Institute for Theoretical Physics
University of Cologne

Prof. Dr. S. Trebst
Q. Preiss, F. Eckstein, M. Pütz

Quantum Computational Physics

Exercise Sheet 8

Winter Term 2024/25

Due date: Friday, 07.02.2025 **Discussion**: Friday, 31.01.2025

Website: thp.uni-koeln.de/trebst/Lectures/2024-QuantCompPhys.shtml

In this last exercise sheet, we want to study **measurement-only circuit models** which eliminate all unitary gates from (deep) random circuits that we previously considered.

Exercise 14: The monitored transverse-field Ising model

Our first exercise will consider two different types of (non-commuting) measurements that compete with one another, such as projective single qubit measurements and two qubit (parity) checks, in different bases. As already discussed in the lecture, such a measurement-only model can be set up to be the monitored version of the well-known transverse-field Ising model (TFIM).

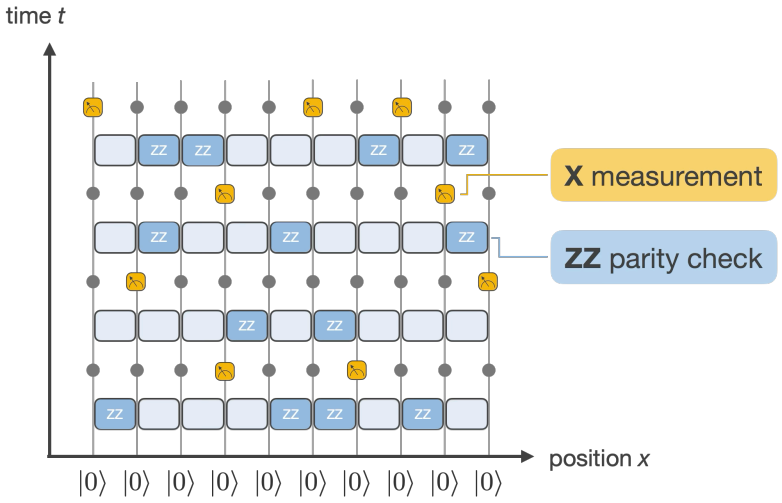

Figure 1 – Visualization of four layers of the monitored transverse-field Ising model circuit. Note that the circuit should have a last layer of ZZ parity measurements at the end (which is not shown here).

The circuit construction for this model can be broken down into the following steps:

1. Start in the $+1$ eigenstate of the X operator on L qubits, i.e., $|++\cdots+\rangle$.
2. With probability $1-p$, apply ZZ parity measurements between neighboring qubits (consider periodic boundary conditions).
3. With probability p, apply single qubit X basis measurements to each qubit.
4. Repeat from step 2 for a sufficiently deep circuit, e.g., $\mathcal{O}(2000)$ times.

5. End the circuit with a last layer of step 2.

To study the entanglement of the model, we want to again calculate the entanglement entropy S between different sized subsystems and the rest of the system.

a) Implement the circuit of the monitored transverse-field Ising model using `QuantumClifford.jl`

b) Simulate the circuit with $L = 50$ qubits for different values of $0 \leq p \leq 1$ and compute the entanglement entropy $S(L/2)$ between two equal sized subsystems, i.e.:

julia
```
S_half = entanglement_entropy(state, 1:L÷2, Val(:rref))
```

Plot the entanglement entropy $S(L/2)$ against the measurement probability p. Can you see a phase transition? What is the critical measurement probabtility p_c?

Hint: To get a clean plot, it is necessary to average the calculated entanglement entropy $S(L/2)$ over multiple, e.g., $\mathcal{O}(1000)$, circuit realizations at the same measurement probability p.

c) Simulate the circuit for $L = 50$ at $p = p_c$ as well as some more p values above and below p_c and again calculate the entanglement entropy $S(L/2)$. Furthermore, also compute the entanglement entropy $S(l)$ for different subsystem sizes $0 \leq l \leq L$, i.e.:

julia
```
for (i, l) in enumerate(0:n:L)
    S[i] = entanglement_entropy(state, 1:l, Val(:rref))
end
```

the step size n can be chosen larger than 1 to reduce the computation time. Plot the entanglement entropy $S(l)$ against the subsystem size l. Can you tell which entanglement arc coresponds to the critical measurement probability p_c? For the other values of p, what does the entanglement arc tell you about the nature of the entanglement inside the two phases?

Hint: Like in part b), to get a clean plot, it is necessary to average the calculated entanglement entropy S over multiple, e.g. $\mathcal{O}(10,000)$, circuit realizations.

d) For $p = p_c$ use the data from part c), to plot the relative entanglement entropy $\Delta S(l) = S(L/2) - S(l)$ against the subsystem size l. Confirm that the dynamics of the model can be described by a non-unitary conformal field theory (see lecture), by fitting the Cardy-Calabrese entanglement arc

$$\Delta S(l'/L) = \frac{c}{3} \log_2 \left(\sin \left(\frac{\pi l'}{L} \right) \right) \tag{1}$$

for $0 < l' < L/2$ and comparing the central charge-like prefactor c to the predicted value of $c = \frac{3\sqrt{3}}{2\pi} \ln 2 \approx 0.573$.

To get even nicer results, you could increase the system size to, e.g., $L = 500$ qubits or improve the

statistics by increasing the averaging steps. However, you will notice that this becomes computationally heavy very fast and is most likely not suited for a laptop but a compute cluster, where averaging steps can be parallelized over hundreds to thousands of CPUs.

Exercise 15: The monitored Kitaev honeycomb model

Another example of a measurement-only circuit is the monitored Kitaev model. Constructing the circuit for this model is somewhat more complicated, since the qubit geometry is not one but two-dimensional, or more concretely, a honeycomb lattice. This means that every qubit has three (compared to two in the one-dimensional case) neighbors. The Kitaev model is characterized by the fact that the bonds of the honeycomb lattice get three colored, such that all bonds in the lattice that are connected to the same qubit have different colors. The Kitaev coloring is not the only such coloring, but arguably the geometrically simplest. For the monitored Kitaev model, the coloring is relevant because the type of parity measurement applied to neighboring qubits depends on the type (color) of the bond between them.

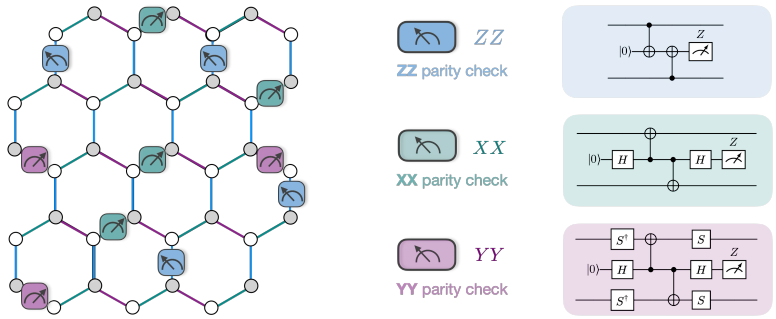

Figure 2 – Visualization of the monitored Kitaev honeycomb model geometry.

To simplify the implementation of the monitored Kitaev model circuit, you can use a Julia package called `MonitoredQuantumCircuits.jl`, which is however not in the Julia registry and thus has to be installed from the GitHub repository:

julia
```
] add https://github.com/J-C-Q/MonitoredQuantumCircuits.jl.git
```

Now using `MonitoredQuantumCircuits.jl` constructing the circuit can be done as follows:

1. Create a qubit geometry, where the qubits are the sites of a periodic (torus) honeycomb lattice with a size of $L \times L$ hexagons, i.e., $2L^2$ qubits.

   julia
   ```
   geometry = PeriodicHoneycombGeometry(L, L)
   ```

2. Initialize a circuit data structure to store information about the gates to apply.

   julia
   ```
   circuit = Circuit(geometry)
   ```

3. Pick a random qubit index out of the geometry.

```julia
qubit = random_qubit(geometry)
```

4. Pick a bond type (X, Y or Z) with the given probabilities p_x, p_y and p_z

5. Depending on the type of bond chosen, find the neighbor of the randomly picked qubit. For example, when the X type bond was chosen:

```julia
neighbor = kitaevX_neighbor(geometry, qubit)
```

6. Apply a parity measurement between the qubits with the same basis as the type of bond picked. For example, for an X type bond:

```julia
apply!(circuit, XX(), qubit, neighbor)
```

7. Repeat from step 3 for a sufficiently deep circuit. One time step consists of $2L^2$ measurements.

8. Simulate the circuit using `QuantumClifford.jl`.

```julia
result = execute(circuit, QuantumClifford.TableauSimulator(nQubits(circuit)))
```

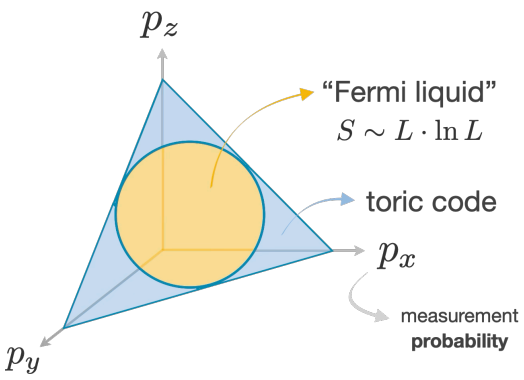

Figure 3 – Visualization of the monitored Kitaev honeycomb model phase diagram.

a) For a system size of $L = 12$, compare the purification of the initially mixed state under the circuit dynamics in different parts of the phase diagram. Consider the four relative bond type measurement probabilities

1. $p_x = p_y = p_z = 1/3$ (Fermi liquid phase)
2. $p_x = 0.652$, $p_y = p_z = 0.174$ (phase transition)
3. $p_x = 0.8$, $p_y = p_z = 0.1$ (toric code phase)
4. $p_x = p_y = 0.5$, $p_z = 0$ (percolation)

Calculate the relative state entropy S after different depths:

```julia
S = QuantumClifford.state_entropy(result)
```

Hint: To continue the simulation, you can pass an initial state to the `TableauSimulator`

```julia
result = execute(circuit, QuantumClifford.TableauSimulator(last_result))
```

b) Plot the state entropy against the circuit depth. Can you see differences in the purification time?

To start the circuit in a pure state, one needs to find $2L^2$ independent commuting stabilizers. One way of doing this is to perform a Z basis parity measurement on all Z type bonds, followed by six qubit parity measurements of every hexagon plaquette. This leaves two stabilizers. To complete the pure state, one can perform a $2L$ qubit parity measurement along two periodic cycles around the torus. `MonitoredQuantumCircuits.jl` provides convenient functions to construct these measurements.

1. Get all Z type bonds (as qubit index tuples)

   ```julia
   z_bonds = kitaevZ(geometry)
   ```

   and apply a Z basis parity measurement between the qubits in every Z type bond.

2. Get all hexagon plaquettes (as lists of qubit indices)

   ```julia
   hexagons = plaquettes(geometry)
   ```

   and apply a six qubit `nPauli` measurement

   ```julia
   nPauli(Y(),X(),Z(),Y(),X(),Z())
   ```

3. Get the two long cycles (as lists of qubit indices)

julia

```
loops = long_cycles(geometry)
```

and apply a $2L$ qubit `nPauli` measurement in the Z basis for the first loop

julia

```
nPauli(fill(Z(), length(loops[1]))...)
```

and the Y basis for the second loop

julia

```
nPauli(fill(Y(), length(loops[2]))...)
```

c) Start the circuit in a pure state by measuring the parity measurements mentioned above. Then continue the circuit as before. Simulate the four points from part a) again and calculate the entanglement entropy $S(l)$ for different sized subsystems.

julia

```
for (i,l) in enumerate(0:L)
    S[i] = QuantumClifford.entanglement_entropy(result, 1:(2L*l))
end
```

d) Plot the entanglement arcs generated in part c) against the subsystem size. Can you see the different phases in the entanglement arcs?