Using IBM Qiskit to Simulate a Quantum Computer: How the Deutsch-Jozsa Algorithm Demonstrates the Increased Computational Power of Quantum Computers over their Classical Counterpart

6636836
Department of Physics, University of Surrey, Guildford, GU2 7XH, UK

Quantum computers have been causing large discussion since the 1970s. In 1992, David Deutsch and Richard Jozsa proposed an algorithm that demonstrates the higher computational power of quantum bits, or qubits, over classical bits. The algorithm utilised some of the functions of quantum particles such as superposition and entanglement. It describes a function that acts upon a register of qubit inputs. It is not known whether the function is constant, returns uniform outputs (all $|0\rangle$ or all $|1\rangle$) or balanced, 50% of returning $|0\rangle$ or $|1\rangle$. Hadamard gates were used to place all input qubits into a superposition state of $|0\rangle$ and $|1\rangle$, and the oracle was then queried. The outputs of the oracle were then acted upon by another set of Hadamard gates to return the qubits to a determined state of $|0\rangle$ or $|1\rangle$. In this project, the algorithm was coded using Python and Qiskit in a Jupyter Notebook, and circuits were constructed and simulated within a for loop, recording the simulated results for each new unique system. The quantum solution to the algorithm proved to be successful in assessing the nature of the oracle 100% of the time, which agrees with the literature from Deutsch and Jozsa. IBM offer the option to run your quantum circuits on real quantum computers using their software GUI, which is a logical next step in this work as it would truly put the code constructed in this project, to the test. An investigation into Shor's and Grover's algorithms would also be ideal prospective work concepts as they tackle more advanced problems using more complex circuits.

I. INTRODUCTION & THEORY

Computers have developed hugely over the last 3 decades. In 1981, IBM mass produced a microcomputer designed to be used by the everyday consumer within households and workplaces, aptly named the 'IBM Personal Computer' (or model 5150).[1] The baseline model had 16 KB of RAM and had an 8-bit system. When comparing the power of the model 5150 to modern day PCs, it is startling to see the sheer improvements and innovative solutions to the machines, with the latest models featuring 64 GB RAM. As computers get more powerful, the faster certain problems and algorithms can be solved. Computer bits can be either 0 or 1, known as binary, and the combinations of these bits in different states are the buildings blocks of almost all computers. Separate from the world of computer science, the study of quantum mechanics and the apparent behaviours of subatomic particles had grown substantially since the workings of Schrödinger, Einstein, and Dirac. Quantum mechanics demonstrates that a particle, such as an atom or an electron, can exist in different energy states, or even a superposition of states.[2] This superposition of states can be mathematically shown to exist until a measurement is made, leading to the collapse of the superposition and one of the energy levels is measured, with probabilities on each level. In the 1970s and 1980s, research was conducted, investigating the concept of using quantum systems for decision making models (Holevo, 1973)[3] or discussing the implementation of quantum systems into classical Turing machine models (Benioff, 1982).[4] The field of quantum computing had been established but there had not yet been a true comparison between a quantum algorithm and a classical algorithm.

In 1992, David Deutsch and Richard Jozsa proposed an algorithm that features a *'black-box'* function (oracle) that takes inputs of bits and outputs either 0 or 1. The exact action the oracle conducts upon the bits is unknown and is either 'constant' or 'balanced'. The former description would produce all 0s or all 1s no matter the input, and the latter oracle produces a 50/50 output of 0 and 1. The aim of the algorithm is to uncover whether the function is constant or balanced. This query proved to be the first example of quantum systems providing a definite solution to a problem that a classical system has no analytical solution for and it is practically impossible with large inputs.

CLASSICAL ORACLE

Prior to the 1992 proposal of the Deutsch-Jozsa algorithm, David Deutsch conducted work in 1985[5] discussing a more specific problem that the subsequent algorithm generalises. Deutsch's earlier algorithm describes a classical oracle which takes an input of one bit, and outputs one result, 0 or 1.

$$f: \{0,1\} \rightarrow \{0,1\} \qquad [1]$$

To understand the nature of the oracle, whether it is constant or balanced, the system would have to be ran at least twice to query the oracle. The Deutsch-Jozsa algorithm builds upon this idea and discusses the problem with multiple, or n, number of inputs.

$$f: \{0,1\}^n \rightarrow \{0,1\} \qquad [2]$$

In this more general case, a classical solution appears to become unfeasible as n grows large. The potential number of outputs of the oracle with n inputs is 2^n. Uncovering the nature of the function would require querying the oracle a maximum of 2^{n-1} +1 times, equalling 1 + half of all output possibilities. With a small n, this is achievable, but there is no analytical solution that provides confidence when n gets larger.

QUANTUM ADVANTAGE

Qubits can be in states 0 or 1, just like classical bits, however they can also be in a superposition of states, composed of certain amounts of either state. A qubit can be described in the form seen in equation [3].

$$|\psi\rangle = \alpha\,|0\rangle + \beta\,|1\rangle \qquad [3]$$

where α and β represent probabilities of observing that state when a measurement is made and the wavefunction collapses. This quantum property of superposition allows for multiple calculations and computations to be done upon multiple states at once. When 2 qubits are in a system, they can be entangled within a combined wavefunction, with their states becoming dependent on the other. This is shown through a tensor product of 2 wavefunctions, or qubits, creating a combined wavefunction, $|\Psi\rangle$, in equation [4].

$$|\Psi\rangle = |\psi_1\rangle \otimes |\psi_2\rangle \qquad [4]$$

Qubits can be entangled in many ways, but some of the most widely known examples of an entangled state are the Bell States. The unique property of a quantum Bell state is the nature in which when a measurement is made upon one of the qubits, the state of the other is immediately known also. See equation [5] for an example of this.

$$|\Psi\rangle = \frac{1}{\sqrt{2}}(\,|0,1\rangle - |1,0\rangle) \qquad [5]$$

From this Bell state, we can observe that if a measurement is made on qubit 1, whether it is in state $|0\rangle$ or $|1\rangle$ also confirms the state of qubit 2. This means that with one measurement of the system, information about 2 qubits is returned. These properties alone grant huge potential for quantum systems to process more information using the same, or less, bits and resources than a classical computer. However, for this to be enabled, efficient and clever use of logic gates in quantum systems must be carried out. Classical logic gates include AND, OR, NOT gates that flip and amend the binary state of classical bits in unique ways. Quantum systems have similar gates that can perform actions upon the state of the qubits in a system, but the main difference seen is that all quantum gates must be unitary operators, or *reversible*. This fact implies that in a quantum system or circuit, no information is lost in either direction of time, allowing an input to be uncovered given the output or vice versa, if the operators are known. There are numerous quantum gates, but the key ones needed for the Deutsch-Jozsa Algorithm are:

- NOT (Pauli-X) gate – A single qubit gate. Flips the state of the qubit. (0→1 or 1→0). It performs a 180° rotation in the axis of the Bloch Sphere.
- CNOT gate – A 2 qubit gate. Enacts a NOT gate on a target qubit if the control qubit is in state $|0\rangle$.
- Hadamard gate (H) – A single qubit gate. Places the qubit in a 50/50 superposition of states.
$$|\psi\rangle = \frac{1}{\sqrt{2}}(\,|0\rangle \pm \,|1\rangle)$$
If the qubit was in state $|0\rangle$, it is +, if it was in state $|1\rangle$, it is -.

THE 1 QUBIT ALGORITHM

To best describe the fundamentals of the algorithm, the 1 qubit situation will be discussed (Deutsch Algorithm). This example is shown in further detail in Mark Fox's book.[6] Imagine 2 qubits, q_1 and q_2 and both are within the same system. q_1 begins in state $|0\rangle$ and q_2 in state $|1\rangle$. See Figure [1] for a quantum circuit diagram of the algorithm.

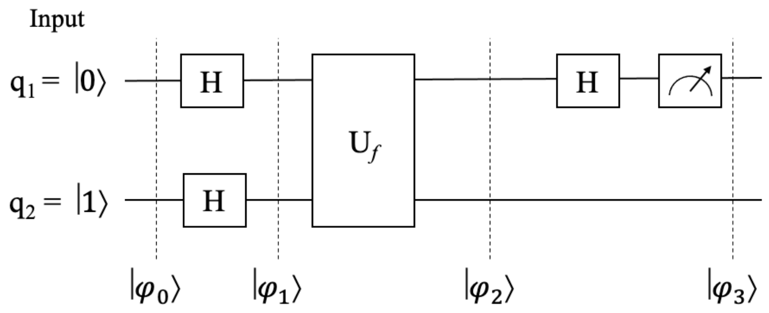

Figure [1] Quantum circuit diagram showing the 1 qubit situation in the Deutsch-Jozsa algorithm (equivalent to Deutsch algorithm, 1985). Despite 2 qubits involved in the circuit, only q_1 is measured at the end of the circuit. H gates are Hadamard gates, U_f is the quantum oracle which is what we are investigating to see if it's constant or balanced. The dial symbol at the end of the q_1 symbolises a measurement being taken of that qubit.

$|\varphi_n\rangle$ denotes the combined wavefunction of the qubit system at that point in the circuit. At $|\varphi_0\rangle$, the combined wavefunction is $|0,1\rangle$, with the 2 states representing q_1 and q_2 respectively. Hadamard gates are then applied to both qubits to place them in a superposition of states:

$$q_1 = \frac{1}{\sqrt{2}}(\,|0\rangle + \,|1\rangle)$$

$$q_2 = \frac{1}{\sqrt{2}}(\,|0\rangle - \,|1\rangle)$$

After the Hadamard gates, the combined wavefunction is:

$$|\varphi_1\rangle = \frac{1}{2}(\,|0\rangle + \,|1\rangle)(\,|0\rangle - \,|1\rangle)$$

$$|\varphi_1\rangle = \frac{1}{2}(\,|0,0\rangle - \,|0,1\rangle + \,|1,0\rangle - \,|1,1\rangle)$$

The operation within the oracle, U_f performs a function only upon q_2, where the function is denoted as $q_2 \oplus f(q_1)$. The function that is performed upon q_2 depends on the value of q_1. Following the oracle, the combined wavefunction becomes:

$$|\varphi_2\rangle = \frac{1}{2}(\,|0, f(0)\rangle - \,|0,1 \oplus f(0)\rangle + \,|1, f(1)\rangle - \,|1,1 \oplus f(1)\rangle)$$

With a constant oracle, the output is always the same, meaning the input of q_1 does not impact the function, so all function terms can be made equivalent. For a balanced oracle, $f(1) \neq f(0)$, and therefore $f(1) = 1 \oplus f(0)$.

$$|\varphi_2\rangle^{constant} = \frac{1}{2}(\,|0\rangle + \,|1\rangle)(\,|f(0)\rangle - \,|1 \oplus f(0)\rangle)$$

$$|\varphi_2\rangle^{balanced} = \frac{1}{2}(\,|0\rangle - \,|1\rangle)(\,|f(0)\rangle - \,|1 \oplus f(0)\rangle)$$

A Hadamard gate is then applied on q_1, resulting in the wavefunctions for constant and balanced oracles respectively:

$$|\varphi_3\rangle^{constant} = \,|0\rangle \frac{1}{\sqrt{2}}(\,|f(0)\rangle - \,|1 \oplus f(0)\rangle)$$

$$|\varphi_3\rangle^{balanced} = \,|1\rangle \frac{1}{\sqrt{2}}(\,|f(0)\rangle - \,|1 \oplus f(0)\rangle)$$

It can be seen clearly that when q_1 is measured, if the result is $|0\rangle$, we can conclude the oracle (function) was constant, and if the result is $|1\rangle$ it is a balanced function. Only one measurement is needed to assess the nature of the oracle, which as opposed to at least 2 measurements in a classical circuit, in this 1 qubit example.

The Deutsch-Jozsa algorithm is the same principle of a single measurement of the system, but instead it is generalised to accommodate for n input qubits. Following the mathematics in Thomas Wong's book,[7] we expect the output of the qubits in the Deutsch-Jozsa algorithm to be $|0,0,\ldots,0\rangle$ if the oracle is constant. Therefore, if the output isn't $|0,0,\ldots,0\rangle$, it is a balanced oracle.

II. COMPUTATIONAL DETAILS

Python was used in a Jupyter Notebook to construct and assess the Deutsch-Jozsa algorithm, and the IBM Qiskit library was used. IBM are one of the largest companies that are actively working towards assembling and optimising physical quantum computers. They introduced the '*IBM Quantum Platform*'[8] in 2016, allowing public use of cloud-based quantum computing. The main functions of Qiskit used in this research are the circuit building and simulation of the circuits. However, there are many different sub-packages that cater to certain disciplines. It has a huge community that will only grow larger in the coming years. The code constructed in this work is based upon the IBM webpage describing the algorithm.[9]

Within the code for this research, the *QuantumCircuit, Aer* and *transpile* functions were imported form the Qiskit library. Also, from the *qiskit.visualisation* sub-package, *plot_distribution* was also imported. The QuantumCircuit is the function that takes the number of qubits as an argument and proceeds to create a quantum circuit, like the style of Figure [1]. The properties of these circuits were trialled with, using the *.x(), .h(), .cx()* functions. These correspond to NOT, Hadamard and CNOT gates respectively.

Following the testing of the Qiskit functions to gain an understanding of the mechanics of the package, 2 functions were constructed: Oracle and Circuit. The former function creates a quantum oracle that is either constant or balanced. It takes as arguments, the type of oracle (*'constant'* or *'balanced'*), and the number of qubits in the circuit. Before discussing the circuit details, in quantum circuits, the inputs of all qubits are deemed to be in state $|0\rangle$, unless specified otherwise. For this work, we assume all qubits begin in state $|0\rangle$.

If the oracle is requested to be *'balanced'*, a string of qubit outputs is generated using NumPy .random.randint() function. This output string is then iterated through using *for* loops. First, where there is a '1' in the output string, a quantum X (NOT) gate is imposed on that qubit. CNOT gates are then applied to all qubits in the circuit. The target qubit is acted upon by each of the control qubits one at a time, entangling their states together. Finally, another X gate is applied to any qubits that had an original value of '1' in the output string. There are many ways an oracle can be coded to be 'balanced', but for this work, the sequence specified here is the form of the 'balanced' oracles.

If the oracle is requested to be *'constant'*, the construction is much simpler. Due to the nature of a constant function being independent of the input, there are no CNOT gates involved. A random output of 0 or 1 is obtained via .random.randint(). If it is 1, an X gate is placed upon the answer qubit. The oracle is returned at the end of the function using the Qiskit function of .to_gate() which turns a QuantumCircuit object into an object that can be placed within another circuit.

Another function was made which creates the overall circuit object. It takes the arguments of an oracle, and the number of qubits (n). It creates a QuantumCircuit object consisting of n+1 qubits. It then places an X gate and H gate upon the n^{th} qubit (answer qubit), putting it into a negative superposition state. All other qubits are run through H gates, putting them in a positive superposition state. The oracle is appended within the circuit, deploying additional gates depending on the 'constant' or 'balanced' nature of it. Following the oracle, all qubits, other than the n^{th}, are ran through H gates again. The Qiskit function .measure() conducts a measurement on the qubits, causing the wavefunctions of the qubits to collapse. This measurement, and the circuit, can be simulated by the Qiskit *Aer* sub-package, which is a quantum computing high-resolution simulation. This simulator was downloaded via the *Aer.get_backend('aer_simulator')* command. The Transpile sub-package can then be used to associate our quantum circuit with the Aer simulator and is then ran to obtain results of the circuit. An evaluation can then be made between the results and the definition of the oracle made by the user, to observe if the quantum solution to the algorithm correctly assesses the oracle.

III. RESULTS

To formally assess the repeatability of the quantum solution for the Deutsch-Jozsa algorithm, a *for* loop is used which randomly assigns a qubit number along with the nature of the quantum oracle. The loop uses random.randint(1,10) to assign a qubit number, and then random.randint(2) and an *if* statement is used to decide the oracle type, constant or balanced. See Figures [2] and [3] for constant and balanced functions plotted from Qiskit and Matplotlib. Both quantities are compiled in lists for further assessment. Oracle and circuit objects are created using the functions described in the previous section, and are then ran on the Aer simulator, with the results being recorded and appended to a list.

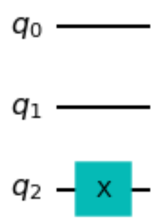

Figure [2] Constant function produced from Qiskit and plotted with MatPlotLib. 3-qubit example. X gate shown on q_2

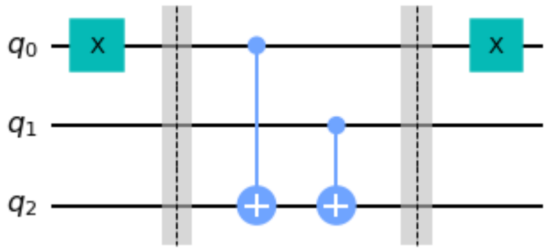

Figure [3] Balanced function produced from Qiskit and plotted with MatPlotLib. 3-qubit example. X gates on q_0 and CNOT gates acting upon q_2, based on the states of q_0 and q_1.

These oracles are examples of constant and balanced types, on a 3-qubit system. They are appended to a larger quantum circuit consisting of the additional gates mentioned prior. See figure [4] for an example of a full quantum circuit demonstrating the Deutsch-Jozsa algorithm.

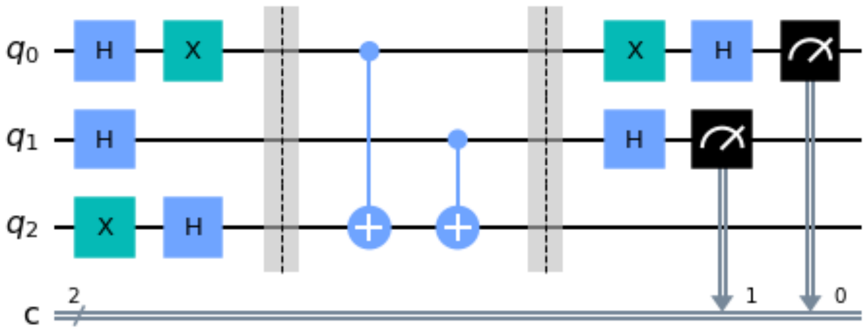

Figure [4] Complete quantum circuit constructed form qiskit.QuantumCircuit(). This example is for a 3-qubit system, and features a balanced oracle, the same one seen within Figure [3]. The c_2 lines featured at the bottom of the circuit represent classical bits, used to measure, and interact with the qubits, shown with the measurement dials at the end of the circuit.

The circuit in Figure [4] is purely an example circuit produced from the constructed code and not all were plotted, despite being simulated via the Aer simulator. The loop created 10 of these circuits, all with random qubit numbers, and randomly selected oracle. The results were compiled within Table [1] which shows the qubit number, known nature of the quantum oracle, and the actual simulated output of the circuit.

The Deutsch-Jozsa algorithm is already known to have a 100% accuracy when solved via the quantum solution as seen in this work, shown by Deutsch and Jozsa in their 1992 work. But it is also important to demonstrate this via multiple iterations of the algorithm to provide further confidence on this claim. It can be seen from Table 1 that on each iteration, despite the random qubit number or oracle nature, the quantum solution successfully queried and assessed the nature of the oracle 100% of the time. The qubit number selection in the iterations could have been heightened, but for the purposes of demonstrating the success rate of the quantum solution, 10 is sufficient. With these lower levels of qubits, a classical computer could eventually solve the oracle nature. However, from the theory, and the simulations ran during this work, it is clear to see that when the qubit number is raised, the classical computational time grows exponentially, making it practically impossible to solve. The simulations in this work cannot confirm this computational speed increase as we are simulating quantum computing on a *classical* computer.

Qubit Number	Oracle Type (known)	Deutsch-Jozsa Quantum Circuit Output	Oracle Type (Output)
2	Constant	00	Constant
8	Constant	00000000	Constant
6	Balanced	111111	Balanced
3	Constant	000	Constant
1	Balanced	1	Balanced
8	Constant	00000000	Constant
1	Constant	0	Constant
4	Constant	0000	Constant
6	Balanced	111111	Balanced
6	Balanced	111111	Balanced

Table [1] Inputs for quantum oracle and quantum circuit functions arguments (number of qubits and oracle nature), and the circuit output and the corresponding oracle type. Qubits were set to be chosen randomly (1-10) on each iteration and a 50/50 chance of the function being *constant* or *balanced*. The circuit output was recorded for each iteration.

Despite this, IBM offer users to run their circuits on a real quantum computer using their IBM Quantum Interface. Figure [5] is a bar chart provided by IBM Quantum, found on the Deutsch-Jozsa page.[9] It plots the results from running the Deutsch-Jozsa algorithm on their real quantum computers, and shows the spread of results. Physical qubit computers are prone to errors due to the coherence of the system falls off quickly after the gate functions are carried out on the qubits. However, one result is most common, *'1111'*. The nature of the oracle that was ran on the quantum computer was balanced. The most common result from this physical test corresponds and agrees with the literature, and simulations seen in this work.

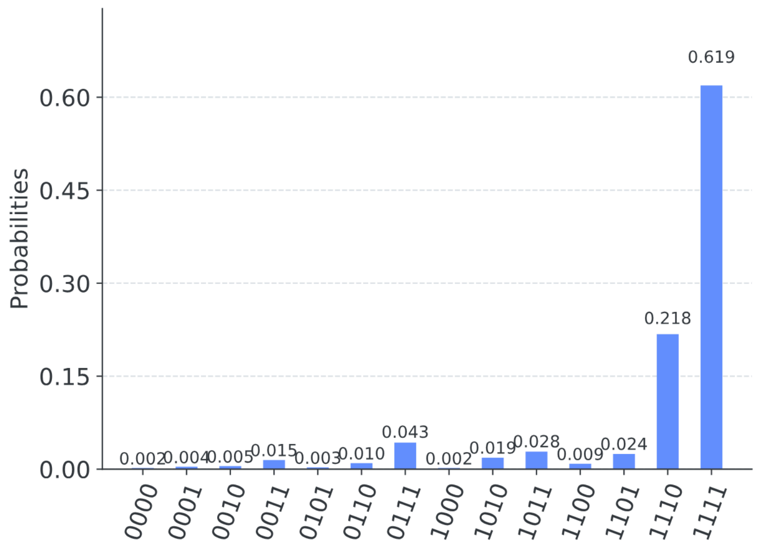

Figure [5] Plot from IBM Quantum Platform: *Deutsch-Jozsa Algorithm*[9] showing results of the quantum solution for the algorithm when ran on a real IBM quantum computer. The spread of results is expected as due to decoherence from the environment, qubits are error-prone and cannot hold superposition states for substantial periods of time. The oracle was balanced, and the results show this with '*1111*' being the most prominent result.

IV. CONCLUSION

Since the Deutsch-Jozsa algorithm was conceptualised in 1992, the world of quantum computing has grown immensely. IBM and Google are constantly updating their quantum computing models, with the latter announcing in August 2023 that the Google Sycamore Quantum Processor now has 70 qubits, a large jump from 53 qubits the year prior.[10] As the field of research and technology expand, it becomes ever more important to recall upon the early fundamental theorems that paved the way for these monumental discoveries. The Deutsch-Jozsa algorithm is a prime example. Being ahead of its time, it eloquently demonstrates the exponential increase of power quantum computation on certain tasks over classical computers. The algorithm is a great introduction to the field as it tackles core quantum principles such as superposition and entanglement, quantum logic gates (X, CNOT, Hadamard) and visualises the flow of quantum circuits and how they differ from classical circuits.

The work in this project aimed to demonstrate the power of the quantum solution, and that was achieved, showing the 100% success rate on various sized circuits. As said prior, the 'run-time' of the quantum circuit could not be tested within the simulations ran via Qiskit as it was being ran on a classical processor that is only replicating the theory behind quantum systems. However, as future work, running the simulations on actual IBM quantum processors would add further confidence in the algorithm and the code constructed in this work. In addition, the investigation into more complex algorithms such as Grover's algorithm, or Shor's algorithm would be a logical next step in this work, to assess whether Qiskit can accurately and reliably replicate these more advanced theories. Understanding these theorems is key in educating the future generations of the computers that will be used more and more frequently in the decades to come, and it is work such as this which inspires said generation to pursue this curiosity further, joining an international effort in developing the computers of tomorrow.

REFERENCES

[1] Big IBM's Little Computer, *The New York Times*, sect. D, pg.1 (13/08/1981) (https://www.nytimes.com/1981/08/13/business/big-ibm-s-little-computer.html)
[2] P. Dirac, The Principles of Quantum Mechanics (vers. 2) (1947)
[3] A. Holevo, Statistical Decision Theory for Quantum Systems, *Journal of Multivariate Analysis*, Vol.3, 4, pg. 337-394 (1973)
[4] P. Benioff, Quantum Mechanical Hamiltonian Models of Turing Machines, *Journal of Statistical Physics*, Vol. 29, 3 pg. 515-546 (1982)
[5] D. Deutsch, Quantum Theory, the Church-Turing Principle and the universal quantum computer, *Proceedings of the Royal Society of London*, Vol. 400, 1818 (1985)
[6] M. Fox, Quantum Optics: An Introduction, *Oxford University Press* (2006)
[7] T. G. Wong, Introduction to Classical and Quantum Computing, *publ. by Rooted Grove* (2022)
[8] IBM Quantum Platform: Dashboard (https://quantum.ibm.com/)
[9] Github: Qiskit, *Deutsch-Jozsa Algorithm* (https://github.com/Qiskit/textbook/blob/main/notebooks/ch-algorithms/deutsch-jozsa.ipynb)
[10] The Quantum Insider, *Google Claims Latest Quantum Experiment Would Take Decades On Classical Computer* (https://thequantuminsider.com/2023/07/04/google-claims-latest-quantum-experiment-would-take-decades-on-classical-computer/)