Homework 3

Quantum Games in QuNetSim

Necessary reading: Chapters 1 and 2
In this exercise, we will implement a quantum game in QuNetSim, specifically the Mermin-Ardehali game. The aim of this exercise is to have an introduction to QuNetSim as well as understand quantum games more deeply. To begin, we point you to the QuNetSim documentation to install it:

- https://tqsd.github.io/QuNetSim/

For more information regarding QuNetSim, you can see our paper:

- https://arxiv.org/abs/2003.06397

The Mermin-Ardehali Game

The Mermin-Ardehali Game is an "XOR" game where the quantum strategy outperforms any classical strategy. The game works as follows:

1. There is a single referee and N players in the game
2. The referee sends a single (uniform) random bit to each player $x_1, ...x_N \in \{0, 1\}$ and decides the winning condition W where
   - $W = 0$ if $\sum_{i=1}^{N} x_i \mod 4 \in \{0, 1\}$
   - $W = 1$ of $\sum_{i=1}^{N} x_i \mod 4 \in \{2, 3\}$
3. The players receive their x_i and each send the referee a bit $a_i \in \{0, 1\}$ back.
4. The game is won if $W = \bigoplus_{i=1}^{N} a_i$
5. The players cannot communicate amongst themselves during the game, but before the game, in the quantum case, they can share quantum entanglement in the form of a GHZ state with N qubits and decide on a strategy.

In the quantum case, the best chance of winning is $\frac{1}{2} + \frac{1}{2\sqrt{2}}$ where on the other hand, the best possible classical strategy can achieve a winning probability of $\frac{1}{2} + \frac{1}{2^{\lceil (N+1)/2 \rceil}}$. In this exercise we will test these bounds by simulating the strategies.

The Classical Strategy

For the classical strategy, the best chance of winning for the case of $N = 8$ which is the number of players we deal with in this exercise, is to simply ignore the input and respond with $a_i = 0$.

Homework 3

The Quantum Strategy

With $\gamma \coloneqq \frac{(2N+1) \mod 8}{4N} \cdot \pi$ and shared GHZ state $|\Psi\rangle = \frac{1}{\sqrt{2}}(|0\rangle^{\otimes N} + |1\rangle^{\otimes N})$, the optimal strategy is then achieved when each player applies

$$\frac{1}{\sqrt{2}} \begin{pmatrix} e^{i(\frac{\pi}{2}+\gamma)} & 1 \\ -1 & e^{-i(\frac{\pi}{2}+\gamma)} \end{pmatrix} \text{ if } x_i = 0 \text{ and } \frac{1}{\sqrt{2}} \begin{pmatrix} e^{i\gamma} & 1 \\ -1 & e^{-i\gamma} \end{pmatrix} \text{ if } x_i = 1$$

to their part of the GHZ state. They then measure their part of the GHZ state in the computation basis and send the outcome a_i to the referee.

Exercise 1

For this homework exercise, the goal is to set up QuNetSim and implement the Mermin-Ardehali game. We will provide you with the template of the file and your job will be to fill in the missing pieces of the code and then collect data to verify the optimal strategies meet roughly the theoretical limit.

Exercise 2

The qubit simulator used in QuNetSim, EQSN, allows for custom unitaries to be applied to qubits. In some qubit simulators, one only has access to standard gates and one would need to decompose a custom unitary using rotation operators. The unitaries in the quantum strategy can be written in a combination of the rotations unitaries below

$$R_z(\theta) \coloneqq \begin{pmatrix} e^{-i\frac{\theta}{2}} & 0 \\ 0 & e^{i\frac{\theta}{2}} \end{pmatrix}$$

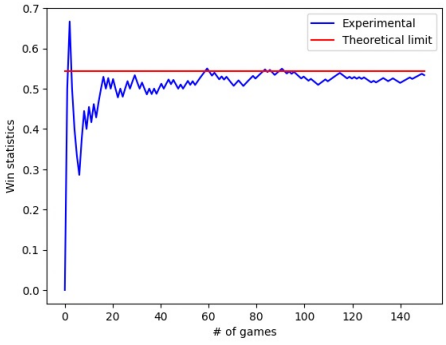
$$R_y(\theta) \coloneqq \begin{pmatrix} \cos(\theta/2) & -\sin(\theta/2) \\ \sin(\theta/2) & \cos(\theta/2) \end{pmatrix}$$

$$R_x(\theta) \coloneqq \begin{pmatrix} \cos(\theta/2) & -i\sin(\theta/2) \\ -i\sin(\theta/2) & \cos(\theta/2) \end{pmatrix}$$

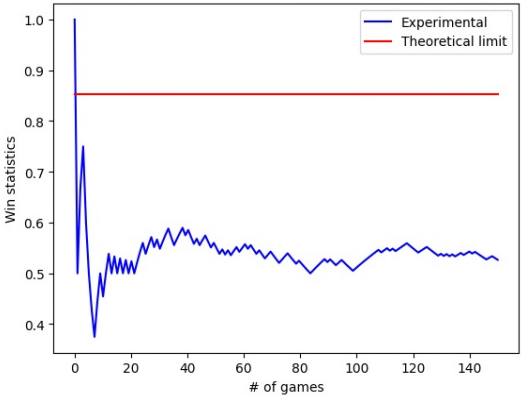
Write the two unitaries in the quantum strategy above as a series of the rotation unitaries. Run the simulation again using the modified unitary and verify the results are similar.

Ex 1 :

classical strategy result :



Quantum strategy result :



Ex 2 :

Decomposing the custom unitary with the help of the following script:

```python
import numpy as np
from qiskit import QuantumCircuit

n=8
angle = ((2*n + 1) % 8)/4/n*np.pi
a = np.exp(1j*angle)
U = (1 / np.sqrt(2)) * np.array([[a, 1], [-1, 1 / a]])
qc = QuantumCircuit(1)
qc.unitary(U, qubits = qc.qubits, label = "U")

target_basis = ['rx', 'ry', 'rz']
decomposed = transpile(qc,
                       basis_gates=target_basis,
                       optimization_level=0)

for gate in decomposed.data:
    print('\ngate name:', gate[0].name)
    print('qubit(s) acted on:', gate[1])
    print('other paramters (such as angles):', gate[0].params)

decomposed.draw()
```

[22] ✓ 0.0s Python

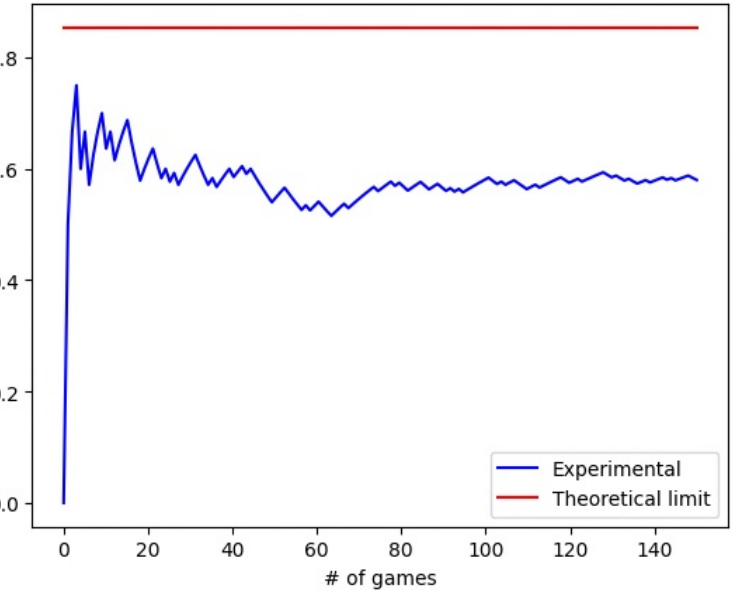
```
gate name: rz
qubit(s) acted on: [Qubit(QuantumRegister(1, 'q'), 0)]
other paramters (such as angles): [3.0434178831651124]

gate name: ry
qubit(s) acted on: [Qubit(QuantumRegister(1, 'q'), 0)]
other paramters (such as angles): [1.5707963267948966]

gate name: rz
qubit(s) acted on: [Qubit(QuantumRegister(1, 'q'), 0)]
other paramters (such as angles): [3.0434178831651124]

global phase: π
q: ─ Rz(3.0434) ─ Ry(π/2) ─ Rz(3.0434) ─
```

Results of the quantum strategy without unitary decomposition:



See Code implementation in the attached python script