

Implementing the Klein Gordon Equation on a Quantum Computer

Desiree Vogt-Lee, 44354471

10 June, 2019

All components of this project including code, Jupyter Notebooks, LaTeX files and images can be found in my [Github repository](#), please check this out for reference.

1 The Klein Gordon Equation

motivation

2 Background

2.1 Quantum Computing

Quantum computing is the melding together of quantum mechanics and classical computation. The computer chips used in quantum computers use the same fundamental components as that of their classical counterparts, however are pieced together in a way which allow quantum-bits, or qubits, to be used instead of bits.

A fundamental flaw in the current design of quantum computers is the large amount of error that is created due to noise and decoherence, which increases with the number of qubits used. This severely limits the computation that can be performed with a negligible amount of error, as at some point, the current error correcting codes will have little effect.

Many believe that quantum computation is the future. Due to the superposition and entanglement properties these qubits can possess, all sorts of optimisation and new algorithms can be created, which would have significant impact on many fields of technology. For example, modern cryptography is centered around the assumption that creating incredibly long numbers take current computers an impossibly long time to factorise. Using Shor's algorithm, a quantum computer would be able to factorise a number in polynomial time, exponentially faster than the best classical algorithm, and thus being able to break modern cryptography. [1]

At present, this is all merely a premonition; as although there exists many quantum algorithms, the error is too great and the number of qubits too little to be able to use them.

2.2 Gates

To modify the state that a qubit is in, a gate or procession of gates can be applied to it. The quantum gates are similar to the logic gates present in classical computing, but work with the quantum nature of the qubit. To easily conceptualise the effect of a quantum gate, a geometric representation called the Bloch sphere is used, where the poles of the sphere represent the standard basis vectors $|0\rangle$ and $|1\rangle$.

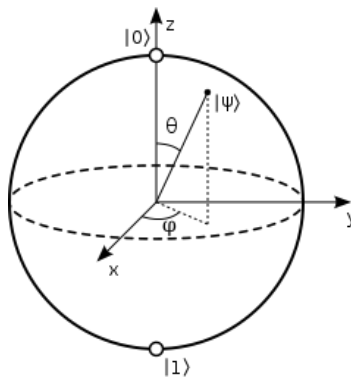


Figure 1: Bloch Sphere [2]

In classical computation, the NOT gate simply flips a given bit state: a 1 will be transformed to a 0 and vice versa. In quantum computing there is an equivalent X gate which when applied to a qubit in the $|0\rangle$ state, performs a rotation about the x axis on a Bloch sphere by π radians resulting in the $|1\rangle$ state when measured (for more detailed explanation on the fundamental gates and terminology, please see my [blog post](#)).

Here I will briefly explain the gates that will be used to create the circuit that implements the Klein Gordon equation (minus those explained in my blog post):

- **T** T: performs a $\frac{\pi}{4}$ rotation about the z axis, which is equivalent to a quarter of one Z gate operation. This gate acts on the phase of the qubit and is represented by the matrix:

$$T = \begin{pmatrix} 1 & 0 \\ 0 & e^{i\frac{\pi}{4}} \end{pmatrix}$$



- **T^\dagger** T^\dagger : simply the opposite of the T gate and applies a $-\frac{\pi}{4}$ rotation about the z axis [3].
- **U_1** U_1 : performs a unitary rotation on the state vector where the continuous phase can be specified. It is represented by the following matrix and as can be seen, is of a similar to

that of the T gate, suggesting that a $U_1(\frac{\pi}{4})$ gate is equivalent to a T gate.

$$U_1(\lambda) = \begin{pmatrix} 1 & 0 \\ 0 & e^{i\lambda} \end{pmatrix}$$

- **U3** U_3 : used to perform arbitrary unitary operation, where the θ , ψ and λ position of the vector on the Bloch sphere can be modified to any value. This is the most general form of a single qubit unitary operation and is represented in matrix form as [4]:

$$U = \begin{pmatrix} \cos(\theta/2) & -e^{i\lambda} \sin(\theta/2) \\ e^{i\phi} \sin(\theta/2) & e^{i\lambda+i\phi} \cos(\theta/2) \end{pmatrix}$$

-  Measurement: used at the end of the circuit to measure the outcome of the states in the z basis.
-  Barrier: directives which inform the compiler not to optimise across barriers. They serve the purpose of keeping gates, especially measurement gates, in their place and are as well used as a visualisation tool to separate parts of the circuit.

3 The Circuits

In the code there is an option to choose to run the circuit on a real IBM quantum computer or the simulator.

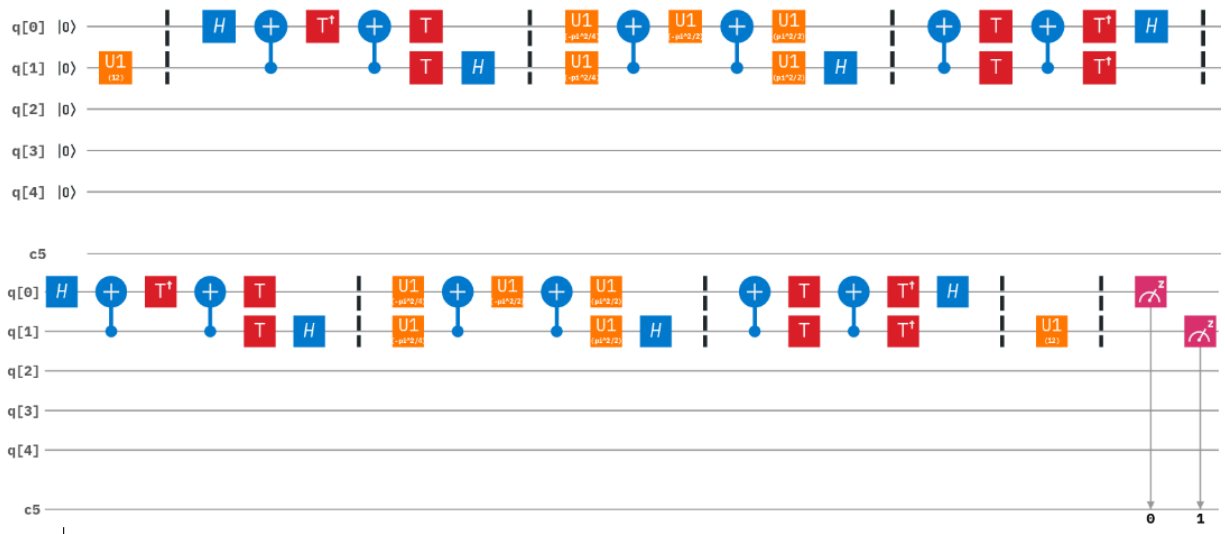


Figure 2: Circuit 1

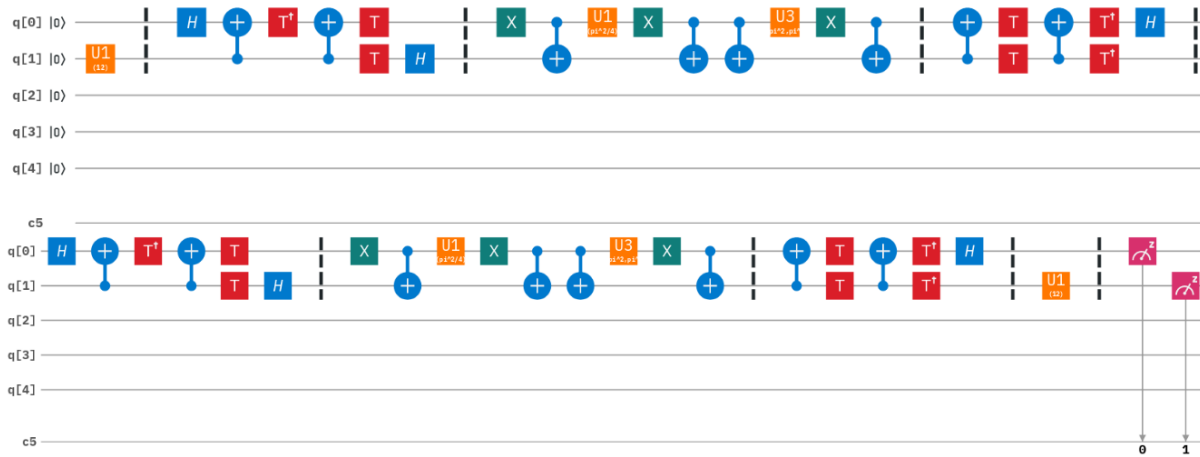


Figure 3: Circuit 2

3.1 Quantum Fourier Transform and its Inverse

4 Results

4.1 Kapil, M. et al. (2018) Results

4.2 My Results

References

- [1] H. Reich, “How quantum computers break encryption — shor’s algorithm explained.” <https://www.youtube.com/watch?v=lvTqbM5Dq4Q>, 2019.
- [2] S. Meister, “Bloch sphere.” https://en.wikipedia.org/wiki/File:Bloch_sphere.svg, 2009.
- [3] IBM, “Introducing qubit phase.” https://quantumexperience.ng.bluemix.net/proxy/tutorial/full-user-guide/002-The_Weird_and_Wonderful_World_of_the_Qubit/003-Introducing_qubit_phase.html, nd.
- [4] IBM, “Advanced single-qubit gates.” https://quantumexperience.ng.bluemix.net/proxy/tutorial/full-user-guide/002-The_Weird_and_Wonderful_World_of_the_Qubit/004-advanced_qubit_gates.html, nd.