Journal of Information Systems and Telecommunication
Vol.11, No.2, April-June 2023, 135-148

http://jist.acecr.org
ISSN 2322-1437 / EISSN:2345-2773

Representing a Novel Expanded Version of Shor′s Algorithm and a Real-Time Experiment using IBM Q-Experience Platform

Sepehr Goodarzi[1] , Afshin Rezakhani[1 *] , Mehdi Maleki[1]

[1]. Department of Computer Engineering, Ayatollah Boroujerdi University, Boroujerd, Iran

Received: 24 Dec 2021/ Revised: 16 Apr 2022/ Accepted: 17 May 2022

Abstract

The data are stored on the memory of the classical computer in small units of classical bits, which could be either 0 or 1. However, on a Quantum Computer, The Quantum States of each Quantum Bit (Qbit), would be every possible number between 0 and 1, including themselves. By placing the photons on a special state, which is a spot located at the middle of the two-dimensional space vectors $\begin{pmatrix}1\\0\end{pmatrix}$ and $\begin{pmatrix}1\\1\end{pmatrix}$ on the Unit Circle, which is called Superposition and we can take advantage of properties of this state when we place lots of vectors of N-dimensional spaces in superposition and we can do a parallelization and factorization for getting significant speedup. In fact, in Quantum Computing we are taking advantage of Quantum Dynamic Principles to process the data, which Classical Computers lack on, by considering the limitations of logical concepts behind them. Through this paper, we expand a quantum algorithm for the number of n Qbits in a new way and by implementing circuits using IBM-Q Experience, we are going to have some practical results, which are more obvious to be demonstrable. By expanding the Quantum Algorithms and using Linear Algebra, we can manage to achieve the goals at a higher level, the ones that Classical Computers are unable to perform, as machine learning problems with complicated models and by expanding the subject we can mention majors in different sciences like Chemistry (predicting the Structure of proteins with higher percentage accuracy in less period), Astronomy and so on.

Keywords: Quantum Computer; Quantum Dynamics; Unit Circle; N-dimensional Space; IBM-Q Experience.

1- Introduction

Nowadays, the act of factorization in mathematics is so popular among scientists and also engineers due to the extensive applications coming with it. Applications such as the optimization of the processing algorithms, which are exclusively written to be processed on Graphics Processing Units, which can run a lot faster than Central Processing Units due to their special parallelization and so on [1, 2]. Another capability of factorization is that it can help us solve differential matrix equations on a large-scale and would be practical in Linear Algebra [3], therefore it would be practical in computer science by considering the foundation of computers, which every classical bit is defined by matrices and the operation on them would be matrices and matrices are coming from Linear Algebra [4], therefore it is directly related to computer science. For example, it has been proved we can pull out more functional properties of some specific data across non-linear mappings, negative valued data processing, and also reviewing the data with only known relationships with them and all the three mentioned properties have been obtained by using the non-negative matrix factorization in a publication [5]. This subject is not related just to some specific branches of science and it is useful in many subjects. As an example, the simulation of materials is so important, given that if it would be possible to simulate the exact construction of a specific protein, then we would be able to cure many diseases since many of them are caused by the lack of some protein or shortages in the structure proteins [6]. By considering the amino acids, which are the main formational units of proteins, they can create many different types of proteins, according to the chemical structure they have and the types of chemical structure they can form. By expanding the data range, we are

✉ **Afshin Rezakhani**
rezakhani@abru.ac.ir

going to need much more power and resources to process the data, but we can use some factorization methods to factor and classify data, which would be easier for computers to process and simulate but sometimes it becomes impossible to do such calculations using the classical computers, the ones which are working with classical bits [7].

Classical computers are too weak to process a huge amount of data using some algorithms and it may take thousands of years to get the results, so it would be useless in that case. This occurs due to their classical properties and their physical limitations [8]. we are going to be more precise on this subject, we are going to check a paper, which has been published and it is concentrated on reviewing some machine learning methods to calculate the protein secondary structure prediction. Proteins are molecules, which carry out essential subordinates in almost all operations in the human body. They are made up of amino acid macromolecules and there are about 500 amino acids out there but 20 of them are coded by the genome to construct the necessary proteins for the human body. In summary, by considering a collection of n amino acids, 2n types of proteins with different constructions are possible to build. As a result, for 250 amino acids, we are going to have, 2250 proteins, which is a large number of species and it needs a very long time, so many powerful hardware and so much power source to calculate every possible state, which is not efficient [9].

Decades ago, a scientist presented another method of computation varying from classical computation and it is called Quantum Computation, which uses Qbits rather than Classical bits. As we know, classical bits can be in either two states of 0 or 1, which are presented by $\begin{pmatrix}1\\0\end{pmatrix}$ as 0 and $\begin{pmatrix}0\\1\end{pmatrix}$ as 1 and there are the only states, which a classical bit can be at a time, but in quantum computation a Qbit, which has the same functionality as the bits in Classical Computing, but it can be in more states at a time. A Qbit can be 0 or 1 or both the 0 and 1 together, which is a quantum property of the underlying atomic particles. This is called "Super Position" in Quantum Mechanics [10]. Scientists are using these properties to their advantage. Superposition means, that an Electron could be in both of the 0 and 1 states at once. According to the cause, Quantum Bits include Classical Bits as special states. These underlying atomic particles have more weird properties such as "Entanglement" and "supremacy" [11], which we are not going to talk about and the superposition is the most important part of our research. Another useful feature of a Quantum Computer is, that it is reversible, which works this way, because of its physical properties and limitations and we are going to take advantage of this property later on. There are four primary operations, which can be done on a single bit of information including, Identity, Negation, Constant-0 and Constant-1, which we are going to have more focus on, later in this paper. For an introduction, Identity and Negation are reversible, but Constant-0 and Constant-1 are irreversible and we are going to have to write them in a reversible way. By knowing a factor like being reversible or irreversible of an unknown function, we can do the factorization to get more properties of that function, which is very useful, as described earlier. In a classical computer, it can take 2 queries to process it by having one bit as an input to know if our unknown function is irreversible or not, whereas a quantum computer would do it on a single query, which is a massive speedup. As a result, for every 2 bits, we are going to need a single query, which is half the time and at the end, we are going to need $\sqrt{(2\text{^}n)}$ queries for n bits.

At first, we had an abstract of the whole paper. After that, we considered four sections in the introduction. In the first part of the introduction, we checked the importance and the capabilities of the factorization operation, then we reviewed the lack of classical computers in calculating the properties of unknown functions in order to factor and classify those properties. After that, by addressing the quantum mechanics and quantum computer properties, we claimed we are able to do a calculation to get the mentioned property on a quantum computer with half the time for getting the same result on a classical computer. Now we are reviewing the sections of the paper and then we are going to have the algorithm of doing the calculation on a quantum computer on paper and the result of an experiment on a real quantum computer using the cloud-base framework IBM has introduced for the researchers to have access to real quantum computers by aiming the real experiments on real quantum computers. At last, we are going to have a discussion and then the references. By using the mentioned methods and, at most of them, we must do the whole calculations of the existing entities of the function to get to know more about the features of the function in order to do some factorizations regarding to increase the pace of further calculations.

2- Related Works

Major In general, mathematical thinking and computation has been doing an important role in our lives and the advancement of various sciences. We are going to discuss a number of the related research jobs, which has been done in this area a few years ago and we are going to know the importance of the discussed subject in the following section. Our special subject in

the discussion would be the pros of mathematical factorization on algorithms and the approach to it.
A recent work, which has been done earlier, shows the benefits of Maximization-Factorization Statics on an advance in computer science, which is Blockchain Technology and also Internet-of-Things, which has been published in IEEE Internet of Things Journal. The pros include using less memory and power and also they are the less iteration to converge to the consensus solution and easiness to configure the complete mathematical model as per the requirement [12]. Another work has been published on PPoPP '21: Proceedings of the 26th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming, in which the authors have proposed a method of deriving parallel I/O lower bounds for the programs and they derive COnfLUX, an LU algorithm with the parallel I/O cost of N3/([EQUATION]) communicated elements per processor - only 1/3× over our established lower bound, which would be considered as a massive speedup [13]. Through another one, the authors show, that the D-Wave 2X can be effectively used as part of an unsupervised machine learning method. The used method takes a matrix as input and produces two low-rank matrices as output—one containing latent features in the data and another matrix describing how the features can be combined to approximately reproduce the input matrix. Despite the limited number of bits in the D-Wave hardware, this method is capable of handling a large input matrix. The D-Wave only limits the rank of the two output matrices. They applied their method to learn the features from a set of facial images and compare the performance of the D-Wave to two classical tools [14]. Another job, which has been published on IEEE Access, proposes two techniques for overcoming load-imbalance encountered when implementing so-called look-ahead mechanisms in relevant dense matrix factorizations for the solution of linear systems. The first technique promotes worker sharing (WS) between the two tasks, allowing the threads of the task that completes first to be reallocated for use by the costlier task. The second technique allows a fast task to alert the slower task of completion, enforcing the early termination (ET) of the second task, and a smooth transition of the factorization procedure into the next iteration [15]. In another research, a person named Reid Atcheson, who is a Ph.D. in Computational and Applied Mathematics, proposed a way to use non-Euclidean norms to formulate a QR-like factorization which can unlock interesting and potentially useful properties of non-Euclidean norms - for example, the ability of l1 norm to suppresses outliers or promote sparsity. At the end of his paper, he confirmed the results using python [16]. Through another work, the authors propose a novel solution to this problem: at the mathematical level, we reduce the computational requirement by exploiting the data sparsity structure of the matrix off-diagonal tiles utilizing low-rank approximations; and, at the programming-paradigm level, we integrate PaRSEC, a dynamic, task-based runtime to reach unparalleled levels of efficiency for solving extreme-scale linear algebra matrix operations. The paper has been published on PASC '20: Proceedings of the Platform for Advanced Scientific Computing Conference [17]. In another research, the authors have proposed to factorize the matrix using a “lattice HH-matrix” format that generalizes the BLR format by storing each of the blocks (both diagonals and off-diagonals) in the HH-matrix format. These blocks stored in the HH-matrix format are referred to as lattices. Thus, this lattice format aims to combine the parallel scalability of BLR factorization with the near-linear complexity of HH-matrix factorization. At the very first step, they compared factorization performances using the HH-matrix, BLR, and lattice HH-matrix formats under various conditions on a shared-memory computer. The performance results show that the lattice format has storage and computational complexities similar to those of the HH-matrix format, and hence a much lower cost of factorization than BLR [18]. In another research, the authors describe efficient algorithms for computing rank-revealing factorizations of matrices that are too large to fit in RAM, and must instead be stored on slow external memory devices such as solid-state or spinning disk hard drives (out-of-core or out-of-memory). They propose separate methods. The first is a blocked version of column pivoted Householder QR, organized as a "left-looking" method to minimize the number of write operations (which are more expensive than reading operations on a spinning disk drive). The second method results in a so-called UTV factorization which expresses a matrix A as A=UTV× where U and V are unitary, and T is triangular. This method is organized as an algorithm-by-blocks, in which floating-point operations overlap read and write operations [19]. In another work, the authors proposed a distributed high-performance parallel implementation of the BPMF using Gibbs sampling on shared and distributed architectures. They have shown by using efficient load balancing using work-stealing on a single node, and by using asynchronous communication in the distributed version they beat state-of-the-art implementations [20]. Through another paper, which has been published in IEEE Transactions on Services Computing, they addressed a privacy bug in clouds by presenting a novel outsourced scheme for NMF (O-NMF), which aims to lessen clients' computing burden and tackle security problems faced by outsourcing NMF [21]. Through another research, which has been published in 2017 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW), Batch matrix operations address the case of

solving the same linear algebra problem for a very large number of very small matrices. They focused on implementing the batch Cholesky factorization in CUDA, in single-precision arithmetic, for NVIDIA GPUs and also the benefits of using noncanonical data layouts, where consecutive memory locations store elements with the same row and column index in a set of consecutive matrices [22]. In another work, the authors provide a comprehensive survey of mixed-precision numerical linear algebra routines, including the underlying concepts, theoretical background, and experimental results for both dense and sparse linear algebra problems [23]. Through another work, the authors provide techniques for Supernodal Sparse Cholesky factorization on a hybrid multicore platform consisting of a multicore CPU and GPU. The techniques are the subtree algorithm, pipelining and, multithreading. The subtree algorithm minimizes PCIe transmissions by storing an entire branch of the elimination tree in the GPU memory (the elimination tree is a tree data structure describing the workflow of the factorization) and also reduces the total kernel launch time by launching BLAS kernels in batches [24]. In another job, the authors highlight the necessary development of new instrumentation tools within the PaRSE task-based runtime system to leverage the performance of low-rank matrix computations. They demonstrate the benefits of these amenable tools while assessing the performance of TLR Cholesky factorization from data distribution, communication-reducing, and synchronization-reducing perspectives. The mentioned tool-assisted performance analysis results in three major contributions: a new hybrid data distribution, a new hierarchical TLR Cholesky algorithm, and a new performance model for tuning the tile size. The new TLR Cholesky factorization achieves an 8X performance speedup over existing implementations on massively parallel supercomputers, toward solving large-scale 3D climate and weather prediction applications [25]. And finally, in the last mentioning paper, the authors present a multithreaded method for Supernodal Sparse Cholesky factorization on a hybrid multicore platform consisting of a multicore CPU and GPU. The mentioned algorithm can utilize concurrency at different levels of the elimination tree by using multiple threads in both the CPU and the GPU. The elimination tree is a tree data structure describing the workflow of the factorization. The results on a platform consisting of an Intel multicore processor along with an Nvidia GPU indicate a significant improvement in performance and energy over a single-threaded Supernodal algorithm [26].

Table 1: Presenting the Related Works with Details

Number	***Publication***	***Description***	***Efficiency***
1	Kumar, G. et. al.,	Discussing an efficient statistical method with a proof-of-work consensus approach for cloud and fog computing	Less iteration to converge to the consensus solution and easiness to configure the complete mathematical model as per the requirement and also less energy and memory are needed
2	Kwasniewski, G. et. al.,	Presenting a method of deriving parallel I/O lower bounds for Dense linear algebra programs	Running on 1,024 nodes of Piz Daint, COnfLUX communicates 1.6× less than the second-best implementation and is expected to communicate 2.1× less on a full-scale and run on Summit
3	Lang, N. et. al.,	Proposing efficient algorithms for solving large-scale matrix differential equations	better performance of the proposed methods compared to earlier formulations
4	Likharev, K.K. et. al.,	The discussion of Fundamental limitations on the energy dissipated during one elementary logical operation	The limits due to classical and quantum statistics are shown to lie well below the earlier estimates, k B T and ηϑ, respectively
5	Malley, D.O. et. al.,	Representing a novel computational architecture and have attracted significant interest	The D-Wave 2X can be effectively used as part of an unsupervised machine learning method

6	Ostrouchov. et. al.,	Explaining the observed performance of sparse matrix factorization algorithms on parallel computers	Finishing with a parameterized model that is capable of reproducing the full range of behavior within these bounds, including the speedups observed in practice
7	Prijatmoko, D. et. al.,	Quantifying changes in body composition and compare methods for measuring body composition in alcoholic cirrhosis	With increasing severity of cirrhosis, total body water increased, whereas total body protein decreased with a significant decrease in serum albumin levels
8	Steffen, P. et. al.,	Updating the compiler to include a parallel backend, launching a large number of independent threads	Speedups ranging from 6.1× to 25.8× on an Nvidia GTX 280 through the CUDA libraries
9	MengTang. et. al.,	Presenting a multithreaded method for Supernodal sparse Cholesky factorization on a hybrid multicore platform consisting of a multicore CPU and GPU	Improvement in performance and energy over single-threaded Supernodal algorithm
10	MengTang. et. al.,	presenting techniques for Supernodal sparse Cholesky factorization on a hybrid multicore platform consisting of a multicore CPU and GPU	Improvement in performance and energy over CHOLMOD
11	Theurer, T. et. al.,	Introducing a rigorous resource theory framework for the quantification of superposition of a finite number of linear independent states	Establishing a strong structural connection between superposition and entanglement
12	VanderAa, T. et. al.,	Proposing a distributed high-performance parallel implementation of the BPMF using Gibbs sampling on shared and distributed architectures	Beating the state of the art implementations by using efficient load balancing and asynchronous communication
13	Wardah, W. et. al.,	Improving protein secondary structure prediction accuracy using neural networks	Producing better protein secondary structure prediction from higher architecture complexity
14	Windley, P. F. et. al.,	The problem of transposing a matrix in the store of a computer	Storing the data on the memory of a computer

15	Yamazaki, I. et. al.,	Parallelizing the LU factorization of a hierarchical low-rank matrix (HH-matrix) on a distributed-memory computer	Lower cost of factorization, which leads to a faster one
16	Zhang, D. et. al.,	Extend the original non-negative matrix factorization to kernel NMF	Extracting more efficient features, dealing with data through knowing the relations, processing negative values

3- Proposed Method

3-1- Problem Statement

As we all know, the four primary operations on a single bit of information are Identity, Negation, and Constants, including Constant-0, and Constant-1. As an intro, Identity and Negation are reversible but the Constants are irreversible. The proof is specified. A bit of information, which is either 0 or 1 and is stored on a Classical Computer and are considered as vectors. They are presented by $\begin{pmatrix}1\\0\end{pmatrix}$ and $\begin{pmatrix}0\\1\end{pmatrix}$, which are presenting the state 0 and 1 in order. The operations are also considered by matrices. By considering the multiplication of an operation into a bit, we are going to have a vector, which is the new state of the mentioned bit. Regardless of the state of the entry bit, the Constants Matrices and the results are going to be the same in every possible way, due to the essence of their construction, so they are irreversible and the vice versa applies to Identity and Negation. We can do some factorization by knowing the features of a function, which would give us the ability of faster processing. It can even make some impossibilities, possible due to the physical limitations of our processing units. To know if a function is reversible or not, it can take up to thousands of years to do the whole process on a classical computer. Imagine we have a single bit of entry information, it would take two queries for us to get to know if the function is reversible or not (by considering that we do not much information of our function, which would be so valuable in many areas in different sciences as Analytical Chemistry since it could help us generating new constructions of materials by solving useful equations). We should input 0 and 1 each time to get the results and then we can recognize if it is reversible or not by comparing the outputs of each query. For two bits, again we should test every possible state of each bit, which would be 4 queries. In summary, we should have two to the power of n (2n) for n bits of entry information. Now consider we have a function with thousands of entries and we want to know if it is reversible or not. In a simple word, we cannot calculate it, because it takes thousands of years to be processed and we are going to represent a way to calculate the mentioned factor of a function with n bits of entry information in a single query on a quantum computer, which would take infinite years of calculations on a classical one [27].

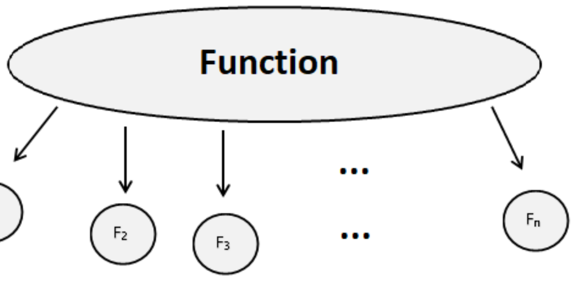

Fig. 1 Proposed beam former

3-2- Necessary Basics of Quantum Computing

In quantum computing, unlike the classical one, we have more than two states for each bit. A Quantum bit could be in the states of 0 and 1 and both and every possible number between them as states. According to this definition, the states of a classical bit would be a part of the states of a quantum bit, which is more recognizable in figure 2.

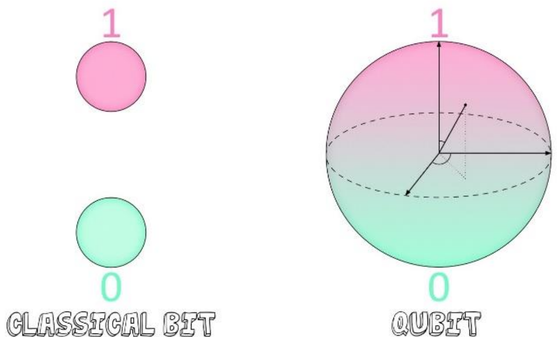

Fig. 2 The comparison of a classical bit (Cbit) with a quantum bit (Qbit)

3-3- Tensor Product

We represent the quantum state of 0 by |0⟩ and 1 by |1⟩. As we discussed earlier, we represent states of a bit using vectors, which is an m×1 matrix, in which m=2^n and we are able to represent it as the Tensor Products of n 2×1 vectors, which would be a lot helpful later on [28]. A Tensor Product of n vectors is:

$$\begin{pmatrix} a0 \\ a1 \\ \dots \\ an \end{pmatrix} \otimes \begin{pmatrix} b0 \\ b1 \\ \dots \\ bn \end{pmatrix} \otimes \dots \otimes \begin{pmatrix} z0 \\ z1 \\ \dots \\ zn \end{pmatrix} = \begin{pmatrix} a0 * b0 * \dots * z0 \\ a1 * b1 * \dots * z1 \\ \dots \\ an * bn * \dots * zn \end{pmatrix} \quad (1)$$

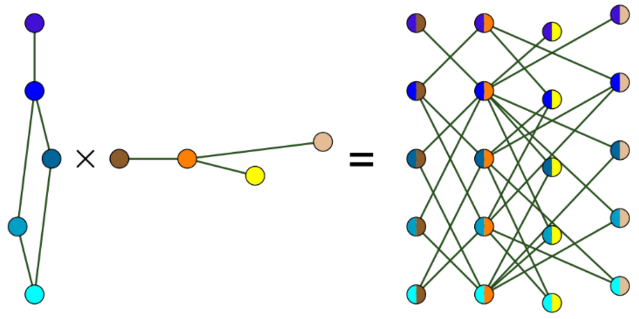

Fig. 3 The representation of Tensor Product using shapes (which is similar to the chemical structure of materials)

3-4- Quantum Gates

There are a number of quantum gates out there, but we need a few of them and we will discuss them [29].

3-4-1 CNOT Gate

CNOT gate is a 4×4 matrix and it changes the last two elements of the 4×1 matrix (Tensor Product of two vectors) [30]. It is represented by:

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix} \quad (2)$$

3-4-2 Bit-Flip Operator

It is the Negation in Classical Computing and is used to produce the symmetry of the input state [31]. The matrix is:

$$\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \quad (3)$$

3-4-3 Hadamard Gate

This gate takes the input into a superposition, which is one of the four states:

$$\begin{pmatrix} \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{pmatrix}, \begin{pmatrix} \frac{-1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{pmatrix}, \begin{pmatrix} \frac{1}{\sqrt{2}} \\ \frac{-1}{\sqrt{2}} \end{pmatrix}, \begin{pmatrix} \frac{-1}{\sqrt{2}} \\ \frac{-1}{\sqrt{2}} \end{pmatrix} \quad (4)$$

3-4-4 Identity and Constants

These are represented and called as the same as Classical Computing environment. Identity, Constant-0 and Constant-1 are represented in order:

$$\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}, \begin{pmatrix} 1 & 1 \\ 0 & 0 \end{pmatrix}, \begin{pmatrix} 0 & 0 \\ 1 & 1 \end{pmatrix} \quad (5)$$

3-5- Irreversible Quantity

Quantum Computers are irreversible devices, which means we do not have the ability to perform irreversible operations in the same way of reversible ones, as we were acting on classical computers [32]. It is not permitted by the physical reality according to the quantum mechanics and properties of a quantum computer. To solve this problem, we can have two Qbits as an entry instead of one.

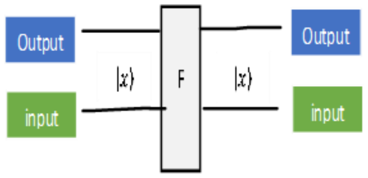

Fig. 4 The solution for having irreversible functions on a reversible computer

The input, which is labeled by output could be either 0 or 1 and there is not much difference between them, because the answer would be symmetry if we use the other one. In this particular subject, we use |0⟩. The important output to us would be the one, which has been labeled by output, which the function would operate on. Now we can reversibly use Constants.

3-6- Method

Now we are going to calculate an algorithm on a single Qbit using one query to get to know if the unknown function is reversible or not in which we should have two queries on a classical computer [33]. We are going to need two Qbits as entries for each Qbit. At first, we are going to get them through a Bit-Flip operator and then a Hadamard Gate, and after that we will apply our unknown function and at the end another Hadamard Gate and finally measuring the results. So it would look like this:

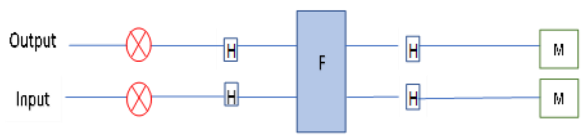

Fig. 5 The implemented Quantum Gates

4- Results and Discussion

At first, we apply the Bit-Flip and then the Hadamard Gate which gives the down results (by considering both the inputs at the state of $|0\rangle$):

Output[1]:

$[(0,1)(1,0)]^T \times [1,0]^T = [0,1]^T$, $[(1/\sqrt{2}, 1/\sqrt{2})(1/\sqrt{2}, -1/\sqrt{2})]^T \times [0,1]^T = [1/\sqrt{2} , -1/\sqrt{2}]^T$

(6)

Input:

$[(0,1)(1,0)]^T \times [1,0]^T = [0,1]^T$, $[(1/\sqrt{2}, 1/\sqrt{2})(1/\sqrt{2}, -1/\sqrt{2})]^T \times [0,1]^T = [1/\sqrt{2} , -1/\sqrt{2}]^T$

(7)

Now we apply each of the four primary gates and the Hadamard on them separately.

Constant-0:

Output:

$[1/\sqrt{2} , -1/\sqrt{2}]^T$ (No changes are applied) , $[(1/\sqrt{2}, 1/\sqrt{2})(1/\sqrt{2}, -1/\sqrt{2})]^T \times [1/\sqrt{2} , -1/\sqrt{2}]^T = [0,1]^T = |1\rangle$

(8)

Input:

$[1/\sqrt{2} , -1/\sqrt{2}]^T$ (No changes are applied) , $[(1/\sqrt{2}, 1/\sqrt{2})(1/\sqrt{2}, -1/\sqrt{2})]^T \times [1/\sqrt{2} , -1/\sqrt{2}]^T = [0,1]^T = |1\rangle$

(9)

So the result of measuring Constant-0 would be $|11\rangle$.

<u>Constant-1</u>:

Output:

$[(0,1)(1,0)]^T \times [1/\sqrt{2} , -1/\sqrt{2}]^T = [-1/\sqrt{2} , 1/\sqrt{2}]^T$,

$[(1/\sqrt{2}, 1/\sqrt{2})(1/\sqrt{2}, -1/\sqrt{2})]^T \times [-1/\sqrt{2} , 1/\sqrt{2}]^T = [0,1]^T = |1\rangle$

(10)

Input:

$[1/\sqrt{2} , -1/\sqrt{2}]^T$ (No changes are applied) , $[(1/\sqrt{2}, 1/\sqrt{2})(1/\sqrt{2}, -1/\sqrt{2})]^T \times [1/\sqrt{2} , -1/\sqrt{2}]^T = [0,-1]^T = |1\rangle$

(11)

As we see, the result of measuring Constant-1 would be $|11\rangle$ too.

Identity:

Output:

$[1/\sqrt{2} , -1/\sqrt{2}]^T$ (No changes are applied) , $[(1/\sqrt{2}, 1/\sqrt{2})(1/\sqrt{2}, -1/\sqrt{2})]^T \times [-1/\sqrt{2} , 1/\sqrt{2}]^T = [0,1]^T = |1\rangle$

(12)

Input:

$[(1,0,0,0)(0,1,0,0)(0,0,0,1)(0,0,1,0)] \times [[1/\sqrt{2}, -1/\sqrt{2}]^T \otimes [1/\sqrt{2}, -1/\sqrt{2}]^T]^T$

$= [1/\sqrt{2}, 1/\sqrt{2}]^T \otimes [1/\sqrt{2}, -1/\sqrt{2}]^T$, $[(1/\sqrt{2}, 1/\sqrt{2})(1/\sqrt{2}, -1/\sqrt{2})]^T \times [1/\sqrt{2}, 1/\sqrt{2}]^T = [0,1]^T = |0\rangle$

(13)

As a result, the measurement result of Identity was $|01\rangle$.

Negation:

Output:

$[(0,1)(1,0)]^T \times [1/\sqrt{2}, -1/\sqrt{2}]^T = [-1/\sqrt{2}, 1/\sqrt{2}]^T$, $[(1/\sqrt{2}, 1/\sqrt{2})(1/\sqrt{2}, -1/\sqrt{2})]^T \times [-1/\sqrt{2}, 1/\sqrt{2}]^T = [0,-1]^T = |1\rangle$

(14)

Input:

$[(1,0,0,0)(0,1,0,0)(0,0,0,1)(0,0,1,0)] \times [[1/\sqrt{2}, -1/\sqrt{2}]^T \otimes [1/\sqrt{2}, -1/\sqrt{2}]^T]^T$

$= [1/\sqrt{2}, 1/\sqrt{2}]^T \otimes [1/\sqrt{2}, -1/\sqrt{2}]^T$, , $[(1/\sqrt{2}, 1/\sqrt{2})(1/\sqrt{2}, -1/\sqrt{2})]^T \times [1/\sqrt{2}, 1/\sqrt{2}]^T = [0,1]^T = |0\rangle$

(15)

The result was the same as Identity, which is $|01\rangle$. As you saw, the results of the Identity and Negation were equal ($|01\rangle$), and the same happened for Constants ($|11\rangle$). It is now proved that we can run the whole process on a single query, in which we need two queries on a classical computer. For one entry Qbit if the result comes out $|01\rangle$, then the function is reversible, but if we get $|11\rangle$, then the function would be irreversible. We are going to discuss the n Qbits through the next table.

[1] A matrix can be represented in several ways. The best way to do this on a paper is to write it linearly. For example, the matrix $\begin{pmatrix} a11 & a12 & \dots & a1n \\ b21 & b22 & \dots & b2m \\ \dots & \dots & \dots & \dots \\ zn1 & zn2 & \dots & znm \end{pmatrix}$, which is a m × n matrix and it could be represented by:
[(a11,a12,…,a1m) (b21,b22,…,b2m)… (zn1,zn2,…,znm)]T, which saves a lot of space.

Table 2: Expanding the algorithm for n Qbits

Number of Qbits	***Reversible***	***Irreversible***
1	|01⟩	|11⟩
2	|0011⟩	|1111⟩
3	|000111⟩	|111111⟩
…	…	…
n	|00…011…1⟩	|11…1⟩

By considering the calculations, we can say for n Qbits as entry, we are going to need 2×n Qbits of memory due to the properties, which discussed earlier, and then we would apply the algorithm and we will face two conditions, a state including 2×n of 1s which is |11…1⟩ and we can conclude that the function is irreversible, but if we face a state including n×0s and n×1s which would be |00…011…1⟩, then the function is reversible. We got it in a single query.

5- Main Difference Between the Method and Shor’s Algorithm

As you have observed, we have proven it is possible for the purposed method to function as well as Shor’s algorithm, by the difference that other existing methods require much more complicated circuits than the method given in this paper. We will be comparing a few of them with the method brought up in this paper at the following table. The table is consisting of the best existing algorithms by considering required random access memory and the estimated time to implement the algorithm in order to get the hidden features of the given function, and as it is observable, the one brought up on this paper is going to get us going less than a minute by considering the implemented circuit and also the number of qubits, which we are going to need and in this case it would be two Qbits.

Table 3: Publications in this area

Publication	***Number of bits/Qbits required for the implementation of n bits/Qbits***	***Estimated time (s)***
Kumar, G. et. al.,	1024	(≈) > 150
Kwasniewski, G. et. al.,	512	(≈) > 70
Lang, N. et. al.,	128	(≈) > 20
Likharev, K.K. et. al.,	16	(≈) > 4
Our proposed method	2	(≈) > 1

6- Applying the Method on a Real Quantum Computer

IBM is a high-tech industry, which does an important role in the United States economy [34]. The concentration of such high-tech companies is on the combination of Science, Technology, Engineering, and Mathematics, which is called STEM for short [35]. It is one of the fewest industries, which has Quantum Computers. They provided a framework for scientists and engineers to do their research on real Quantum Computers online. It is called Qiskit, which has been written in python and is open source and free to use [36]. By creating an account on the IBM website, we got access to real Quantum Computers and have done a short process, due to the number of processes and also physical limitations, so we could do it on two Qbits, which has been considered for a single bit of entry. We have done it by defining the necessary gates and entries to get it into a lineup, which has been established by IBM and when it is our turn, after getting checked by their agents, they processed it on a real Quantum Computer and sent us back the results. We have calculated the results of each primary operation on a single Qbit, in which we need two of them and we have exported all the data we entered including the source codes and the graphical figure of the implemented gates and also we exported the results we got from the calculations including the plots of the probabilities and amplitudes on computational basis states and an extra 3D Q-sphere, which is the exact state of the Qbit and it is represented in a 3D plot but we are unable to put it on the paper, due to its 3D feature, but we put a 2D version of it on the paper.

6-1- Constant-0

The source code of the four designed circuits is available at:
https://github.com/sepehrgoodarzi6/The-four-primary-operations-on-a-Qbit.git
By measuring the results of the Qbits, we got the results showing on plots a 3D-sphere. According to the probability states and also amplitude, the measured probability of states of Qbits is a hundred percent |11⟩[1] (which we have demonstrated in the previous section).

[1] There is no certainty in Quantum Mechanics and in fact, the probabilities are all we got, which are able to measure them so much faster than the certain results using classical computers and the point is that we are able to get to a point, which we can ensure we found the result we are looking for by repeating the calculation a few more time, which is still so much faster than any Classical Algorithm implemented on Classical Computers.

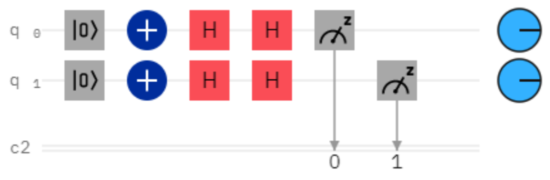

Fig. 8 The implemented circuit for the Constant-0 test

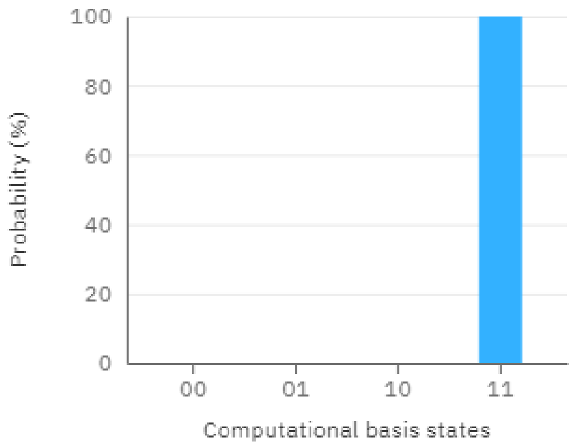

Fig. 9 The prediction of the probability states of Qbits of Constant-0 test

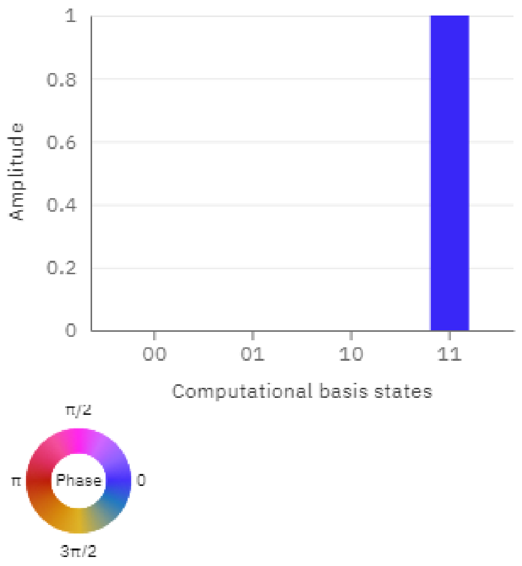

Fig. 10 The amplitude on computational basis states of Constant-0 test

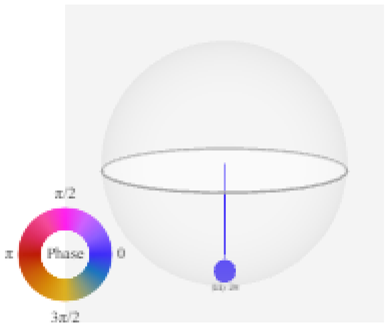

Fig. 11 2D version of the 3D-sphere of Constant-0 test

6-2- Constant-1

According to the probability states and also amplitude, the measured probability of states of Qbits is a hundred percent |11⟩, exactly the same as the results we got from testing Constant-0.

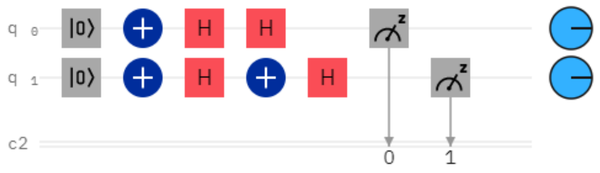

Fig. 12 The implemented circuit for the Constant-1 test

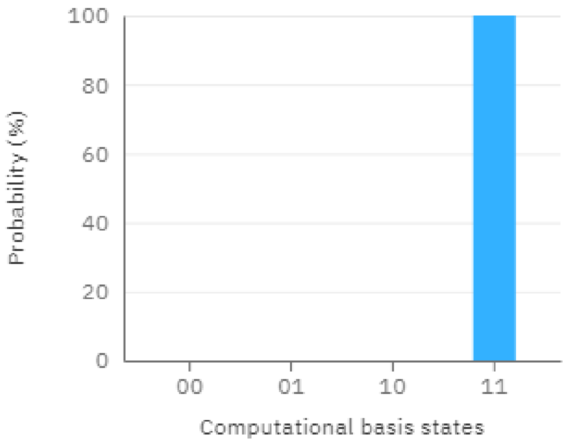

Fig. 13 The prediction of the probability states of Qbits of Constant-1 test

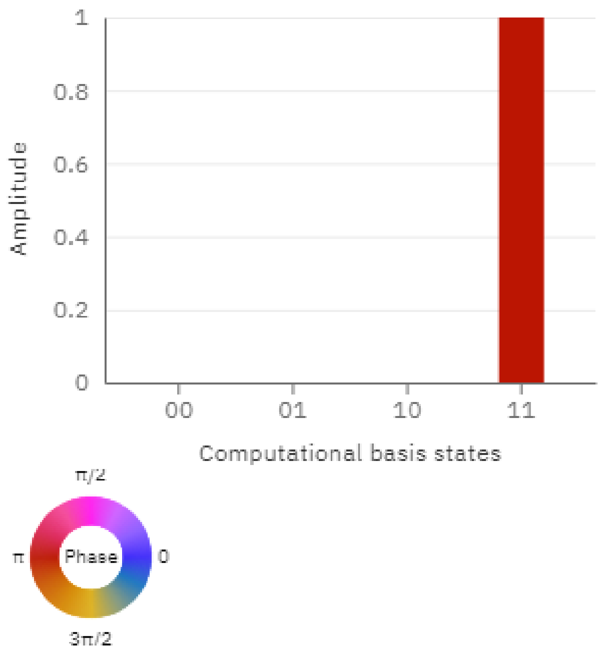

Fig. 14 The amplitude on computational basis states of Constant-1 test

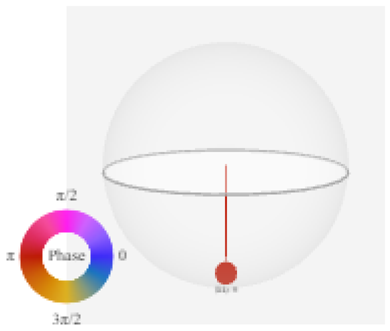

Fig. 15 2D version of the 3D-sphere of Constant-1 test

6-3- Identity

According to the probability states and also amplitude, the measured probability of states of Qbits is a hundred percent |10⟩, which varies from the results of the Constants.

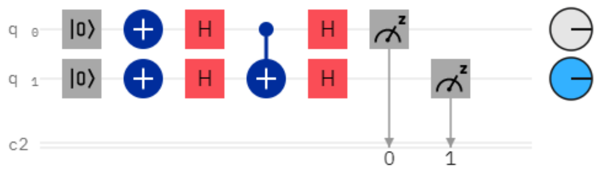

Fig. 16 The implemented circuit for the Identity test

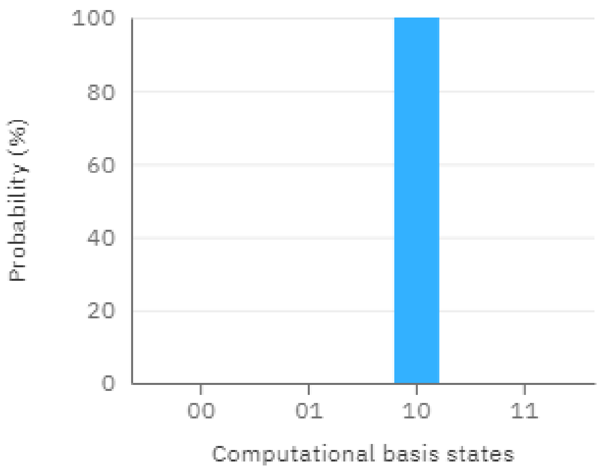

Fig. 17 The prediction of the probability states of Qbits of Identity test

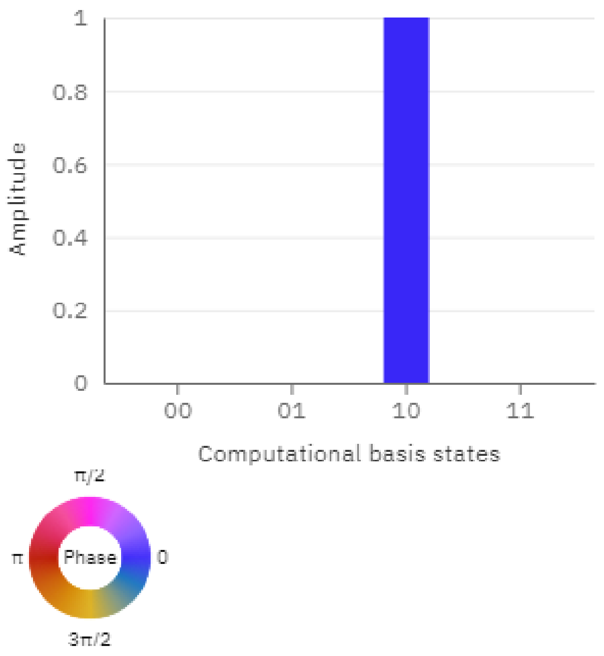

Fig. 18 The amplitude on computational basis states of Identity test

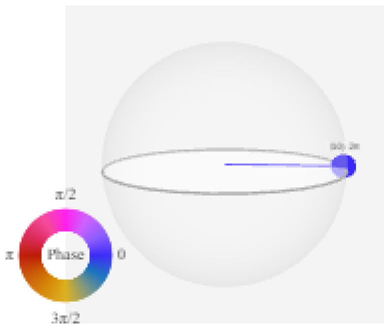

Fig. 19 2D version of the 3D-sphere of Identity test

6-4- Negation

According to the probability states and also amplitude, the measured probability of states of Qbits is a hundred percent |10⟩, which varies from the results of the Constants and is the same as Identity.

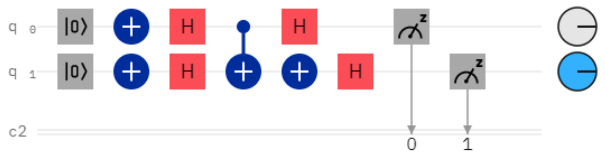

Fig. 20 The implemented circuit for the Negation test

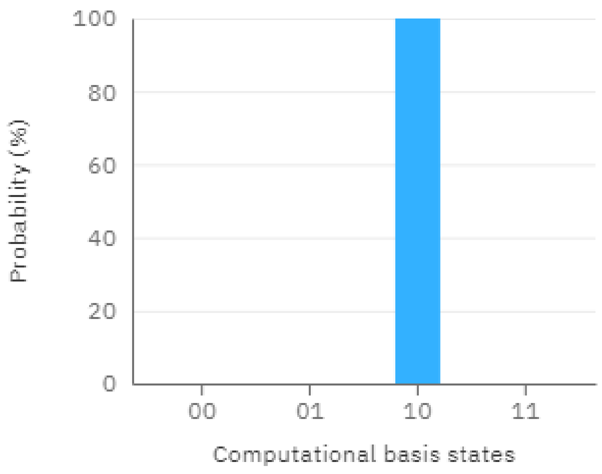

Fig. 21 The prediction of the probability states of Qbits of Negation test

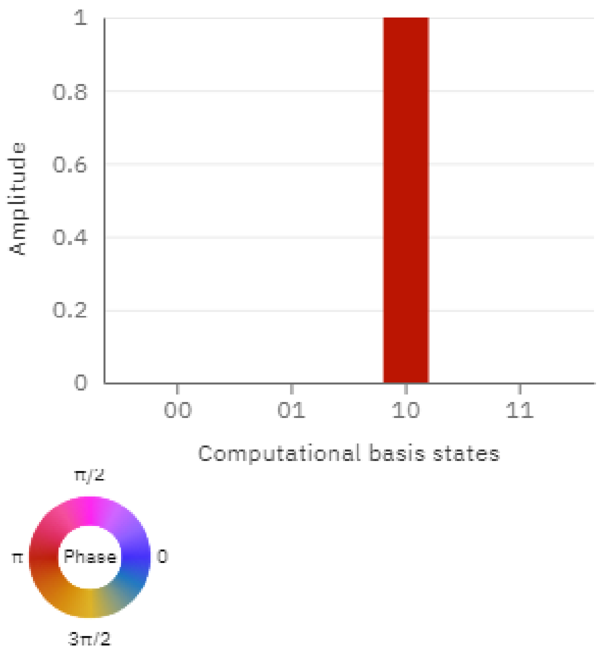

Fig. 22 The amplitude on computational basis states of Negation test

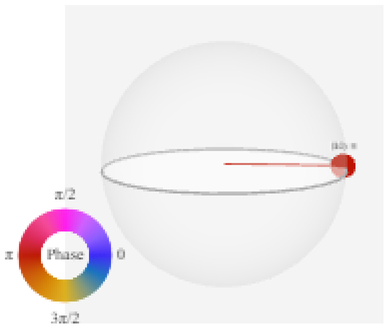

Fig. 23 2D version of the 3D-sphere of Negation test

6-5- Results Review

By measuring the results of the Qbits, we got the results showing on plots a 3D-sphere. According to the probability states and also amplitude, the measured probability of states of Qbits is a hundred percent |11⟩ (which we have demonstrated in the previous sections). According to the probability states and also amplitude, the measured probability of states of Qbits is a hundred percent |11⟩, exactly the same as the results we got from testing Constant-0. By considering the probability states and also amplitude, the measured probability of states of Qbits of Negation is a hundred percent |10⟩, which varies from the results of the Constants.

According to the probability states and also amplitude, the measured probability of states of Qbits is a hundred percent |10⟩, which varies from the results of the Constants and is the same as Identity.
By having this information, we are going to be able to predict the results of these sorts of functions by testing two opponents of them.

7- Conclusions

In summary, we can model every single quantum bit of information existing all around the world, including the quantum world or many observable entities in a mathematical way, which would be Linear Algebra and by taking advantage of the principles of Quantum Dynamics, we can speed up the calculations by outperforming the best classical algorithms, which are implemented on the classical computers. By having these kinds of information, we are going to be able to predict the results of these sorts of functions by testing two opponents of them instead of calculating the whole entities, and by this method, we are going to be able to increase the pace of the calculations at a high rate. By speeding up the calculations, we can achieve the results of the most complex problems, which we are dealing with now, the ones we were not able to solve even in years and they took thousands of years like the ability to predict every possible Secondary Structure of Proteins and so on. But since the middle of the late 90th century, scientists have proved we can process them in a short amount of time if we have the real Quantum Computers with real Quantum Processors and now, we are achieving that goal. The only restriction of this method is that we cannot use it for Non-Quantum processing, and also they are not applicable in all subject of areas. In a few years, we are going to have a massive revolution in all the subject areas of Science and Technology.

References

[1] P. Steffen, R. Giegerich, and M. Giraud, "GPU parallelization of algebraic dynamic programming," in International Conference on Parallel Processing and Applied Mathematics, 2009: Springer, pp. 290-299.
[2] S. Asano, T. Maruyama, and Y. Yamaguchi, "Performance comparison of FPGA, GPU and CPU in image processing," in 2009 international conference on field programmable logic and applications, 2009: IEEE, pp. 126-131.
[3] N. Lang, H. Mena, and J. Saak, "On the benefits of the LDLT factorization for large-scale differential matrix equation solvers," Linear Algebra and its Applications, vol. 480, pp. 44-71, 2015.
[4] P. Windley, "Transposing matrices in a digital computer," The Computer Journal, vol. 2, no. 1, pp. 47-48, 1959.
[5] D. Zhang, Z.-H. Zhou, and S. Chen, "Non-negative matrix factorization on kernels," in Pacific Rim International Conference on Artificial Intelligence, 2006: Springer, pp. 404-412.
[6] D. Prijatmoko et al., "Early detection of protein depletion in alcoholic cirrhosis: role of body composition analysis," Gastroenterology, vol. 105, no. 6, pp. 1839-1845, 1993.
[7] L. S. Ostrouchov, M. Heath, and C. Romine, "Modeling speedup in parallel sparse matrix factorization," Oak Ridge National Lab., TN (USA), 1990.
[8] K. Likharev, "Classical and quantum limitations on energy consumption in computation," International Journal of Theoretical Physics, vol. 21, no. 3, pp. 311-326, 1982.
[9] W. Wardah, M. G. Khan, A. Sharma, and M. A. Rashid, "Protein secondary structure prediction using neural networks and deep learning: A review," Computational biology and chemistry, vol. 81, pp. 1-8, 2019.
[10] T. Theurer, N. Killoran, D. Egloff, and M. B. Plenio, "Resource theory of superposition," Physical review letters, vol. 119, no. 23, p. 230401, 2017.
[11] G. Gour and C. M. Scandolo, "Dynamical Entanglement," Physical Review Letters, vol. 125, no. 18, p. 180505, 2020.
[12] G. Kumar, R. Saha, M. K. Rai, R. Thomas, and T.-H. Kim, "Proof-of-work consensus approach in blockchain technology for cloud and fog computing using maximization-factorization statistics," IEEE Internet of Things Journal, vol. 6, no. 4, pp. 6835-6842, 2019.
[13] G. Kwasniewski, T. Ben-Nun, A. N. Ziogas, T. Schneider, M. Besta, and T. Hoefler, "On the parallel I/O optimality of linear algebra kernels: near-optimal LU factorization," in Proceedings of the 26th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming, 2021, pp. 463-464.
[14] D. O'Malley, V. V. Vesselinov, B. S. Alexandrov, and L. B. Alexandrov, "Nonnegative/binary matrix factorization with a d-wave quantum annealer," PloS one, vol. 13, no. 12, p. e0206653, 2018.
[15] S. Catalán, J. R. Herrero, E. S. Quintana-Ortí, R. Rodríguez-Sánchez, and R. Van De Geijn, "A case for malleable thread-level linear algebra libraries: The LU factorization with partial pivoting," IEEE access, vol. 7, pp. 17617-17633, 2019.
[16] R. Atcheson, "A Generalization of QR Factorization To Non-Euclidean Norms," arXiv preprint arXiv:2101.09830, 2021.
[17] Q. Cao et al., "Extreme-scale task-based cholesky factorization toward climate and weather prediction applications," in Proceedings of the Platform for Advanced Scientific Computing Conference, 2020, pp. 1-11.
[18] I. Yamazaki, A. Ida, R. Yokota, and J. Dongarra, "Distributed-memory lattice h-matrix factorization," The International Journal of High Performance Computing Applications, vol. 33, no. 5, pp. 1046-1063, 2019.
[19] N. Heavner, P.-G. Martinsson, and G. Quintana-Ortí, "Computing rank-revealing factorizations of matrices stored out-of-core," arXiv preprint arXiv:2002.06960, 2020.
[20] T. Vander Aa, I. Chakroun, and T. Haber, "Distributed Bayesian probabilistic matrix factorization," Procedia Computer Science, vol. 108, pp. 1030-1039, 2017.
[21] A. Fu, Z. Chen, Y. Mu, W. Susilo, Y. Sun, and J. Wu, "Cloud-based outsourcing for enabling privacy-preserving

large-scale non-negative matrix factorization," IEEE Transactions on Services Computing, 2019.

[22] M. Gates, J. Kurzak, P. Luszczek, Y. Pei, and J. Dongarra, "Autotuning batch Cholesky factorization in CUDA with interleaved layout of matrices," in 2017 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW), 2017: IEEE, pp. 1408-1417.

[23] A. Abdelfattah et al., "A survey of numerical linear algebra methods utilizing mixed-precision arithmetic," The International Journal of High Performance Computing Applications, p. 10943420211003313, 2021.

[24] M. Tang, M. Gadou, S. Rennich, T. A. Davis, and S. Ranka, "Optimized sparse Cholesky factorization on hybrid multicore architectures," Journal of computational science, vol. 26, pp. 246-253, 2018.

[25] Q. Cao et al., "Performance analysis of tile low-rank cholesky factorization using parsec instrumentation tools," in 2019 IEEE/ACM International Workshop on Programming and Performance Visualization Tools (ProTools), 2019: IEEE, pp. 25-32.

[26] M. Tang, M. Gadou, and S. Ranka, "A Multithreaded Algorithm for Sparse Cholesky Factorization on Hybrid Multicore Architectures," Procedia Computer Science, vol. 108, pp. 616-625, 2017.

[27] M. Green, K. Glover, D. Limebeer, and J. Doyle, "AJ-Spectral Factorization Approach to H_∞," SIAM Journal on Control and Optimization, vol. 28, no. 6, pp. 1350-1371, 1990.

[28] P. Smolensky, "Tensor product variable binding and the representation of symbolic structures in connectionist systems," Artificial intelligence, vol. 46, no. 1-2, pp. 159-216, 1990.

[29] D. P. DiVincenzo, "Quantum gates and circuits," Proceedings of the Royal Society of London. Series A: Mathematical, Physical and Engineering Sciences, vol. 454, no. 1969, pp. 261-276, 1998.

[30] D. M. Zajac et al., "Resonantly driven CNOT gate for electron spins," Science, vol. 359, no. 6374, pp. 439-442, 2018.

[31] D. Riste et al., "Detecting bit-flip errors in a logical qubit using stabilizer measurements," Nature communications, vol. 6, no. 1, pp. 1-6, 2015.

[32] F. Benatti and R. Floreanini, Irreversible quantum dynamics. Springer Science & Business Media, 2003.

[33] S. Gulde et al., "Implementation of the Deutsch–Jozsa algorithm on an ion-trap quantum computer," Nature, vol. 421, no. 6918, pp. 48-50, 2003.

[34] M. Wolf and D. Terrell, "The high-tech industry, what is it and why it matters to our economic future," 2016.

[35] A. P. Carnevale, N. Smith, and M. Melton, "STEM: Science Technology Engineering Mathematics," Georgetown University Center on Education and the Workforce, 2011.

[36] A. Cross, "The IBM Q experience and QISKit open-source quantum computing software," in APS March Meeting Abstracts, 2018, vol. 2018, p. L58. 003.