Mathematical approaches for modified quantum calculation

Nikolay Raychev

Abstract - in this report is proposed a programming technique for presentation of an operation modifier as an operation and are examined some of the mathematics around this programming technique.

Keywords: boolen function, circuit, composition, encoding, gate, phase, quantum.

1. INTRODUCTION

In this report is examined some alternative constructions of qubit operators, which include more than one control or target bit, based on a formalized qubit operator. This work is part of the developed from the author formalized system for design of algorithmic models for quantum circuits, based on phase encoding, decoding and parameterization of primitive quantum operators. In previous publications of the author [6, 7, 8] were defined several sets of operators on the n qubit, which generalize certain classical characteristics: identity and logical negation. It has been proven that the space is enough to capture BQP, the class of tasks, efficiently solvable by quantum computations [1, 2]. Moreover, some ways were explored in which can be constructed operators as linear combinations of elements from those sets. Such combinations capture the partial application of an operator together with another operator that in a broader sense is its logical negation.

2. OPERATIONS AND CIRCUITS AS MATRICES

Each operation on a circuit, whether the circuit is classical, probabilistic or quantum, can be presented as a matrix.

The presentation of operations as matrices facilitates the finding of intuitive solutions. In this way easily could be achieved the overall effect of a circuit into a single operation: Simply the matrices should be multiplied together. Also the combination of independent operations, which are applied to different lines becomes straightforward: the Kronecker's product must be used.

Intuitively the Kronecker's product $\boldsymbol{A}\otimes\boldsymbol{B}$ works by superimposing $\boldsymbol{B}$ inside of $\boldsymbol{A}$, then each tile is scaled by the coefficient it was paired with. For example, let's assume that must be applied a Hadamard gate $\boldsymbol{H}$ to one line and a NOT gate $\boldsymbol{X}$ to another line, similar to the following:

—H—
—X—

The overall matrix of the circuit is computed in the following way:

$$H = \frac{1}{\sqrt{2}}\begin{bmatrix}1 & 1\\ 1 & -1\end{bmatrix}$$

$$X = \begin{bmatrix}0 & 1\\ 1 & 0\end{bmatrix}$$

$$X \otimes H = \frac{1}{\sqrt{2}}\begin{bmatrix}0\otimes H & 1\otimes H\\ 1\otimes H & 0\otimes H\end{bmatrix}$$

$$= \frac{1}{\sqrt{2}}\begin{bmatrix}0_2 & H\\ H & 0_2\end{bmatrix}$$

$$= \frac{1}{\sqrt{2}}\begin{bmatrix}0 & 0 & 1 & 1\\ 0 & 0 & 1 & -1\\ 1 & 1 & 0 & 0\\ 1 & -1 & 0 & 0\end{bmatrix}$$

Also the Kronecker's product should be used when the other line doesn't have an operation, to expand the

IJSER

matrix of the operation, so as to be applied to the larger vector of the state of the larger circuit. (To avoid affecting the state of the other lines, must be used the Kronecker's product against the identity matrix.)

Another way to apply an operation to more lines is to be controlled.

Control matrix

Controlled operations are operations conditioned to only occur if a certain control line is ON. In the diagrams the control line is displayed by covering it with a small black circle and connecting it to the other operation with a straight line:

```
──•──
  │
──X──
```

Sometimes when creating quick ASCII diagrams of circuits is omitted the connecting line. A side effect of this is that the control appears as an independent operation:

```
──•──
──X──
```

When someone looks to the upper diagram, his first thought is "What is the matrix for this strange operation •?"

Of course, there is no matrix for •, since the controls are not operations. The controls are operation modifiers. The attempt to compute the matrix for a controlled operation by calculating $X\otimes\ \bullet$ is a simply a type error. Even it is not completely wrong...

What would happen if the rules of the arithmetics are changed a little bit, so there is no matrix for the so-called "gate •"?

The programming technique which is used includes introducing a special value called **μ**. In the code **μ** is only an instance of the Complex class. There is a real part 1 and imaginary part 0. In this way it acts as a normal 1 anywhere. This is the case with the exception of the code for the Kronecker's product, which is an exceptional case:

```
def q_kronecker_product(m1, m2):
    w1, h1 = len(m1), len(m1[0])
    w2, h2 = len(m2), len(m2[0])
    return [[
        q_controlled_product(m1[i1][j1], m2[i2][j2], i1, i2, j1,
j2)
        for i1 in range(w1), i2 in range(w2)]
        for j1 in range(h1), j2 in range(h2)]

def q_controlled_product(v1, v2, i1, i2, j1, j2):
    if v1 is Q_SPECIAL_CONTROL_ONE:
        return Q_SPECIAL_CONTROL_ONE if i2==j2 else 0
    if v2 is Q_SPECIAL_CONTROL_ONE:
        return Q_SPECIAL_CONTROL_ONE if i1==j1 else 0
    return v1*v2
```

In practice the upper code is saying: when one matrix is put into another, each nested matrix, which gets paired with **μ**, is replaced with a matrix with **μ** along the diagonal. In other words, $\boldsymbol{\mu}\otimes\boldsymbol{U}$ is defined to be different from $\boldsymbol{\mu}\cdot\boldsymbol{U}$. Instead of $\boldsymbol{\mu}\otimes\boldsymbol{U}=\boldsymbol{\mu}\cdot\boldsymbol{U}$ is available $\boldsymbol{\mu}\otimes\boldsymbol{U}=\boldsymbol{\mu}\cdot\boldsymbol{I}$. For example,

$$X \otimes \mu = \frac{1}{\sqrt{2}} \begin{bmatrix} \mu & 0 \\ 0 & \mu \end{bmatrix}$$

If the new value **μ** is given, it is easy to be made a "gate •". When the input line is OFF, the operations are replaced by the identity matrix, so that it can be scaled with Kronecker by **μ**. When the input line is ON, the operations are applied, so that it can be scaled with Kronecker by 1. Thus the matrix of the control gate is defined to be:

$$C = \begin{bmatrix} \mu & 0 \\ 0 & 1 \end{bmatrix}$$

The introducing of **μ** and **C** is a useful programming technique, because going around the Kronecker's product is comparatively little resource intensive compared with adding logic for marking and generating controlled operations.

Marked numbers

We can do more things with this value **μ**. It can be added, multiplied, squared, etc.

IJSER

A lot of interesting numerical systems begin by introducing a new value, with special rules related to squaring. If a value **i** is entered, whose square is -1, the complex numbers are obtained. The complex numbers are useful for working with rotational quantities in 2d. If, instead of this is entered a value **ϵ**, whose square is 0, are obtained the double numbers. The double numbers make the numerical differentiation very easy, because $f(x+\epsilon)-f(x)=\epsilon\frac{\mathrm{d}}{\mathrm{dx}}f(x)$. If a value **j** is entered, whose square is +1, are obtained the hyperbolic numbers. The hyperbolic numbers behave as the time and space in a special relativity.

So the squaring looks good, when a behavior should be defined. In the case of **μ** the semantics, which is looked for, is an approximate restriction in a certain approximation. **μ** can be used as a marker, by means of which after multiplication it can be said what was was controlled and what was not. Taking this into account is defined $\boldsymbol{\mu^2}$ to be again **μ**.

For now there is no standard name for a numeric system, created by adding **μ**, such that $\boldsymbol{\mu^2}=\boldsymbol{\mu}$. The lack of name may be due to the fact that it is a basis change different from isomorphic to hyperbolic numbers. In this article these numbers will be called Marked numbers, since **μ** changes the values in a way that can not be canceled.

Each time when a numeric system is defined, the first thing to examine is how the typical operations behave. Is the multiplication commutative? Or associative? Does dividing have borderline cases in which it can not be divided? Do functions like $\boldsymbol{e^x}$ make something new?

For example, let's look at the raising of the marked number $a+b\mu$ to the power **n.**

First $(a+b\mu)^n$ is expanded with the aid of the binomial theorem:

$$=\sum_{i=0}^{n}\binom{n}{i}a^i(b\mu)^{n-1}$$

Now let's subtract the only member that does not receive coefficient **μ**:

$$=a^n+\mu\sum_{i=0}^{n-1}\binom{n}{i}a^ib^{n-1}$$

And then fill the hole in the sum:

$$=a^n+\left(-a^n+\sum_{i=0}^{n}\binom{n}{i}a^ib^{n-1}\right)\mu$$

After eliminating the application of the binomial theorem is obtained the answer:

$$(a+b\mu)^n=a^n+(a+b)^n\mu-a^n\mu$$

Another good operation for testing is the exponential function. A definition of $\boldsymbol{e^x}$ is selected, usually the Taylor series $\boldsymbol{e^x}=\sum_{n=0}^{\infty}\frac{x^n}{n!}$ works very well, and is seen what happens after applying the definition to $\boldsymbol{e^{a+b\mu}}$

The same thing, which makes Euler, to show that $\boldsymbol{e^{\pi i}=-1}$ $\boldsymbol{e^{a+b\mu}}$ is revealed in:

$$=\sum_{n=0}^{\infty}\frac{(a+b\mu)^n}{n!}$$

The numerator of the addendums can be simplified by using the raised to n-th power equivalence:

$$=\sum_{n=0}^{\infty}\frac{a^n+(a+b)^n\mu-a^n\mu}{n!}$$

Now that there are additions (and subtractions) in the infinite sum, it may be broken up into three infinite sums. This is not always a safe step (caution should be taken for conditionally convergent series):

$$=\sum_{n=0}^{\infty}\frac{a^n}{n!}+\mu\sum_{n=0}^{\infty}\frac{(a+b)^n}{n!}-\mu\sum_{n=0}^{\infty}\frac{a^n}{n!}$$

Each of the sums matches the definition of the series of $\boldsymbol{e^x}$. After the sums are replenished back in the form of $\boldsymbol{e^x}$, is obtained a good solution:

$$\boldsymbol{e^{a+b\mu}}=\boldsymbol{e^a}+(\boldsymbol{e^{a+b}}-\boldsymbol{e^a})\mu$$

It should be noted that the exponentiating and raising to a power are influenced in similar ways when they are generalized to work on Marked numbers. In both cases is obtained

$$f(a+b\mu)=f(a)+(f(b)-f(a))\mu$$

That's not a coincidence.

Let's take into account the matrices $I=\begin{bmatrix}1&0\\0&1\end{bmatrix}$ and $M=\begin{bmatrix}0&0\\0&1\end{bmatrix}$. It must be noted that $I\cdot M=M$ and $M\cdot M=M$, just like $1\cdot\mu=\mu$ и $\mu\cdot\mu=\mu$. The adding, scaling, and

multiplication of $\boldsymbol{I}$ and $\boldsymbol{M}$ also behave isomorphically relative to their behavior for **1** and **μ**. This means that $\boldsymbol{I}$ and $\boldsymbol{M}$, and their linear combinations may be used to represent Marked numbers.

The number $a + b\mu$ can be translated into the matrix $\begin{bmatrix} a & 0 \\ 0 & a+b \end{bmatrix}$, after which to be translated facts for such type of matrix back to facts for Marked numbers.

The eigenvalues of the matrix $\begin{bmatrix} a & 0 \\ 0 & a+b \end{bmatrix}$ are just **a** and **a + b**. A good rule of thumb for applying functions to a matrix is to decompose the matrix into its eigenvalues/vector parts, to transform its eigenvalues with the function in question, and then to assemble the matrix again. Therefore $f(a + \mu b)$ ends up from the point of $f(a)$ and $f(a + b)$ because **a** and **a + b** are the eigenvalues, which are transformed. The backing up of the **μ** part, the new **b**, requires subtracting off the added part **a**. From there the pattern

$$f(a + \mu b) = f(a) + \mu(f(a + b) - f(a)).$$

With this ends the tangent to the main abstract algebra. Let's again examine the operations on circuits.

Merging operations into controls

When there is a circuit such as this:

```
──H─X─H──
   │
──H─•─H──
```

The Hadamard gates on the top line definitely can not be merged into a NOT gate. This would mean that they are controlled by the bottom line, which changes the behavior of the circuit. But what would happen if the bottom Hadamard gates are multiplied in the control? Then it would be obtained:

$$H \cdot C \cdot H$$

$$= \frac{1}{2}\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \cdot \begin{bmatrix} \mu & 0 \\ 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$$

$$= \frac{1}{2}\begin{bmatrix} 1 \cdot \mu + 1 \cdot 0 & 1 \cdot 0 + 1 \cdot 1 \\ 1 \cdot \mu - 1 \cdot 0 & 1 \cdot 0 - 1 \cdot 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$$

$$= \frac{1}{2}\begin{bmatrix} \mu & 1 \\ \mu & -1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$$

$$= \frac{1}{2}\begin{bmatrix} \mu + 1 & \mu - 1 \\ \mu - 1 & \mu + 1 \end{bmatrix}$$

Let's now have a "strange control" with Marked numbers for all of its entries. What would happen if the strange control is combined with **X**, using the Kronecker's product with a special **μ** case?

$$X \otimes (H \cdot C \cdot H)$$

Let's first outline the familiar solution in the matrix of the **X** gate:

$$= \frac{1}{2}\begin{bmatrix} 0 \otimes \begin{bmatrix} \mu + 1 & \mu - 1 \\ \mu - 1 & \mu + 1 \end{bmatrix} & 1 \otimes \begin{bmatrix} \mu + 1 & \mu - 1 \\ \mu - 1 & \mu + 1 \end{bmatrix} \\ 1 \otimes \begin{bmatrix} \mu + 1 & \mu - 1 \\ \mu - 1 & \mu + 1 \end{bmatrix} & 0 \otimes \begin{bmatrix} \mu + 1 & \mu - 1 \\ \mu - 1 & \mu + 1 \end{bmatrix} \end{bmatrix}$$

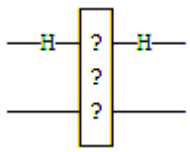
The alignment of the upper expression into a single matrix is a little complicated. **μ** are on the right side this time, so the diagonal, where **μ** must be placed, is more difficult to be seen. Mainly all **μ** in the top-left and bottom-right sections remain **μ**, while **μ** in the top-right and bottom-left are replaced with 0. It is also confusing, that 1 are added/subtracted from **μ**, using the normal rules for Kronecker's product instead of the special case for following the diagonal:

$$= \frac{1}{2}\begin{bmatrix} \mu & \mu & 1 & -1 \\ \mu & \mu & -1 & 1 \\ 1 & -1 & \mu & \mu \\ -1 & 1 & \mu & \mu \end{bmatrix}$$

Since it is not necessary to track what is controlled, **μ** may be eliminated by applying the function: $cleaning(a + \mu b) = a + b$

$$= \frac{1}{2}\begin{bmatrix} 1 & 1 & 1 & -1 \\ 1 & 1 & -1 & 1 \\ 1 & -1 & 1 & 1 \\ -1 & 1 & 1 & 1 \end{bmatrix}$$

The circuit looks like this:



In order to obtain the matrix for the entire circuit, must be multiplied the last two Hadamard gates:

$$(H \otimes H) \cdot (X \otimes C) \cdot (H \otimes H)$$

$$= (H \otimes I) \cdot (X \otimes (H \cdot C \cdot H)) \cdot (H \otimes I)$$

IJSER

$$= \frac{1}{2}\begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & -1 & 0 \\ 0 & 1 & 0 & -1 \end{bmatrix} \cdot \frac{1}{2}\begin{bmatrix} 1 & 1 & 1 & -1 \\ 1 & 1 & -1 & 1 \\ 1 & -1 & 1 & 1 \\ -1 & 1 & 1 & 1 \end{bmatrix} \cdot \frac{1}{2}\begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & -1 & 0 \\ 0 & 1 & 0 & -1 \end{bmatrix}$$

Which is equivalent to:

$$= \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

What is the circuit:

In other words: The surrounding of a controlled NOT with operations of Hadamard from all sides will swap on which line are the control and the NOT. This is actually a well-known programming technique, but the fact that the computing is correct, shows that the merging of operations into controls is safe. It was shown that $(C \otimes U) \cdot (V \otimes I) = (C \cdot V) \otimes U$ (and this is preserved, when all $\otimes$ members and/or all $\cdot$ members are reversed).

For the merging of operations into controls can be thought as a modification of the controls for applying in a different basis. For example, since the Hadamard gate swaps between the bases **X** and **Z**, the merging into a Hadamard operation on each side of the control causes the control to be applied to **X** observable instead to **Z** observable (the **Z** observable is the usual computing base).

Multiple controls

Do things continue to work when there are multiple controls? Let's examine a Toffoli gate:

First the controls are combined with each other:

$C \otimes C$

$= C^{\otimes 2}$

$$= \begin{bmatrix} \mu \otimes \begin{bmatrix} \mu & 0 \\ 0 & 1 \end{bmatrix} & 0 \otimes \begin{bmatrix} \mu & 0 \\ 0 & 1 \end{bmatrix} \\ 0 \otimes \begin{bmatrix} \mu & 0 \\ 0 & 1 \end{bmatrix} & \mu \otimes \begin{bmatrix} \mu & 0 \\ 0 & 1 \end{bmatrix} \end{bmatrix}$$

$$= \begin{bmatrix} \mu & 0 & 0 & 0 \\ 0 & \mu & 0 & 0 \\ 0 & 0 & \mu & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

It should be noted that the entire diagonal is built from **μ**, with the exception of the bottom-right value (1). This pattern continues for all Kronecker powers $C^{\otimes n}$ of *C*. (The resulting matrix can be set briefly in a bracket notation:

$$C^{\otimes n} = \mu I_{2^n} + (1 - \mu) \, |2^n - 1\rangle \langle 2^n - 1|$$

Let's calculate the matrix of a Toffoli gate:

$C^{\otimes n} \otimes X$

$$= \begin{bmatrix} \mu & 0 & 0 & 0 \\ 0 & \mu & 0 & 0 \\ 0 & 0 & \mu & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \otimes X$$

$$= \begin{bmatrix} \mu \otimes X & 0 & 0 & 0 \\ 0 & \mu \otimes X & 0 & 0 \\ 0 & 0 & \mu \otimes X & 0 \\ 0 & 0 & 0 & 1 \otimes X \end{bmatrix}$$

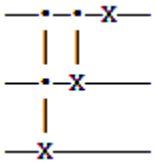
$$= \begin{bmatrix} \mu & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & \mu & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \mu & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \mu & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \mu & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & \mu & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}$$

The above matrix is in fact the correct matrix for the Toffoli gate (or will be after applying ***cleaning***), i.e. the things continue to work when there are multiple controls. Let's now attempt to merge larger operations into larger controls.

Merging multiple operations into multiple controls multiple times

A frequent task in the computing is the incrementation. Fortunately the circuits that increment, are quite simple for obtaining from controlled NOTs. Here is one that increments three bits:

IJSER



The pattern continues just as it is expected. In order to carry out an incrementation on more bits, simply are added slightly larger controlled-NOTs in front.

This particular pattern of gates used for incrementation from controlled NOTs, is particularly interesting, since each operation has controls on all the lines, affected by the smaller operations. From here, it follows that these smaller operations can be merged into the controls and to simplify

$(I \otimes I \otimes X) \cdot (I \otimes X \otimes C) \cdot (X \otimes C \otimes C)$ to use fewer large matrix multiplications.

First it must be noticed that the Kronecker's product is distributed over the matrix multiplication. This allows to be simplified the sub-expression $(I \otimes I \otimes X) \cdot (I \otimes X \otimes C)$ in $I \otimes \big((I \otimes X) \cdot (X \otimes C)\big)$. Also it may be suggested that $(I \otimes X) \cdot (X \otimes C)$ is simplified in $X \otimes (X \cdot C)$

Let's calculate this simplified sub-expression:

$X \otimes (X \cdot C)$

$$= X \otimes \begin{bmatrix} 0 & 1 \\ \mu & 0 \end{bmatrix}$$

$$= \begin{bmatrix} 0 \otimes \begin{bmatrix} 0 & 1 \\ \mu & 0 \end{bmatrix} & 1 \otimes \begin{bmatrix} 0 & 1 \\ \mu & 0 \end{bmatrix} \\ 1 \otimes \begin{bmatrix} 0 & 1 \\ \mu & 0 \end{bmatrix} & 0 \otimes \begin{bmatrix} 0 & 1 \\ \mu & 0 \end{bmatrix} \end{bmatrix}$$

$$= \begin{bmatrix} 0 & 0 & 0 & 1 \\ \mu & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & \mu & 0 \end{bmatrix}$$

In the above matrix, the output (the row with non-zero entry) is always one more than the input column. This is a matrix for incrementing 2 bits.

Knowing the 2-bit case, it can be calculated the case with 2 bits from the start:
$(I \otimes I \otimes X) \cdot (I \otimes X \otimes C) \cdot (X \otimes C \otimes C)$

The distributed **I** is subtracted:
$= (I \otimes ((I \otimes X) \cdot (X \otimes C))) \cdot (X \otimes C^{\otimes 2})$

X is subtracted by merging the operations in the mentioned **X** controls:
$= X \otimes ((I \otimes X \cdot (X \otimes C))) \cdot C^{\otimes 2})$

The inner **X** by merging the operations in the mentioned **X** controls:
$= X \otimes ((X \otimes (X \cdot C))) \cdot C^{\otimes 2})$

The 2-bit increment gate is expanded:

$$= X \otimes \begin{bmatrix} 0 & 0 & 0 & 1 \\ \mu & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & \mu & 0 \end{bmatrix} \cdot C^{\otimes 2}$$

The matrix multiplication is calculated:

$$= X \otimes \begin{bmatrix} 0 & 0 & 0 & 1 \\ \mu & 0 & 0 & 0 \\ 0 & \mu & 0 & 0 \\ 0 & 0 & \mu & 0 \end{bmatrix}$$

The Kronecker's product is calculated:

$$= \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ \mu & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & \mu & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \mu & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \mu & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & \mu & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \mu & 0 \end{bmatrix}$$

And with this, the process ends.

If thinking more abstractly, the original construction for the matrix was to be made a triangle of controlled NOT operators. More specifically:
$Inc(n) = \prod_{i=1}^{n} (I^{\otimes i-1} \otimes X \otimes C^{\otimes i-1})$

The goal of the new construction is to make a smaller increment but merged into a new controlled NOT gate. More specifically: $Inc(n) = X \otimes (Inc(n-1) \cdot C^{\otimes i-1})$.

The reason this to work comes down to **μ**. First, the smaller increment gate is multiplied by the controls This causes all non-zero elements to become **μ**, but leaves 1 only in the top-right corner. After this the Kronecker's product expands all those **μ** into $\begin{bmatrix} \mu & 0 \\ 0 & \mu \end{bmatrix}$, by duplicating them in the top-left and bottom-right quadrants of the new larger increment matrix. The only breakthrough in the new diagonal is the top-right corner of the bottom-left quadrant, but it is filled from the top-right 1, being expanded into $\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$. This is how a correct increment matrix is achieved from a smaller increment.

The benefit of this strategy for the computing is in the size of the multiplications. Initially, fondly, were

IJSER

performed $\mathbf{n-1}$ matrix multiplications of size $2^{\mathbf{n}} \times 2^{\mathbf{n}}$. By performing the most multiplications in a less recursive step, instead of this is done one 2 x 2 dot matrix multiplication, one 4 x 4 dot matrix multiplication, one 8 x 8 dot matrix multiplication and so on until $2^{\mathbf{n}} \times 2^{\mathbf{n}}$. This is a factor for speeding up the runtime for performing $\mathbf{n}$ compared to the naive strategy. This speeding up can be achieved even automatically by introducing a optimization of type "merging operations into controls when possible".

Applicability of the time optimization

Let's examine the time optimization, achieved by merging operations into controls. A better way to achieve this optimization would be simply to recognize that an operation with $\mathbf{m}$ controls affects at most $2^{\mathbf{n-m}}$ amplitudes. By taking only this subset of amplitudes is achieved the same speedup in a much simpler way.

In addition, if the focus is on optimization, matrix multiplications for X gates should not be used. For example, incrementing is just an operation rotation of an array by 1. At a circular array the time is constant! Another problem is that the optimizations due to $\boldsymbol{\mu}$ will be combined difficultly with other optimizations, because $\boldsymbol{\mu}$ violates certain mathematical identities. For example, it is no longer the case that
$(X \otimes Y) \cdot (Z \otimes T) = (X \cdot Z) \otimes (Y \cdot T)$

Each optimization that implicitly relies on it for correctness, will have to check for $\boldsymbol{\mu}$ and $\mathbf{C}$ before that. Whether or not it's useful, $\boldsymbol{\mu}$ is definitely interesting for experimentation.

3. SUMMARY

By introducing a special value $\boldsymbol{\mu}$, which the Kronecker's product uses in special cases, the concept of using a line as a control for that operation can be turned into an operation.

$\boldsymbol{\mu}$ is a neat way to determine controlled operations, but in the end, it is not very practical, because it violates some useful mathematical identities.

REFFERENCES

[1] M. A. Nielsen and I. L. Chuang, *Quantum Computation and Quantum Information*, 1st ed. (Cambridge University Press, Cambridge, UK, 2000).
[2] P. W. Shor, SIAM Journal on Computing 26, 1484 (1997).
[3] L. K. Grover, Physical Review Letters 79, 325 (1997).
[4] C. H. Bennett and G. Brassard, in *Proceedings of IEEE international Conference on Computers, Systems and Signal Processing, Bangalore, India* (IEEE Press, New York, 1984), p. 175.
[5] A. K. Ekert, Phys. Rev. Lett. 67, 661 (1991).
[6] C. Elliott, New Journal of Physics 4, 46 (2002). 12
[7] C. Elliott, D. Pearson, and G. Troxel, in *Proceedings of the ACM SIGCOMM 2003 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication, August 25-29, 2003, Karlsruhe, Germany.* (PUBLISHER, ADDRESS, 2003), pp. 227–238.
[8] C. Elliott, IEEE Security & Privacy 2, 57 (2004).
[9] C. Elliott *et al.*, in *Current status of the DARPA quantum network (Invited Paper)*, edited by E. J. Donkor, A. R. Pirich, and H. E. Brandt (SPIE, ADDRESS, 2005), No. 1, pp. 138–149.
[10] J. H. Shapiro, New Journal of Physics 4, 47 (2002).
[11] B. Yen and J. H. Shapiro, IEEE Journal of Selected Topics in Quantum Electronics 9, 1483 (2003).
[12] S. Lloyd *et al.*, SIGCOMM Comput. Commun. Rev. 34, 9 (2004).
[13] I.-M. Tsai and S.-Y. Kuo, IEEE Transactions on Nanotechnology 1, 154 (2002).
[14] S.-T. Cheng and C.-Y. Wang, IEEE Transactions on Circuits and Systems I: Regular Papers 53, 316 (2006).
[15] J. C. Garcia-Escartin and P. Chamorro-Posada, Phys. Rev. Lett. 97, 110502 (2006).
[16] M. Oskin, F. T. Chong, and I. L. Chuang, Computer 35, 79 (2002).
[17] D. Copsey *et al.*, IEEE Journal of Selected Topics in Quantum Electronics 9, 1552 (2003).
[18] C. H. Bennett and S. J. Wiesner, Physical Review Letters 69, 2881 (1992).
[19] X. S. Liu, G. L. Long, D. M. Tong, and F. Li, Phys. Rev. A 65, 022304 (2002).
[20] A. Grudka and A. W´ojcik, Phys. Rev. A 66, 014301 (2002).
[21] C.-B. Fu *et al.*, JOURNAL OF THE KOREAN PHYSICAL SOCIETY 48, 888891 (2006).
[22] A. Winter, IEEE Transactions on Information Theory 47, 3059 (2001).
[23] H. Concha, J.I.; Poor, IEEE Transactions on Information Theory 50, 725 (2004).
[24] M. Fujiwara, M. Takeoka, J. Mizuno, and M. Sasaki, Physical Review Letters 90, 167906 (2003).
[25] J. R. Buck, S. J. van Enk, and C. A. Fuchs, Phys. Rev. A 61, 032309 (2000).
[26] M. Huang, Y. Zhang, and G. Hou, Phys. Rev. A 62, 052106 (2000).
[27] B. J. Yen and J. H. Shapiro, in *Two Problems in Multiple Access Quantum Communication*, edited by S. M. Barnett *et al.* (AIP, ADDRESS, 2004), No. 1, pp. 25–28.
[28] B. J. Yen and J. H. Shapiro, Physical Review A (Atomic, Molecular, and Optical Physics) 72, 062312 (2005).
[29] B. Sklar, IEEE Communications Magazine 21, 6 (1983).
[30] B. Sklar, *Digital Communications*, 2nd ed. (Prentice Hall, Upper Saddle River, New Jersey 07458, 2000).

IJSER

[31] P. D. Townsend, Nature 385, 47 (1997).
[32] V. Fernandez *et al.*, in *Quantum key distribution in a multi-user network at gigahertz clock rates*, edited by G. Badenes, D. Abbott, and A. Serpenguzel (SPIE, ADDRESS, 2005), No. 1, pp. 720–727.
[33] Nikolay Raychev. Dynamic simulation of quantum stochastic walk. In International jubilee congress (TU), 2012.
[34] Nikolay Raychev. Classical simulation of quantum algorithms. In International jubilee congress (TU), 2012.
[35] Nikolay Raychev. Interactive environment for implementation and simulation of quantum algorithms. CompSysTech'15, DOI: 10.13140/RG.2.1.2984.3362, 2015
[36] Nikolay Raychev. Unitary combinations of formalized classes in qubit space. International Journal of Scientific and Engineering Research 04/2015; 6(4):395-398. DOI: 10.14299/ijser.2015.04.003, 2015.
[37] Nikolay Raychev. Functional composition of quantum functions. International Journal of Scientific and Engineering Research 04/2015; 6(4):413-415. DOI:10.14299/ijser.2015.04.004, 2015.
[38] Nikolay Raychev. Logical sets of quantum operators. International Journal of Scientific and Engineering Research 04/2015; 6(4):391-394. DOI:10.14299/ijser.2015.04.002, 2015.
[39] Nikolay Raychev. Controlled formalized operators. In International Journal of Scientific and Engineering Research 05/2015; 6(5):1467-1469, 2015.
[40] Nikolay Raychev. Controlled formalized operators with multiple control bits. In International Journal of Scientific and Engineering Research 05/2015; 6(5):1470-1473, 2015.
[41] Nikolay Raychev. Connecting sets of formalized operators. In International Journal of Scientific and Engineering Research 05/2015; 6(5):1474-1476, 2015.
[42] Nikolay Raychev. Indexed formalized operators for n-bit circuits. International Journal of Scientific and Engineering Research 05/2015; 6(5):1477-1480, 2015.
[43] Nikolay Raychev. Converting the transitions between quantum gates into rotations. International Journal of Scientific and Engineering Research 06/2015; 6(6): 1352-1354. DOI:10.14299/ijser.2015.06.001, 2015.
[44] Nikolay Raychev. Quantum algorithm for non-local coordination. International Journal of Scientific and Engineering Research 06/2015; 6(6):1360-1364. DOI:10.14299/ijser.2015.06.003, 2015.
[45] Nikolay Raychev. Universal quantum operators. International Journal of Scientific and Engineering Research 06/2015; 6(6):1369-1371. DOI:10.14299/ijser.2015.06.005, 2015.
[46] Nikolay Raychev. Ensuring a spare quantum traffic. International Journal of Scientific and Engineering Research 06/2015; 6(6):1355-1359. DOI:10.14299/ijser.2015.06.002, 2015.
[47] Nikolay Raychev. Quantum circuit for spatial optimization. International Journal of Scientific and Engineering Research 06/2015; 6(6):1365-1368. DOI:10.14299/ijser.2015.06.004, 2015.
[48] Nikolay Raychev. Encoding and decoding of additional logic in the phase space of all operators. International Journal of Scientific and Engineering Research 07/2015; 6(7): 1356-1366. DOI:10.14299/ijser.2015.07.003, 2015.
[49] Nikolay Raychev. Measure of entanglement by Singular Value decomposition. International Journal of Scientific and Engineering Research 07/2015; 6(7): 1350-1355. DOI:10.14299/ijser.2015.07.004, 2015.
[50] Nikolay Raychev. Quantum algorithm for spectral diffraction of probability distributions. International Journal of Scientific and Engineering Research 08/2015; 6(7): 1346--1349. DOI:10.14299/ijser.2015.07.005, 2015.
[51] Nikolay Raychev. Reply to "The classical-quantum boundary for correlations: Discord and related measures". Abstract and Applied Analysis 11/2014; 94(4): 1455-1465, 2015.
[52] Nikolay Raychev. Reply to "Flexible flow shop scheduling: optimum, heuristics and artificial intelligence solutions". Expert Systems; 25(12): 98-105, 2015.
[53] Nikolay Raychev. Classical cryptography in quantum context. Proceedings of the IEEE 10/2012, 2015.

IJSER