $B=P^{-1}AP$ or $PBP^{-1}=A$
5	Matrices *A* and *B* have the same eigenvalues
6	$PB_{egv}=A_{egv}$, where B_{egv} and A_{egv} are the eigenvectors of the matrices *A* and *B*

The proof of above six properties of similar matrices is very simple and can be found in the course of linear algebra.

By applying six properties of similar matrices to the CNOT matrices (24) and (26), we will find that the properties second, fourth, fifth and six are not satisfied between CNOT matrices (24) and (26). Thus, the matrices (24) and (26) are not similar.

Physical interpretation of similar matrices

The similar matrices are very important part of the linear algebra in quantum computation. For example, by finding all similar matrices, we can get all the CNOT matrices or operators, which will reduce our mathematical and physical realization of quantum computing. If we apply all the six properties of similar matrices to CNOT matrices (26), we will see that matrices (26) are similar matrices. Now, if we apply operators C_{C1} and C_{C2} to the state $|\psi\rangle=a|0,0\rangle+b|0,1\rangle+c|1,0\rangle+d|1,1\rangle$, of Hamiltonian (23). We will obtain

$\lvert\psi_1\rangle$	=	$C_{C1}\lvert\psi\rangle$	=	$\begin{pmatrix}1&0&0&0\\0&1&0&0\\0&0&0&1\\0&0&-1&0\end{pmatrix}\begin{pmatrix}a(0,0)\\b(0,1)\\c(1,0)\\d(1,1)\end{pmatrix}$	=	$\begin{pmatrix}a(0,0)\\b(0,1)\\d(1,1)\\-c(1,0)\end{pmatrix}$		
$\lvert\psi_2\rangle$	=	$C_{C2}\lvert\psi\rangle$	=	$\begin{pmatrix}1&0&0&0\\0&0&0&1\\0&0&1&0\\0&-1&0&0\end{pmatrix}\begin{pmatrix}a(0,0)\\b(0,1)\\c(1,0)\\d(1,1)\end{pmatrix}$	=	$\begin{pmatrix}a(0,0)\\d(1,1)\\c(1,0)\\-b(1,0)\end{pmatrix}$		(27)

the state $\left|\psi_{1,2}\right\rangle$, which is obtained by operating C_{C1} and C_{C2}, and which gives us simple picture of the state before and after the CNOT operation. We can find all the *CNOT* matrices or similar matrices of (26).

Remark. Similar matrices of *CNOT* matrix (24) are have additional rotation, which complicates *CNOT* operation and which is not required for the *CNOT* operation. So, we are will find the similar matrices (26), which minimize *CNOT* operation in quantum computing.

CNOT similar matrices. As it is seen from the Eqs (27) that the state $\left|0,0\right\rangle$ in $\left|\psi\right\rangle$ is not used by the *CNOT* matrix. It means, we have still more options to use the possibility to get other *CNOT* matrices or similar matrices. This can be done by considering the six similar matrix properties.

C_{C11}	$=$	$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & -i \\ 0 & 0 & -i & 0 \end{pmatrix}$	C_{C51}	$=$	$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & -i \\ 0 & 0 & 1 & 0 \\ 0 & -i & 0 & 0 \end{pmatrix}$
C_{C22}	$=$	$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & -i \\ 0 & 0 & 1 & 0 \\ 0 & -i & 0 & 0 \end{pmatrix}$	C_{C52}	$=$	$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & i \\ 0 & 0 & i & 0 \end{pmatrix}$
C_{C31}	$=$	$\begin{pmatrix} 0 & 0 & -i & 0 \\ 0 & 1 & 0 & 0 \\ -i & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$	C_{C61}	$=$	$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix}$
C_{C32}	$=$	$\begin{pmatrix} 0 & -i & 0 & 0 \\ -i & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$	C_{C71}	$=$	$\begin{pmatrix} 0 & -1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$
C_{C11}	$=$	$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & -i \\ 0 & 0 & -i & 0 \end{pmatrix}$	C_{C72}	$=$	$\begin{pmatrix} 0 & 0 & -1 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$
C_{C41}	$=$	$\begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$	C_{C81}	$=$	$\begin{pmatrix} 0 & i & 0 & 0 \\ i & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$
C_{C42}	$=$	$\begin{pmatrix} 0 & 1 & 0 & 0 \\ -1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$	C_{C82}	$=$	$\begin{pmatrix} 0 & 0 & i & 0 \\ 0 & 1 & 0 & 0 \\ i & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$

We have obtained 16 *CNOT* matrices or operators, which are obtained by using the similar matrices properties. It means we can perform the CNOT operation on qubits with different pulse rotations around the different axes according to this convenience. The technique of similar matrices is very closely related to the

tensor product, which was above described. The similar matrices method can be applied to the other branches of quantum theory.

The Quantum Fourier Transform and Algorithms Based on It: From DFT to QFT

The classical discrete Fourier transform (DFT) on *N* input values (28), is defined as

$$y_k = \frac{1}{\sqrt{N}} \sum_{j=0}^{N-1} x_j \omega_N^{jk}, 0 \le k \le N-1, \tag{28}$$

with $\omega_N := e^{2\pi i/N}$ denoting the N^{th} root of unity. Its inverse transformation is $x_j = \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} y_k \omega_N^{-jk}$.

Remark. In some quotations the coefficient is $1/N$ instead of $1/\sqrt{N}$. In this case, the inverse transformation has no factor $1/\sqrt{N}$. If the x_j are values of a function f at some sampling points, it is often written f_j instead of x_j and $\hat{f}_k$ instead of y_k.

Considering the different notations the quantum Fourier transform (QFT) corresponds exactly to the classical DFT in (28). Here, the QFT is defined to be the linear operator with the following action on an arbitrary *n*-qubit state: $\sum_{j=0}^{N-1} x_j |j\rangle \xrightarrow{QFT} \sum_{k=0}^{N-1} y_k |k\rangle$

Remark. In literature the definition of the Fourier transform is not consistent. The DFT and its inverse transform are interchanged. In most original papers the QFT is defined as above. However, both the classical Fourier transform and its inverse transform can be deduced from the general form of a Fourier transform with $N = 2^n$. The y_k are the Fourier transforms of the amplitudes x_j as defined in Eq. (28). In product representation, the QFT looks like

$$|j_1,\ldots,j_n\rangle \to \frac{\left(|0\rangle + e^{2\pi i 0.j_n}|1\rangle\right)\left(|0\rangle + e^{2\pi i 0.j_{n-1}j_n}|1\rangle\right)\cdots\left(|0\rangle + e^{2\pi i 0.j_1 j_2 \cdots j_n}|1\rangle\right)}{2^{n/2}}. \tag{29}$$

This leads to an efficient circuit for the QFT which is shown in Fig. 1.

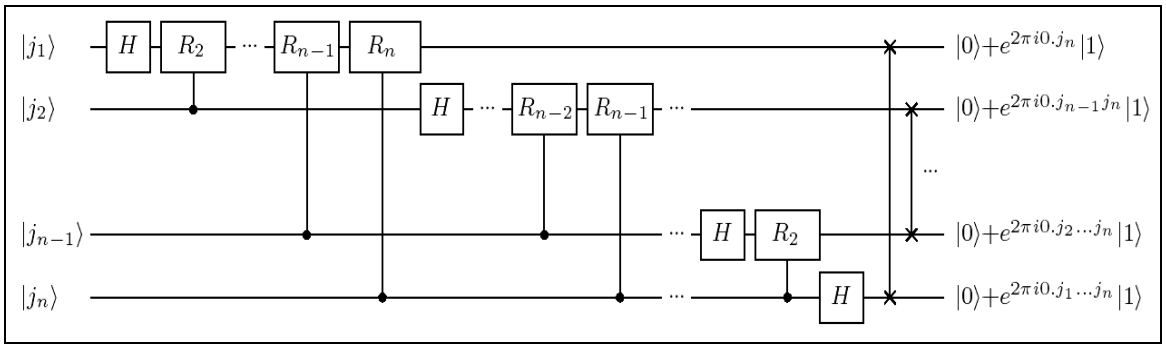

Figure 1. A Quantum circuit for QFT

The gate R_k corresponds to the phase gate $PH\left(2\pi/2^k\right)$. At the end of the circuit, swap gates reverse the order of the qubits to obtain the desired output. The circuit uses $n(n-1)/2$ Hadamard and conditional phase gates and at most $n/2$ swap gates in addition. Therefore, this *n*-qubit QFT circuit has a $\Theta\left(n^2\right) = \Theta\left(\log^2 N\right)$ runtime. Of course, QFT^{-1} can be performed in $\Theta\left(n^2\right)$ steps, too. In contrast, the best classical algorithms (like the Fast Fourier Transform) need $\Theta\left(n2^n\right)$ classical gates for computing the DFT

on 2^n elements. That is, the speedup of the QFT compared with the best classical algorithms is exponential. Note that the size of the circuit can be reduced to $O\left(npoly\left(\log n\right)\right)$ by neglecting the phase shifts of the first few bits as this amounts only requires only $O\left(n\log n\right)$. Nevertheless, there are two major problems concerning the use of the QFT: first, it is not known how to efficiently prepare any original state to be Fourier transformed, and second, the Fourier transformed amplitudes cannot be directly accessed by measurement.

Phase Estimation

An important application of the QFT is the phase estimation on which in turn many other applications are based. Given a unitary operator *U* with eigenvector (or eigenstate) $\left|u\right\rangle$, the phase estimation problem is to estimate the value ϕ of the eigenvalue $e^{2\pi i\phi}$ $\left(U\left|u\right\rangle = e^{2\pi i\phi}\left|u\right\rangle\right)$. To perform the estimation it is assumed that black-boxes (so-called oracles) are available which prepare the state $\left|u\right\rangle$ and perform the controlled-U^{2^l} operation for integers $l > 0$.

Figure 2 shows the phase estimation procedure. It works on two registers, a *n*-qubit register initially in state $\left|0\right\rangle$ (*n* depends on the number of digits of accuracy in the estimate for ϕ and the probability that the procedure is successful) and a register in the initial state $\left|u\right\rangle$.

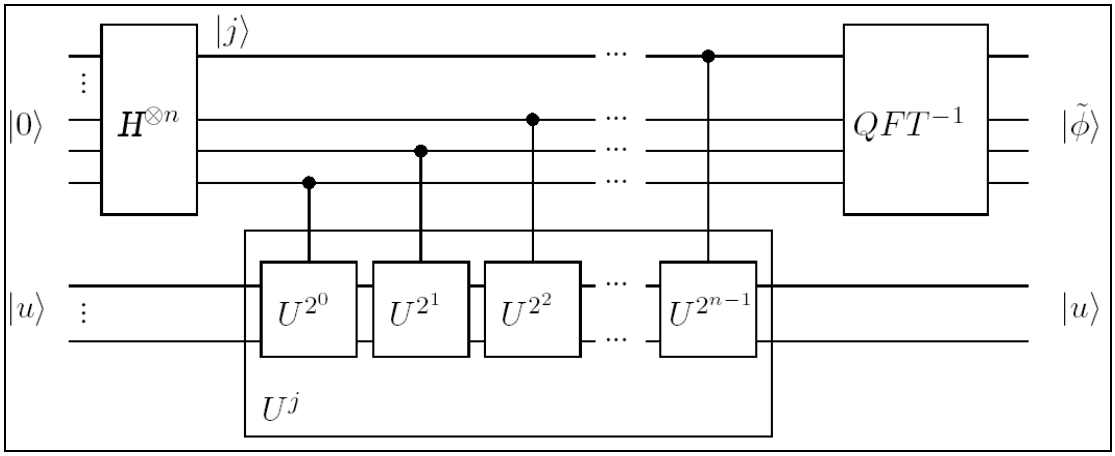

Figure 2. A quantum circuit for the face estimation procedure

The circuit begins by applying the *n*-qubit Hadamard transform $H^{\otimes n}$ to the first quantum register, follows by the application of controlled *U*-operations on the second register, with *U* raised to successive power of two. This sequence maps

$$\left|0\right\rangle\left|u\right\rangle \rightarrow \frac{1}{2^{n/2}}\sum_{j=0}^{2^n-1}\left|j\right\rangle U^j\left|u\right\rangle = \frac{1}{2^{n/2}}\sum_{j=0}^{2^n-1}e^{2\pi i\phi j}\left|j\right\rangle\left|u\right\rangle .$$

With $\phi = 0.\phi_1...\phi_n$ this state may be rewritten in product from which is equivalent to the product representation (Eq. (29)). If ϕ is exactly expressed in *n* qubits the application of the inverse QFT to the first qubit register leads to the state $\left|\phi\right\rangle = \left|\phi_1...\phi_n\right\rangle$, or a good estimator $\left|\tilde{\phi}\right\rangle$ otherwise.

Computing QFT^{-1} can be seen as the actual *phase estimation step*.

Quantum Algorithm: *Quantum phase estimation*	
1.	Create the equally weighted superposition: $\left|0\right\rangle\left|u\right\rangle \underline{H} \frac{1}{2^{n/2}}\sum_{j=0}^{2^n-1}\left|j\right\rangle\left|u\right\rangle$

2.	Apply the quantum oracle: $\xrightarrow{U^j} \frac{1}{2^{n/2}}\sum_{j=0}^{2^n-1}\|j\rangle U^j\|u\rangle = \frac{1}{2^{n/2}}\sum_{j=0}^{2^n-1} e^{2\pi i\phi j}\|j\rangle\|u\rangle$
3.	Calculate the inverse quantum Fourier transform and measure the first register: $\overset{QFT^{-1}}{\rightarrow}\|\tilde{\phi}\rangle\|u\rangle\overset{M_1}{\rightarrow}\tilde{\phi}$

A final measurement of the first register yields to the result ϕ or $\tilde{\phi}$ respectively. To estimate ϕ with *k*-bit accuracy and probability of success at least $1-\epsilon$, choose $n = k + \lceil \log(2+\frac{1}{2\epsilon}) \rceil$.

This algorithm, summarized below, computes the approximation $\tilde{\phi}_u$ to ϕ_u using $O(n^2)$ operations and one query to the controlled-U^j oracle gate.

Even if the eigenvector $|u\rangle$ is unknown and cannot be prepared, running the phase estimation algorithm on an arbitrary state $|\psi\rangle = \sum_u c_u|u\rangle$ (written in terms of eigenstate $|u\rangle$) yields a state $\sum_u c_u|\tilde{\phi}_u\rangle|u\rangle$, where $\tilde{\phi}_u$ is a good approximation to ϕ_u. Assuming that *n* is chosen as specified above, the probability for measuring ϕ_u with *k*-bit accuracy is at least $|c_u|^2(1-\epsilon)$.

Order-Finding and Other Applications

The *order-finding problem* reads as follows: for $x, N \in \mathbb{N}, x < N$, $\gcd(x,N)=1$, determine the least $r \in \mathbb{N}$, such that $x^r = 1 \bmod N$. It is believed to be a hard problem on classical computers. Let $n = \lceil \log N \rceil$ be the number of bits needed to specify *N*. The quantum algorithm for order-finding is just the phase estimation algorithm applied to the unitary operator $U|y\rangle \equiv |xy \bmod N\rangle$ with $y \in \{0,1\}^n$ and $|u\rangle = |1\rangle$, a superposition of eigenstates of *U* (for $N \le y \le 2^n - 1$, define $xy \bmod N := y$).

The entire sequence of controlled-U^{2^l} operations can be implemented efficiently using *modular exponentiation*: It is

$$|\mathrm{z}\rangle \mathrm{U}^{\mathrm{z}}|\mathrm{y}\rangle = |\mathrm{z}\rangle \mathrm{U}^{\mathrm{z}_{t-1}2^{t-1}} \ldots \mathrm{U}^{\mathrm{z}_0 2^0} = |\mathrm{z}\rangle\left|\left(\mathrm{x}^{\mathrm{z}^{t-1}2^{t-1}} \times \ldots \times \mathrm{x}^{\mathrm{z}_0 2^0}\right) \mathrm{ymod\,N}\right\rangle = |\mathrm{z}\rangle\left|\mathrm{x}^2 \mathrm{ymodN}\right\rangle.$$

In a first step modular multiplication is used to compute successively (by squaring) $x^{2j} \bmod N$ from x^{2j-1} for all $j = 1 \ldots t-1$. In a second step $x^z \bmod N$ is calculated by multiplying the *t* terms $\left(x^{z^{t-1}2^{t-1}} \bmod N\right) \ldots \left(x^{z_0 2^0} \bmod N\right)$. On the basis of this procedure the construction of the quantum circuit computing $U^z : |z\rangle|y\rangle \rightarrow |z\rangle|x^z \bmod N\rangle$ is straightforward, using $O(n^3)$ gates in total.

To perform the phase estimation algorithm, an eigenstate of *U* with a nontrivial eigenvalue or a superposition of such eigenstates has to be prepared. The eigenstates of *U* are

$$|u_s\rangle \equiv \frac{1}{\sqrt{r}}\sum_{k=0}^{r-1} \omega_r^{-sk}\left|x^k \bmod N\right\rangle,$$

for $0 \le s \le r-1$. Since $1/\sqrt{r}\sum_{s=0}^{r-1}|u_s\rangle = |1\rangle$ it is sufficient to choose $|u\rangle = |1\rangle$.

Now, applying the phase estimation algorithm on $t = 2n+1+\lceil \log(2+1/2\epsilon) \rceil$ qubits in the first register and a second quantum register prepared to $|1\rangle$ leads with a probability of a least $(1-\epsilon)/r$ to an estimate of the phase $\phi \approx s/r$ with $2n+1$ bits accuracy.

From this result, the order r can be calculated classically using an algorithm, known as the *continued fraction expansion*. It efficiently computes the nearest fraction of two bounded integers to ϕ, i.e. two integers s', t' with no common factor, such that $s'/r' = s/r$. Under certain conditions this classical algorithm can fail, but there are other methods to circumvent this problem.

An application of the order-finding quantum subroutine is Shor's factorization algorithm. Other problems which can be solved using the order-finding algorithm are period-finding and the discrete logarithm problem. These and some other problems can be considered in a more general context, which will be explained in the following two subsections.

Quantum Algorithm: *Order-finding*	
1.	Create superposition: $$\left\|0\right\rangle\left\|1\right\rangle \xrightarrow{H^{\otimes n}} \frac{1}{2^{t/2}}\sum_{j=0}^{2^t-1}\left\|j\right\rangle\left\|1\right\rangle$$
2.	Apply the black-box oracle $U_{x,N}:\left\|j\right\rangle\left\|k\right\rangle \to \left\|x^j k \bmod N\right\rangle$, with *x* co-prime to *N*: $$\xrightarrow{U_{x,N}} \frac{1}{2^{t/2}}\sum_{j=0}^{2^t-1}\left\|j\right\rangle\left\|x^j \bmod N\right\rangle \approx \frac{1}{2^{t/2}\sqrt{r}}\sum_{s=0}^{r-1}\sum_{j=0}^{2^t-1} e^{2\pi i s j/r}\left\|j\right\rangle\left\|u_s\right\rangle$$
3.	Calculate the inverse QFT of the first register and measure this: $$\xrightarrow{QFT^{-1}} \frac{1}{\sqrt{r}}\sum_{j=0}^{r-1}\left\|s/r\right\rangle\left\|u_s\right\rangle \xrightarrow{M_1}\left\|s/r\right\rangle$$

Fourier Transform on Arbitrary Groups

The more general definition of the Fourier transforms FT_G on an arbitrary group *G* needs some background in algebra and in representation theory over finite groups.

Let *G* be a finite group of order *N* and $f: G \to \mathbb{C}$. The *Fourier transform of f at the irreducible representation* ρ of *G* (denoted $\hat{f}(\rho)$) is defined to be the $d_\rho \times d_\rho$ matrix

$$f(\rho) = \sqrt{\frac{d_\rho}{N}}\sum_{g\in G} f(g)\rho(g). \tag{30}$$

Remark. A representation ρ of a finite group *G* is a homomorphism $\rho: G \to GL(V)$, where *V* is a $\mathbb{C}$-vector space. The dimension of *V* is called the *dimension of the representation* ρ, denoted d_ρ. ρ is said to be irreducible, if no other subspaces *W* are *G*-invariant, i.e. $\rho(g)W \subseteq W, \forall g \in G$, expect 0 and *V*. Up to isomorphism, a finite group has a finite number of irreducible representations. G denoted such a set of irreducible representations, which is a complete system of representatives of all isomorphism classes.

The collection of matrices $\left\langle f(\rho)\right\rangle_{\rho\in G}$ is designated as the *Fourier transform of f*. Thus, the Fourier transform maps *f* into $|G|$ matrices of varying dimensions which have totally $\sum_\rho d_\rho^2 = |G|$ entries. This means that the $|G|$ complex numbers $\left\langle f(g)\right\rangle_{g\in G}$ are mapped into $|G|$ complex numbers organized into

matrices. Furthermore, the Fourier transform is linear in $f\left(f_1+f_2(\rho)=f_1(\rho)+f_2(\rho)\right)$ and $f(\rho)$ is unitary for all $\rho\in G$.

If *G* is a finite Abelian group, all irreducible representations have dimension one. Thus, the representations correspond to the $|G|$ (irreducible) characters $\chi: G\to\mathbb{C}^{\times}$ of *G*. It is $(G,+)\simeq(\hat{G},\cdot)$. This is quite useful to simplify the notation by choosing an isomorphism $\mathfrak{g}\mapsto\chi_{\mathfrak{g}}$.

With the operation $(\chi_1\cdot\chi_2)(\mathfrak{g})=\chi_1(\mathfrak{g})\cdot\chi_2(\mathfrak{g})$ the set $\hat{G}$ of all characters of *G* is an Abelian group, called the dual group of *G*. Any value $\chi(\mathfrak{g})$ is a $|G|^{th}$ root of unity, i.e. $\chi(\mathfrak{g})^{|G|}=1$. For instance, if $G=\mathbb{Z}_N$ then the group of characters is the set $\{\chi_k(j)=\omega_N^{jk}\,|\,j,k=0...N-1\}$ and the Fourier transform corresponds to the classical DFT. For $G=\mathbb{Z}_2^n$ the Fourier transform $FT_{\mathbb{Z}_2^n}$ corresponds to the Hadamard transform $H^{\otimes n}$.

By the structure theorem of infinite Abelian groups, *G* is isomorphic to products of cyclic groups Z_{p_i} of prime power order with addition modulo p_i being the group operation i.e. $G\simeq\mathbb{Z}_{p_1}\times...\times\mathbb{Z}_{p_m}$. Any $\mathfrak{g}\in G$ can be written equivalently as $(\mathfrak{g}_1,...,\mathfrak{g}_m)$, where $\mathfrak{g}_i\in\mathbb{Z}_{p_i}$. This naturally leads to the description of the characters $\mathfrak{g}_i\in\mathbb{Z}_{p_i}$ on G as the product of the characters $\chi^{\mathbb{Z}_{p_i}}$ on $\mathbb{Z}_{p_i}$. It is

$$\chi_h^G(\mathfrak{g})=\prod_{i=1}^{m}\chi_{h_i}^{\mathbb{Z}_{p_i}}(\mathfrak{g}_i)=\prod_{i-1}^{m}e^{2\pi i\left(\sum_i \mathfrak{g}_i h_i q_i\right)},$$

Remark. The following mapping is used: $(\mathfrak{g}_1,...,\mathfrak{g}_m)\mapsto\prod_{i=1}^{m}\chi_{h_i}^{\mathbb{Z}_{p_i}}(\mathfrak{g}_i)$. With $h=(h_1,...,h_m)$ and $q_i:=N/p_i$. Note that $\chi_h(\mathfrak{g})=\chi_{\mathfrak{g}}(\mathfrak{h})$. With Eq. (30) the Fourier transform of a function $f: G\to\mathbb{C}$ may be written as $\hat{f}(h)=\frac{1}{\sqrt{N}}\sum_{\mathfrak{g}\in G}f(\mathfrak{g})\chi_{\mathfrak{g}}(h)$, with $h\in G$. Using the conventional notation, the QFT on an Abelian group *G* acts on the basis states as follows: $QFT_G|h\rangle=\frac{1}{|G|}\sum_{\mathfrak{g}\in G}\chi_{\mathfrak{g}}(h)|\mathfrak{g}\rangle$.

The Hidden Subgroup Problem

Nearly all know problems that have a quantum algorithm which provides an exponential speedup over the best known classical algorithm can be formulated as a *hidden subgroup problem* (HSP). Problem instances are for example order-finding, period-finding and discrete logarithm. The HSP is: Let *G* be a finitely generated group, $H<G$ a subgroup, *X* a finite set and $f: G\to X$ a function such that *f* is constant on cosets $\overline{\mathfrak{g}}=\mathfrak{g}H\in G/H$ and takes distinct values on distinct cosets, i.e. $f(\overline{\mathfrak{g}}_1)\neq f(\overline{\mathfrak{g}}_2)$ for $\overline{\mathfrak{g}}_1\neq\overline{\mathfrak{g}}_2$. Find a generating set for *H*.

Definite $H^{\circ}:=\{\mathfrak{g}\in G:\chi_{\mathfrak{g}}(h)=1,\forall h\in H\}$, also called the *annihilator group of H*. Note that H° is equivalent to the subgroup $H^{\perp}=\{\chi: H\subseteq\ker\chi\}$, which in turn is isomorphic to the dual of $H^{\perp}=\{\chi: H\subseteq\ker\chi\}$. The algorithm for solving the HSP uses the following two properties of the Fourier transform over *G*.

- The FT of the convolution $*$ of two vectors is the pointwise product of the FT of each vector:

$$QFT\left(\sum_k \alpha_k |k\rangle * \sum_l \beta_l |l\rangle\right) = \sqrt{|G|}\left[QFT\left(\sum_k \alpha_k |k\rangle\right) \cdot QFT\left(\sum_l \beta_l |l\rangle\right)\right]. \quad (31)$$

- The superposition state, uniform on *H*, is mapped to H°:

$$\frac{1}{\sqrt{|H|}} \sum_{h\in H} |h\rangle \xrightarrow{QFT} \sqrt{\frac{|H|}{|G|}} \sum_{k\in H^\circ} |k\rangle. \quad (32)$$

Using this, it follows for a coset state $1/\sqrt{|H|}\sum_{h\in H}|\mathrm{g}_0 h|$:

$$\frac{1}{\sqrt{|H|}} \sum_{h\in H} |\mathrm{g}_0 h| = |\mathrm{g}_0\rangle * \frac{1}{\sqrt{|H|}} \sum_{h\in H} |h\rangle$$

$$\xrightarrow{QFT_{GH}, EQ(32)} \sqrt{\frac{|H|}{|G|}}\left[\left(\sum_{\mathrm{g}\in G} \chi_{\mathrm{g}}(\mathrm{g}_0)|\mathrm{g}\rangle\right)\cdot\left(\sum_{k\in H^\circ}|k\rangle\right)\right] = \sqrt{\frac{|H|}{|G|}}\sum_{k\in H^\circ}\chi_k(\mathrm{g}_0)|k\rangle.$$

So, the QFT takes a coset state to the annihilator group states of the corresponding subgroup where the coset is encoded in the phase of the basis vectors.

Assume that *G* is Abelian. A hybrid algorithm to solve the Abelian HSP consists of the following quantum subroutine:

Quantum Algorithm: *Abelian HSP*	
1.	Create a random coset state: $\|0\rangle\|0\rangle \xrightarrow{QFT_G\otimes I} \frac{1}{\sqrt{\|G\|}}\sum_{\mathrm{g}\in G}\|\mathrm{g}\rangle\|0)\rangle \xrightarrow{U_f} \frac{1}{\sqrt{\|G\|}}\sum_{\mathrm{g}\in G}\|\mathrm{g}\rangle\|f(\mathrm{g})\rangle \xrightarrow{M_2} \frac{1}{\sqrt{\|H\|}}\sum_{h\in H}\|\mathrm{g}_0 h\rangle$
2.	Fourier sample the coset state: $\frac{1}{\sqrt{\|H\|}}\sum_{h\in H}\|\mathrm{g}_0 h\rangle \xrightarrow{QFT_G} \sqrt{\frac{\|H\|}{\|G\|}}\sum_{k\in H^\circ}\chi_k(\mathrm{g}_0)\|k\rangle \xrightarrow{M_1} k_i \in H^\circ$

M_i denotes the measurement of the *i*-th quantum register. The process of computing the Fourier transform over a group *G* and measuring subsequently is also called *Fourier sampling*. After applying M_2 , it is sufficient to observe the first register. The last measurement results in one out of $|G|/|H|$ elements $k_i \in H^\circ$ with probability $\frac{|H|}{|G|}$. Repeating this process $t = poly(\log|G|)$ times leads to a set of element which describe the $\chi_i \in H^\perp$. Thus, H can be classically computed as the intersection of the kernels of $\chi_i : H = \bigcap_{i=0}^{t-1} \ker \chi_i$.

The efficient quantum circuit for the Abelian HSP is shown in Fig. 3.

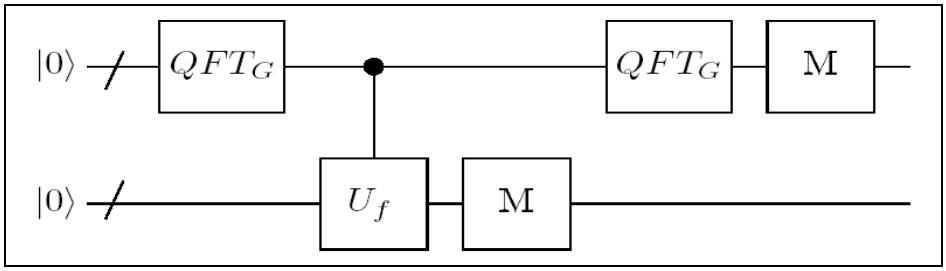

Figure 3. A quantum circuit for the Abelian HSP

Generalizations of the Abelian HSP quantum algorithm to the non-Abelian case have been attempted by many authors, unfortunately only with limited success. Up to now, the problem is still open, except for only a few particular instances.

A coarse outline on QFT-based algorithms

Table 2 shows an overview on problems efficiently solvable by means of quantum algorithm based on the QFT.

Table 2. The quantum Fourier transform and algorithms based on it.

Problem	Runtime
(discrete) QFT	$\Theta(n^2)$ or $O(n \log n)$ resp.
Deutsch †	1 oracle query
Deutsch-Jozsa †	1 oracle query
Bernstein-Vazirani	1 oracle query
Simon †	$O(n)$ repet. with 1 oracle query each
Period-finding † f with $f(x+r) = f(x)$, $x, r \in \mathbb{N}_0, 0 < r < 2^n$, a periodic function, output r!	1 oracle query, $O(n^2)$ operations
Phase estimation	$O(n^2)$ +1 oracle query
Order-finding †	$O(n^3)$
Factoring †	$O(n^3)$ operations, $O(n^2 \log n \log\log n)$
Discrete logarithm † given: $a, b = a^s \bmod N$ determine s	polyn. time QA
Hidden linear function problems †	polyn. time QA
Abelian stabilizer †	polyn. time QA
Shifted Legendre symbol problem and variants	polyn. time QA
Computing orders of finite solvable groups	polyn. time QA
Decomposing Finite Abelian Groups	polyn. time QA
Pell's Equation & Principal Ideal Problem	polyn. time QA

Problems marked with † are special instances of the HSP. The parameter n is the input length of the given problem.

The following explanations and annotations refer to single problems given below in the table.

Abelian group stabilizer problem: Let G be an Abelian group acting on a set X. Find the stabilizer $G_x := \{g \in G \mid g \cdot x = x\}$ for $x \in X$.

Decomposing finite Abelian groups: Any finite Abelian group G is isomorphic to a product of cyclic groups. Find such a decomposition. For many groups no classical algorithm is known which performs this task efficiently.

The quantum algorithm mainly uses the quantum algorithm for HSP to solve a partial problem. The rest is done classically.

An application of this algorithm is the efficient computing of class numbers (assuming the generalized Riemann Hypothesis).

Deutsch's problem: Determine whether a black-box binary function $f:\{0,1\}\to\{0,1\}$ is constant (or balanced).

Hidden linear function problem: Let $f:\mathbb{Z}^k\to S$ be a function for an arbitrary range S with $f(x_1,...,x_k)=h(x_1+\alpha_2 x_2+...+\alpha_k x_k)$ for a function h with period q and $\alpha_i\in\mathbb{Z}$. Recover the values of all the $\alpha_i \pmod q$ form an oracle for f. This problem is an instance of the HSP.

Inner product problem (Bernstein-Vazirani): For $a\in\{0,1\}^n$, let $f_a:\{0,1\}^n\to\{0,1\}$ be defined by $f_a(x)=a\cdot x$. Calculate a. This problem is not directly an instance of the HSP. Nevertheless, Fourier sampling helps finding a solution, too.

Orders of finite solvable groups: The problem is described.

Pell's Equation: Given a positive non-square integer d, Pell's equation is $x^2-dy^2=1$. Find integer solutions. The quantum algorithm calculates the regulator of the ring $\mathbb{Z}\left|\sqrt{d}\right|$, which is a closely related problem. The quantum step in this algorithm is a procedure to efficiently approximate the irrational period S of a function in time polynomial in $\ln S$.

Principal ideal problem: Given an ideal I, determine (if existing) an $\alpha\in\mathbb{Q}\left(\sqrt{d}\right)$ such that $I=\alpha\mathbb{Z}\left|\sqrt{d}\right|$. The algorithm reduces to a discrete log type problem.

Shifted Legendre symbol problem (SLSP): Given a function f_s and an odd prime p such that $f_s(x)=\left(\frac{x+s}{p}\right)$, for all $x\in\mathbb{Z}_p$, find s! Variants of this problem are the shifted Jacobi symbol problem and the shifted version of the quadratic character χ over finite fields $\mathbb{F}_q$ (shifted quadratic character problem). The classical complexities of these problems are unknown.

Simon's problem: Let $a\in\{0,1\}^n$ and $f:\{0,1\}^n\to\{0,1\}^n$ a function with $f(x\oplus a)=f(x)$ ($\oplus$: bitwise XOR). Calculate a. Simon's problem was the first that was shown to have an expected polynomial time quantum algorithm but no polynomial time randomizes algorithm. Brassard and Hoyer an exact quantum polynomial-time algorithm to solve this problem.

Two other transformations which can be recovered from the DFT are the discrete cosine transformation and the discrete sine transformation. Klappenecker and Rotteler show that both transformations of size $N\times N$ and types I-IV can be realized in $O\left(\log^2 N\right)$ operations on a quantum computer instead of $O(N\log N)$ on a classical computer. Another class of unitary transforms, the wavelet transforms, are efficiently implementable on a quantum computer. For a certain wavelet transform, a quantum algorithm is designed using the QFT.

Quantum Search Algorithms

Search problems are well-known and intensively investigated in computer science. Generally spoken, a search problem is to find one or more elements in a (finite or infinite, structured or unstructured) search space, which meet certain properties.

Suppose, a large database contains $N \gg 1$ items in a random order. On a classical computer it takes $O(N)$ comparisons to determine the items searched for. However, there is a quantum search algorithm, also called Grover's algorithm, which required only $O(\sqrt{N})$ operations.

Grover's algorithm is optimal for unstructured search problems.

Also in some cases of structured search spaces quantum algorithms can do better than classical as demonstrated by Hogg's algorithm for 1-SAT and highly constrained k-SAT.

Grover's Algorithm

It is assumed that $N = 2^n$. The database is represented by a function f which takes as input an integer $x, 0 \le x \le N-1$ (the database index), with $f(x) = 1$ if x is a solution to the search problem and $f(x) = 0$ otherwise. Let $\mathcal{L}$ be the set of solutions, i.e., $\mathcal{L} = \{x | f(x) = 1, x \in \{0, ..., N-1\}\}$ and $M = |\mathcal{L}|$ the number of solutions. Furthermore, let be $|\alpha\rangle = \frac{1}{\sqrt{N-M}} \sum_{x \notin \mathcal{L}} |x\rangle$, $|\beta\rangle = \frac{1}{\sqrt{M}} \sum_{x \in \mathcal{L}} |x\rangle$ and

$$|\phi\rangle = \sqrt{\frac{N-M}{N}} |\alpha\rangle + \sqrt{\frac{M}{N}} |\beta\rangle = \frac{1}{\sqrt{N}} \sum_{x=0}^{N-1} |x\rangle .$$

This algorithm works on two quantum registers $|x\rangle|y\rangle$, where $|x\rangle$ is the index register and $|y\rangle$ is a single ancilla qubit. Let U_α be the black-box which computes f. Then applied to the state $1/\sqrt{2}\,|x\rangle(|0\rangle - |1\rangle)$, the oracle "marks" all solutions by shifting the phase of each solution. Ignoring the single qubit register, the action of U_α may be written as $|x\rangle \to (-1)^{f(x)} |x\rangle$ or $U_\alpha = I - 2|\beta\rangle\langle\beta|$.

Geometrically, U_α induces a reflection about the vector $|\alpha\rangle$ in the plane defined by $|\alpha\rangle$ and $|\beta\rangle$. Another important operation, denoted with U_ϕ, is the so-called inversion about the mean:

$$U_\phi = H^{\otimes n} (2|0\rangle\langle 0| - I) H^{\otimes n} = 2|\phi\rangle\langle\phi| - I . \quad (33)$$

Applying U_ϕ to a general state $|\alpha\rangle = \sum_k \alpha_k |k\rangle$ leads to

$$U_\phi |\alpha\rangle = 2|s\rangle\langle s|a\rangle - |a\rangle = 2\sum_k A|k\rangle - \sum_k a_k |k\rangle = \sum_k (2A - a_k)|k\rangle$$

using state $\langle s|a\rangle = \frac{1}{\sqrt{2^n}} \sum_k a_k = \sqrt{2^n} A$, where $A = \frac{1}{\sqrt{2^n}} \sum_k a_k$ is the mean of the amplitude. Thus, the amplitudes are transformed as $U_\phi : a_k \to 2A - a_k$, i.e., the coefficient of $|k\rangle$ is reflected about the mean value of the amplitude. In other words, U_ϕ is a reflection about the vector ϕ in the spanned by $|\alpha\rangle$ and $|\beta\rangle$ The combination of both operator U_α and U_ϕ leads to the Grover operator, defined to be

$$U_G = U_\phi U_\alpha = -H^{\otimes n} (2|0\rangle\langle 0| - I) H^{\otimes n} (2|\beta\rangle\langle\beta| - I) \quad (34)$$

(see Fig. 4).

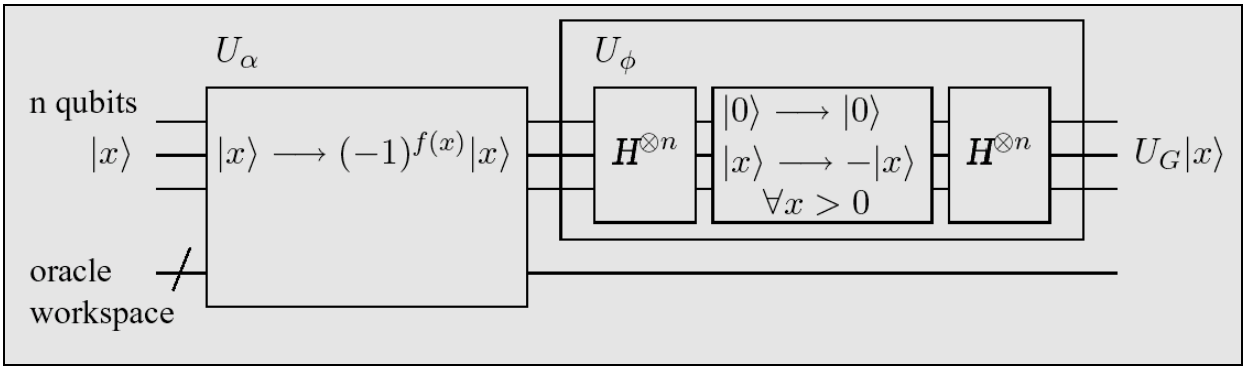

Figure 4. A quantum circuit for the Grover operator

Thus, the product of the two reflections U_α and U_ϕ is a rotation in the two-dimensional subspace spanned by $|\alpha\rangle$ and $|\beta\rangle$ rotating the space by an angle θ, defined by $\sin\theta/2 = \sqrt{M/N}$, as shown in Fig.5.

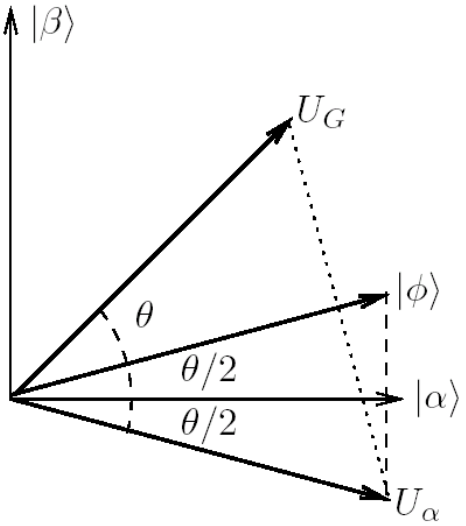

Figure 5. Geometric visualization of the single step of the Grover iteration

Repeating the Grover operator $T \simeq \pi/4\sqrt{N/M} = O\left(\sqrt{N/M}\right)$ times rotated the initial system state ϕ close to $|\beta\rangle$. Observation of the state I the computational basis yields a solution to the search problem with probability at least $p \geq 1/2$. When $M \ll N$ this probability is at least $1 - M/N$, that is nearly 1.

Figure 6 illustrates the entire quantum search algorithm. It should be mentioned that the quantum search algorithm is optimal, i.e. no quantum algorithm can perform the task of searching *N* items using fewer than $\Omega\left(\sqrt{N}\right)$ accesses to the search oracle.

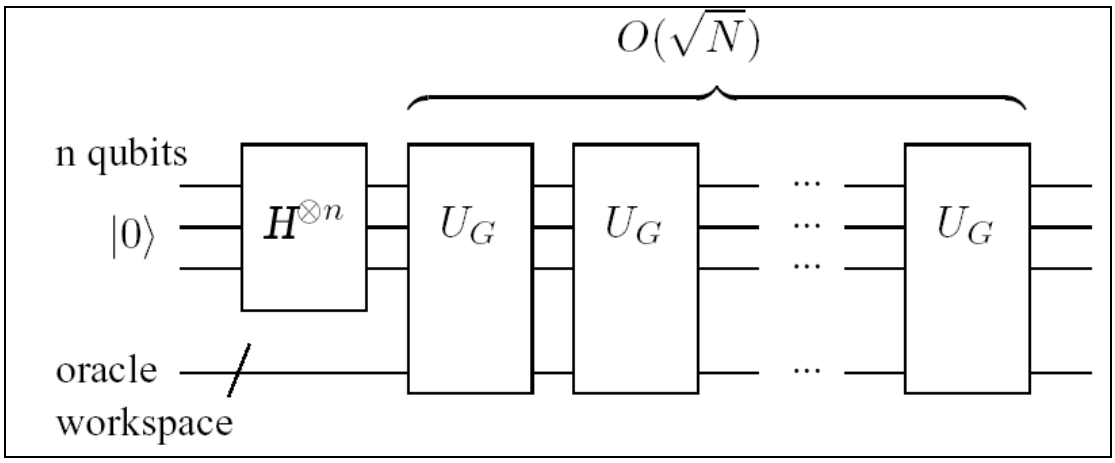

Figure 6. Quantum circuit for the QSA

A concluding note: In the basis $\{|\alpha\rangle,|\beta\rangle\}$ the Grover operator may also be written as

$$U_G = \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix}$$

where $0 \le \theta \le \pi/2$, assuming without limitation that $M \le N/2$. Its eigenvalues are $e^{\pm i\theta}$. If it is not known whether $M \le N/2$, this can be achieved for certain by adding a single qubit to the search index, doubling the number of items to be searched to $2N$. A new augmented oracle ensured that only those items are marked which are solutions and whose extra bit is set to zero.

Remark. A generalization of the Grover iteration to boost the probability of measuring a solution state is called amplitude amplification. According to Gruska, there are at least two (only slightly different) methods which meet this condition, one proposed by Grover, the other by Brassard et al. The most general version of a Grover operator is presented by Biham at al. All methods have in common, that they use an arbitrary unitary transformation *U* instead of the Hadamard transformation $H^{\otimes n}$ as in the original Grover operator (Eq. (1.7)). The following transformation might be deemed as the *generalized Grover operator*:

$$U_G = -UI_s^{\beta}U^{-1}I_f^{\gamma},$$

where $I_s^{\beta} = I - \left(1 - e^{i\beta}\right)|s\rangle\langle s|$ is a rotation of a fixed basis state $|s\rangle$ by an angle β and $I_f^{\gamma} = \sum_x e^{i\gamma f(x)}|x\rangle\langle x|$ is a rotation by an arbitrary phase γ.

Quantum Counting: Combining Grover Operator and Phase Estimation.

Provided that *M* is known, Grover's algorithm can be applied as described above. If *M* is not known in advance, it can be determined by applying the phase estimation algorithm to the Grover operator U_G, estimating one of its phases $\pm\theta$.

Figure 7 shows a quantum circuit for performing approximate quantum counting.

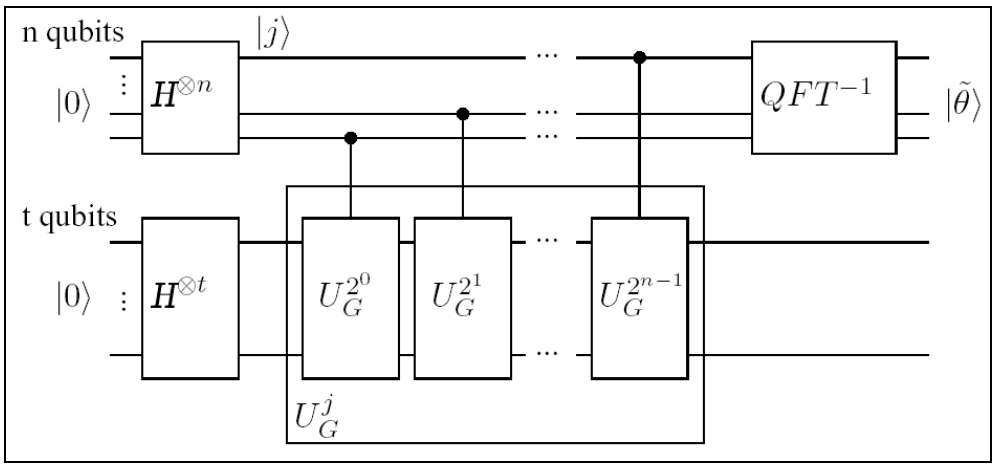

Figure 7. Circuit for the quantum counting algorithm

An application of the phase estimation procedure to estimate the eigenvalues of the Grover operator U_G which enables to determine the number of solutions M to the search problem. The first register contains n qubits and the second register contains t qubits, sufficient to implement Grover's operator on the augmented search space of size $2N$. The state of the second register is initialized to $\sum_x |x\rangle$ by $H^{\otimes n}$.

But this is a superposition of the two eighenvectors of U_G with corresponding eighenvalues $e^{i\theta}$ and $e^{i(2\pi-\theta)}$. Therefore applying the phase estimation procedure, results in the estimate for θ or $2\pi-\theta$ which is equivalent to the estimate for $\theta\left(\sin^2(\theta/2)\right)=\sin^2((2\pi-\theta)/2)$ From the equation $\sin^2\left(\theta/2\right)=M/2N$ and the estimate for θ it follows as estimate for the number of solutions M.

Remark. The search space is expanded to $2N$ to ensure that $M \le N/2$. As already described above this is done by adding a single qubit to the search space.

Further analysis shows, the error in this estimate for M is less than $\left(\sqrt{2MN}+\frac{N}{2^{k+1}}\right)2^{-k}$, provided that θ has a k-bit accuracy. Summarizing, the algorithm requires $O\left(\sqrt{N}\right)$ oracle calls to estimate M to high accuracy. Finding a solution to a search problem when M is unknown requires to apply both algorithms, first the quantum counting and then the quantum search algorithm. Errors arisen in the estimate for θ and M affect the total probability to find a solution to the search problem. However, the probability can be increased close to 1 by a few repetitions of the combined counting-search algorithm.

Counting the number of solutions and, consequently, determining the solvability of a search problem has many applications, including decision variants of NP-complete problems.

Applications of Grover's Algorithm

The quantum search algorithm or at least its main operator can be applied to solve many kinds of search problems. Table 3 presents a survey of these problems. Some of them are now described briefly.

Table 3. Algorithms based on quantum searching.

Problem	Runtime
Quantum database search (Grover's algorithm)	$O\left(\sqrt{N}\right)$ quadratic speedup
Quantum counting	$O\left(\sqrt{N}\right)$
Scheduling problem	$O\left(N^{3/4}\log N\right)$

Minimum-finding	$O\left(\sqrt{N}\right)$
Database retrieval	$N/2+O\left(\sqrt{N}\right)$
String matching	$O\left(\sqrt{n}\log\sqrt{\frac{n}{m}}\log m+\sqrt{m}\log^2 m\right)$
Weighing matrix problem	2 oracle queries
Element distinctness Collision funding Claw finding Triangle-finding	comparison complexity always better than classical

Claw finding: Given two functions $f:X\to Z$ and $g:Y\to Z$, find a pair $(x,y)\in X\times Y$ such that $f(x)=g(y)$.

Collisions finding: Given a function $f:X\to Y$, find two different $x_1,x_2\ (x_1\neq x_2)$, such that $f(x_1)=f(x_2)$ under the promise that such a pair exists.

Database retrieval: Given a quantum oracle which returns $|k,y\otimes X_k\rangle$ on an $n+1$ qubit query $|k,y\rangle$, the problem is to obtain all $N=2^n$ bits of X_k.

Element distinctness: given a function $f:X\to Y$, decide whether *f* maps different $x\in X$ to different $y\in Y$.

Minimum-finding: Let $T[0...N-1]$ be an unsorted table of *N* (distinct) items. Find the index y of an item such that $T[y]$ is minimal.

Scheduling problem: Given two unsorted lists of length N each. Find the (promised) single common entry. The complexity is measured in quantum memory accesses and accesses to each list.

String matching: Determine whether a given pattern p of length m occurs in a given text t of length n. The algorithm combines quantum searching algorithms with deterministic sampling, a technique from parallel string matching. Grover's search algorithm is applied twice in conjunction with a certain oracle in each step.

Triangle-finding: Given an undirected graph $G=(V,E)$, find distinct vertices $a,b,c\in V$ such that $(a,b),(a,c),(b,c)\in E$.

Weighing matrix problem: Let M be a $W(n,k)$ weighing matrix.

Remark. A matrix $M\in\{-1,0,+1\}^{n\times n}$ is called a weighing matrix, iff $M\cdot M^T=k\cdot I_n$ for some $k,0\le k\le n$.

A set of n functions $f_s^M\in\{1,...,n\}\to\{-1,0,+1\}$ for $s\in\{1,...,n\}$ is defined by $f_s^M(i):=M_{si}$. Determine s.

Quantum Search and NP Problems

Solving problems in the complexity class NP may also be speed up using quantum search. The entire search space of the problem (e.g. orderings graph vertices) is represented by a string of qubits. This strings has to be read as defined by the problem specifications (e.g. the string consists of blocks of the same length

storing as index of a single vertex). In order to apply the quantum search algorithm, the oracle which depends on the problem instance must be designed and implemented. It marks those qubit strings which represent a solution. As the verification of whether a potential solution meets the requirements is much easier than the problem itself, even on a classical computer, it is sufficient to convert the classical circuit to a reversible circuit. Now, this circuit can be implemented on a quantum computer. Thus, the quantum counting algorithm which determines whether or not a solution to the search problem exists requires the square root of the number of operations that the classical "brute-force" algorithm requires. Roughly speaking, the complexity changes from $O\left(2^{\theta(n)}\right)$ to $O\left(2^{\theta(n)/2}\right)$, where θ is some polynomial in n. Nonetheless, the complexity is still exponential and in the sense of complexity theory these problems are not efficiently solved.

Hogg's *Algorithm*

Certain structured combinatorial search problems can be solved effectively on a quantum computer as well, even outperforming the best classical heuristics. Hogg introduced a quantum search algorithm for 1-SAT and highly constrained *k*-SAT.

The satisfiability problem (SAT). A satisfiability problem (SAT) consists of a logical formula in *n* variables $v_1,...,v_n$ and the requirement to find an assignment $a=\left(a_1,...,a_n\right)\in\left\{0,1\right\}^n$ for the variables that makes the formula true. For *k*-SAT the formula is given as a conjunction of m clauses, where each clause is a disjunction of *k* literals v_i or $\overline{v}$ respectively with $i\in\left\{1...n\right\}$. Obviously there are $N=2^n$ possible assignments for a given assignment are called conflicts. Let $c\left(a\right)$ denote the number of conflicts for assignment a. A k-SAT problem is called maximally constrained if the formula has the largest possible number of clauses for which a solution still exists. Thus, any conceivable additional clause will prevent the satisfiability of the formula. For $k\geq 3$ the satisfiability problem is NP-complete and in general, the computational costs grow exponentially with the number of variables *n*. For $k=1, k=2$, and some k-SAT problems with a certain problem structure, there are classical algorithms which require only $O\left(n\right)$ *search steps*, that is, the number of sequentially examined assignments. Also, quantum algorithms can exploit the structure of these problems to improve the general quantum search (ignoring any problem structure) as perform better than classical algorithms.

Hogg's quantum algorithm for 1-SAT. Hogg's quantum algorithm for 1-SAT primarily requires *n* qubits, one for each variable. A basis state specifies the value assigned to each variable and consequently, assignments and basis state correspond directly to ach other. Information about the particular problem to be solved is accessible by a special diagonal matrix *R* (of dimension $2^n\times 2^n$), where

$$R_{aa}=i^{c(a)} \tag{35}$$

is the matrix entry at position (*a*,*a*). Thus, the problem description is entirely encoded in this input matrix by the number of conflicts in all 2^n possible assignments to the given logical formula. For the moment it is assumed that such a matrix can be implemented efficiently. Further implementation details will be given in the next paragraph.

Figure 8 exemplifies the input matrix *R* for a 1-SAT problem with $n=3$ variables.

$$\begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & i & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & i & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & i & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & -i \end{pmatrix}$$

Figure 8. Input matrix for he local formula $f(v_1, v_2, v_3) = \bar{v}_1 \wedge \bar{v}_2 \wedge \bar{v}_3$. *The matrix has diagonal coefficients* ($i^{c(000)}, ..., i^{c(111)}$). *Off instance the assignment a=101* ($v_1 = true, v_2 = false, v_3 = true$) *makes two clauses false, i.e. c(101)=2 and* $i^2 = -1$

Furthermore, let U be the matrix defined by

$$U_{rs} = 2^{-n/2} e^{i\pi(n-m)/4} (-i)^{d(r,s)},$$

where $d(r,s)$ is the Hamming distance between two assignments r and s, viewed as bit strings. *U* is independent *f* the problem and its logical formula. The first step of the quantum algorithm is the preparation of an unbiased, i.e. equally weighted, superposition. An already demonstrated, this is achieved by applying the Hadamard gate on the *n* qubits $(H^{\otimes n})$ initially is state $|0\rangle$.

Let $|\psi\rangle$ denote the resulting quantum state. Then, applying *R* and *U* to $|\psi\rangle$ gives $|\phi\rangle = UR|\psi\rangle$.

It was proven that the final quantum state is the equally weighted superposition of all assignments *a* with $c(a) = 0$ conflicts. Thus, considering a soluble 1-SAT problem, a final measurement will lead with equal probability to one of the 2^{n-m} solutions. If there is no solution of the problem, the measurement will return a wrong result. Hence, finally the resulting assignment has to be verified.

Implementation. The matrices *R* and *U* have to be decomposed into elementary quantum gates to build a practical and implementable algorithm. The operation *R* can be performed using a reversible (quantum) version of the classical algorithm to count the number of conflicts of a 1-SAT formula and a technique to adjust the phases which are powers of i. Therefore, two ancillary qubits are necessary which are prepared in the superposition

$$|\Psi\rangle = \frac{1}{2}(|00\rangle - i|01\rangle - |10\rangle + i|11\rangle),$$

where the local phases correspond to the four possible values of R_{aa} for any assignment *a*. As suggested by Hogg, the superposition $|\Psi\rangle$ can be constructed by applying *H* on qubit 1 and $iR_x(3/4\pi)$ on qubit 0, both of the qubits being initially prepared to $|1\rangle$. Then, the reversible operation

$$F : |a, x\rangle \to |a, x + c(a) \bmod 4\rangle$$

acts on $|\psi, \Psi\rangle$ as follows

$$|\psi, \Psi\rangle \to 2^{-n/2} \sum_a i^{c(a)} |a\rangle |\Psi\rangle$$

performing the required operation *R*. To see this, further intermediate calculation steps are necessary. Note that after applying *F*, the ancillary qubits reappear in the original superposition form and can therefore be dropped, since they do not influence *U*. In a single application of $F, c(a)$ is evaluated once.

The matrix *U* can be implemented in terms of two simpler matrices $H^{\otimes n}$ and Γ, where Γ is diagonal with $\Gamma_{aa} = i^{|a|}$. Here $|a|$ is the number of 1-bits in the string of assignment *a*. Using this definition, it is $U \equiv H^{\otimes n} \Gamma H^{\otimes n}$. For implementing Γ, Hogg suggests to use similar procedures to those for the implementation of the matrix *R*, using a quantum routine for counting the number of 1-bits in each assignment instead of the number of conflicts. Thus, the elements of the matrix Γ can be computed easily and the operation *U* is efficiently computable in $O(n)$ bit operations: 2^n Hadamard gates plus another $C \cdot n$ elementary single qubit gates, with a constant $C \geq 1$, fir the computation of Γ (to give a rough estimation). It is shown as a result of the GP evolution of a 1-SAT quantum algorithm that $U = R_x(3/4\pi)^{\otimes n}$

, and consequently it can be implemented even more efficiently by n elementary rotation gates. Once again, Hogg's algorithm consists of the following steps:

Quantum Algorithm: *1-SAT / maximally constrained k-SAT*	
1.	Apply $H^{\otimes n}$ on $\|0\rangle$.
2.	Compute the number of conflicts with the constraints (clauses) of the problem into the phases by applying the input gate *R*.
3.	Applying $U = H^{\otimes n}\Gamma H^{\otimes n}$ results in an equally weighted superposition of solution states.

An experimental implementation of Hogg's 1-SAT algorithm for logical formulas in three variables was demonstrated.

Performance. The matrix operations and the initialization of $|\psi\rangle$ contribute $O(n)$ bit operations to the overall costs. Evaluating the number of conflicts results in costs of $O(m)$ for a k-SAT problem with m clauses. In total the costs of the quantum algorithm amount to $O(n+m)$. This corresponds to the costs of a single search step of a classical search algorithm which is based on examinations of neighbors of assignments. While the quantum algorithm examines all assignments in a single step, the best (local search based) classical algorithm needs $O(n)$ search steps.

A slide modification of Hogg's algorithm for 1-SAT can also be applied to maximally and highly constrained *k*-SAT problems for arbitrary *k* to find a solution with high probability in a single step. For all 1-SAT and also maximally constrained 2-SAT problems, Hogg's algorithm finds a solution with probability one. Thus, an incorrect result definitely indicates the problem is not soluble.

Note that a comparison of Hogg's algorithm with any classical algorithms according to the number of search steps is only permissible, if the classical algorithms are based on local search, especially on the examination of the number of conflicts. Otherwise, the comparison must be based on a more fundamental measure. Local search is not always the best solution. For instance, it is unreasonable to solve 1-SAT using local search since it is trivial to find an assignment satisfying the given Boolean formula in $O(m)$.

Quantum Simulation

The third class of quantum algorithms consists of those algorithms which simulate quantum mechanical systems. Commonly, the problem of simulating a quantum system is classically (at least) as difficult as simulating a quantum computer. This is due to the exponential growth of the Hilbert space which comprises the quantum states of the system. Therefore, simulation of quantum systems by classical computers is possible, but in general only very inefficiently. A quantum computer can perform the simulation of some dynamical systems much more efficiently, but unfortunately not all information from the simulation is accessible. The simulation leads inevitable to a final measurement collapsing the usually superimposed simulation state into a definite basis state. Nevertheless, quantum simulation seems to be an important application of quantum computers.

A rough outline of the quantum simulation algorithm is presented now. Further reading matter are the original papers dealing with the simulation of quantum physical systems on a quantum computer.

Simulating a quantum system means "predicting" the state of the system at some time t_f (and/or position) as accurately as possible given the initial system state. Simulation is mainly based on solving differential equations. Unfortunately, the number of differential equations increases exponentially with the dimension of the system to be simulated. The quantum counterpart to the simple differential equation $dy/dt = f(y)$ in classical simulations is $i\,d|\psi\rangle/dt = H|\psi\rangle$ for a Hamiltonian *H*. Its solution for a time-independent *H* is $|\phi(t)\rangle = e^{-iHt}|\phi(0)\rangle$. However, e^{-iHt} is usually difficult to compute. High order solutions

are possible especially for those Hamiltonians which can be written as sums over local interactions: $H = \sum_{k-1}^{L} H_k$, acting on a *n*-dimensional system, $L = poly(n)$ where each Hamiltonian H_K acts on a small subsystem. Now, the single terms e^{-iHt} are much easier to approximate by means of quantum circuits than e^{-iHt}. How to calculate e^{-iHt} from $e^{-iH_k t}$?

In general $e^{-iHt} \neq \prod_k e^{-iH_k t}$ because of $\left[H_j, H_k\right] \neq 0$. Instead, approximate e^{-iHt} with e.g.

$$e^{i(A+B)\Delta t} = e^{iA\Delta t/2} e^{iB\Delta t} e^{iA\Delta t/2} + O\left(\Delta t^3\right).$$

This and other approximations are derived from the *Trotter formula*

$$\lim_{n\to\infty}\left(e^{iAt/n} e^{iBt/n}\right)^n = e^{i(A+B)t}.$$

Now, let the *n*-qubit state $\left|\tilde{\psi}\right\rangle$ approximate the system. Assume further, that the operators $e^{-iH_k \Delta t}$ have efficient quantum circuit approximations. Then, the approximation of $e^{i\left(\sum_k H_k\right)\Delta t}$ can be efficiently implemented by a quantum circuit $U_{\Delta t}$. With it, the quantum simulation algorithm can be formulated as follows:

Quantum Algorithm: *Quantum simulation algorithm*	
1.	Initialization: $\left\|\tilde{\psi}_0\right\rangle$ (initial system state at time $t = 0$), $j = 0$;
2.	Iterative evolution: **do** $\left\|\tilde{\psi}_{j+1}\right\rangle = U_{\Delta t}\left\|\tilde{\psi}_j\right\rangle$; $j = j+1$; until $j\Delta t \geq t_f$
3.	Output: $\left\|\tilde{\psi}\left(t_f\right)\right\rangle = \left\|\tilde{\psi}_j\right\rangle$

Speedup Limits for Quantum Algorithms

This section summarizes some theoretical results about the limitations of quantum computing. The methodologies to prove lower bounds for the complexity and the speedup of quantum algorithms are not the subject matter, but they can be read up in the given literature.

An interesting result on limitations of quantum algorithm refers to black-box algorithms. Many quantum algorithms use oracle or black-box algorithm. Accessing the black-box can be thought of as a special subroutine call or query whose invocation only costs unit time. Speaking more formally, let $X = \left(x_0, ..., x_{N-1}\right)$ be a such an oracle, containing *N* Boolean variables $x_i \in \{0,1\}$. On input i the oracle returns x_i. A property *f* of *X* which is any Boolean function $f : \{0,1\}^N \to \{0,1\}$ is only determinable by oracle queries. The complexity of a black-box algorithm is usually rated by the number of queries necessary to compute the property. Then, Beals et al. prove that black-box quantum algorithm for which no inner structure is known, i.e., for which no promises are made restricting the function. Then, the function is said to be total. Can achieve maximally a polynomial speedup compared to probabilistic and deterministic classical algorithm. In addition, they specify exact bounds for the (worst case) quantum complexity of AND, OR, PARITY and MAJORITY for different error models (assuming exact -, Las Vegas – and Monte-Carlo algorithms). For the exact setting (the algorithm returns the correct result with certainty) the quantum complexities are N for OR and AND, N/2 for PARITY and $\Theta(N)$ for MAJORITY. The bound for the parity problem was independently obtained by Farhi et al.

As shown by Buhrman at al. the following problems cannot be solved more efficiently on a quantum computer:

Parity-collision problem: Given a function $f : X \to Y$, find the parity of the cardinality of the set $\left\{\left(x_1, x_2\right) \in X \times X \,\middle|\, x_1 < x_2 \wedge f\left(x_1\right) = f\left(x_2\right)\right\}$.

No-Collision problem: Given a function $f : X \to Y$, find an element $x \in X$ that is not involved in a collision, i.e., $f^{-1}\left(f\left(x\right)\right) = \left\{x\right\}$.

No-range problem: Given a function $f : X \to Y$, find an element $y \in Y$ such that $y \notin f\left(X\right)$.

Hoyer at al. determined the quantum complexity for further problems.

Ordered searching: Given a sorted list of *N* numbers $\left(x_0, x_1, ..., x_{N-1}\right), x_i \le x_{i+1}$ and a number $y \le x_{N-1}$, find the minimal index *i* such that $y \le x_i$. This problem can be regarded as a non-Boolean promise problem. The best known quantum lower bound is $1/\pi\left(\ln\left(N\right)-1\right)$ queries. A quantum algorithm using $\log_3\left(N\right)+O\left(1\right) \approx 0.631\log_2\left(N\right)$ queries is presented. A slightly better algorithm ($0.526\log_2\left(N\right)$ queries) is given.

Sorting: Given a list of numbers $\left(x_0, x_1, ..., x_{N-1}\right)$, output a permutation σ on the index set such that the list $\left(x_{\sigma_0}, x_{\sigma_1}, ..., x_{\sigma_{N-1}}\right)$ is in non-decreasing order. The quantum lower bound for this problem is given by $\Omega\left(N \log N\right)$ binary comparisons.

Element distinctness: Given a list of numbers $\left(x_0, x_1, ..., x_{N-1}\right)$. Decide whether they are all distinct. The problem has a quantum lower bound of $\Omega\left(\sqrt{N} \log N\right)$ binary comparisons.

References

1. Gruska J. Quantum computing. – Advanced Topics in Computer Science Series, McGraw-Hill Companies, London. – 1999.
2. Nielsen M.A. and Chuang I.L. Quantum computation and quantum information. – Cambridge University Press, Cambridge, Englandю – 2000.
3. Hirvensalo M. Quantum computing. – Natural Computing Series, Springer-Verlag, Berlinю – 2001.
4. Hardy Y. and Steeb W.-H. Classical and quantum computing with C++ and Java Simulations. – Birkhauser Verlag, Basel. – 2001.
5. Hirota O. The foundation of quantum information science: Approach to quantum computer (in Japanese). – Japan. – 2002.
6. Pittenberg A.O. An introduction to quantum computing and algorithms. – Progress in Computer Sciences and Applied Logic. – Vol. 19. – Birkhauser. – 1999.
7. Brylinski F.K. and Chen G. (Eds). Mathematics of quantum computation. – Computational Mathematics Series. – CRC Press Co. – 2002.
8. Lo H.-K., Popescu S. and Spiller T. (Eds). Introduction to quantum computing and information. – World Scientific Publ. Co. – 1998.
9. Berman G.P., Doolen G.D., Mainieri R. and Tsifrinovich V.I. Introduction to quantum computers. – World Scientific Publ. Co. – 1999.
10. Rieffel E. and Polak W. An introduction to quantum computing for non-physicists // ACM Computing Surveys. – 2000. – Vol. 32. – No 3. – pp. 300 – 335.
11. Hogg T., Mochon C., Polak W. and Rieffel E. Tools for quantum algorithms // International Journal of Modern Physics. – 1999. – Vol. C10. – No 7. – pp. 1347 – 1361.
12. Uesaka Y. Mathematical principle of quantum computation (in Japanese). – Corona Publ. Co. Ltd. – 2000.

13. Marinescu D.C. and Marinescu G.M. Approaching quantum computing. – Pearson Prentice Hall, New Jersey. – 2005.
14. Benenti G., Casati G. and Strini G. Principles of quantum computation and information. –Singapore: World Scientific. – Vol. I. – 2004; – Vol. II. – 2007.
15. Nakahara M. and Ohmi T. Quantum computing: From Linear Algebra to Physical Realizations. – Taylor & Francis. – 2008.
16. Stenholm S. and Suominen K.-A. Quantum approach to informatics. – Wiley- Interscience. A J. Wiley&Sons, Inc. – 2005.
17. Jaeger G. Quantum Information: An Overview. – N.Y.: Springer Verlag. – 2007.
18. McMahon D. Quantum computing explained. – Wiley- Interscience. A J. Wiley&Sons, Inc. – 2008.
19. Steeb W.-H. and Hardy Y. Matrix Calculus and Kronecker Product: A Practical Approach to Linear and Multilinear Algebra, 2nd edition. – World Scientific, Singapore. – 2011.
20. Hardy Y, Steeb W.-H. and Kemp G. Matrices, Fermi operators and applications // arXiv:1708.05289v1 [quant-ph], 15 Aug 2017.