Lab IX
Simulating quantum spin chain.

Jakub Tworzydlo

Institute of Theoretical Physcis, UW
Jakub.Tworzydlo@fuw.edu.pl

21/04/2026 Pasteura 5, Warszawa

The physical model: 1D Transverse Field Ising Model

Today we will simulate the dynamics of a 1D spin chain under the Transverse Field Ising Model (TFIM). The Hamiltonian consists of two competing terms:

$$H = H_{zz} + H_x = -J \sum_{i=1}^{N-1} \sigma_i^z \sigma_{i+1}^z - h \sum_{i=1}^{N} \sigma_i^x$$

- σ^z, σ^x are the Pauli spin matrices representing spin
- J – spin-spin interaction energy (tending to align spins)
- h – external magnetic field (tending to flip spins)

Observable: **average magnetization** along the z-axis

$$m = \frac{1}{N} \sum_{i=1}^{N} \sigma_i^z.$$

In our simulation, we will measure the qubits to find the spin states, mapping classical bit `0` $\rightarrow +1$ (spin up) and `1` $\rightarrow -1$ (spin down).

The simulation: Trotterization

We want to simulate the time evolution of the system $U(t) = e^{-iHt}$. However, because the interaction and the magnetic field do not commute ($[H_{zz}, H_x] \neq 0$), we cannot simply split the exponent.

Instead, we use the **Suzuki-Trotter approximation**. For a small time step Δt:

$$e^{-i(H_{zz}+H_x)\Delta t} \approx e^{-iH_{zz}\Delta t} e^{-iH_x\Delta t}$$

By repeating this small step N_{iter} times, we can simulate the continuous evolution. In our digital quantum computer, we must map these physical time-evolution operators into standard logic gates (R_X, R_Z, and CZ).

Translating physics to logic gates

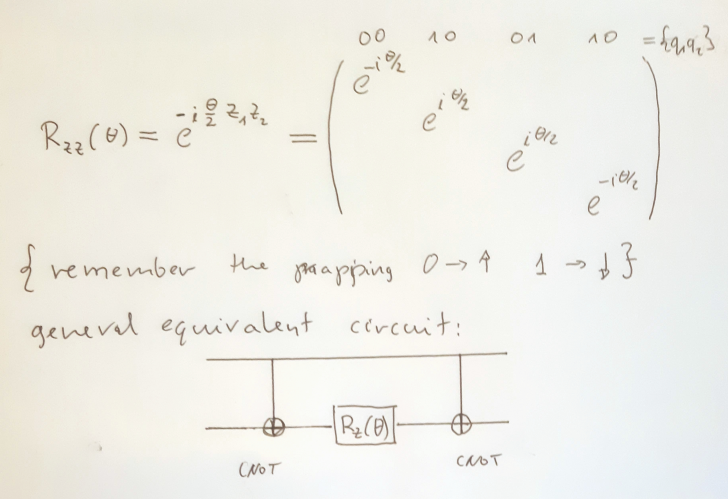
How do we build the circuit for a single Trotter step?

The magnetic field (H_x)
The operator $e^{ih\Delta t \sigma_i^x}$ translates directly to an R_X gate. Since $R_X(\theta) = e^{-i\frac{\theta}{2}X}$, we map the physical parameter to the gate angle:

$$\theta_h = -2h\Delta t \quad \longrightarrow \quad \text{Apply } R_X(\theta_h) \text{ to all qubits.}$$

The spin-spin interaction (H_{zz})
The operator $e^{iJ\Delta t \sigma_i^z \sigma_{i+1}^z}$ acts on two neighboring spins. It can be constructed using R_Z rotation and two Controlled-X (CX) gates.

Implementing RZZ gate



Simplification

simplification for $\theta = -\frac{\pi}{2}$

$$R_z\left(-\tfrac{\pi}{2}\right) = \begin{pmatrix} e^{i\pi/4} & \\ & e^{-i\pi/4} \end{pmatrix}$$

$$R_{zz}\left(-\tfrac{\pi}{2}\right) = \begin{pmatrix} e^{i\pi/4} & & & \\ & e^{-i\pi/4} & & \\ & & e^{-i\pi/4} & \\ & & & e^{i\pi/4} \end{pmatrix}$$

let's check:

$$CZ\left(R_z\left(-\tfrac{\pi}{2}\right) \otimes R_z\left(-\tfrac{\pi}{2}\right)\right) = \begin{pmatrix} 1 & & & \\ & 1 & & \\ & & 1 & \\ & & & -1 \end{pmatrix}\begin{pmatrix} e^{i\pi/4}e^{i\pi/4} & & & \\ & e^{i\pi/4}e^{-i\pi/4} & & \\ & & e^{-i\frac{\pi}{4}}e^{i\pi/4} & \\ & & & e^{-i\frac{\pi}{4}}e^{-i\frac{\pi}{4}} \end{pmatrix}$$

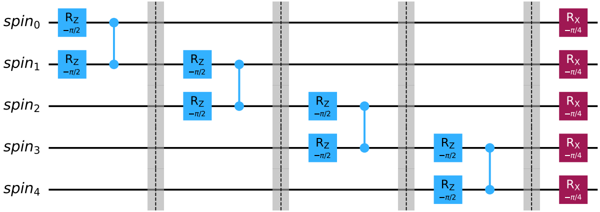
$$= \begin{pmatrix} 1 & & & \\ & 1 & & \\ & & 1 & \\ & & & -1 \end{pmatrix}\begin{pmatrix} i & & & \\ & 1 & & \\ & & 1 & \\ & & & -i \end{pmatrix} = \begin{pmatrix} i & & & \\ & 1 & & \\ & & 1 & \\ & & & i \end{pmatrix} = e^{i\frac{\pi}{4}} R_{zz}\left(-\tfrac{\pi}{2}\right)$$

DONE!

One step of the simulation

Setting $\theta_z = -2J\Delta t = -\frac{\pi}{2}$:

Apply $R_Z(\theta_z)$ to qubit i and $i+1$, then apply $CZ(i, i+1)$.



Task 1: Quantum circuit simulation

Goal: Simulate the Trotterized TFIM on a 5-qubit chain and measure the magnetization using Qiskit's `Sampler`.

- Set your physical parameters: $J = \pi/4$, $h = \pi/8$, and $\Delta t = 1$.
- Gate angles: $\theta_z = -2J\Delta t$ and $\theta_h = -2h\Delta t$.
- Create a loop to build circuits for $\text{iter} = 0, 1, \ldots, 20$ Trotter steps. In each step, apply the H_{zz} interaction gates to all neighboring pairs, followed by the H_x rotation gates to all qubits.
- Execute the batch of circuits with 10000 shots each.
- Write a function to parse the `counts` dictionary. Map the bits to spins (`0`$\rightarrow +1$, `1`$\rightarrow -1$), calculate the average magnetization for each iteration, and plot the result.

Task 2: Exact matrix verification

Goal: Verify your circuit results by calculating the exact matrices for the Trotter steps using linear algebra in Python.

- Use `numpy.kron` to generate the full physical σ_i^z and σ_i^x matrices (size $2^N \times 2^N$) for all $N = 5$ spins.
- Construct the Hamiltonians H_{zz} and H_x.
- Use `scipy.linalg.expm` to calculate the unitary matrices for a single step: $U_{zz} = e^{-iH_{zz}\Delta t}$ and $U_x = e^{-iH_x\Delta t}$.
- Start with the state vector $|00000\rangle$. Iteratively apply $U_x U_{zz}$ to the state vector for 20 steps.
- At each step, calculate the expected magnetization $m = \frac{1}{N}\sum \sigma_i^z$ and plot it on the **same graph** as your Task 1 results. They should match perfectly!

Scoring: Either Task 1 or Task 2 separately – 2pts. Both in agreement – full 5pts.

Hint: representing spins with matrices

- representation of spin $1/2$ with Pauli matrices

$$\hat{\sigma}^x = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix},\ \hat{\sigma}^y = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix},\ \hat{\sigma}^z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$$

- states as vectors

$$|\uparrow\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix}, |\downarrow\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$$

- spin operator $\sigma_i^{x,z}$ using Kronecker product $\hat{\sigma}_i^{x,z} = \mathbf{1} \otimes \ldots \otimes \hat{\sigma}^{x,z} \otimes \ldots \otimes \mathbf{1}$ (with matrix $\hat{\sigma}^{x,z}$ at position i)
- states using Kronecker product e.g.

$$|\uparrow \ldots \uparrow\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \otimes \ldots \otimes \begin{pmatrix} 1 \\ 0 \end{pmatrix}$$

Presentation or extra – suggestions

Experiment with the size of the chain: how large N can you reasonably simulate with the matrix simulation, circuit simulation, or circuit with noise model?

Experiment with real devices: does it agree with noisy simulation? Can you get N sizes going beyond our digital simulation?

Discuss the exact solution of the one dimensional Transverse Field Ising Model (reference: SciPost Lecture Notes).

Simulation with noise: extra

```
## Imports for running a simulator
from qiskit_ibm_runtime import Sampler
from qiskit_aer import AerSimulator
from qiskit_ibm_runtime.fake_provider import FakeBrisbane as Fake


aer_backend = AerSimulator.from_backend( Fake() )

## Generate ISA circuit transpiled for a given backend
from qiskit.transpiler.preset_passmanagers import generate_preset_pass_manager

pm = generate_preset_pass_manager(backend=aer_backend)
isa_circuits = pm.run(circuits)

sampler = Sampler(aer_backend)
job = sampler.run(isa_circuits, shots=10000)
results = job.result()
```