Week 9

Lecture 17 – Introduction to IBM Quantum Experience

Introduction to IBM Quantum Experience: Guest Lecture

Lecture 18 – IBM and Optimisations

14.1 QUI compared to IBM

14.2 QASM and QISKit

14.3 Optimizing circuits

Lab 9

Using the IBM system

1

PHYC90045 Introduction to Quantum Computing

IBM and Optimisation

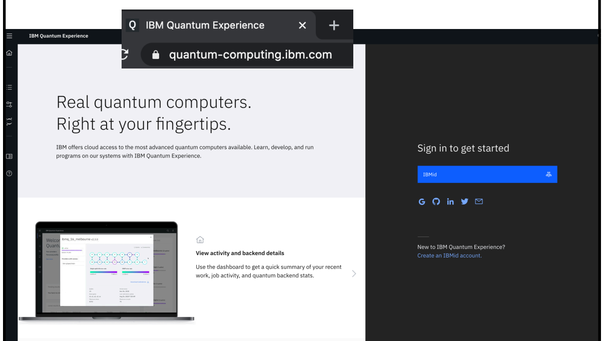
Physics 90045

Lecture 18

2

PHYC90045 Introduction to Quantum Computing

The IBM Quantum Computing System

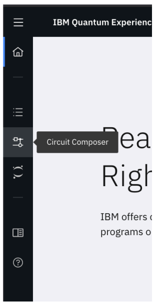


Sign up using your university email before Thursday!

3

THE UNIVERSITY OF MELBOURNE

IBM's Circuit Composer



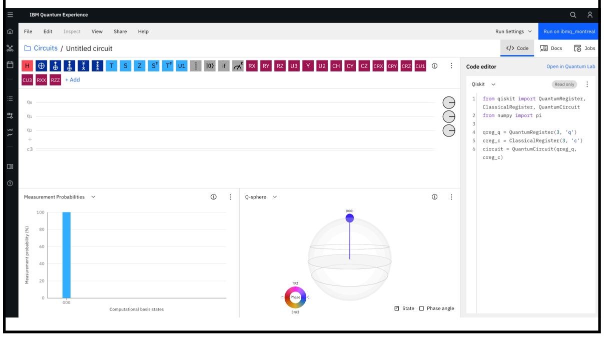
Access through the left sidebar

4

THE UNIVERSITY OF MELBOURNE

Creating a Bell State Circuit

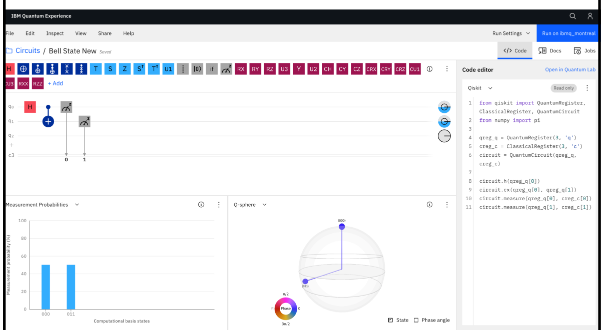
Drag and drop gates onto your circuit:



5

THE UNIVERSITY OF MELBOURNE

Bell State



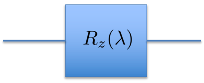
6

U1

U1 is a rotation around Z by angle lambda, which is equivalent to a rotation around the z-axis by an angle lambda

$$U_1 = \begin{bmatrix} 1 & 0 \\ 0 & \exp i\lambda \end{bmatrix}$$

Most easily understood as:



In the QUI, to emultate these z-rotations, use a global phase of lambda/2.
No global phase for the y-rotation.

7

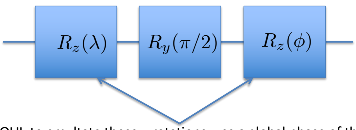
PHYC90045 Introduction to Quantum Computing

U2

The U2 operation is given by

$$U_2 = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & -\exp(i\lambda) \\ \exp(i\phi) & \exp(i\lambda + i\phi) \end{bmatrix}$$

Which can be represented as:



In the QUI, to emultate these z-rotations, use a global phase of theta/2.
No global phase for the y-rotation.

8

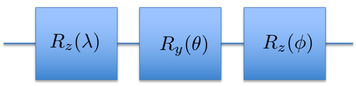
PHYC90045 Introduction to Quantum Computing

U3

The matrix of a U3 rotation is:

$$U_3 = \frac{1}{\sqrt{2}} \begin{bmatrix} \cos\theta/2 & -\exp(i\lambda)\sin(\theta/2) \\ \exp(i\phi)\sin(\theta/2) & \exp(i\lambda + i\phi)\cos(\theta/2) \end{bmatrix}$$

As a circuit:



9

Euler Angle Decomposition

THE UNIVERSITY OF MELBOURNE

Any rotation can be represented as a rotation around orthogonal axes:

$R_n(\alpha)$ = $R_z(\lambda)$ $R_y(\theta)$ $R_z(\phi)$

QUI IBM Quantum Experience

10

Converting to and from Euler angles

THE UNIVERSITY OF MELBOURNE

General form of arbitrary rotation about an unit axis n=(n_x, n_y, n_z):

$$R_n(\alpha) = \cos\frac{\alpha}{2} I - i\sin\frac{\alpha}{2}\hat{n}\cdot\sigma$$

$$= \left[\begin{array}{cc} \cos\frac{\alpha}{2} - in_z\sin\frac{\alpha}{2} & \sin\frac{\alpha}{2}(-in_x - n_y) \\ \sin\frac{\alpha}{2}(-in_x + n_y) & \cos\frac{\alpha}{2} + in_z\sin\frac{\alpha}{2} \end{array}\right]$$

Euler angle rotations (with global phase = 0):

$$U_3 = \left[\begin{array}{cc} e^{-i(\lambda+\phi)/2}\cos(\theta/2) & -e^{i(\lambda-\phi)/2}\sin(\theta/2) \\ e^{i(-\lambda+\phi)/2}\sin(\theta/2) & e^{i(\lambda+\phi)/2}\cos(\theta/2) \end{array}\right]$$

Write out the matrix (with determinant 1) and equate elements.

11

Equating angles

THE UNIVERSITY OF MELBOURNE

$$\cos\frac{\alpha}{2} = \cos\frac{\lambda+\phi}{2}\cos\frac{\theta}{2}$$

$$\sin\left(\frac{\alpha}{2}\right)n_x = \sin\frac{\lambda-\phi}{2}\sin\frac{\theta}{2}$$

$$\sin\left(\frac{\alpha}{2}\right)n_y = \cos\frac{\lambda-\phi}{2}\sin\frac{\theta}{2}$$

$$\sin\left(\frac{\alpha}{2}\right)n_z = \sin\frac{\lambda+\phi}{2}\cos\frac{\theta}{2}$$

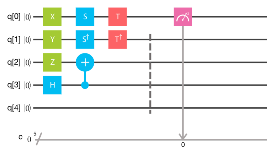
12

PHYC90045 Introduction to Quantum Computing

QASM – Quantum Assembly language

```
include "qelib1.inc";
qreg q[5];
creg c[5];

x q[0];
y q[1];
z q[2];
h q[3];

s q[0];
sdg q[1];

cx q[3],q[2];
t q[0];
tdg q[1];

barrier q[1],q[2],q[3],q[4];
measure q[0] -> c[0];
```

Import QASM   Download QASM



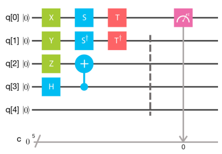
13

PHYC90045 Introduction to Quantum Computing

QASM Syntax

Semi-Colons

```
include "qelib1.inc";

// This is a comment
qreg q[5];
creg c[5];

x q[0];
y q[1];
z q[2];
h q[3];

s q[0];
sdg q[1];

cx q[3],q[2];
t q[0];
tdg q[1];

barrier q[1],q[2],q[3],q[4];
measure q[0] -> c[0];
```



Comment

14

PHYC90045 Introduction to Quantum Computing

QASM

Hidden first line: **OPENQASM 2.0;**

```
include "qelib1.inc";
qreg q[5];
creg c[5];

x q[0];
y q[1];
z q[2];
h q[3];

s q[0];
sdg q[1];

cx q[3],q[2];
t q[0];
tdg q[1];

barrier q[1],q[2],q[3],q[4];
measure q[0] -> c[0];
```

Include standard definitions

Declare quantum register

Declare classical register

Single qubit gates

CNOT gate
(control first parameter, target second)

Dagger indicated by "dg"

Barrier (don't optimize across it)

Measure qubits to classical register

15

PHYC90045 Introduction to Quantum Computing

QISKit

There is also a Python interface to IBM Quantum Experience.

It is required to make use of the larger machines.

You can:

- Authenticate with the system
- Construct circuits (ie. python which translates to QASM)
- Submit jobs, and check for results
- Receive the results of jobs

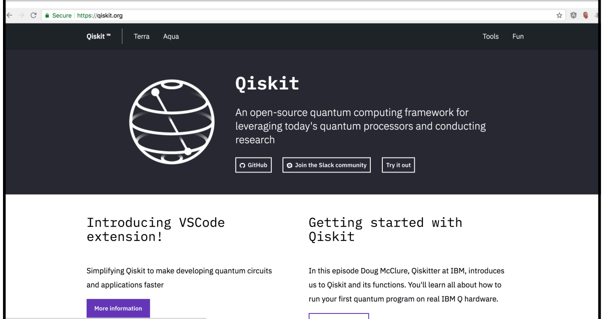
Python works well with Jupyter interface.

We will use this later when we use the 16 qubit quantum computer.

16

PHYC90045 Introduction to Quantum Computing

QISKit



Lots of examples in the github repository.

17

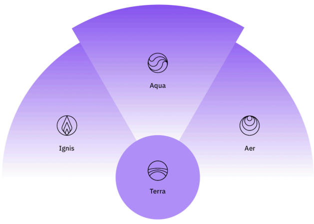
PHYC90045 Introduction to Quantum Computing

IBM's Qiskit

18

PHYC90045 Introduction to Quantum Computing

Terra, Aer, Ignis, Aqua

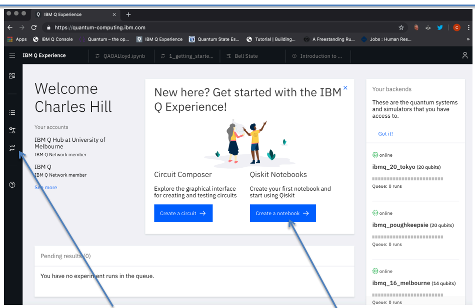


Tera (Earth): Access to IBM Q Devices through python interface
Aer (Air): Classical simulation of quantum algorithms/circuits
Ignis (Fire): Characterisation of errors, tomography
Aqua (Water): Large selection of quantum algorithms

19

PHYC90045 Introduction to Quantum Computing

Starting a new Workbook



To see your existing notebooks, tutorials or start a new one. Select "QISKIT Notebooks"

Or click on the button to create a new notebook

20

PHYC90045 Introduction to Quantum Computing

We will loosely follow the "Getting Started with Qiskit" tutorial from IBM,

Qiskit

Getting Started with Qiskit

Here, we provide an overview of working with Qiskit. Qiskit provides the basic building blocks necessary to program quantum computers. The fundamental unit of Qiskit is the **quantum circuit**. A workflow using Qiskit consists of two stages: **Build** and **Execute**. **Build** allows you to make different quantum circuits that represent the problem you are solving, and **Execute** allows you to run them on different backends. After the jobs have been run, the data is collected. There are methods for putting this data together, depending on the program. This either gives you the answer you wanted, or allows you to make a better program for the next instance.

In [1]:
```
import numpy as np
from qiskit import *
%matplotlib inline
```

21

Python Primer (if required)

22

Some Python Basics

```
In [2]: a=6
        b=7
        life = a*b
        life

Out[2]: 42
```

Similar to many other imperative languages you may know for numerical work: (C/C++, MATLAB, R, FORTRAN, Julia) and often used for data processing.

23

Defining and calling Functions

def keyword indicates a new function

No types on parameters

Colon

Comment

```
def square(x):
    # This is a comment
    return x*x
```

Whitespace is significant in python. Indentation indicates a new block.

No semicolons. Newline is the end of a statement

Calling a function:

```
square(4)
square(x=4)
```

Named parameters

24

Lists and for loops

Lists store a sequence of values. Square brackets indicate a list:

```
["This", "is", "a", "list"]

primes = [2, 3, 5, 7, 11]
```

Eg. For loops often use lists:

```
for p in primes:
    print(p)
```

```
2
3
5
7
11
```

Accessing an individual element. 0-based!

```
primes[2]
```

```
5
```

25

Dictionaries

Dictionaries store key-value pairs.

Curly braces indicate a dictionary

key

value

```
me = {"name": "Charles", "height":1.79, "favourite_food": "pizza"}
me["favourite_food"]
```

```
'pizza'
```

```
me["favourite_food"] = "sweet and sour pork"
me["favourite_food"]
```

```
'sweet and sour pork'
```

26

Importing other libraries

```
import numpy as np

X = np.matrix([[0,1],
               [1,0]])
```

Importing a module ("as np" is optional). numpy gives similar functionality to MATLAB

Calling functions from that module. Here creating an X matrix.

Or import individual functions and classes:

```
from qiskit import QuantumProgram
from qiskit import available_backends, execute, get_backend, compile
from qiskit import QuantumCircuit, ClassicalRegister, QuantumRegister, QISKitError
```

qiskit is an Python library/API for interacting with IBM's quantum computers remotely.

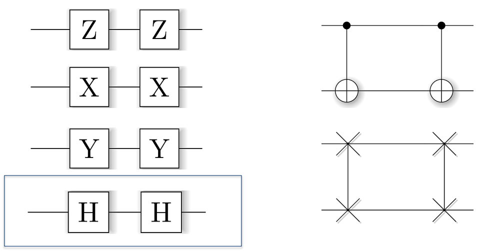
27

Circuit Optimisation

28

Many gates square to identity

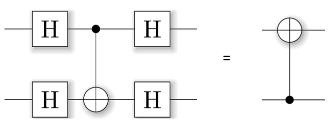
Pro Tip: Most physicists looking at quantum circuit diagrams aren't multiplying matrices in their head. They're identifying common patterns.



All of these combinations square to the identity (do nothing)

29

Circuit identity: Inverted CNOT

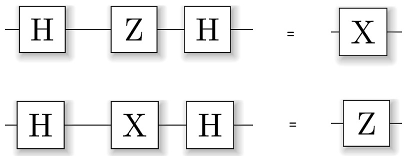


Exercise: You can verify this by writing out the matrices and multiplying!

30

PHYC90045 Introduction to Quantum Computing

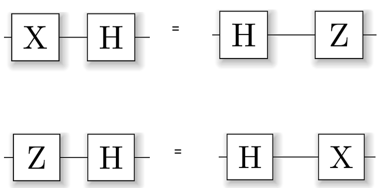
Conjugating with Hadamard



31

PHYC90045 Introduction to Quantum Computing

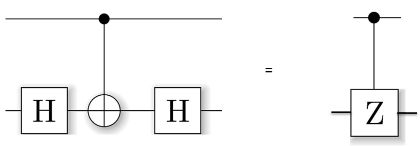
Commuting through Hadamard



32

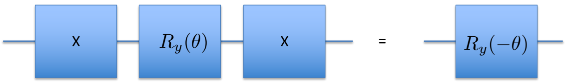
PHYC90045 Introduction to Quantum Computing

Control-Z from CNOT



33

Conjugation with Pauli



$$
\begin{aligned}
XR_y(\theta)X &= X\left(\cos\frac{\theta}{2}I - i\sin\frac{\theta}{2}Y\right)X \\
&= \cos\frac{\theta}{2}XX - i\sin\frac{\theta}{2}XYX \\
&= \cos\frac{\theta}{2}I + i\sin\frac{\theta}{2}Y \\
&= R_y(-\theta)
\end{aligned}
$$

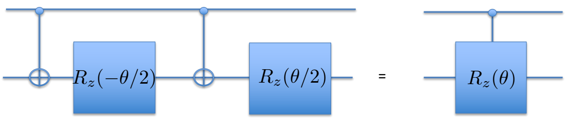
Paulis **anticommute**
XY = -YX

Works with any two orthogonal axes.

34

PHYC90045 Introduction to Quantum Computing

Controlled Angle Rotation



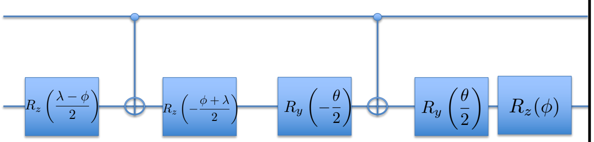
If the control is a zero, the rotations cancel.
If the control is one, the rotations add.

35

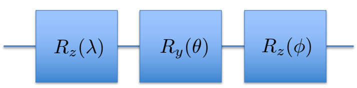
PHYC90045 Introduction to Quantum Computing

Any Controlled U

For U3 Euler angle rotation (on IBM's system):



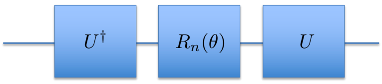
Controlled version of a U3 gate:



36

Conjugation with Rotation

Conjugation with a rotation:



Changes the axis of rotation, but not the rotation angle.

This rotates the axis itself

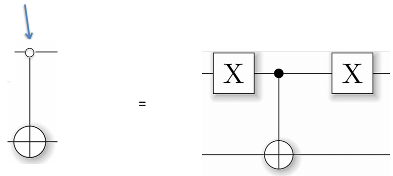
$$
\begin{aligned}
SR_x(\theta)S^\dagger &= \cos\left(\frac{\theta}{2}\right) I - i\sin\left(\frac{\theta}{2}\right) SXS^\dagger \\
&= \cos\left(\frac{\theta}{2}\right) I - i\sin\left(\frac{\theta}{2}\right) Y \\
&= R_y(\theta)
\end{aligned}
$$

Conjugation with Hadamard is a special case of this.

37

Control from “0” state

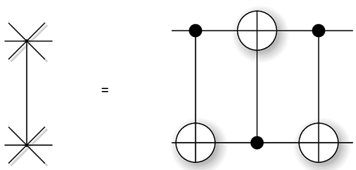
Open circle =
Only apply
when the
control is “0”



We’ve seen this trick in labs: for example in the oracle for Grover’s algorithm.

38

Swap gate from three CNOTs



Let’s check:

00 -> 00
01 -> 10
10 -> 01
11 -> 11

39

PHYC90045 Introduction to Quantum Computing

Square root of SWAP

Swap

$$U_{Swap} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

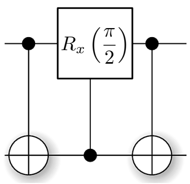
Square root of swap

$$U_{SS} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \frac{1+i}{2} & \frac{1-i}{2} & 0 \\ 0 & \frac{1-i}{2} & \frac{1+i}{2} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

40

PHYC90045 Introduction to Quantum Computing

Square Root Swap Construction

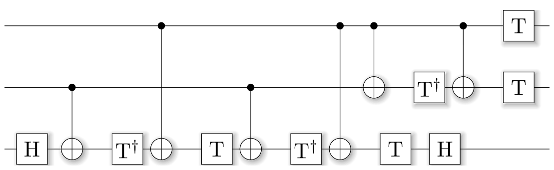


Similar to swap
Global phase of pi/4

41

PHYC90045 Introduction to Quantum Computing

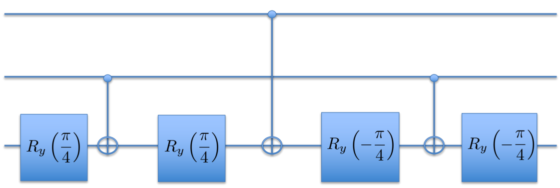
Toffoli from CNOTS



42

PHYC90045 Introduction to Quantum Computing

Toffoli with Incorrect Phase

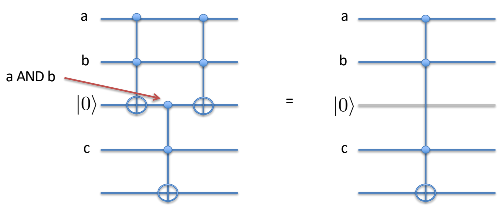


If neither control is 1, then no net rotation.
If only first control is a 1, then becomes Z rotation (Ry by pi).
If only second control qubit is a 1, then both halves cancel.
If both controls are 1, then bottom qubit flips because of middle CNOT.

43

PHYC90045 Introduction to Quantum Computing

Multiply Controlled Gates



We can use Toffoli and an ancilla qubit to define multiply controlled gates.

44

PHYC90045 Introduction to Quantum Computing

Mocking up gates

If you use controlled operations on all qubits except the target, then you isolate a single 2x2 subspace, with the rest of the matrix untouched.

$$CU = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & a & b \\ 0 & 0 & c & d \end{bmatrix}$$

Or controlled off the zero state:

$$C_0 U = \begin{bmatrix} a & b & 0 & 0 \\ c & d & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

45

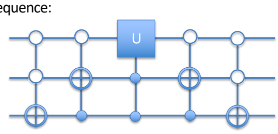
Using Gray codes

Imagine we wanted a 2x2 matrix between the 000 and 111 states:

A	B	C
0	0	0
0	0	1
0	1	1
1	1	1

Gray code

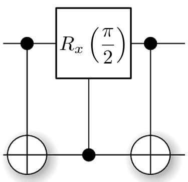
Corresponding sequence:



In this way, complicated multi-qubit gates can be built, piece by piece.

46

Square Root Swap Construction



Similar to swap
Global phase of pi/4

47

Week 9

Lecture 17 – Introduction to IBM Quantum Experience
Introduction to IBM Quantum Experience: Guest Lecture

Lecture 18 – IBM and Optimisations
14.1 QUI compared to IBM
14.2 QASM and QISKit
14.3 Optimizing circuits

Lab 9
Using the IBM system

48

49

Defining a new Function/Gate

Keyword: gate

```
// Controlled Phase gate
gate cz a,b
{
   h b;
   cx a,b;
   h b;
}
```

"cz" is name of gate

a and b are parameters

This gate can then be used like a native gate:

```
cz q[3],q[2];
```

50

QASM Header File

```
// Quantum Experience (QE) Standard Header
// file: qelib1.inc

// --- QE Hardware primitives ---

// 3-parameter 2-pulse single qubit gate
gate u3(theta,phi,lambda) q { U(theta,phi,lambd
// 2-parameter 1-pulse single qubit gate
gate u2(phi,lambda) q { U(pi/2,phi,lambda) q; }
// 1-parameter 0-pulse single qubit gate
gate u1(lambda) q { U(0,0,lambda) q; }
// controlled-NOT
gate cx c,t { CX c,t; }
// idle gate (identity)
gate id a { U(0,0,0) a; }

// --- QE Standard Gates ---

// Pauli gate: bit-flip
gate x a { u3(pi,0,pi) a; }
// Pauli gate: bit and phase flip
gate y a { u3(pi,pi/2,pi/2) a; }
// Pauli gate: phase flip
gate z a { u1(pi) a; }
// Clifford gate: Hadamard
gate h a { u2(0,pi) a; }
```

Also defines:

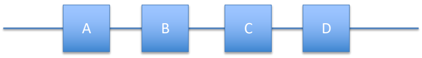
Rotations
rx, ry, rz

Toffoli
ccx

Controlled rotations
cu1, cu2, cu3, crz, ch

51

Product of single qubit unitaries



Euler angle rotations (with global phase = 0):

$$U_3 = \left[\begin{array}{cc} e^{-i(\lambda+\phi)/2}\cos(\theta/2) & -e^{i(\lambda-\phi)/2}\sin(\theta/2) \\ e^{i(-\lambda+\phi)/2}\sin(\theta/2) & e^{i(\lambda+\phi)/2}\cos(\theta/2) \end{array} \right]$$

Write out the matrix and equate elements.

52