Exact Quantum Ising Model Simulation

Qian-Rui Li, *109022115,* Jiun-Cheng Jiang, *109022126*
Department of Physics, National Tsing Hua University, Hsinchu 30043, Taiwan, Republic of China

***Abstract*—This project aims to simulate the exact solution of the quantum Ising model through quantum computers and quantum circuits. In recent years, with the continuous advancement of quantum technology, quantum computers have also begun to be able to be put into practical applications. Among them, the use of quantum computers to simulate quantum many-body systems is one of the main applications. There have been many related articles discussing the use of quantum computers to simulate and solve the quantum Ising model, however, those articles often evade relevant mathematical derivations and even have many errors and inconsistencies in mathematical details. This often makes it confusing and painful for a beginner who wants to get started with quantum simulation. Therefore, we try to use a more detailed mathematical process, especially in details that are prone to error, to solve a one-dimensional quantum Ising model with only 4 spins. We hope to use this relatively simple example, together with Qiskit, a Python package provided by IBM to facilitate quantum circuit design, to help someone who wants to get in touch with quantum simulation.**

***Index Terms*—Ising Model, Quantum Simulation, physics education**

I. Introduction

THE purpose of our project is to simulate and obtain the exact solution of the transverse-field Ising model. The best way to do it is through the quantum simulation. Since the complexity class of calculation is #P-hard problem [1], however, in a quantum computer, we just need the number of qubits, the same as the number of particles, to solve the problem and which complexity class is BQP problem, a quantum version P problem, that we can solve it much faster than a classical computer.

In this project, we are going to make a quantum simulation of the 4 spins chain transverse-field Ising model (TFIM). Our project is based on reproducing the result of 《Exact Ising model simulation on a quantum computer》 [2], but, where some mathematical formulations of techniques used in the origin paper made us confused. And therefore, we rewrite the mathematical formulations of these useful techniques in a more reasonable way.

We will first make a brief introduction to the techniques we used and the realization will be in the next section. Finally, we will show the results that we got from our oracle of the solution to TFIM.

A. Transverse-field Ising model

Transverse-field Ising model is used to solving Ising model when we considering the non-commuting observable quantities, i.e., the order of x-direction and z-direction where we measure the spin will affect the result we obtain. Hence, we need a quantum version Ising model instead of classical Ising model and what we used is Transverse-field Ising model.

Generally, TFIM can be expressed as following Hamiltonian:

$$H = -J \sum_{<i,,j>} X_i X_j + g \sum_j Z_j \tag{1}$$

where the first summation $\sum_{<i,,j>}$ is done over pairs of the nearest neighbor sites and J is a pre-factor determined by interaction and g is coupling coefficient that determines the strength of the external transverse field. Since, the sites in TFIM are fermions which they have to obey anti-commutation relation as:

$$\{c_i, c_j\} = 0 \tag{2}$$

where we will use later.

In our project, we used antiferromagnet where $J = -1$ and we are interested in the magnetization in the z-axis, thus, the measuring direction will be focus on z-direction. Also, what we constructed is Ising chain which is 1D Ising model. Therefore, the Hamiltonian of TFIM we used would be:

$$H = \sum_i \sigma_i^x \sigma_{i+1}^x + \lambda \sum_i \sigma_i^z \tag{3}$$

where λ is the relative strength of the external transverse field compared to the nearest neighbor interaction, which leads to different phases.

For $\lambda \gg 1$, the system is in disordered phase. The second term leads the Hamiltonian, where

$$H \simeq \lambda \sum_i \sigma_i^z \tag{4}$$

which has a unique ground state $|\psi\rangle = |\cdots \uparrow\uparrow\uparrow \cdots\rangle$ with $|\uparrow\rangle = \frac{1}{\sqrt{2}}(|\rightarrow\rangle + |\leftarrow\rangle)$ where $|\rightarrow\rangle$ and $|\leftarrow\rangle$ are +x and -x direction respectively. And we can see that in disordered phase, the system is quantum ferromagnetic.

On the other hand, for $\lambda \ll 1$, the system is in ordered phase. The first term leads the Hamiltonian, where

$$H \simeq \sum_i \sigma_i^z \sigma_{i+1}^z \tag{5}$$

the ground states of it would be anti-parallel configuration as the form of $|\psi\rangle = |\cdots \leftarrow\rightarrow\leftarrow\rightarrow \cdots\rangle$. In this case, the system is quantum antiferromagnetic.

As for $\lambda = 1$, the system will have quantum phase transition, at which the characteristic from ferromagnetic

turn into antiferromagnetic or from antiferromagnetic turn into ferromagnetic determined by the way of λ changes.

B. Jordan-Wigner Transformation

Jordan-Wigner transformation is a mathematical technique that mapping the spin operators (Pauli matrice) on to fermionic operators.

In this way, we can have more convenient matrix representation to obtain the result from quantum computing. And in our method, this is also a necessary step to obtain the exact solution of 1D TFIM. For details of mathematical formalism, we put it in the next section.

C. Quantum Fast Fourier Transform

In class, we have discussed how to turn discrete Fourier transform (DFT) into quantum Fourier transform (QFT).

Still, we have a mathematical technique that makes DFT into sparse matrix and improves the efficient which is called fast Fourier transform (FFT). With this concept, we can build a quantum version of it which is so-called quantum fast Fourier transform (QFFT) [3].

We can also compare the computational complexity of them. If we consider a binary data sequence with ($N = 2^n$) to operate, then we have [4] [3]:

$$\mathcal{C}_{DFT} = \mathcal{O}(N^2) = \mathcal{O}(2^{2n}) \tag{6}$$
$$\mathcal{C}_{FFT} = \mathcal{O}(N \log N) = \mathcal{O}(n2^n) \tag{7}$$
$$\mathcal{C}_{QFT} = \mathcal{O}(log^2 N) = \mathcal{O}(n^2) \tag{8}$$
$$\mathcal{C}_{QFFT} = \mathcal{O}(logN) = \mathcal{O}(n) \tag{9}$$

As a result, the complexity of QFFT is much more lower than QFT in the case of that n, numbers of bits, is huge. Where if we used QFFT will improved lots of performance and reduced the spending time.

In next section, we will show how to construct a QFFT gate in quantum circuits.

D. Bogoliubov Transformation

The Bogoliubov transformation is another useful mathematical technique which is used to diagonalizing Hamiltonian and we can obtain the stationary solution to the corresponding Schrödinger equation.

The Bogoliubov transformation is commonly applied to systems described by second-quantized operators, which are operators that represent the creation and annihilation of particles in a quantum field.

The transformation involves rewriting the original creation and annihilation operators in terms of new operators, known as Bogoliubov operators. The Bogoliubov operators can be thought of as a linear combination of the original operators. By choosing appropriate coefficients, the transformed Hamiltonian would become diagonal, which simplifies the analysis of the system.

The general form of the Bogoliubov transformation for a fermionic system can be expressed as:

$$\begin{aligned} c_j &= U_{j\mu} b_\mu + V^*_{j\mu} b^\dagger_\mu \\ c^\dagger_j &= U^*_{j\mu} b^\dagger_\mu + V_{j\mu} b_\mu \end{aligned} \tag{10}$$

where c_j and $c^\dagger_j$ are the annihilation and creation operators for the original fermionic particles, b_μ and $b^\dagger_\mu$ are the annihilation and creation operators for the Bogoliubov quasiparticles, $U_{j\mu}$ and $V_{j\mu}$ are complex coefficients.

The coefficients $U_{j\mu}$ and $V_{j\mu}$ are determined by the properties of the system and can be obtained by solving the Bogoliubov-de Gennes (BdG) equations. These equations are derived by substituting the Bogoliubov transformation into the original Hamiltonian and requiring the transformed Hamiltonian to be diagonal.

In next section, we will show how to apply the Bogoliubov transformation on TFIM and how we implemented on quantum circuits.

II. Method

A. Jordan-Wigner Transformation

This transformation maps the spin-raising and lowering operators into fermion creation and annihilation operators. For detail, raising and lowering operators $S^\pm$ work as

$$\begin{aligned} S^+|\downarrow\rangle &= |\uparrow\rangle \\ S^-|\uparrow\rangle &= |\downarrow\rangle \end{aligned} \tag{11}$$

and the creation and annihilation operators $c^\pm$ work as

$$\begin{aligned} c^+|0\rangle &= |1\rangle \\ c^-|1\rangle &= |0\rangle \end{aligned} \tag{12}$$

the state $|0\rangle$ means there is no fermion, and state $|1\rangle$ means there is a fermion. Now, let us see some properties of raising and lowering operators. First, in matrix form, the operators look like this (here we let $\hbar = 1$)

$$\begin{aligned} S^+ = S^x + iS^y = \frac{\sigma^x + i\sigma^y}{2} = \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix} \\ S^- = S^x - iS^y = \frac{\sigma^x - i\sigma^y}{2} = \begin{pmatrix} 0 & 0 \\ 1 & 0 \end{pmatrix} \end{aligned} \tag{13}$$

and the relation between $S^\pm$ is

$$\begin{aligned} \left[S^\pm, S^\pm\right] &= 0 \\ \{S^-, S^+\} &= 1 \end{aligned} \tag{14}$$

The whole story about Jordan-Wigner transformation is to find the relations about

$$\begin{aligned} S^+ &\to c^+_j = c^\dagger_j \\ S^- &\to c^-_j = c_j \end{aligned} \tag{15}$$

After the transformation, we require that c_j should obey the following rules. Suppose $|\Omega_c\rangle$ is the vacuum state which looks like $|00\cdots00\rangle$.

$$\{c_i, c_j\} = 0 \tag{16}$$
$$\{c_i, c_j^\dagger\} = \delta_{ij} \tag{17}$$
$$c_i|\Omega_c\rangle = 0 \tag{18}$$

Notice that $S^\pm$ are local operators but c_j, $c_j^\dagger$ are global operators, the antisymmetry property $\{c_i, c_j\} = 0$ is the most import difference between $S^\pm$ and c_j, $c_j^\dagger$. In order to satisfy this condition, the transformation is

$$c_j = e^{\left(-i\pi\Sigma_{k=1}^{j-1} S_k^+ S_k^-\right)} S_j^- \tag{19}$$
$$c_j^\dagger = S_j^+ e^{\left(+i\pi\Sigma_{k=1}^{j-1} S_k^+ S_k^-\right)} \tag{20}$$
$$c_j^\dagger c_j = S_j^+ S_j^- \tag{21}$$

and due to the following relation

$$S_j^z = S_j^+ S_j^- - \frac{1}{2}\mathbb{I} \tag{22}$$

the Jordan-Wigner transformation can be also written as

$$c_j = \left(\bigotimes_{l<j} -\sigma_l^z\right) \otimes S_j^- \otimes \left(\bigotimes_{j<l} \mathbb{I}_l\right) \tag{23}$$
$$c_j^\dagger = \left(\bigotimes_{l<j} -\sigma_l^z\right) \otimes S_j^+ \otimes \left(\bigotimes_{j<l} \mathbb{I}_l\right) \tag{24}$$

Note that the $S_j^\pm$ in equation(23) and equation(24) represents the local operator which only works on jth spin. Other $S_j^{\pm,x,y,z}$ or operators with subscript j are defaulted global operators, the most import difference is the size of the matrix. The inverse transformation is given by

$$S_j^- = e^{\left(+i\pi\Sigma_{k=1}^{j-1} S_k^+ S_k^-\right)} c_j \tag{25}$$
$$S_j^+ = c_j^\dagger e^{\left(-i\pi\Sigma_{k=1}^{j-1} S_k^+ S_k^-\right)} \tag{26}$$
$$S_j^z = c_j^\dagger c_j - \frac{1}{2}\mathbb{I} \tag{27}$$

Let's start with the simplest antiferromagnetic Ising Hamiltonian with transverse field

$$H = \sum_{i=1}^{n} \sigma_i^x \sigma_{i+1}^x + \lambda \sum_{i=1}^{n} \sigma_i^z \tag{28}$$

To simplify the difficulty of the problem, we require cyclic boundary conditions and stipulate that there can only be an even number of spins. In this way, after the Jordan—Wigner transformation, there is no need to consider the boundary terms in the Hamiltonian. That is $c_{n+1} = c_1$ and $c_{n+1}^\dagger = c_1^\dagger$. The first step of transformation is to use the "local $S_j^\pm$" to rewrite the Hamiltonian

$$\begin{aligned} H &= \sum_{j=1}^{n} (S_j^- + S_j^+)(S_{j+1}^- + S_{j+1}^+) + \lambda \sum_{j=1}^{n} 2S_j^+ S_j^- - \mathbb{I} \\ \Rightarrow H &= \sum_{j=1}^{n} (S_j^+ S_{j+1}^+ + S_j^+ S_{j+1}^- + S_j^- S_{j+1}^+ + S_j^- S_{j+1}^-) \\ &\quad + 2\lambda \sum_{j=1}^{n} S_j^+ S_j^- - \frac{1}{2}\mathbb{I} \end{aligned} \tag{29}$$

According to equations (25) $\sim$ (27), we can know that

$$\begin{aligned} S_j^+ S_{j+1}^+ &= c_j^\dagger e^{\left(-i\pi S_j^+ S_j^-\right)} c_{j+1}^\dagger = c_j^\dagger (1 - 2c_j^\dagger c_j) c_{j+1}^\dagger \\ \Rightarrow S_j^+ S_{j+1}^+ &= c_j^\dagger c_{j+1}^\dagger \end{aligned} \tag{30}$$

By analogy, we can know the following transformation relations

$$S_j^+ S_{j+1}^- = c_j^\dagger c_{j+1} \tag{31}$$
$$S_j^- S_{j+1}^+ = c_j (1 - 2(1 - c_j c_j^\dagger)) c_{j+1}^\dagger = c_{j+1}^\dagger c_j \tag{32}$$
$$S_j^- S_{j+1}^- = c_j (1 - 2(1 - c_j c_j^\dagger)) c_{j+1} = c_{j+1} c_j \tag{33}$$

Now the Hamiltonian reads

$$\begin{aligned} H_c &= \sum_{j=1}^{n} (c_j^\dagger c_{j+1}^\dagger + c_j^\dagger c_{j+1} + c_{j+1}^\dagger c_j + c_{j+1} c_j) \\ &\quad + 2\lambda \sum_{j=1}^{n} c_j^\dagger c_j - \frac{1}{2}\mathbb{I} \end{aligned} \tag{34}$$

Ignore the constant, the Hamiltonian becomes

$$\begin{aligned} H_c &= \sum_{j=1}^{n} (c_j^\dagger c_{j+1}^\dagger + c_j^\dagger c_{j+1} + c_{j+1}^\dagger c_j + c_{j+1} c_j) \\ &\quad + 2\lambda \sum_{j=1}^{n} c_j^\dagger c_j \end{aligned} \tag{35}$$

Finally, this transformation takes a state of spin-1/2 particles and turns it into a fermionic state. In terms of the wave function

$$\begin{aligned} |\psi\rangle &= \sum_{i_1,\cdots,i_n=0,1} \psi_{i_1\cdots i_n} |i_1 \cdots i_n\rangle \\ &= \sum_{i_1,\cdots,i_n=0,1} \psi_{i_1\cdots i_n} (c_1^\dagger)^{i_1} \cdots (c_n^\dagger)^{i_n} |0\cdots 0\rangle \end{aligned} \tag{36}$$

It can be found that the coefficients of the wave function have not changed after the transformation, so the Jordan—Wigner transformation itself does not require any quantum gates to operate. The only thing to pay attention to is the antisymmetric nature of the fermions, so it needs to be modified when executing the SWAP gate. We call this modified SWAP gate fermionic SWAP (fSWAP)

$$fSWAP = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & -1 \end{pmatrix} \tag{37}$$

The corresponding quantum circuit is in fig(1)

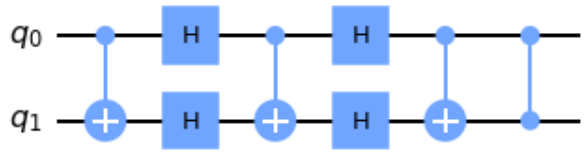

Fig. 1. fermionic SWAP gate in quantum circuit.

B. Fourier Transform

The next step to solving the Ising model consists of getting the fermionic modes to momentum space using the DFT

$$f_k^\dagger = \frac{1}{\sqrt{n}} \sum_{j=1}^{n} exp\left(i\frac{2\pi k}{n} j\right) c_j^\dagger \tag{38}$$

$$c_j^\dagger = \frac{1}{\sqrt{n}} \sum_{k} exp\left(-i\frac{2\pi j}{n} k\right) f_k^\dagger \tag{39}$$

Because the boundary condition we use is $c_{n+1}^\dagger = c_1^\dagger$, hence the standard choice for the k is

$$k = 1 - \frac{n}{2}, \cdots, 0, \cdots \frac{n}{2} \tag{40}$$

and due to the boundary condition, we have the following properties

$$\begin{aligned} f_k^\dagger &= \frac{1}{\sqrt{n}} \sum_{j+1=1}^{j+1=n} exp\left(i\frac{2\pi k}{n}(j+1)\right) c_{j+1}^\dagger \\ \Rightarrow f_k^\dagger &= \frac{1}{\sqrt{n}} \sum_{j=0}^{n-1} exp\left(i\frac{2\pi k}{n} j\right) exp\left(i\frac{2\pi k}{n}\right) c_{j+1}^\dagger \\ \Rightarrow f_k^\dagger &= e^{i\frac{2\pi k}{n}} c_1^\dagger + \frac{1}{\sqrt{n}} \sum_{j=1}^{n} e^{i\frac{2\pi k}{n} j} e^{i\frac{2\pi k}{n}} c_{j+1}^\dagger - e^{i\frac{2\pi k}{n}} c_{n+1}^\dagger \\ \Rightarrow f_k^\dagger &= \frac{1}{\sqrt{n}} \sum_{j=1}^{n} e^{i\frac{2\pi k}{n} j} e^{i\frac{2\pi k}{n}} c_{j+1}^\dagger \\ \Rightarrow e^{-i\frac{2\pi k}{n}} f_k^\dagger &= \frac{1}{\sqrt{n}} \sum_{j=1}^{n} e^{i\frac{2\pi k}{n} j} c_{j+1}^\dagger \end{aligned} \tag{41}$$

and hence

$$c_{j+1}^\dagger = \frac{1}{\sqrt{n}} \sum_{k=1-n/2}^{n/2} e^{-i\frac{2\pi k}{n} j} e^{-i\frac{2\pi k}{n}} f_k^\dagger \tag{42}$$

The DFT work as

$$\begin{aligned} &\sum_{j=1}^{n} (c_j^\dagger c_{j+1}^\dagger + c_j^\dagger c_{j+1} + c_{j+1}^\dagger c_j + c_{j+1} c_j) + 2\lambda c_j^\dagger c_j \\ &= \sum_{j=1}^{n} (c_j^\dagger c_{j+1}^\dagger + c_j^\dagger c_{j+1} + H.c.) + 2\lambda c_j^\dagger c_j \\ &\Rightarrow \frac{1}{n} \sum_{j=1}^{n} \sum_{k} e^{-i\frac{2\pi j}{n} k} f_k^\dagger \sum_{k'} e^{-i\frac{2\pi k'}{n} j} e^{-i\frac{2\pi k'}{n}} f_{k'}^\dagger \\ &\quad + \sum_{k} e^{-i\frac{2\pi j}{n} k} f_k^\dagger \sum_{k'} e^{i\frac{2\pi k'}{n} j} e^{i\frac{2\pi k'}{n}} f_{k'} + H.c. \\ &\quad + 2\lambda \sum_{k} e^{-i\frac{2\pi j}{n} k} f_k^\dagger \sum_{k'} e^{i\frac{2\pi k'}{n} j} f_{k'} \end{aligned} \tag{43}$$

use the definition of delta function

$$\delta_{(k,k')} = \frac{1}{n} \sum_{j=1}^{n} e^{-i\frac{2\pi j}{n}(k-k')} \tag{44}$$

hence, we can rewrite equation(43) as

$$\begin{aligned} &\sum_{k} \sum_{k'} \delta_{(k,-k')} e^{-i\frac{2\pi k'}{n}} f_k^\dagger f_{k'}^\dagger + \delta_{(k,k')} e^{i\frac{2\pi k'}{n}} f_k^\dagger f_{k'} \\ &\quad + \delta_{(k,k')} e^{-i\frac{2\pi k'}{n}} f_k^\dagger f_{k'} + \delta_{(-k,k')} e^{i\frac{2\pi k}{n}} f_k f_{k'} \\ &\quad + 2\lambda \sum_{k} \sum_{k'} \delta_{(k,k')} f_k^\dagger f_{k'} \\ &\Rightarrow \sum_{k} e^{i\frac{2\pi k}{n}} f_k^\dagger f_{-k}^\dagger + e^{i\frac{2\pi k}{n}} f_k^\dagger f_k + e^{-i\frac{2\pi k}{n}} f_k^\dagger f_k \\ &\quad + \sum_{k} e^{i\frac{2\pi k}{n}} f_k f_{-k} + 2\lambda \sum_{k} f_k^\dagger f_k \end{aligned} \tag{45}$$

Note that in the equation (45) we are summing over index k', but it is also equivalent to summing over index k. So a little trick can be played where one part sums over index k and one part sums over index k', that is

$$\begin{aligned} &\sum_{k'} \frac{1}{2} \left(\sum_{k} \delta_{(k,-k')} e^{-i\frac{2\pi k'}{n}} f_k^\dagger f_{k'}^\dagger + \delta_{(-k,k')} e^{i\frac{2\pi k}{n}} f_k f_{k'} \right) \\ &+ \sum_{k} \frac{1}{2} \left(\sum_{k'} \delta_{(k,-k')} e^{-i\frac{2\pi k'}{n}} f_k^\dagger f_{k'}^\dagger + \delta_{(-k,k')} e^{i\frac{2\pi k}{n}} f_k f_{k'} \right) \\ &\quad + e^{i\frac{2\pi k}{n}} f_k^\dagger f_k + e^{-i\frac{2\pi k}{n}} f_k^\dagger f_k + 2\lambda f_k^\dagger f_k \end{aligned} \tag{46}$$

then we get

$$\begin{aligned} &\sum_{k'} \frac{1}{2} e^{-i\frac{2\pi k'}{n}} \left(f_{-k'}^\dagger f_{k'}^\dagger + f_{-k'} f_{k'} \right) \\ &+ \sum_{k} \frac{1}{2} e^{i\frac{2\pi k}{n}} \left(f_k^\dagger f_{-k}^\dagger + f_k f_{-k} \right) \\ &\quad + e^{i\frac{2\pi k}{n}} f_k^\dagger f_k + e^{-i\frac{2\pi k}{n}} f_k^\dagger f_k + 2\lambda f_k^\dagger f_k \end{aligned} \tag{47}$$

Using k=k' and considering the antisymmetry of fermions, we finally get the Hamiltonian in momentum space

$$H_p = \sum_k 2\left(\lambda + \cos\left(\frac{2\pi k}{n}\right)\right) f_k^\dagger f_k + i\sin\left(\frac{2\pi k}{n}\right)\left(f_k^\dagger f_{-k}^\dagger + f_k f_{-k}\right) \tag{48}$$

Now, it's time to design our quantum gates to perform the DFT. As mentioned earlier, to simplify the boundary conditions of the model, it is stipulated that there can only be an even number of spins. Here we will continue to use this feature to simplify the DFT. First divide all spins into two parts with odd and even labels, and due to the convention, we shift the range of label j from [1, n] to [0, n-1]

$$\begin{aligned}&\frac{1}{\sqrt{n}}\sum_{j=0}^{n-1} exp\left(i\frac{2\pi k}{n}j\right)c_j^\dagger \\ =&\frac{1}{\sqrt{n}}\left(\sum_{j=0}^{n-1} e^{i\frac{2\pi k}{n}even}c_{even}^\dagger + e^{i\frac{2\pi k}{n}odd}c_{odd}^\dagger\right) \\ =&\frac{1}{\sqrt{2}\sqrt{n/2}}\left(\sum_{j=0}^{n/2-1} e^{i\frac{2\pi k}{n}(2j)}c_{2j}^\dagger + e^{i\frac{2\pi k}{n}(2j+1)}c_{2j+1}^\dagger\right)\end{aligned} \tag{49}$$

Then we can define the $f_{k(even)}^\dagger$ and $f_{k(odd)}^\dagger$ as

$$f_{k(even)}^\dagger = \frac{1}{\sqrt{n/2}}\sum_{j=0}^{n/2-1} exp\left(i\frac{2\pi k}{n}(2j)\right)c_{2j}^\dagger \tag{50}$$

$$f_{k(odd)}^\dagger = \frac{1}{\sqrt{n/2}}\sum_{j=0}^{n/2-1} exp\left(i\frac{2\pi k}{n}(2j)\right)c_{2j+1}^\dagger \tag{51}$$

so the QFT can be split to two parts

$$f_k^\dagger = \frac{1}{\sqrt{2}}\left(f_{k(even)}^\dagger + exp\left(i\frac{2\pi k}{n}\right)f_{k(odd)}^\dagger\right) \tag{52}$$

and the annihilation operator follows the same rules

$$f_k = \frac{1}{\sqrt{2}}\left(f_{k(even)} + exp\left(-i\frac{2\pi k}{n}\right)f_{k(odd)}\right) \tag{53}$$

What we have done is like the famous fast Fourier Transform(FFT), the spirit of which is to split the DFT matrix into smaller matrices. (Because we are using quantum computers to perform this task, it is more appropriate to call it QFFT.) Take n=4 for example, the DFT matrix can be written as

$$\begin{pmatrix} f_{-1}^\dagger \\ f_0^\dagger \\ f_1^\dagger \\ f_2^\dagger \end{pmatrix} = \frac{1}{\sqrt{4}}\begin{pmatrix} \omega^0 & \omega^{-1} & \omega^{-2} & \omega^{-3} \\ \omega^0 & \omega^0 & \omega^0 & \omega^0 \\ \omega^0 & \omega^1 & \omega^2 & \omega^3 \\ \omega^0 & \omega^2 & \omega^4 & \omega^6 \end{pmatrix}\begin{pmatrix} c_0^\dagger \\ c_1^\dagger \\ c_2^\dagger \\ c_3^\dagger \end{pmatrix} \tag{54}$$

where now ω is

$$\omega = exp\left(i\frac{2\pi k}{4}\right) = i \tag{55}$$

so the Fourier matrix becomes

$$\frac{1}{2}\begin{pmatrix} 1 & -i & -1 & i \\ 1 & 1 & 1 & 1 \\ 1 & i & -1 & -i \\ 1 & -1 & 1 & -1 \end{pmatrix}\begin{pmatrix} c_0^\dagger \\ c_1^\dagger \\ c_2^\dagger \\ c_3^\dagger \end{pmatrix} \tag{56}$$

Next we separate $c_k^\dagger$ into odd and even part, then swap the second and third columns of the matrix to keep the result unchanged.

$$\frac{1}{2}\begin{pmatrix} 1 & -1 & -i & i \\ 1 & 1 & 1 & 1 \\ 1 & -1 & i & -i \\ 1 & 1 & -1 & -1 \end{pmatrix}\begin{pmatrix} c_0^\dagger \\ c_2^\dagger \\ c_1^\dagger \\ c_3^\dagger \end{pmatrix} \tag{57}$$

Since the DFT projects $c_k^\dagger$ into the column space of the Fourier matrix, the results will not be affected after the rows of the Fourier matrix are rotated. Thus, we can find

$$\frac{1}{2}\left(\begin{array}{cc|cc} 1 & 1 & -1 & -1 \\ 1 & -1 & -i & i \\ \hline 1 & 1 & 1 & 1 \\ 1 & -1 & i & -i \end{array}\right)\begin{pmatrix} c_0^\dagger \\ c_2^\dagger \\ c_1^\dagger \\ c_3^\dagger \end{pmatrix} \tag{58}$$

Note the Hadamard matrix $\mathbb{H}$ hidden in it, and further rewrite the matrix as

$$\begin{aligned}&\frac{1}{\sqrt{2}}\frac{1}{\sqrt{2}}\left(\begin{array}{cc|cc} \begin{pmatrix}1&1\\1&-1\end{pmatrix} & & \begin{pmatrix}1&0\\0&i\end{pmatrix}\begin{pmatrix}1&1\\1&-1\end{pmatrix} & \\ \hline \begin{pmatrix}1&1\\1&-1\end{pmatrix} & & \begin{pmatrix}-1&0\\0&-i\end{pmatrix}\begin{pmatrix}1&1\\1&-1\end{pmatrix} & \end{array}\right) \\ \Rightarrow &\frac{1}{\sqrt{2}}\begin{pmatrix} \begin{pmatrix}1&0\\0&1\end{pmatrix} & \begin{pmatrix}1&0\\0&i\end{pmatrix} \\ \begin{pmatrix}1&0\\0&1\end{pmatrix} & \begin{pmatrix}-1&0\\0&-i\end{pmatrix} \end{pmatrix}\begin{pmatrix} \mathbb{H} & 0 \\ 0 & \mathbb{H} \end{pmatrix}\end{aligned} \tag{59}$$

When the matrix is applied to $c_k^\dagger$, it becomes

$$\frac{1}{\sqrt{2}}\begin{pmatrix} \begin{pmatrix}1&0\\0&1\end{pmatrix} & \begin{pmatrix}1&0\\0&i\end{pmatrix} \\ \begin{pmatrix}1&0\\0&1\end{pmatrix} & \begin{pmatrix}-1&0\\0&-i\end{pmatrix} \end{pmatrix}\begin{pmatrix} \mathbb{H}\begin{pmatrix}c_0^\dagger\\c_2^\dagger\end{pmatrix} \\ \mathbb{H}\begin{pmatrix}c_1^\dagger\\c_3^\dagger\end{pmatrix} \end{pmatrix} \tag{60}$$

Let us focus on the effect of $\mathbb{H}$ on even part of $c_k^\dagger$

$$\begin{aligned}&\mathbb{H}\begin{pmatrix}c_0^\dagger\\c_2^\dagger\end{pmatrix} = \frac{1}{\sqrt{2}}\begin{pmatrix}1&1\\1&-1\end{pmatrix}\begin{pmatrix}c_0^\dagger\\c_2^\dagger\end{pmatrix} \\ \Rightarrow &\begin{cases}\frac{1}{\sqrt{2}}c_0^\dagger + \frac{1}{\sqrt{2}}c_2^\dagger \\ \frac{1}{\sqrt{2}}c_0^\dagger - \frac{1}{\sqrt{2}}c_2^\dagger\end{cases}\end{aligned} \tag{61}$$

According to what we mentioned earlier about the Jordan-Wigner transformation, now we change the creation and annihilation operators back to the basis that the computer can understand (spin-up or spin-down basis, and expressed in binary).

$$\begin{cases} \frac{1}{\sqrt{2}}c_0^\dagger|00\rangle + \frac{1}{\sqrt{2}}c_2^\dagger|00\rangle \\ \frac{1}{\sqrt{2}}c_0^\dagger|00\rangle - \frac{1}{\sqrt{2}}c_2^\dagger|00\rangle \end{cases} \Rightarrow \begin{cases} \frac{1}{\sqrt{2}}|10\rangle + \frac{1}{\sqrt{2}}|01\rangle \\ \frac{1}{\sqrt{2}}|10\rangle - \frac{1}{\sqrt{2}}|01\rangle \end{cases} \tag{62}$$

where $|01\rangle = [0100]^T$, $|10\rangle = [0010]^T$. Thus, when Hadamard matrix $\mathbb{H}$ acts on the binary basis, the form of the matrix is

$$\begin{aligned} F_0^\dagger \equiv \mathbb{H}_2 &= \begin{cases} |00\rangle\langle 00| \\ \frac{1}{\sqrt{2}}|10\rangle\langle 10| + \frac{1}{\sqrt{2}}|01\rangle\langle 10| \\ \frac{1}{\sqrt{2}}|10\rangle\langle 01| - \frac{1}{\sqrt{2}}|01\rangle\langle 01| \\ -|11\rangle\langle 11| \end{cases} \\ &= \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \frac{-1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & 0 \\ 0 & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & 0 \\ 0 & 0 & 0 & -1 \end{pmatrix} \end{aligned} \tag{63}$$

where the fermionic anticommutation relation has been taken into account in the -1 element of the $F_0^\dagger$ matrix. We seem to be missing a step here because $|1\rangle = [1,0]^T$ in mathematical calculations is the opposite of the representation of qubits $|0\rangle = [1,0]^T$. But since we are considering 1D TFIM with cyclic boundary conditions, the symmetry in our model allows us to ignore this issue. Specifically, if the above problems are considered, the position of the negative sign in equation(63) will be changed from (1,1) to (2,2). You can experiment with the F_0 gate we provided in the appendix, just swap the order of the input qubits and you will get the effect that the negative sign exchanging between the (1,1) and the (2,2) position. The reason why this does not affect the simulation of our system is that the label of the spin is artificially determined. Due to the cyclic boundary conditions, the spin chain with the original label order (0, 1, 2, 3) is completely equal to (2, 3, 0, 1), thus, the two positions in the middle are exchanged for FFT to obtain (0, 2, 1, 3) and (2, 0, 3, 1) respectively. All in all, although the mathematical expression of the spin quantum state is different from the mathematical expression of the qubit, the symmetry in our system can be used to compensate for the difference between the two.

Now, note that what we actually need on the quantum circuit is the inverse operation of $F_0^\dagger$, the details will be discussed in later "Disentangle Gate" section. The result after realizing F_0 with quantum circuits is in fig(2). After continuing to decompose the Fourier matrix in the same way, we can get another matrix $F_1^\dagger$, but as mentioned above, what we really need in the quantum circuit is the inverse QFFT, so let us focus on F_1

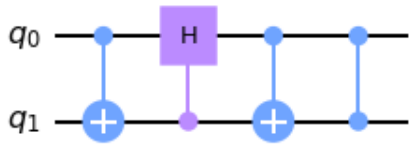

Fig. 2. Inverse gate of $F_0^\dagger$, that is F_0

$$F_1 \equiv \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \frac{i}{\sqrt{2}} & \frac{-i}{\sqrt{2}} & 0 \\ 0 & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & 0 \\ 0 & 0 & 0 & i \end{pmatrix} \tag{64}$$

The result after realizing F_1 with quantum circuits is in fig(3)

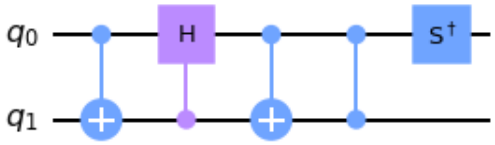

Fig. 3. F_1 gate, which is inverse gate of $F_1^\dagger$.

In the example of performing n=4, the sub-matrix required by the QFFT can be completed only by $F_0^\dagger$, $F_1^\dagger$ and fSWAP gates. The following fig(4) is a schematic diagram of the complete QFFT on a quantum circuit.

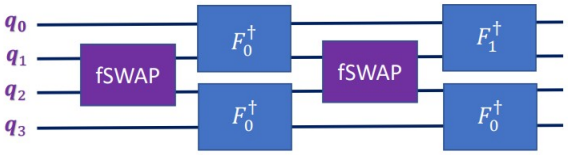

Fig. 4. QFFT represented by quantum circuit schematic.

Again, what we actually need is the inverse transformation in fig(5)

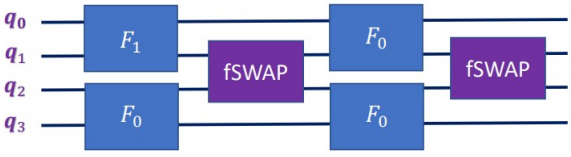

Fig. 5. QFFT represented by quantum circuit schematic.

The inverse QFFT actually achieved by quantum gates is in fig(6)

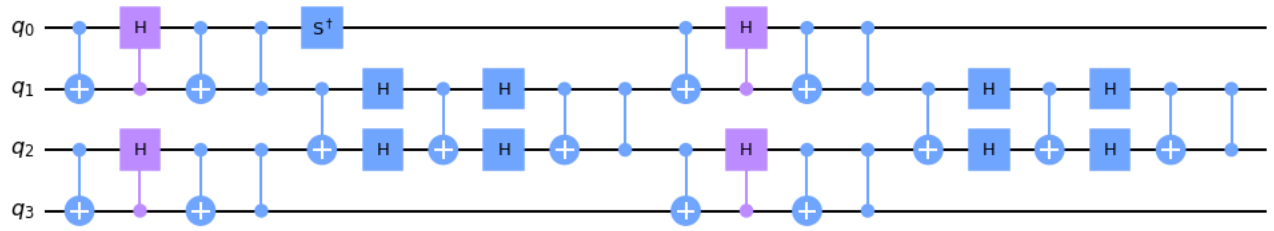

Fig. 6. Inverse QFFT represented by quantum gates.

C. Bogoliubov Transformation

To diagonalize the Hamiltonians, we need to use the Bogoliubov transformation. We now define a fermionic two-component spinor.

$$\Psi_k = \begin{pmatrix} f_k \\ f_{-k}^\dagger \end{pmatrix}$$
$$\Psi_k^\dagger = (f_k^\dagger, f_{-k}) \tag{65}$$

with anti-commutation relations($\alpha = 1, 2$stands for the two components of ψ_k)

$$\{\Psi_{k\alpha}, \Psi_{k'\alpha'}^\dagger\} = \delta_{\alpha,\alpha'}\delta_{k,k'} \tag{66}$$

Take advantage of the anti-commutation relations, then we have following properties

$$\sum_k 2\cos\left(\frac{2\pi k}{n}\right) f_k^\dagger f_k$$
$$= \sum_k \cos\left(\frac{2\pi k}{n}\right)\left(f_k^\dagger f_k - f_{-k} f_{-k}^\dagger\right) \tag{67}$$
$$\sum_k \left(2 f_k^\dagger f_k - 1\right)$$
$$= \sum_k \left(f_k^\dagger f_k - f_{-k} f_{-k}^\dagger\right) \tag{68}$$

Thus, Hamiltonian in momentum space (equation(48)) can be written as

$$H_p = \sum_k \left(\lambda + \cos\left(\frac{2\pi k}{n}\right)\right)\left(f_k^\dagger f_k - f_{-k} f_{-k}^\dagger\right) + i\sin\left(\frac{2\pi k}{n}\right)\left(f_k^\dagger f_{-k}^\dagger - f_{-k} f_k\right) \tag{69}$$

in matrix form, Hamiltonian can be written as

$$H_p = \sum_k \Psi_{k\alpha}^\dagger H_{p\alpha\alpha'} \Psi_{k\alpha'}$$
$$= \sum_k \Psi_k^\dagger \begin{pmatrix} \left(\lambda + \cos\left(\frac{2\pi k}{n}\right)\right) & i\sin\left(\frac{2\pi k}{n}\right) \\ -i\sin\left(\frac{2\pi k}{n}\right) & -\left(\lambda + \cos\left(\frac{2\pi k}{n}\right)\right) \end{pmatrix} \Psi_k \tag{70}$$

By solving the 2×2 eigenvalue problem for the Hamiltonian we find the eigenvalues $\omega_{k\pm} = \pm\omega_k$

$$\omega_k = \sqrt{\left(\lambda + \cos\left(\frac{2\pi k}{n}\right)\right)^2 + \sin^2\left(\frac{2\pi k}{n}\right)} \tag{71}$$

The diagonalized Hamiltonian can be written as

$$H_b = \sum_k \Psi_k^\dagger U \begin{pmatrix} \omega_k & 0 \\ 0 & -\omega_k \end{pmatrix} U^\dagger \Psi_k$$
$$= \sum_k \Psi_k^\dagger \begin{pmatrix} u_k & -v_k^* \\ v_k & u_k^* \end{pmatrix} \begin{pmatrix} \omega_k & 0 \\ 0 & -\omega_k \end{pmatrix} \begin{pmatrix} u_k^* & v_k^* \\ -v_k & u_k \end{pmatrix} \Psi_k \tag{72}$$

After comparing and calculating with the original matrix, it can be found that

$$|u_k|^2 = \frac{1+\phi}{2} \tag{73}$$
$$|v_k|^2 = \frac{1-\phi}{2} \tag{74}$$

where ϕ is

$$\phi = \frac{\left(\lambda + \cos\left(\frac{2\pi k}{n}\right)\right)}{\sqrt{\left(\lambda + \cos\left(\frac{2\pi k}{n}\right)\right)^2 + \sin^2\left(\frac{2\pi k}{n}\right)}} \tag{75}$$

Now we let

$$u_k = \cos\left(\frac{\theta_k}{2}\right) \quad , \quad v_k = -i\sin\left(\frac{\theta_k}{2}\right) \tag{76}$$

where θ_k defined as

$$\theta_k = \arccos\left(\frac{\left(\lambda + \cos\left(\frac{2\pi k}{n}\right)\right)}{\sqrt{\left(\lambda + \cos\left(\frac{2\pi k}{n}\right)\right)^2 + \sin^2\left(\frac{2\pi k}{n}\right)}}\right) \tag{77}$$

Thus, the Bogoliubov transformation is

$$\begin{pmatrix} b_k \\ b_{-k}^\dagger \end{pmatrix} = \begin{pmatrix} \cos\left(\frac{\theta_k}{2}\right) & i\sin\left(\frac{\theta_k}{2}\right) \\ i\sin\left(\frac{\theta_k}{2}\right) & \cos\left(\frac{\theta_k}{2}\right) \end{pmatrix} \begin{pmatrix} f_k \\ f_{-k}^\dagger \end{pmatrix} \tag{78}$$

The final diagonalized Hamiltonian is

$$H_b = \sum_k \omega_k b_k^\dagger b_k - \omega_k b_{-k} b_{-k}^\dagger \tag{79}$$
$$= \sum_k \omega_k (b_k^\dagger b_k - b_{-k} b_{-k}^\dagger)$$
$$= \sum_k \omega_k (b_k^\dagger b_k + b_{-k}^\dagger b_{-k} - 1)$$
$$= \sum_k 2\omega_k (b_k^\dagger b_k - \frac{1}{2}) \tag{80}$$

Now, take n = 4 for example, the ground state (Bogoliubov vacuum) energy can be easily calculated

$$\sum_{k=-1}^{2} -\omega_k$$
$$= \sum_{k=-1}^{2} -\sqrt{\left(\lambda + \cos\left(\frac{2\pi k}{4}\right)\right)^2 + \sin^2\left(\frac{2\pi k}{4}\right)}$$
$$= -\left(\sqrt{\lambda^2+1} + \sqrt{(\lambda+1)^2} + \sqrt{\lambda^2+1} + \sqrt{(\lambda-1)^2}\right)$$
$$= -\left(2\sqrt{\lambda^2+1} + |\lambda+1| + |\lambda-1|\right) \tag{81}$$

As we projected the FFT submatrix onto the spin-up or spin-down binary basis in equation(63), after the projection, the Bogoliubov transformation matrix in equation(78) becomes

$$B_k^n \equiv \begin{pmatrix} \cos\left(\frac{\theta_k}{2}\right) & 0 & 0 & i\sin\left(\frac{\theta_k}{2}\right) \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ i\sin\left(\frac{\theta_k}{2}\right) & 0 & 0 & \cos\left(\frac{\theta_k}{2}\right) \end{pmatrix} \tag{82}$$

Remember the boundary condition and the range of k in equation(40). In n=4 example, the k are just -1, 0, 1, and 2. Thus, the sum of the phase in momentum space is zero

$$\sum_{k=-1}^{2} e^{i\frac{\pi}{2}k} = i^{-1} + i^0 + i^1 + i^2 = 0 \tag{83}$$

So k=1 and k=-1 are opposite momentum modes, k=0 and k=2 are also opposite momentum modes. Because what Bogoliubov transformation has done is to decouple the modes with opposite momentum, therefore, we only need to use B_0^4 and B_1^4 to complete the transformation. The θ_k now is equal to

$$\theta_k = \arccos\left(\frac{\left(\lambda + \cos\left(\frac{\pi k}{2}\right)\right)}{\sqrt{\left(\lambda + \cos\left(\frac{\pi k}{2}\right)\right)^2 + \sin^2\left(\frac{\pi k}{2}\right)}}\right) \tag{84}$$

and we get θ_0 and θ_1 are (consider $\lambda \geq 1$)

$$\theta_0 = \arccos\left(\frac{(\lambda+1)}{\sqrt{(\lambda+1)^2}}\right) = 0 \tag{85}$$
$$\theta_1 = \arccos\left(\frac{\lambda}{\sqrt{\lambda^2+1}}\right) \tag{86}$$

so the B_0^4 and B_1^4 are

$$B_0^4 = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{87}$$
$$B_1^4 = \begin{pmatrix} \cos\left(\frac{\theta_1}{2}\right) & 0 & 0 & i\sin\left(\frac{\theta_1}{2}\right) \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ i\sin\left(\frac{\theta_1}{2}\right) & 0 & 0 & \cos\left(\frac{\theta_1}{2}\right) \end{pmatrix} \tag{88}$$

It can be found that B_0^4 actually does nothing, so we only need $B_1^4 = B_1$ to perform Bogoliubov transformation. Actually, the modes with k=0 and k=2 can be dropped in the thermodynamic limit. Again, what we need in quantum circuits is the inverse transform

$$U_{Bog} = B_1^\dagger = \begin{pmatrix} \cos\left(\frac{\theta_1}{2}\right) & 0 & 0 & -i\sin\left(\frac{\theta_1}{2}\right) \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -i\sin\left(\frac{\theta_1}{2}\right) & 0 & 0 & \cos\left(\frac{\theta_1}{2}\right) \end{pmatrix} \tag{89}$$

The fig(7) shows the quantum gates needed to construct B_1.

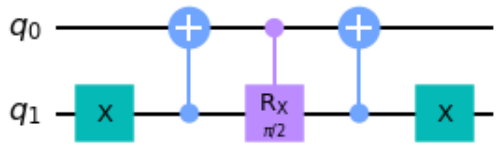

Fig. 7. The inverse Bogoliubov transform gate, here we choose $\lambda = 0$ so $\theta_1 = \pi/2$

D. Disentangle Gate

We went through three transformations to diagonalize the Hamiltonian, the Jordan-Wigner transform $U_{JW}^\dagger$, the QFFT $U_{QFFT}^\dagger$, and the Bogoliubov transform $U_{Bog}^\dagger$.

$$H = U_{JW} H_c U_{JW}^\dagger$$
$$= U_{JW} U_{QFFT} H_p U_{QFFT}^\dagger U_{JW}^\dagger$$
$$= U_{JW} U_{QFFT} U_{Bog} H_b U_{Bog}^\dagger U_{QFFT}^\dagger U_{JW}^\dagger \tag{90}$$

and we let U_{dis} as

$$U_{dis} = U_{JW} U_{QFFT} U_{Bog} \tag{91}$$

Note that U_{JW} doesn't actually do anything as we mentioned earlier. Now, we can solve the problem on H_b basis.

$$\langle\psi_b|H_b|\psi_b\rangle = \langle\psi_b|U_{dis}^\dagger H U_{dis}|\psi_b\rangle \tag{92}$$

The eigenstates of H_b, $|\psi_b\rangle$, are separable states, so they are very easy to prepare in a quantum computer. And by

applying the U_{dis} gate on the $|\psi_b\rangle$ state, we can get the Ising energy spectrum ω_i through measurement.

$$H_b|\psi_b\rangle = HU_{dis}|\psi_b\rangle = \omega_i|U_{dis}\psi_b\rangle \tag{93}$$

The order of execution in a quantum circuit is

$$|initial\rangle \rightarrow U_{Bog} \rightarrow U_{QFFT} \rightarrow measurement \tag{94}$$

and U_{dis} achieved by quantum gates can be seen in fig(8)

E. Phase Transition

A major feature of the one-dimensional quantum Ising model is the existence of quantum phase transitions. From the Hamiltonian in equation(28), we can simply see that after we calculate the expected value of energy, we only need to differentiate the expected value of energy to λ and divide it by the total number of spins to get the expected value of magnetization.

$$\langle\sigma_z\rangle = \frac{1}{n}\frac{\partial\langle H\rangle}{\partial\lambda} \tag{95}$$

Therefore, according to the energy of the ground state calculated by us in equation(81), if we ignore the negative sign and differentiate the energy of the ground state to λ, the magnetization can be obtained as

$$\langle\sigma_z\rangle = \frac{1}{4}\left(\frac{2\lambda}{\sqrt{\lambda^2+1}} + \frac{d}{d\lambda}|\lambda+1| + \frac{d}{d\lambda}|\lambda-1|\right) \tag{96}$$

We can easily find that there is a singular point when $\lambda = \pm 1$, that is, a phase transition point. Unfortunately, our current method cannot allow quantum circuits to simulate the phase transition point. Although we have related methods, however, consider the space, we will ignore the steps of simulating the phase transition point and directly tell the position of the phase transition point to the quantum circuit. We adjust the ground state in the H_b basis (diagonal basis) depends on λ value (here we only consider the condition that $\lambda > 0$)

$$|ground\rangle = \begin{cases} |0001\rangle & for \quad \lambda < 1 \\ |0000\rangle & for \quad \lambda > 1 \end{cases} \tag{97}$$

Finally, the whole quantum circuit which maps qubits to a quantum Ising model is shown in fig(9)

F. time evolution

Schrödinger equation describes the time evolution of a wave function

$$i\hbar\frac{d}{dt}|\psi(t)\rangle = H|\psi(t)\rangle \tag{98}$$

If the Hamiltonian is not time-dependent, the wave function can be written as

$$|\psi(t)\rangle = exp\left(-\frac{i}{\hbar}Ht\right)|\psi(0)\rangle \tag{99}$$

And we can use the energy basis to expand the propagator(here we still let $\hbar = 1$)

$$exp\left(-\frac{i}{\hbar}Ht\right) = \sum_i e^{-i\omega_i t}|E_i\rangle\langle E_i| \tag{100}$$

In Heisenberg picture, the time evolution of a observable physical quantity $\mathbb{O}$ can be written as

$$i\hbar\frac{d}{dt}\mathbb{O} = [\mathbb{O}, H] \tag{101}$$

and it's clear that $[\sigma_z, H] \neq 0$, so we can observe the evolution of $\langle\sigma_z\rangle$ over time. First, we calculate the time evolution on H_b basis

$$|\psi_b(t)\rangle = \sum_i e^{-i\omega_i t}|\psi_{bi}(0)\rangle \tag{102}$$

then we implement the U_{dis} gate to map the state to the Ising basis

$$|\psi(t)\rangle = U_{dis}|\psi_b(t)\rangle = \sum_i e^{-i\omega_i t}U_{dis}|\psi_{bi}(0)\rangle \tag{103}$$

It can be found that in equation(102), since each state is an eigenstate of H_b, so if we directly measure the quantum state, we will only get results that will not change with time. But in equation(103), since the eigenstates in the H_b basis are not the eigenstates in the original H, thus, after the operation of U_{dis}, we can observe the time-dependent results after the measurement.

Now, let's see the n = 4 example, we choose to observe the time evolution of the state of all spins align in the positive direction of σ_z. This is not an eigenstate of Ising Hamiltonian, since "all spins aligned" is a degenerate state (all spins up or all spins down). First, we need to know what is the corresponding state in the H_b basis (diagonal basis). Note that the convention in quantum computers is to treat spin up $[1, 0]^T$ state as $|0\rangle$ state, thus, the state corresponding to $|\uparrow\uparrow\uparrow\uparrow\rangle$ is $|0000\rangle$ state. Now that we have a complete circuit that can obtain the exact solution of the quantum Ising model, we can use this circuit to find the representation of the $|0000\rangle$ state in the H_b basis. Therefore, we need to take $|0000\rangle$ state as input and pass-through $U_{dis}^\dagger$, and finally measure and observe which states the wave function collapses to. The complete circuit of $U_{dis}^\dagger$ is in fig(12). The result of wave function collapse is $|0000\rangle$ state and $|1100\rangle$ state(see fig(11))

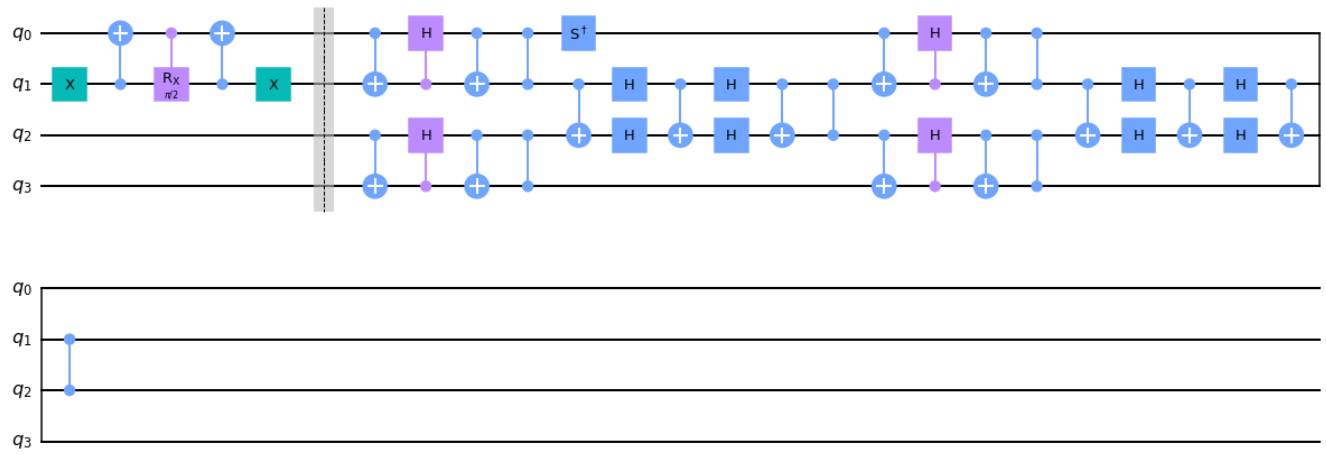

Fig. 8. Execute U_{dis} gates with the quantum circuit. From left to right are U_{Bog} gates and U_{QFFT} gates, which are divided by the barrier. Here we choose the $\lambda = 0$ case.

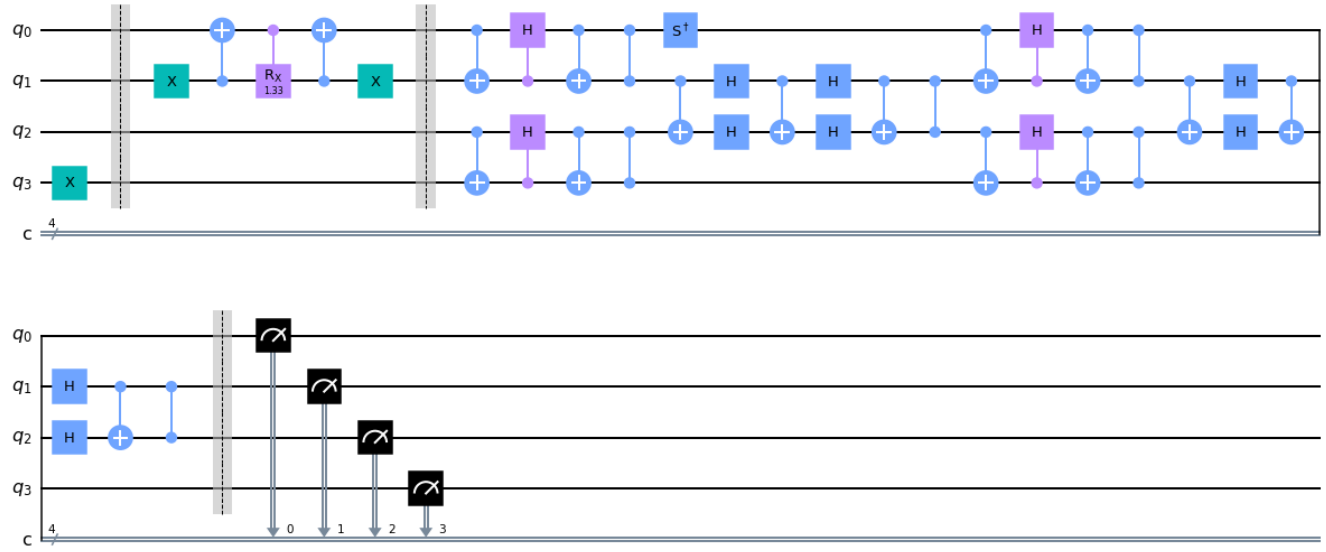

Fig. 9. The complete quantum circuit that maps qubits to a quantum Ising model.

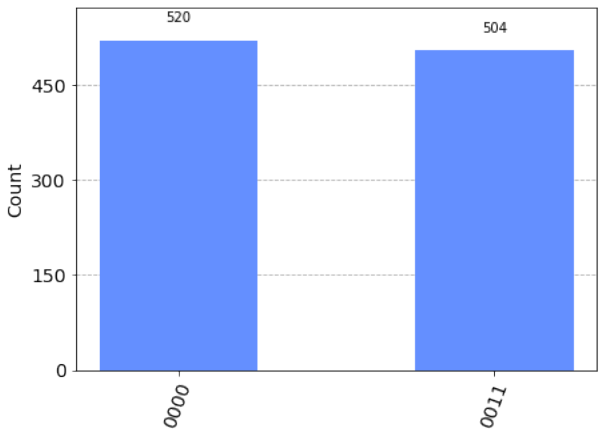

Fig. 11. Measurement result of $|0000\rangle$ state as input and pass-through $U_{dis}^{\dagger}$. We did 1024 samples, and we can see that the number of two state stations is almost the same. This is because we use $\lambda = 0$, so θ_1 can be known from the equation (86) It is equal to half of π, and the two coefficients of Bogoliubov transformation are $\cos(\pi/4)$ and $i\sin(\pi/4)$ respectively.

With the method shown in fig(11)), we can know that the $|0000\rangle$ state represented in H_b basis can be written as

$$|\psi_b(0)\rangle = \cos\left(\frac{\theta_1}{2}\right)|0000\rangle + i\sin\left(\frac{\theta_1}{2}\right)|1100\rangle \tag{104}$$

For simplicity, we let $\theta_1/2 = \phi$. With equation(79) and equation(102), we can write down the $|\psi_b(t)\rangle$ as

$$\begin{aligned}|\psi_b(t)\rangle =& e^{-i2\sqrt{1+\lambda^2}t}\cos(\phi)|0000\rangle \\ &+ e^{i2\sqrt{1+\lambda^2}t} i\sin(\phi)|1100\rangle\end{aligned} \tag{105}$$

Since the global phase does not affect the measurement results, it can be raised and ignored.

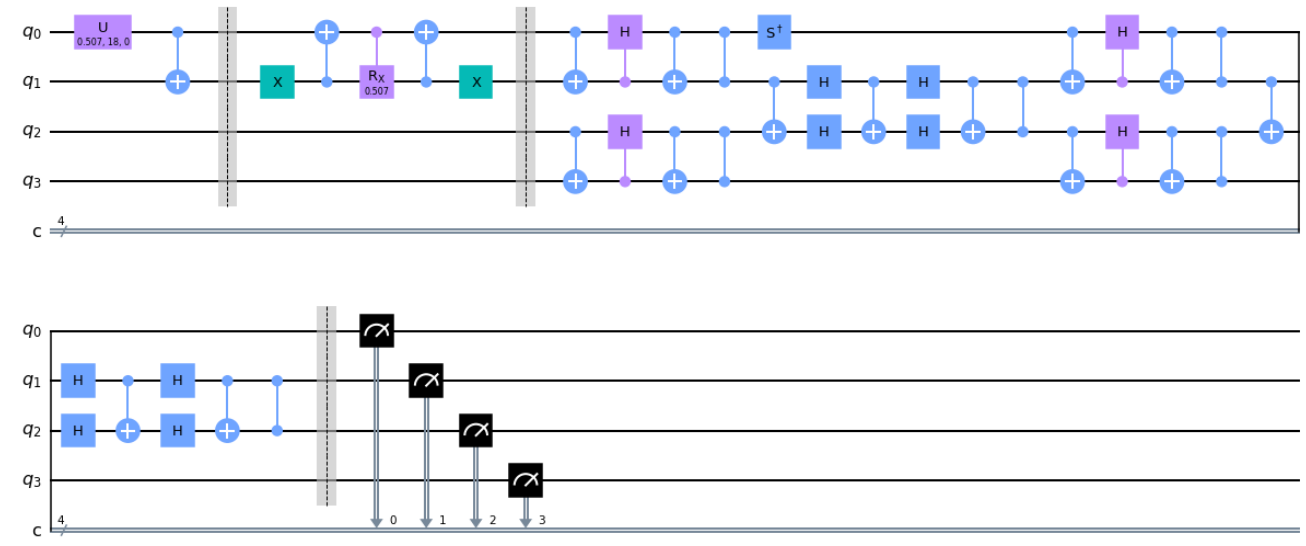

Fig. 10. The complete quantum circuit containing simulated time evolution.

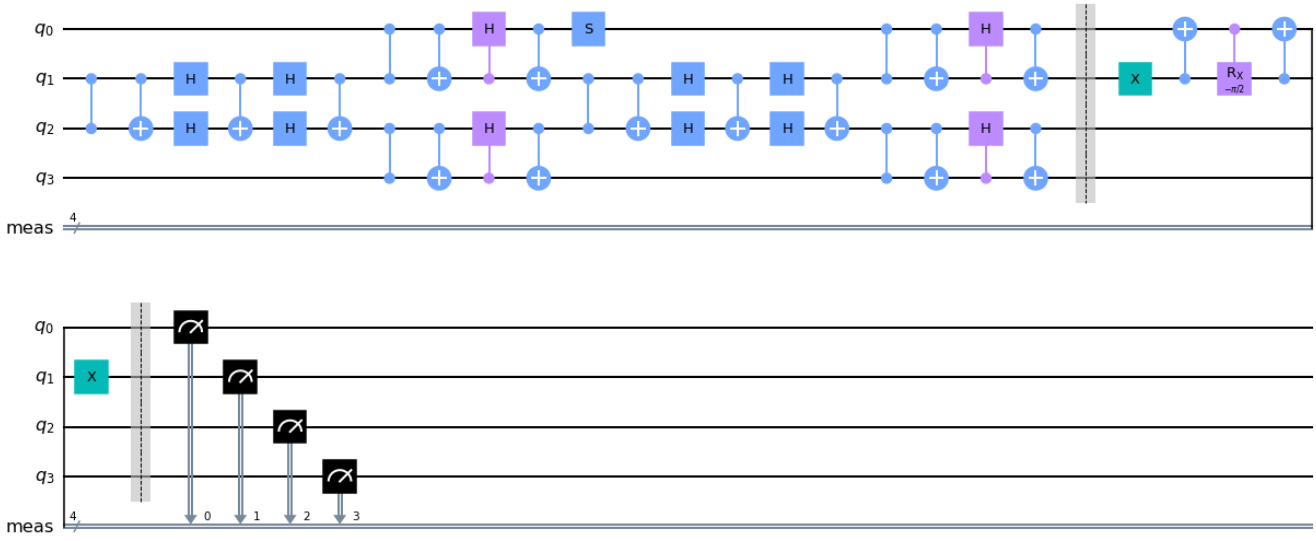

Fig. 12. The complete circuit of $U_{dis}^{\dagger}$. What this circuit does is that we are solving all the processes of the quantum Ising model. The order from left to right is QFFT $U_{QFFT}^{\dagger}$, and the Bogoliubov transform $U_{Bog}^{\dagger}$. Here we choose the $\lambda = 0$ case.

$$|\psi_b(t)\rangle =$$
$$e^{-i2\sqrt{1+\lambda^2}t}\left(\cos(\phi)\,|0000\rangle + e^{i4\sqrt{1+\lambda^2}t} i\sin(\phi)\,|1100\rangle\right)$$
$$\Rightarrow$$
$$|\psi_b(t)\rangle = \cos(\phi)\,|0000\rangle + e^{i4\sqrt{1+\lambda^2}t} i\sin(\phi)\,|1100\rangle \quad (106)$$

here we can use the quantum gate in Qiskit package which called U Gate to prepare the state, in matrix representation it can be written as

$$U(\theta,\phi,\lambda) = \begin{pmatrix} \cos\left(\frac{\theta}{2}\right) & e^{-i\lambda}\sin\left(\frac{\theta}{2}\right) \\ e^{i\phi}\sin\left(\frac{\theta}{2}\right) & e^{i(\phi+\lambda)}\cos\left(\frac{\theta}{2}\right) \end{pmatrix} \quad (107)$$

It can be found very interesting that this quantum gate is very similar to the matrix of the Bogoliubov transformation (equation(78)) in our example. To achieve our purpose, we only need to set the parameters of this gate as

$$U\left(\theta_1, \left(\frac{\pi}{2} + 4\sqrt{1+\lambda^2}t\right), 0\right) \quad (108)$$

Now, use equation(95) to compute the theoretical numerical solution of the magnetization is [2]

$$\langle\sigma_z\rangle = \frac{1 + 2\lambda^2 + \cos(4\sqrt{1+\lambda^2}t)}{2 + 2\lambda^2} \quad (109)$$

the final time evolution quantum circuit is shown in fig(10).

III. Results

A. Ground state simulation

First, we use the circuit in fig(9), which maps qubits to a quantum Ising model, to simulate the change of the average magnetization under different applied transverse magnetic fields (different λ). The magnetization direction we choose to observe is the z direction in the same direction as the transverse magnetic field, that is, $\langle\sigma_z\rangle$. The result is shown in the fig(13)

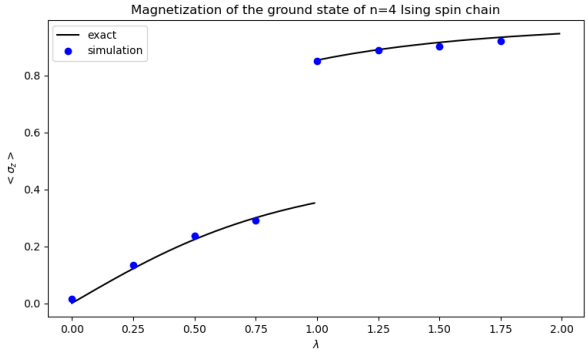

Fig. 13. The change of $\langle\sigma_z\rangle$ with the gradual increase of the transverse magnetic field λ. The solid line in the figure is the solution of the exact numerical calculation, and the blue dots are the results of the simulation using quantum circuits. Each blue dot is the average result after 1024 measurements. In this figure we are using the local ideal simulator(Aer simulator).

Since in fig(13) we use a local simulator instead of a real quantum computer, we can see that the simulation results are quite consistent with the theoretical calculations. In order to be more in line with the real situation, we add specific noise while simulating, the specific added noise are

- When applying a single qubit gate, flip the state of the qubit with probability 1% .
- When applying a 2-qubit gate apply single-qubit errors to each qubit.
- When resetting a qubit reset to 1 instead of 0 with probability 1% .
- When measuring a qubit, flip the state of the qubit with probability 1% .

After adding the above noise, the result of the simulation is shown in fig(14)

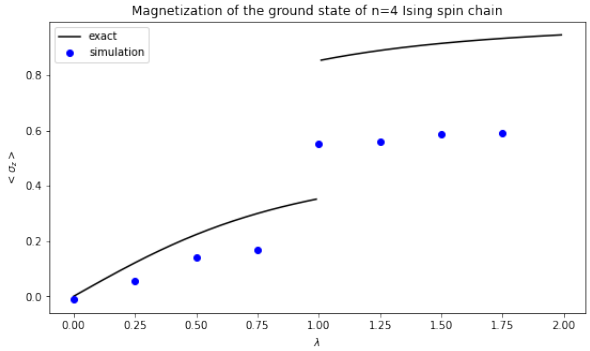

Fig. 14. The simulation results of the magnetization changing with the applied magnetic field strength after considering the noise that may exist in the actual quantum computer.

B. time evolution simulation

We choose the external transverse magnetic field of specific strength, respectively $\lambda = 0.5$, $\lambda = 0.9$ and $\lambda = 1.8$, and observe the change of magnetization $\langle\sigma_z\rangle$ in the same direction as the external transverse magnetic field over time. The initial state used in our simulation is $|\uparrow\uparrow\uparrow\uparrow\rangle$ as mentioned earlier. Consider the simulation results performed with ideal qubits is shown in fig(15)

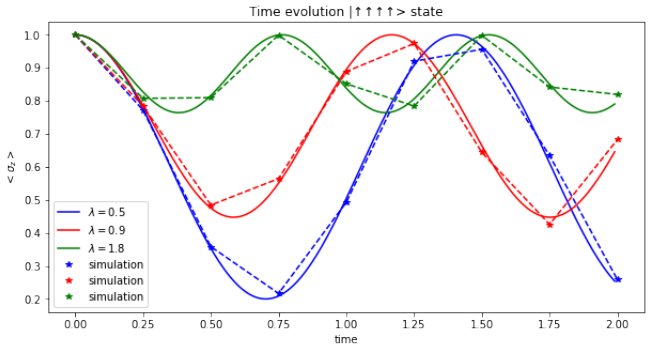

Fig. 15. Using ideal qubits to simulate the time evolution of magnetization $\langle\sigma_z\rangle$ of Ising model start with $|\uparrow\uparrow\uparrow\uparrow\rangle$ state. In the figure, we can see the oscillation of the magnetization under three different applied magnetic field strengths. The solid line is the numerical solution of the theory, and the solid point is the simulation result using a quantum circuit. Each point is the average of 1024 measurement results.

As we mentioned earlier, we consider the possible errors of actual qubits. The results of simulation is shown in the fig(16).

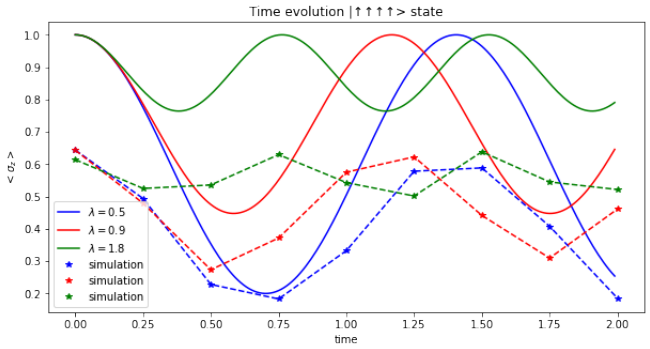

Fig. 16. Results of a simulation using qubits that account for noise. Other conditions are same as we mentioned in fig(15).

IV. CONCLUSION

We use Jordan—Wigner transformation, Fourier transformation, and Bogoliubov transformation to obtain the exact solution of the one-dimensional transverse field Ising model. According to the above mathematical process, we construct the corresponding quantum gates in reverse order and map the qubits into a system of the Ising model. Based on this, we can simulate the ground state of the Ising model or its time evolution by preparing an appropriate initial state of the qubit.

Then we had the local simulations to simulate this circuit with a non-noise model and 1%-error-rate (1%-e/r) model. After simulations, we have the plot of magnetization of the ground state and time evolution of $|\uparrow\uparrow\uparrow\uparrow\rangle$. In a non-noise model simulation, the results are the same as the theoretical exact solution predictions. And in 1%-e/r model simulation, we can see that though every gates have 99% fidelity, the results will be far from the exact solution. This means that to make a good solution time evolution, we need high-fidelity gate operations and high-coherence qubits to ensure the result is accurate enough.

In sum, we made a quantum circuit to simulate the exact solution of the one-dimensional transverse-field Ising model. The result is the same as a theoretical prediction if we have high-fidelity gate operations and high-coherence qubits. And also, we have shown the full mathematical formalism of the concepts and techniques we used in the quantum simulation.

Finally, we presented a feasible method to solve a condensed matter physics problem, which introduces the possibility of simulating other complex models.

References

[1] B. Fefferman, M. Foss-Feig, and A. V. Gorshkov, "Exact sampling hardness of ising spin models," *Phys. Rev. A*, vol. 96, p. 032324, Sep 2017.
[2] A. Cervera-Lierta, "Exact ising model simulation on a quantum computer," *Quantum*, vol. 2, p. 114, dec 2018.
[3] A. J. Ferris, "Fourier transform for fermionic systems and the spectral tensor network," *Phys. Rev. Lett.*, vol. 113, p. 010401, Jul 2014.
[4] D. R. Musk, "A comparison of quantum and traditional fourier transform computations," *Computing in Science & Engineering*, vol. 22, no. 6, pp. 103–110, 2020.
[5] G. B. Mbeng, A. Russomanno, and G. E. Santoro, "The quantum ising chain for beginners," 2020.

Appendix

A. Gate Operation

```
"""This module contains some handy gate operations
    for the quantum circuit."""
from qiskit import QuantumCircuit, QuantumRegister
from qiskit.circuit.library import *
import numpy as np
from qiskit.circuit.quantumregister import Qubit
from typing import Union


def CZ(circuit: QuantumCircuit, qubit0: int, qubit1:
     int):
    "Applies a CZ gate to the specified qubits."

    circuit.append(CZGate(), [qubit0, qubit1])


def fSWAP(circuit: QuantumCircuit, qubit0: Union[int
    , Qubit], qubit1: Union[int, Qubit]):
    "Applies a fSWAP gate to the specified qubits."

    if type(qubit0) is int:
        q0 = circuit.qubits[qubit0]
    elif type(qubit0) is Qubit:
        q0 = qubit0
    else:
        raise TypeError("only int or Qubit type
            accepted")
    if type(qubit1) is int:
        q1 = circuit.qubits[qubit1]
    elif type(qubit1) is Qubit:
        q1 = qubit1
    else:
        raise TypeError("only int or Qubit type
            accepted")

    circuit.cx(q0, q1)
    circuit.h(q0)
    circuit.h(q1)
    circuit.cx(q0, q1)
    circuit.h(q0)
    circuit.h(q1)
    circuit.cx(q0, q1)
    CZ(circuit, q0, q1)


def CH(circuit: QuantumCircuit, qubit0: int, qubit1:
     int):
    "Applies a Controlled_Haddamard gate to the
        specified qubits."

    circuit.append(CHGate(), [qubit0, qubit1])


def RZ(circuit: QuantumCircuit, qubit: int, angle:
    float):
    "Applies a RZ gate to the specified qubit."

    circuit.append(RZGate(angle), [qubit])


def RX(circuit: QuantumCircuit, qubit: int, angle:
    float):
    "Applies a RX gate to the specified qubit."

    circuit.append(RXGate(angle), [qubit])


def RY(circuit: QuantumCircuit, qubit: int, angle:
    float):
    "Applies a RY gate to the specified qubit."

    circuit.append(RYGate(angle), [qubit])
```

```
def CRX(circuit: QuantumCircuit, qubit0: int, qubit1
    : int, angle: float):
    "Applies a Controlled_RX gate to the specified
        qubits."

    circuit.append(CRXGate(angle), [qubit0, qubit1])


def B(circuit: QuantumCircuit, qubit0: Union[int,
    Qubit], qubit1: Union[int, Qubit], thk: float):
    "Applies a Bogoliubov gate to the specified
        qubits."

    if type(qubit0) is int:
        q0 = circuit.qubits[qubit0]
    elif type(qubit0) is Qubit:
        q0 = qubit0
    else:
        raise TypeError("only int or Qubit type
            accepted")
    if type(qubit1) is int:
        q1 = circuit.qubits[qubit1]
    elif type(qubit1) is Qubit:
        q1 = qubit1
    else:
        raise TypeError("only int or Qubit type
            accepted")

    circuit.x(q1)
    circuit.cx(q1, q0)
    CRX(circuit, q0, q1, thk)
    circuit.cx(q1, q0)
    circuit.x(q1)

# Fourier transform gates


def F2(qp, q0, q1):
    qp.cx(q0, q1)
    CH(qp, q1, q0)
    qp.cx(q0, q1)
    CZ(qp, q0, q1)


def F0(qp, q0, q1):
    F2(qp, q0, q1)


def F1(qp, q0, q1):
    F2(qp, q0, q1)
    qp.sdg(q0)
```

B. 1D TFIM Simulation Tools

```
from math import pi
from .gate_operation import *
import matplotlib.pyplot as plt


def digit_sum(n):
    num_str = str(n)
    sum = 0
    for i in range(0, len(num_str)):
        sum += int(num_str[i])
    return sum


def DFT(Udis, q0, q1, q2, q3):
    F1(Udis, q0, q1)
    F0(Udis, q2, q3)
    fSWAP(Udis, q1, q2)
    F0(Udis, q0, q1)
    F0(Udis, q2, q3)
    fSWAP(Udis, q1, q2)


def Udisg(Udis, lam, q0, q1, q2, q3):
    k = 1
    n = 4
    th1 = np.arccos((lam+np.cos(2*pi*k/n)) /
                    np.sqrt((lam+np.cos(2*pi*k/n))
                        **2+np.sin(2*pi*k/n)**2))
    B(Udis, q0, q1, th1)
    Udis.barrier()
    DFT(Udis, q0, q1, q2, q3)


def Initial(qc, lam, q0, q1, q2, q3):
    if lam < 1:
        qc.x(q3)


def Ising(qc, ini, udis, mes, lam, q0, q1, q2, q3,
    c0, c1, c2, c3):
    Initial(ini, lam, q0, q1, q2, q3)
    Udisg(udis, lam, q0, q1, q2, q3)
    mes.measure([q0, q1, q2, q3], [c0, c1, c2, c3])
    qc.compose(ini, inplace=True)
    qc.barrier()
    qc.compose(udis, inplace=True)
    qc.barrier()
    qc.compose(mes, inplace=True)


def exact(lam):
    if lam < 1:
        return lam/(2*np.sqrt(1+lam**2))
    if lam > 1:
        return 1/2+lam/(2*np.sqrt(1+lam**2))
    return None


def plot_Mag_of_ground_state(vexact, mag_sim):
    plt.clf()
    l = np.arange(0.0, 2.0, 0.01)
    l1 = np.arange(0.0, 2.0, 0.25)
    plt.figure(figsize=(9, 5))
    plt.plot(l, vexact(l), 'k', label='exact')
    plt.plot(l1, mag_sim, 'bo', label='simulation')
    plt.xlabel('$\lambda$')
    plt.ylabel('$<\sigma_{z}>$')
    plt.legend()
    plt.title('Magnetization of the ground state of
        n=4 Ising spin chain')
    plt.savefig(
        "./images/1D_n=4_Ising_spin_chain/
            Magnetization_of_the_ground_state_of_n=4
            _Ising_spin_chain.png")
    plt.show()
    plt.clf()


def Initial_time(qc, t, lam, q0, q1, q2, q3):
    qc.u(np.arccos(lam/np.sqrt(1+lam**2)), pi/2.+4*t
        *np.sqrt(1+lam**2), 0., q0)
    qc.cx(q0, q1)


def Ising_time(qc, ini, udis, mes, lam, t, q0, q1,
    q2, q3, c0, c1, c2, c3):
    Initial_time(ini, t, lam, q0, q1, q2, q3)
    Udisg(udis, lam, q0, q1, q2, q3)
    mes.measure([q0, q1, q2, q3], [c0, c1, c2, c3])
    qc.compose(ini, inplace=True)
    qc.barrier()
    qc.compose(udis, inplace=True)
    qc.barrier()
    qc.compose(mes, inplace=True)


def exact_time(lam, tt):
```

```
    Mt = (1 + 2*lam**2 + np.cos(4*tt*np.sqrt(1 + lam
        **2)))/(2 + 2*lam**2)
    return Mt


def plot_Time_evolution_all_up_state(vexact_t,
    magt_sim):
    plt.clf()
    t = np.arange(0.0, 2.0, 0.01)
    tt = np.arange(0.0, 2.25, 0.25)
    plt.figure(figsize=(10, 5))
    plt.plot(t, vexact_t(0.5, t), 'b', label='$\
        lambda=0.5$')
    plt.plot(t, vexact_t(0.9, t), 'r', label='$\
        lambda=0.9$')
    plt.plot(t, vexact_t(1.8, t), 'g', label='$\
        lambda=1.8$')
    plt.plot(tt, magt_sim[0], 'b*', label='
        simulation')
    plt.plot(tt, magt_sim[1], 'r*', label='
        simulation')
    plt.plot(tt, magt_sim[2], 'g*', label='
        simulation')
    plt.plot(tt, magt_sim[0], 'b--')
    plt.plot(tt, magt_sim[1], 'r--')
    plt.plot(tt, magt_sim[2], 'g--')
    plt.xlabel('time')
    plt.ylabel('$<\sigma_{z}>$')
    plt.legend()
    plt.title('Time evolution |↑↑↑↑> state')
    plt.savefig(
        "./images/1D_n=4_Ising_spin_chain/
            Time_evolution_all_up_state.png")
    plt.show()
    plt.clf()
```