Modification of the quantum grover algorithm by using the inversion method around the middle

L Cherckesova¹, O Safaryan¹, I Pilipenko¹, V Porksheyan¹, N Bogdanova¹, and N Beryoza²

¹Don State Technical University, Gagarin sq., 1, Rostov on Don, 344003, Russia
²The Institute Of Service and Business (Branch) Don State Technical University In Shakhty, st. Shevchenko, 147, Shakhty, Rostov region, 346500, Russia

E-mail: safari_2006@mail.ru

Abstract. The article proposes a modification of Grover's quantum oracle quantum search algorithm, which makes it easier to search the database. The algorithm is implemented in the Python programming language using the Reggeti Forest cloud quantum service. The authors of the article use the mean flipping method, which solves the search problem during the iterative order √ (2 ^ n). Development offers great potential for the practical application of the Grover quantum algorithm, as it is characterized by higher performance and speed when performing research. Theoretically, the algorithm provides quadratic acceleration compared to conventional computers. It is not an exponential acceleration, but it remains important for large data carriers. The quantum parallelism of the Grover search algorithm is based on a simultaneous change in the amplitudes of all the inputs. This is done through a superposition of states, which is a purely quantum concept. In addition, the research is carried out globally, which indicates a significant improvement in optimization procedures. Grover's algorithm, on the other hand, is sensitive to the number of iterations. The more iterations, the smaller the amplitude of the correct answer, so the wrong choice of this parameter can digest the solution. In addition, the operation of the algorithm is limited in the case of the introduction of noise into a quantum system, which is real in modern quantum computers.

1. Introduction
Quantum computers can lead to advancement in many thematic areas, because they offer computing power that was not previously available. However, not all problems can be solved on quantum computers. To find the right problems, you need to understand the quantum computer troubleshooting process. Although quantum computers offer an exponential increase in computing power, they cannot be programmed in the same way as conventional computers. They have completely different input data, commands and algorithms, special quantum programming languages are used and the output data after solving the problems must be interpreted in a special way. If in common computers the solutions are searched sequentially, checking the possible results, one for each iteration, the quantum computers work in parallel, that is, they can examine all possible solutions at once. However, there are some complications: for some tasks, it takes a long time and getting the right solution from a quantum computer is problematic, because all the solutions obtained are probabilistic [1,2].

Quantum algorithms are classic algorithms that define the sequence of operations of the unit - ports or valves, with an indication of exactly what the qubits need to perform these operations. The quantum
algorithm is defined in the form of an oral description of such commands or in the form of their graphical representation in the form of a system of gates - an array of quantum gates. As already mentioned, the result of the quantum algorithm will be probabilistic. However, due to a small increase in the number of operations in the algorithm, the probability of obtaining the correct result can be brought closer to unity [3].

2. Purpose of the study
It is believed that any task to be solved by a quantum algorithm can be solved by a classical computer by directly calculating the exponential dimension of the uniform matrix and obtaining a clear form of quantum states. In particular, problems that cannot be solved on classical computers remain unsolved on quantum computers. This direct modeling requires exponential time costs, and therefore it is possible to use quantum parallelism to accelerate and be the work of many classical algorithms.

3. Material and research methods
Based on the foregoing, we will try to demonstrate the advantage of the quantum algorithm over the algorithms of other computational models. Therefore, Grover's algorithm consists of two main parts: the construction of the Hadamard gate and the inversion around the center. Quantum algorithms are still very difficult to understand, so the program code that we presented in the Python programming language visualizes the functioning of the algorithm. Grover's algorithm is a quantum search method whose execution time is many orders of magnitude faster than that of classical algorithms. This algorithm is not intended to simply search for elements in the database, but its purpose is to search for input to the function to check whether the function returns a “true” value for specific input. This method is useful if the function is unknown or extremely complex, and we want to find out for which input values the function returns true or provides the correct vector for solving the equation. Grover's algorithm and its quantum scheme will be implemented on the Python and Riggetti Forest platforms.

From Microsoft's point of view, quantum computers can be used as specialized processing modules (coprocessors) for solving many applied quantum problems. The possibility of a quantum revolution is offered from development to the delivery of full-fledged quantum systems using a technological qubit, such as a qubit, which allows you to resize the entire system. The result is an integrable and scalable solution capable of combining quantum and classical calculus.

The process of creating a complete quantum system includes the creation of the raw materials necessary to create topological quantum devices. There are several approaches to creating photonic computers: the principle of nuclear magnetic resonance in the liquid molecular phase (“quantum computer with liquid NMR”), on atomic ions trapped in Paul (Paul) or Penning (Penning) (“Quantum computer with ion trap”), which uses nuclear magnetic resonance or electron paramagnetic resonance in a solid (“quantum computer NMR / EPR of a solid”), using the phenomenon of superconductivity (“superconducting systems”), on quantum dots in inorganic semiconductor systems (“Computer with quantum dots”), and not on the basis of optical modeling of quantum logic (“Optical computer”) or on the basis of hybrid metal biology (“Quantum Hyum computer”).

To create a quantum computer infrastructure, several competing studies have been conducted. However, Microsoft has the necessary technologies for programming a quantum computer, including the Azure control system, software, development tools and services, a combination that can be considered a complete quantum set [4].

A quantum computer can solve the most complex problems in the world only if high-quality qubits are enough for this. It is said that existing systems provide a large number of qubits. (material from the presentation) However, the quality of these qubits does not always meet the requirements, and this is the main factor for constructing a viable quantum system. This problem is fundamental, because qubits are difficult to bring to a steady state. All are interconnected in sequence. This means that if at least one qubit fails, all subsequent qubits will also lose their operability. The key to solving this problem is topological qubit.

Unlike a traditional qubit, a topological qubit is constructed so that the information contained and
processed in it is protected and / or stored automatically. Due to the fact that traditional qubits are extremely fragile and unstable in nature, such protection can be a breakthrough in improving performance, providing increased stability (stability) and the need to use fewer qubits in general. This is a critical advantage, which allows the scaling of a quantum system [5].

4. Noise identification and free object in quantum systems
Artificial atoms made using solid-state devices (for example, superconducting qubits, quantum dots, and nitrogen vacancy centers) are promising for future quantum information processors. To build more stable and robust quantum circuits, a lot of effort is required to overcome the complex decoherence brought about by their "dirty" environment. By adopting control actions (eg dynamic decoupling or modulation of the environment) to combat decoherence. To characterize and control quantum systems, it is extremely important to obtain a dynamic model of a quantum system in accordance with the measurement result of the system. For closed systems, this requires identification of the Hamiltonian. Otherwise, when the system is open, the scattering effect must also be specified according to the noise associated with the system. Therefore, obtaining information about noise is critical for modeling the dynamics of an open quantum system. Knowledge of noise is also useful for improving the accuracy of state and process tomography from a series of developed quantum measurements. Since the number of measurements required for process tomography rapidly increases with the size of the system, a good model of the dynamics of the measured system will help reduce the cost of measurement based on predicting the actual process matrix.

Physically stationary noise in a quantum system is mainly characterized by its correlation spectrum. In many solid-state systems, this spectrum has nontrivial structures (for example, Lorentz shape, ohmic, 1 / f noise), which corresponds to color noise. Typically, noise causes random frequency shift and scattering effects in system dynamics, which can be described by a generalized general equation or a spectral overlap function with a noise correlation spectrum. Such models have been used to derive a widely used identification strategy based on the Fermi golden rule, which states that the rate of transition of states of a two-level system is proportional to the intensity of the noise spectrum at the resonant frequency. This makes it possible to experimentally analyze noise using a tunable qubit. The spectrum of the phase noise is determined by direct measurement of the asymptotic decay of the nuclear spin qubit in NMR experiments. The strategy requires that the designed π-pulse trains suppress the dominant influence of the low-frequency component of the noise spectrum.

5. Grover's search algorithm
As a result of the action of the gate on the top two qubits, we get an idea that applies to the case of an arbitrary number of n qubits. We start by flipping the sign of the probability amplitude that corresponds to the desired location. Then we perform a flip relative to the average. However, in this case, the amplification of the amplitude is not as significant as in the situation with two qubits. Take eight numbers for example, seven of which are 1 and one is -1. Their sum is 6, and the average is 6/8. After a flip relative to the average, 1 becomes 1/2, and -1 becomes 10/4. As a consequence, with three qubits, by measuring one qubit after amplifying the amplitude, we are more likely to obtain the desired location than others. The problem is that there is a significant chance of getting the wrong answer. We need a higher probability of getting the correct answer - we need to further amplify the amplitude before measuring. The solution is to pass all the qubits back through the circuit. We again flip the sign of the probability amplitude associated with the desired location, and again flip around the mean.

Consider a generalized case. We need to find something that is in one of m possible locations. To find it in the classical way, in the worst case, we have to ask m - 1 questions. The number of questions grows in proportion to m. Grover calculated a formula that determines how many times to use his chain to get the maximum probability of a correct answer. The number that this formula gives grows in proportion to image. This is a quadratic acceleration. At its core, this is the task of unstructured search. If there is an unordered data set ("haystack") and you want to find one element in it that satisfies a specific requirement ("needle"), the Grover algorithm will help, since the alternative is a
classic enumeration of options.

We are given a function \( f : \{0,1\}^n \to \{0,1\} \) in the form of a set of logical operators "AND" and "OR". The function returns “true” for only one binary string of zeros and ones. Encoding such a function can be relatively simple, however, to find out for which combination of zeros and ones the function returns “true”; in the worst case, \( 2^n \) function calls on a classic machine are required. On a quantum computer we can express \( f \) as a valid set of quantum logic elements that make up the oracle (indisputable truth) \( U_f \), and use the Grover search algorithm to find the correct input with very high accuracy only in \( \sqrt{2^n} \) iterations.

For example, imagine a certain function. In the classical style, the function is expressed:

\[
  f(x) = \begin{cases} 
    1, & x = 10 \\
    0, & x \neq 10 
  \end{cases}
\]

(1)

Then we need to introduce a function \( f : \{0,1\}^n \to \{0,1\} \) in the form of a quantum oracle \( U_f \), this task is not simple, since the first assumption \( U_f|\psi\rangle = U_f|\psi\rangle \) it is not a unitary and reversible operation, therefore it is not a quantum input (\( |x\rangle \) consists of two qubits, whereas \( |f(x)\rangle \) from only one qubit). However, there is a method for constructing a quantum oracle by adding a control qubit and preparing it in a state \( |\psi\rangle = \frac{1}{\sqrt{2}} (|0\rangle - |1\rangle) \), so the action \( U_f : \)

\[
  U_f |\psi\rangle |\chi\rangle = (-1)^{f(x)} |\psi\rangle |\chi\rangle
\]

(2)

A rotation of the amplitude sign in the case \( f(x) = 1 \) is necessary. In our analysis, we discard the control qubit and express the action of the quantum oracle as:

The state of a quantum system is established as an overlap of all possible input data, so the probability of finding the desired output signal increases with each iteration of the inversion circuit algorithm around the mean [3,5,6].

We analyze each step of the algorithm on the basis of an example in the Python programming language using the Rigetti Forest platform (Rigetti systems are used in the public and private sectors to accelerate innovation and open solutions for more complex activities). To get started, we import all the necessary libraries and functions:

```python
import numpy as np
From pyquil.quil import Program
From pyquil.api import QVMConnection
From pyquil.gates import H, I
```

6. Applications of Grover's algorithm

There are several problems with the implementation of the algorithm. First, the quadratic speedup is estimated relative to the complexity query. To use an oracle, it must be created, and if you do not treat this task with due care, the number of steps performed by the oracle will outweigh the number of steps that the algorithm saves, and as a result, the algorithm will become slower, not faster than the classical one. Another problem is that when we define speedup, we assume that the dataset is disordered. If a dataset has a specific structure, it is often possible to find a classic algorithm that uses this structure and finds a solution much faster. The last problem has to do with acceleration. Quadratic acceleration is nothing more than the exponential acceleration that we have observed in other algorithms. Can you do more? Let's look at these issues.

Both problems related to the implementation of the oracle and the presence of a structure in the dataset are valid and show that in most cases Grover's algorithm has no practical application for searching the database. But in some situations, the presence of a structure in the data makes it possible to create an oracle that works with high efficiency. In such situations, the algorithm can outperform classical algorithms. The answer to the question of the possibility of achieving greater success has already been given. It has been proven that Grover's algorithm is optimal. There is no quantum algorithm that can solve a problem with more than quadratic acceleration. Quadratic acceleration, while not as impressive as exponential, still has some benefits. When working with large datasets, any
speedup can be valuable [7]. Probably, the main application of Grover’s algorithm will find not for searching, as it was presented above, but for its variations. In particular, the idea of amplifying the amplitude can come in handy.

7. Quantum oracle
Create a quantum oracle that marks the desired string. We continue the previous example with a string of length 10 found [8]. Note that this is for presentation purposes only, the whole point is that a quantum oracle is already assigned to the algorithm, and we are trying to determine for which input the function returns “true”.
It is the simple case n = 2.
SEARCHED_STRING = "10"
N = len(SEARCHED_STRING)
oracle = np.zeros(shape=(2 ** N, 2 ** N))
for b in range(2 ** N):
    if np.binary_repr(b, N) == SEARCHED_STRING:
        oracle[b, b] = -1
    else:
        oracle[b, b] = 1
print(oracle)
The quantum oracle is expressed by the matrix:

\[
U_f = \begin{bmatrix}
1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 \\
0 & 0 & -1 & 0 \\
0 & 0 & 0 & 1
\end{bmatrix}
\]

At this point, it is important to understand that the construction of the quantum oracle for this example has already shown to which input the function returns “true”. This is for presentation purposes only, but in practice, we have a black box quantum oracle that is already assigned to the algorithm[9].

8. Initialization
Let’s continue and create a connection to the Quantum virtual machine, as well as initialize the program [10].

qvm = QVMConnection()
gr_prog = Program()
First, we initialize each of n = 2 qubits in the state | 0 \rangle. The RigettiForest package recognizes qubits in the reverse order; therefore, it is useful to create a list containing qubit indices in descending order [11].
qubits = list(reversed(range(N)))
gr_prog.inst([I(q) for q in qubits])
Our current status: |ψ0⟩=|00⟩
Creating a superposition.
The next step is to bring the system into a superposition with the Hadamard gate:
gr_prog.inst([H(q) for q in qubits])
The state of the system after this step is expressed as follows:

\[
|ψ⟩ = H^\otimes n |0⟩ = \frac{|0⟩+|1⟩}{√2} \otimes \frac{|0⟩+|1⟩}{√2} = \frac{1}{2} (|00⟩ + |01⟩ + |10⟩ + |11⟩) = \frac{1}{1} \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}
\]

The state represents all possible inputs for a quantum oracle in an equally weighted superposition. We
used only 2 qubits to represent 4 input data.

9. The loop

We pass to the part of the algorithm that will be repeated about $\frac{\pi}{2\sqrt{n}}$ times [12]. Two steps will be included in the loop:
1. The application of the quantum oracle $U_f$
2. Inversion around the middle.

A quantum oracle has already been specified as input. Add it to the program:

```
ORACLE_GATE_NAME = "GROVER_ORACLE"
gr_prog.defgate(ORACLE_GATE_NAME, oracle)
```

What does inversion around mean? To understand what inversion around the mean means, let’s see what happens at the first iteration after we apply the quantum oracle [13].

Action $U_f$ - flip the sign of the amplitude of the desired "string 10":

$$|\psi_1^{(1)}\rangle = U_f |\psi_1\rangle = \frac{1}{2}(|00\rangle + |01\rangle - |10\rangle + |11\rangle) = \frac{1}{2} \begin{bmatrix} 1 \\ 1 \\ -1 \\ -1 \end{bmatrix}$$

(5)

The quantum oracle is not enough to recognize the desired input, because the sign of the amplitude does not affect the probability of measurement. We must look for additional quantum gates that increase the absolute value of the amplitude for the desired state [14]. The answer comes with an inversion around the mean (also called the diffusion operator), which is defined as $D=2A-I\otimes2$.

A expresses as:

$$A = \frac{1}{n} \begin{bmatrix} 1/n & \ldots & 1/n \\ \vdots & \ddots & \vdots \\ 1/n & \ldots & 1/n \end{bmatrix}$$

(6)

In our example, this is:

$$A = \frac{1}{2} \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}$$

(7)

and the inversion around the mean is:

$$D = 2A - I\otimes2 = 1/2 \begin{bmatrix} -1 & 1 & 1 & 1 \\ 1 & -1 & 1 & 1 \\ 1 & 1 & -1 & 1 \\ 1 & 1 & 1 & -1 \end{bmatrix}$$

(8)

You can verify that this is a unitary matrix, that is, real quantum gates. System state after applying inversion around the average value in the first iteration:

$$|\psi_2^{(1)}\rangle = D|\psi_1^{(1)}\rangle = \frac{1}{4} \begin{bmatrix} -1 & 1 & 1 & 1 \\ 1 & -1 & 1 & 1 \\ 1 & 1 & -1 & 1 \\ 1 & 1 & 1 & -1 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

(9)

The search string has a probability amplitude of $\frac{\pi}{4\sqrt{2}} \approx 1$ iteration. This means that when you measure qubits, you always get the desired “string 10”, however this is definitely not the case for longer strings (when the number of qubits is more than two).

We implement a loop in Python [15]:

```
# Define quantum oracle
ORACLE_GATE_NAME = "GROVER_ORACLE"
gr_prog.defgate(ORACLE_GATE_NAME, oracle)
```
Define inversion around the mean

```python
DIFFUSION_GATE_NAME = "DIFFUSION"
diffusion = 2.0 * np.full((2**N, 2**N), 1/(2**N)) - np.eye(2**N)
gr_prog.defgate(DIFFUSION_GATE_NAME, diffusion)
```

# Number of algorithm iterations

```python
N_ITER = int(np.pi / 4 * np.sqrt(2**N))
```

# Loop

```python
for i in range(N_ITER):
    # ψ_2^i: Apply Quantum Oracle
    gr_prog.inst(tuple([ORACLE_GATE_NAME] + qubits))
    #print(qvm.wavefunction(gr_prog))
    # ψ_3^i: Apply Inversion around the mean
    gr_prog.inst(tuple([DIFFUSION_GATE_NAME] + qubits))
    #print(qvm.wavefunction(gr_prog))
```

Here's what the response of the RiggetiForrest service will look like when launching the Grover algorithm with some given values (fig. 1):

![Figure 1. Graphic answer.](image)

As we can see, the answer is built using vectors in three-dimensional space, which helps to visually see the operation of the algorithm.

### 10. Measurement

The last step is to measure the qubits and calculate the result:

```python
# ψ_5: Measure
for q in qubits:
gr_prog.measure(qubit_index=q, classical_reg=q)
```

# Run

```python
ret = qvm.run(gr_prog, classical_addresses=qubits)
ret_string = "".join([str(q) for q in ret[0]])
print("The searched string is: {}".format(ret_string)).
```

### 11. Research results and discussion

As already mentioned, for case \( n = 2 \), the desired order is measured with a probability equal to one. You can try to use a longer string, for example, `SEARCHED_STRING = "1011010"`. In this scenario, the probability of determining the correct answer after \( \frac{\pi}{4\sqrt{2^n}} \) iterations is about 99.6%. This is one of the differences between classical and quantum computers. Many quantum algorithms with some probability return the correct answer, whereas classic computers have some calculation results (as long as there is no noise) [6]. This occurs regardless of quantum noise, i.e. H. The uncertainty of the results is an integral property of this quantum algorithm. In order to be 100% sure that quantum computers will react, you have to take several measurements. Repeating the algorithm several times has no significant influence on the
quadratic acceleration for large \( n \) (fig. 2).

12. Conclusion

Theoretically, the algorithm provides square acceleration compared to classic computers. This is not exponential acceleration, but it is still valid for large data vectors. The Grover search algorithm is based on the simultaneous change of amplitudes of all inputs. This is due to the superposition of states, which is a purely quantum concept. In addition, the search is done globally, which indicates a significant improvement in the optimization process. In turn, the Grover algorithm depends on the number of iterations. The more iterations, the lower the amplitude of the correct answer, so the wrong choice of this parameter can digest the solution. In addition, the algorithm's functionality for introducing noise into the quantum system is limited, which is true in modern quantum computers.

References

[1] Radhwan M A, Radhwan B, Ali A A and Abedallah Z A 2019 Procedia Computer Science, 160, 54-60 https://doi.org/10.1016/j.procs.2019.09.443
[2] Dyakonov M I 2019 Solid-State Electronics, 155, 4-6 https://doi.org/10.1016/j.sse.2019.03.004
[3] Cafaro C, Mancini S and Physica A 2012 SMnd its Applications, 391(4), 1610-1625 https://doi.org/10.1016/j.physa.2011.09.018
[4] Luan L, Wang Z and Liu S 2012 Energy Procedia, 16(C), 1701-1706 https://doi.org/10.1016/j.egypro.2012.01.263
[5] Liu Y, Koehler G J 2010 European Journal of Operational Research, 207(2), 620-632 https://doi.org/10.1016/j.ejor.2010.05.039
[6] Pan M, Qiu D, Mateus P and Gruska J 2019 Theoretical Computer Science, 773, pp 138-152 https://doi.org/10.1016/j.tcs.2018.10.001
[7] Hines P 2010 Theoretical Computer Science, 411(11–13), pp 1501-1520 (2010)
[8] Colnaghi T, Mauro D’Ariano G, Facchini S and Perinotti P 2012 Physics Letters A, 376(45), pp 2940-2943 https://doi.org/10.1016/j.physleta.2012.08.028
[9] Shrivastava P, Soni K K and Rasool A 2020 Procedia Computer Science, 167, pp 1849-1860 https://doi.org/10.1016/j.procs.2020.03.204
[10] Cafaro C 2017 Physica A: Statistical Mechanics and its Applications, 470, pp 154-196 https://doi.org/10.1016/j.physa.2016.11.117
[11] Lu X, Yuan J and Zhang W 2013 Computer Physics Communications, 184(9), pp 2035-2041 https://doi.org/10.1016/j.cpc.2013.03.017
[12] Hosoyamada A, Sasaki Y, Tani S and Xagawa K 2020 Theoretical Computer Science, 842, pp 100-117 https://doi.org/10.1016/j.tcs.2020.07.039
[13] Nikahd E, Houshmand M, Saheb M Z and Sedighi M 2015 Microprocessors and Microsystems, 39(3), pp 210-222 https://doi.org/10.1016/j.micpro.2015.02.007,
[14] Avila de A B, Reiser R H S, Pilla M L and Yamin A C 2020 Neurocomputing, 393, pp 223-233 https://doi.org/10.1016/j.neucom.2019.01.118
[15] Izaac J A and Wang J B 2015 Computer Physics Communications, 186, pp 81-92 https://doi.org/10.1016/j.cpc.2014.09.011