Multi qubit controlled rotations

January 7, 2022

[1]:
```
import qiskit as qk
from qiskit import QuantumCircuit, QuantumRegister, Aer, IBMQ
from qiskit import transpile, assemble
from qiskit.visualization import plot_histogram
import math
import numpy as np
from qiskit.circuit import Parameter, ParameterVector
from qiskit.tools.visualization import circuit_drawer
from qiskit.quantum_info import state_fidelity
from math import pi
from qiskit.visualization import array_to_latex
```

[2]:
```
def ckry(k, theta):
    circ = QuantumCircuit()
    qr = QuantumRegister(k+1,'q')   # k cotrolled qubits + one target qubit
    qra = QuantumRegister(k-2,'a') # ancillary qubits
    circ.add_register( qr )        # appending circuits
    circ.add_register( qra )       # appending circuits
    # the below 5 commands implements C^2(U) with controlled qubits q_k-1, a_k-3␣
 ↪and q_k as the target qubit
    circ.cry(theta,qr[k-1],qr[k])
    circ.cx(qr[k-1],qra[k-3])
    circ.cry(-theta,qra[k-3],qr[k])
    circ.cx(qr[k-1],qra[k-3])
    circ.cry(theta,qra[k-3],qr[k])
    for t in range(k-3):
        circ.ccx(qr[k-2-t], qra[k-4-t], qra[k-3-t])
    circ.ccx(qr[0], qr[1], qra[0])
    for t in reversed(range(k-3)):
        circ.ccx(qr[k-2-t], qra[k-4-t], qra[k-3-t])
    # the below 5 commands implements C^2(U) with controlled qubits q_k-1, a_k-3␣
 ↪and q_k as the target qubit
    circ.cry(theta,qr[k-1],qr[k])
    circ.cx(qr[k-1],qra[k-3])
    circ.cry(-theta,qra[k-3],qr[k])
    circ.cx(qr[k-1],qra[k-3])
    circ.cry(theta,qra[k-3],qr[k])
```

```
    # the below two for loops set the ancillary qubits to initialized states
    for t in range(k-3):
        circ.ccx(qr[k-2-t], qra[k-4-t], qra[k-3-t])
    circ.ccx(qr[0], qr[1], qra[0])
    for t in reversed(range(k-3)):
        circ.ccx(qr[k-2-t], qra[k-4-t], qra[k-3-t])
    return circ
```

[3]:
```
n = 4
qc = QuantumCircuit()
qr =QuantumRegister(n+1,'q')
qra = QuantumRegister(n-2,'a')
qc.add_register( qr )
qc.add_register( qra )
for i in range(n):
    qc.h(qr[i])
qc.barrier()
for i in range(n):
    qc.x(qr[i])
for m in range(16):
    var = math.sqrt(m/15)
    theta = math.asin(var)
    temp = format(m,'b')
    p=m.bit_length()
    if p == 0:
        qc.compose(ckry(n,theta), inplace=True)
        qc.barrier()
    else:
        for t in range(p):
            if temp[t] == '1':
                qc.x(p-t-1)
        qc.compose(ckry(n,theta), inplace=True)
        for t in range(p):
            if temp[t] == '1':
                qc.x(p-t-1)
        qc.barrier()
for i in range(n):
    qc.x(qr[i])
#qc.draw()
```

[4]:
```
from qiskit.quantum_info import Statevector
state = Statevector.from_int(0, 2**7)
state = state.evolve(qc)
#draw using latex
state.draw('latex')
```

[4]:

$$\frac{1}{4}|0000000\rangle + \frac{\sqrt{210}}{60}|0000001\rangle + \frac{\sqrt{195}}{60}|0000010\rangle + \frac{\sqrt{5}}{10}|0000011\rangle + \frac{\sqrt{165}}{60}|0000100\rangle + \frac{\sqrt{6}}{12}|0000101\rangle + \ldots + \frac{\sqrt{165}}{60}|0$$

[5]:
```
from qiskit.visualization import array_to_latex
#Alternative way of reprefrom qiskit.quantum_info import Statevector
state = Statevector.from_int(0, 2**7)
state = state.evolve(qc)
#draw using latex
#state.draw('latex')senting in latex
array_to_latex(state)
```

[5]:
$$\begin{bmatrix} \frac{1}{4} & 0.24152 & 0.23274 & 0.22361 & \cdots & 0 & 0 & 0 \end{bmatrix}$$

[6]:
```
qc.measure_all()
aer_sim = Aer.get_backend('aer_simulator')
t_qc = transpile(qc, aer_sim)
qobj = assemble(t_qc, shots=8192)
result = aer_sim.run(qobj).result()
counts = result.get_counts(qc)
print(counts)
plot_histogram(counts)
```

```
{'0000100': 420, '0001101': 58, '0010101': 165, '0011100': 395, '0011111': 488,
'0001010': 166, '0011101': 431, '0000000': 531, '0010111': 236, '0000001': 490,
'0010110': 216, '0011110': 509, '0011010': 340, '0000011': 427, '0000111': 281,
'0011011': 388, '0000010': 409, '0011001': 308, '0001011': 148, '0000101': 331,
'0001000': 233, '0000110': 265, '0001100': 99, '0001110': 36, '0010010': 73,
'0011000': 246, '0001001': 225, '0010100': 145, '0010011': 96, '0010001': 37}
```