Advanced: Quantum Dynamics Simulation

We will perform quantum dynamics (QD) simulation on a quantum computer for the transverse-field Ising model (TFIM) Hamiltonian for two spins,

$$H = -J\sigma_0^z\sigma_1^z - B\sum_{j=0}^{1}\sigma_j^x, \tag{1}$$

where σ_j^z and σ_j^x are Pauli *Z* and *X* matrices acting on the *j*-th spin, *J* is the exchange coupling, and *B* is the magnetic field along the *x* axis.

Time evolution of a two-spin wave function, $|\Psi(t)\rangle = |\psi_0(t)\rangle|\psi_1(t)\rangle$ ($|\psi_j(t)\rangle$ is the wave function of the *j*-th spin at time *t*), for small time step Δt is governed by (*cf.* https://aiichironakano.github.io/phys516/03QD.pdf)

$$|\Psi(t+\Delta t)\rangle = \exp(-iH\Delta t)|\Psi(t)\rangle \tag{2}$$

in the atomic unit. Using Trotter expansion, the time-propagation operator is approximated as

$$\exp(-iH\Delta t) = \exp(i\Delta t J\sigma_0^z\sigma_1^z)\exp(i\Delta t B\sigma_0^x)\exp(i\Delta t B\sigma_1^x) + O(\Delta t^2). \tag{3}$$

Let us first consider the transverse-field propagator $\exp\left(i\Delta t B\sigma_j^x\right)$ acting on the *j*-th spin independent of the other spin. We use the eigendecomposition (see Appendix) of Pauli *X* matrix,

$$\sigma^x = X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}. \tag{4}$$

Note that

$$\sigma^x H = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}\frac{1}{\sqrt{2}}\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} = \frac{1}{\sqrt{2}}\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}\begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} = H\sigma^z, \tag{5}$$

where H is the Hadamard gate (which is column-aligned eigenvectors $(1/\sqrt{2}, \pm 1/\sqrt{2})^T$ of σ^x with respective eigenvalues ± 1), or equivalently

$$\sigma^x = H\sigma^z H, \tag{6}$$

where we have used the fact *H* is a symmetric orthogonal matrix, *i.e.*, $H^{-1} = H^T = H$ and thus

$$H^2 = I \tag{7}$$

(*I* is the identity matrix).

Using Taylor expansion of the time propagator and Eqs. (6) and (7) (the procedure is called telescoping),

$$\exp\left(i\Delta t B\sigma^x\right) = \sum_{n=0}^{\infty}\frac{(i\Delta tB)^n}{n!}\sigma^{x\,n} = \sum_{n=0}^{\infty}\frac{(i\Delta tB)^n}{n!}(H\sigma^z H)^n =$$

$$\sum_{n=0}^{\infty}\frac{(i\Delta tB)^n}{n!}\overbrace{H\sigma^z HH\sigma^z H\cdots H\sigma^z H}^{n\text{ times}}\ (\text{every internal HH product becomes } I) =$$

$$H\sum_{n=0}^{\infty}\frac{(i\Delta tB)^n}{n!}\sigma^{z\,n}H = H\sum_{n=0}^{\infty}\frac{(i\Delta tB)^n}{n!}\begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}^n H = H\begin{pmatrix} \sum_{n=0}^{\infty}\frac{(i\Delta tB)^n}{n!} & 0 \\ 0 & \sum_{n=0}^{\infty}\frac{(-i\Delta tB)^n}{n!} \end{pmatrix}H =$$

$$H\begin{pmatrix} e^{i\Delta tB} & 0 \\ 0 & e^{-i\Delta tB} \end{pmatrix}H = HR_z(-2\Delta tB)H = \frac{1}{2}\begin{pmatrix} e^{i\Delta tB}+e^{-i\Delta tB} & e^{i\Delta tB}-e^{-i\Delta tB} \\ e^{i\Delta tB}-e^{-i\Delta tB} & e^{i\Delta tB}+e^{-i\Delta tB} \end{pmatrix} =$$

$$\begin{pmatrix} \cos(\Delta tB) & i\sin(\Delta tB) \\ i\sin(\Delta tB) & \cos(\Delta tB) \end{pmatrix} = R_x(-2\Delta tB). \tag{8}$$

In terms of the native gates on IBM Q computers, Eq. (8) can be implemented using either rotation around the *z* axis, $R_z(\theta)$, along with Hadamard gate H, or solely using rotation around the *x* axis, $R_x(\theta)$. Here, R_z and R_x gates are defined as

$$R_z(\theta) = \begin{pmatrix} e^{-i\theta/2} & 0 \\ 0 & e^{i\theta/2} \end{pmatrix}, \tag{9}$$

$$R_x(\theta) = \begin{pmatrix} \cos(\theta/2) & -i\sin(\theta/2) \\ -i\sin(\theta/2) & \cos(\theta/2) \end{pmatrix}. \tag{10}$$

(see https://github.com/Qiskit/qiskit-tutorials/blob/master/tutorials/circuits/3_summary_of_quantum_operations.ipynb).

Next, we consider the exchange-coupling propagator $\exp(i\Delta tJ\sigma_0^z\sigma_1^z)$. We first consider a tensor product of operators multiplied by a scalar constant,

$$i\Delta tJ\sigma_0^z \otimes \sigma_1^z = i\Delta tJ \begin{pmatrix} 1\cdot\begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} & 0\cdot\begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} \\ 0\cdot\begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} & -1\cdot\begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} \end{pmatrix} = \begin{pmatrix} i\Delta tJ & 0 & 0 & 0 \\ 0 & -i\Delta tJ & 0 & 0 \\ 0 & 0 & -i\Delta tJ & 0 \\ 0 & 0 & 0 & i\Delta tJ \end{pmatrix}. \tag{11}$$

Since this is a diagonal matrix, it can be exponentiated element by element as

$$\exp(i\Delta tJ\sigma_0^z\sigma_1^z) = \begin{pmatrix} \exp(i\Delta tJ) & 0 & 0 & 0 \\ 0 & \exp(-i\Delta tJ) & 0 & 0 \\ 0 & 0 & \exp(-i\Delta tJ) & 0 \\ 0 & 0 & 0 & \exp(i\Delta tJ) \end{pmatrix} = \begin{pmatrix} R_z(-2\Delta tJ) & 0 \\ 0 & R_z(2\Delta tJ) \end{pmatrix}. \tag{12}$$

Now consider the following sequence of quantum gates operating on two qubits, q_0 and q_1,

$$G = CX(q_0, q_1) \cdot R_1^z(-2\Delta tJ) \cdot CX(q_0, q_1), \tag{13}$$

where

$$CX(q_0, q_1) = \begin{pmatrix} I & 0 \\ 0 & X \end{pmatrix} \tag{14}$$

is the controlled *X* (CNOT) gate, with q_0 and q_1 being the control and target bits, and R_1^z is the R^z gate acting on q_1. When operating on two qubits, R_1^z signifies a tensor product,

$$I \otimes R^z(-2\Delta tJ) = \begin{pmatrix} 1\cdot R^z(-2\Delta tJ) & 0\cdot R^z(-2\Delta tJ) \\ 0\cdot R^z(-2\Delta tJ) & 1\cdot R^z(-2\Delta tJ) \end{pmatrix} = \begin{pmatrix} R^z(-2\Delta tJ) & 0 \\ 0 & R^z(-2\Delta tJ) \end{pmatrix}. \tag{15}$$

Substituting Eqs. (14) and (15) in Eq. (13), we obtain

$$G = \begin{pmatrix} I & 0 \\ 0 & X \end{pmatrix} \begin{pmatrix} R^z(-2\Delta tJ) & 0 \\ 0 & R^z(-2\Delta tJ) \end{pmatrix} \begin{pmatrix} I & 0 \\ 0 & X \end{pmatrix} = \begin{pmatrix} R^z(-2\Delta tJ) & 0 \\ 0 & XR^z(-2\Delta tJ)X \end{pmatrix}. \tag{16}$$

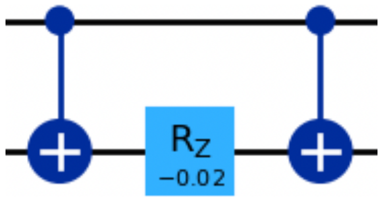
Here,

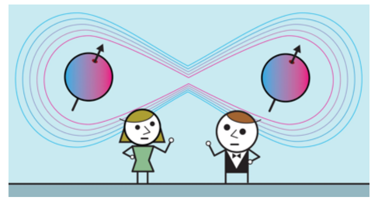
$$XR^z(-2\Delta tJ)X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} \exp(i\Delta tJ) & 0 \\ 0 & \exp(-i\Delta tJ) \end{pmatrix} \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} = \begin{pmatrix} 0 & \exp(-i\Delta tJ) \\ \exp(i\Delta tJ) & 0 \end{pmatrix} \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} = \begin{pmatrix} \exp(-i\Delta tJ) & 0 \\ 0 & \exp(i\Delta tJ) \end{pmatrix} = R^z(2\Delta tJ). \tag{17}$$

Substituting Eq. (17) in Eq. (16) and compare the result with Eq. (12), we arrive at the identity,

$$G = CX(q_0, q_1)R_1^z(-2\Delta tJ)CX(q_0, q_1) = \begin{pmatrix} R^z\,(-2\Delta tJ) & 0 \\ 0 & R^z\,(2\Delta tJ) \end{pmatrix} = \exp(i\Delta tJ\sigma_0^z\sigma_1^z). \quad (18)$$

where the last equality results from Eq. (12). Namely, $G = CX(q_0, q_1) \cdot R_1^z(-2\Delta tJ) \cdot CX(q_0, q_1)$ is a quantum-gate implementation of the exchange-coupling propagator $\exp(i\Delta tJ\sigma_0^z\sigma_1^z)$.





Combining Eqs. (8) and (18) for the transverse-field and exchange-coupling time propagators, respectively, quantum-circuit implementation for a single time step of time evolution for the TFIM model, Eq. (1), is given by

$$\exp(-iH\Delta t) = \exp(i\Delta tJ\sigma_0^z\sigma_1^z)\exp(i\Delta tB\sigma_0^x)\exp(i\Delta tB\sigma_1^x) = CX(q_0, q_1)R_1^z(-2\Delta tJ)CX(q_0, q_1)R_0^x(-2\Delta tB)R_1^x(-2\Delta tB). \quad (18)$$

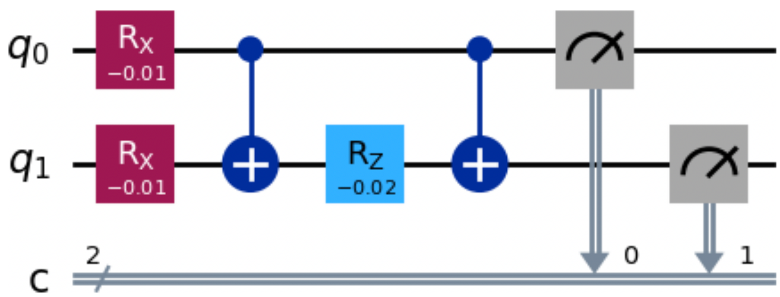


Fig. 1: Quantum circuit for time evolution of TFIM in IBM Quantum Lab.

Hands-on Exercise (try it at https://quantum-computing.ibm.com using IBM Quantum Lab)

Execute the following Qiskit program to perform a single time step of QD simulation. Here, we have used model parameters, $J = 1$, $B = 0.5$ and $\Delta t = 0.01$, in atomic units.

```
##### Single step of Trotter propagation in transverse-field Ising model #####

import numpy as np

# Import standard Qiskit libraries
from qiskit import QuantumCircuit
from qiskit_aer import AerSimulator
from qiskit.visualization import *
from ibm_quantum_widgets import *

### Physical parameters (atomic units) ###
J = 1.0    # Exchange coupling
B = 0.5    # Transverse magnetic field
dt = 0.01  # Time-discretization unit

### Build a circuit ###

circ = QuantumCircuit(2, 2)  # 2 quantum & 2 classical registers

circ.rx(-2*dt*B, 0)  # Transverse-field propagation of spin 0
circ.rx(-2*dt*B, 1)  # Transverse-field propagation of spin 1
circ.cx(0, 1)        # Exchange-coupling time propagation (1)
circ.rz(-2*dt*J, 1)  #                                    (2)
circ.cx(0, 1)        #                                    (3)
circ.measure(range(2), range(2))  # Measure both spins
circ.draw('mpl')
```

This will build a circuit and draw it, which should then be transpiled and run on a simulator as follows.

```
### Simulate on OpenQASM backend ###

# Use Aer simulator
backend = AerSimulator()
# Transpile the quantum circuit to low-level QASM instructions
from qiskit import transpile
circ_compiled = transpile(circ, backend)
# Execute the circuit on the Qasm simulator, repeating 1024 times
job_sim = backend.run(circ_compiled, shots=1024)
# Grab the results from the job
result_sim = job_sim.result()
# Get the result
counts = result_sim.get_counts(circ_compiled)
# Plot histogram
from qiskit.visualization import plot_histogram
plot_histogram(counts)
```

Table I: Qiskit program for single-time-step QD simulation of TFIM: tfim-1step.qiskit (https://aiichironakano.github.io/phys516/src/QComp/tfim-1step.qiskit).

After opening a Qiskit (ipykenel) notebook, you can copy and paste the above code into a cell in the Python notebook. Here, we have used QASM simulator as a backend. Actual quantum dynamics simulation [L. Bassman *et al*., *Phys. Rev. B* **101**, 184305 ('20)] will iterate this unit-time stepping for many time steps. For Python programming underlying Qiskit, see A. Scopatz and K. D. Huff, *Effective Computation in Physics* (O'Reilly, '15).

Appendix: Eigendecomposition

For a 2×2 Hermitian matrix,

$$\mathbf{A} = \begin{bmatrix} a & b \\ b^* & a \end{bmatrix}, \tag{A1}$$

where a and b are real and complex numbers, respectively, consider an eigenvalue problem,

$$\begin{bmatrix} a & b \\ b^* & a \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = \varepsilon \begin{bmatrix} u \\ v \end{bmatrix}. \tag{A2}$$

or equivalently

$$\begin{bmatrix} \varepsilon - a & -b \\ -b^* & \varepsilon - a \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}. \tag{A3}$$

For nontrivial solutions (*i.e.*, other than $u = v = 0$), the determinant of the matrix in Eq. (A3) should be zero. (Otherwise, one can invert Eq. (A3) to get $u = v = 0$.) Hence,

$$\begin{vmatrix} \varepsilon - a & -b \\ -b^* & \varepsilon - a \end{vmatrix} = (\varepsilon - a)^2 - |b|^2 = 0, \text{ Secular (characteristic) equation} \tag{A4}$$

which has two solutions,

$$\varepsilon_\pm = a \pm |b|. \text{ Eigenvalues} \tag{A5}$$

The corresponding eigenvectors can be obtained by solving Eq. (A3) for these eigenvalues

$$\begin{bmatrix} |b| & -b \\ -b^* & |b| \end{bmatrix} \begin{bmatrix} u_+ \\ v_+ \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}; \begin{bmatrix} -|b| & -b \\ -b^* & -|b| \end{bmatrix} \begin{bmatrix} u_- \\ v_- \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \tag{A6}$$

with the answers (note the degeneracy of the two linear equations for each eigenvalue, *e.g.*, $|b|u_+ - bv_+ = 0 \Longrightarrow \left(\times \frac{-b^*}{|b|}\right) -b^*u_+ + |b|v_+ = 0$)

$$\mathbf{w}_\pm = \begin{bmatrix} u_\pm \\ v_\pm \end{bmatrix} = \frac{1}{\sqrt{2}|b|} \begin{bmatrix} b \\ \pm|b| \end{bmatrix}. \text{ Eigenvectors} \tag{A7}$$

In Eq. (A7), we have normalized each eigenvector so that

$$\mathbf{w}_\pm^\dagger \mathbf{w}_\pm = [u_\pm^* \quad v_\pm^*] \begin{bmatrix} u_\pm \\ v_\pm \end{bmatrix} = \frac{\overbrace{b^*b}^{|b|^2} + |b|^2}{2|b|^2} = 1, \tag{A8}$$

where $\mathbf{w}_\pm^\dagger$ denotes the Hermitian conjugate (or conjugate transpose) of $\mathbf{w}_\pm$. Also, the two eigenvectors are orthogonal:

$$\mathbf{w}_\mp^\dagger \mathbf{w}_\pm = [u_\mp^* \quad v_\mp^*] \begin{bmatrix} u_\pm \\ v_\pm \end{bmatrix} = \frac{\overbrace{b^*b}^{|b|^2} - |b|^2}{2|b|^2} = 0. \tag{A9}$$

Now, define a 2×2 matrix composed of column aligned eivenvectors,

$$\mathbf{U} = [\mathbf{w}_+ \quad \mathbf{w}_-] = \begin{bmatrix} u_+ & u_- \\ v_+ & v_- \end{bmatrix} = \frac{1}{\sqrt{2}|b|} \begin{bmatrix} b & b \\ |b| & -|b| \end{bmatrix}, \tag{A10}$$

then

$$\mathbf{U}^\dagger \mathbf{U} = \begin{bmatrix} \mathbf{w}_+^\dagger \\ \mathbf{w}_-^\dagger \end{bmatrix} [\mathbf{w}_+ \quad \mathbf{w}_-] = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} = \mathbf{I}, \tag{A11}$$

where $\mathbf{I}$ is the 2×2 identity matrix and we have used the orthonormalization relations, Eqs. (A8) and (A9). Using the explicit formula for $\mathbf{U}$ in Eq. (A10), we can also verify that $\mathbf{U}\mathbf{U}^\dagger = \mathbf{I}$ and hence $\mathbf{U}$ is a unitary matrix:

$$\mathbf{U}^\dagger\mathbf{U} = \mathbf{U}\mathbf{U}^\dagger = \mathbf{I}. \text{ Unitary} \tag{A12}$$

The two solutions of Eq. (A2) can now be combined into a matrix form as

$$\begin{cases} \begin{bmatrix} a & b \\ b^* & a \end{bmatrix} \begin{bmatrix} u_+ \\ v_+ \end{bmatrix} = \varepsilon_+ \begin{bmatrix} u_+ \\ v_+ \end{bmatrix} \\ \begin{bmatrix} a & b \\ b^* & a \end{bmatrix} \begin{bmatrix} u_- \\ v_- \end{bmatrix} = \varepsilon_- \begin{bmatrix} u_- \\ v_- \end{bmatrix} \end{cases} \Leftrightarrow \underbrace{\begin{bmatrix} a & b \\ b^* & a \end{bmatrix}}_{\mathbf{A}} \underbrace{\begin{bmatrix} u_+ & u_- \\ v_+ & v_- \end{bmatrix}}_{\mathbf{U}} = \underbrace{\begin{bmatrix} u_+ & u_- \\ v_+ & v_- \end{bmatrix}}_{\mathbf{U}} \underbrace{\begin{bmatrix} \varepsilon_+ & 0 \\ 0 & \varepsilon_- \end{bmatrix}}_{\mathbf{D}}, \tag{A13}$$

i.e.,

$$\mathbf{AU} = \mathbf{UD}, \tag{A14}$$

where we have defined a diagonal matrix,

$$\mathbf{D} = \begin{bmatrix} \varepsilon_+ & 0 \\ 0 & \varepsilon_- \end{bmatrix}. \tag{A15}$$

$$\because \begin{bmatrix} u_+ & u_- \\ v_+ & v_- \end{bmatrix} \begin{bmatrix} \varepsilon_+ \\ 0 \end{bmatrix} = \varepsilon_+ \begin{bmatrix} u_+ \\ v_+ \end{bmatrix} \text{ and } \begin{bmatrix} u_+ & u_- \\ v_+ & v_- \end{bmatrix} \begin{bmatrix} 0 \\ \varepsilon_- \end{bmatrix} = \varepsilon_- \begin{bmatrix} u_- \\ v_- \end{bmatrix}$$ 1^{st} & 2^{nd}-column pickers

Multiplying both sides of Eq. (A14) by $\mathbf{U}^\dagger$ from the right hand and using the unitary, Eq. (A12), we obtain

$$\mathbf{A} = \mathbf{U}\mathbf{D}\mathbf{U}^\dagger. \text{ Eigendecomposition} \tag{A16}$$

or more explicitly

$$\begin{bmatrix} a & b \\ b^* & a \end{bmatrix} = \frac{1}{\sqrt{2}|b|} \begin{bmatrix} b & b \\ |b| & -|b| \end{bmatrix} \begin{bmatrix} a + |b| & 0 \\ 0 & a - |b| \end{bmatrix} \frac{1}{\sqrt{2}|b|} \begin{bmatrix} b^* & |b| \\ b^* & -|b| \end{bmatrix}. \tag{A17}$$

(Example) Pauli X matrix, *i.e.*, $a = 0$ and $b = 1$

$$\mathbf{X} = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} = \mathbf{HZH}. \tag{A18}$$

where $\mathbf{H}$ and $\mathbf{Z}$ are matrix representations of Hadamard and Pauli Z gates.