notebook

December 5, 2024

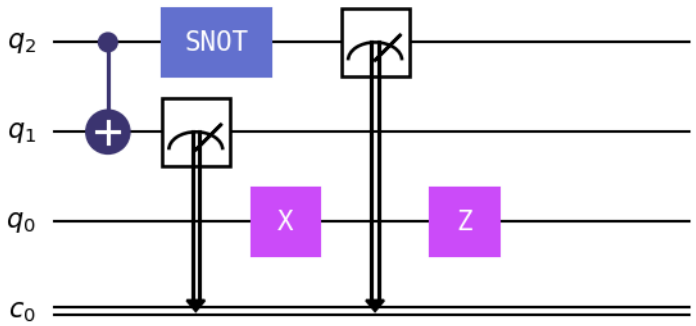
[1]:
```
!pip install qutip-qip
```

```
Requirement already satisfied: qutip-qip in
/home/nathan/anaconda3/lib/python3.9/site-packages (0.4.0)
Requirement already satisfied: qutip>=4.6 in
/home/nathan/anaconda3/lib/python3.9/site-packages (from qutip-qip) (5.0.4)
Requirement already satisfied: numpy>=1.16.6 in
/home/nathan/anaconda3/lib/python3.9/site-packages (from qutip-qip) (1.26.4)
Requirement already satisfied: scipy>=1.0 in
/home/nathan/anaconda3/lib/python3.9/site-packages (from qutip-qip) (1.13.1)
Requirement already satisfied: packaging in
/home/nathan/anaconda3/lib/python3.9/site-packages (from qutip-qip) (21.0)
Requirement already satisfied: pyparsing>=2.0.2 in
/home/nathan/anaconda3/lib/python3.9/site-packages (from packaging->qutip-qip)
(3.0.4)
```

[2]:
```
import qutip as qt
from qutip_qip.circuit import QubitCircuit
import matplotlib.pyplot as plt
import numpy as np
```

[7]:
```
# Problem 2(b)
qc = QubitCircuit(3, num_cbits=1)
qc.add_gate("CNOT", controls=2, targets=1)
qc.add_gate("SNOT", targets=2)
qc.add_measurement("M0", targets=1, classical_store=0)
qc.add_gate("X", targets=0, classical_controls=0)
qc.add_measurement("M0", targets=2, classical_store=0)
qc.add_gate("Z", targets=0, classical_controls=0)

# This throws an error if I try to draw in the LaTeX style (which
# is supposed to look a lot nicer than the matplotlib style)
qc.draw("matplotlib", bulge=False)

for i in range(100):
    # I didn't wanna do symbolic computation with alpha and beta,
    # so instead, just check that teleportation works for a ton
    # of randomly generated qubits (this is super janky)
```

```
    Psi = (np.random.uniform(-10, 10) + np.random.uniform(-10, 10) * 1j) * qt.
↪ket("0") + \
        + (np.random.uniform(-10, 10) + np.random.uniform(-10, 10) * 1j) * qt.
↪ket("1")
    Psi = Psi.unit()
    initial_state = qt.tensor(Psi, qt.bell_state("00"))
    final_state = qc.run(initial_state)
    teleported_state = final_state.ptrace(0)
    # Using ptrace turns the state into a density matrix, so now I
    # need to compare it to the density matrix of the original state
    assert teleported_state == Psi * Psi.dag()

print("It doesn't show up here, but the magenta X and Z gates are both " \
      "controlled by the c_0 register.\nq_2 is the qubit to be teleported, " \
      "and q_0 \otimes q_1 is the bell state \Phi^+")
```

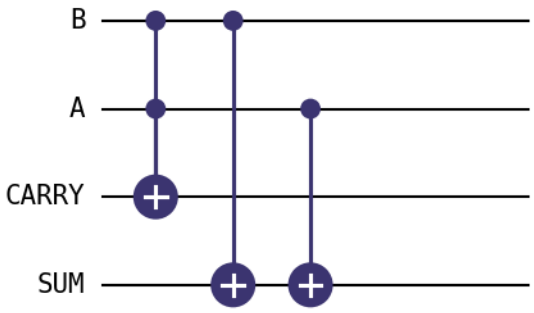



```
It doesn't show up here, but the magenta X and Z gates are both controlled by
the c_0 register.
q_2 is the qubit to be teleported, and q_0 \otimes q_1 is the bell state \Phi^+
```

[4]:
```
# Problem 3
from qutip_qip.operations import Measurement

qc = QubitCircuit(4)
qc.add_gate("TOFFOLI", controls=[3, 2], targets=[1])
qc.add_gate("CNOT", controls=[3], targets=[0])
qc.add_gate("CNOT", controls=[2], targets=[0])

for input_bits in ["00", "01", "10", "11"]:
    initial_state = qt.ket("00" + input_bits)
    final_state = qc.run(initial_state)
```

```
    sum_bit   = Measurement("sum",   targets=[0]).
↪measurement_comp_basis(final_state)[1][1]
    carry_bit = Measurement("carry", targets=[1]).
↪measurement_comp_basis(final_state)[1][1]
    A = input_bits[0]
    B = input_bits[1]
    print(f"{A} + {B} = {int(carry_bit)}{int(sum_bit)}")

qc.draw("matplotlib", bulge=False, wire_label=["SUM", "CARRY", "A", "B"])
```

```
0 + 0 = 00
0 + 1 = 01
1 + 0 = 01
1 + 1 = 10
```



[ ]: