Hands-on Introduction to Qiskit

Stefan Kühn

The Cyprus Institute
RESEARCH•TECHNOLOGY•INNOVATION

Workshop on Quantum Computing and Quantum Sensors
DESY, 18 August 2020

Motivation

On the verge of the NISQ era

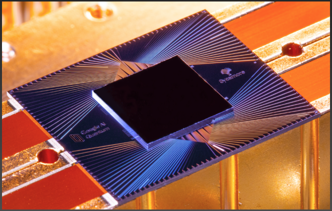
- Noisy Intermediate-Scale Quantum (NISQ) technology is available
- Noise significantly limits the circuit depths that can be executed reliably

J. Preskill, Quantum 2, 79 (2018)

Motivation

On the verge of the NISQ era

- Noisy Intermediate-Scale Quantum (NISQ) technology is available
- Noise significantly limits the circuit depths that can be executed reliably
- Current NISQ devices have already outperformed classical devices



Article

Quantum supremacy using a programmable superconducting processor

https://doi.org/10.1038/s41586-019-1666-5
Received: 22 July 2019
Accepted: 20 September 2019
Published online: 23 October 2019

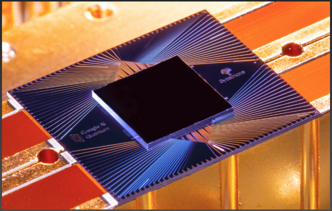
Frank Arute[1], Kunal Arya[1], Ryan Babbush[1], Dave Bacon[1], Joseph C. Bardin[1,2], Rami Barends[1], Rupak Biswas[3], Sergio Boixo[1], Fernando G. S. L. Brandao[1,4], David A. Buell[1], Brian Burkett[1], Yu Chen[1], Zijun Chen[1], Ben Chiaro[5], Roberto Collins[1], William Courtney[1], Andrew Dunsworth[1], Edward Farhi[1], Brooks Foxen[1,5], Austin Fowler[1], Craig Gidney[1], Marissa Giustina[1], Rob Graff[1], Keith Guerin[1], Steve Habegger[1], Matthew P. Harrigan[1], Michael J. Hartmann[1,6], Alan Ho[1], Markus Hoffmann[1], Trent Huang[1], Travis S. Humble[7], Sergei V. Isakov[1], Evan Jeffrey[1], Zhang Jiang[1], Dvir Kafri[1], Kostyantyn Kechedzhi[1], Julian Kelly[1], Paul V. Klimov[1], Sergey Knysh[1], Alexander Korotkov[1,8], Fedor Kostritsa[1], David Landhuis[1], Mike Lindmark[1], Erik Lucero[1], Dmitry Lyakh[9], Salvatore Mandrà[3,10], Jarrod R. McClean[1], Matthew McEwen[5], Anthony Megrant[1], Xiao Mi[1], Kristel Michielsen[11,12], Masoud Mohseni[1], Josh Mutus[1], Ofer Naaman[1], Matthew Neeley[1], Charles Neill[1], Murphy Yuezhen Niu[1], Eric Ostby[1], Andre Petukhov[1], John C. Platt[1], Chris Quintana[1], Eleanor G. Rieffel[3], Pedram Roushan[1], Nicholas C. Rubin[1], Daniel Sank[1], Kevin J. Satzinger[1], Vadim Smelyanskiy[1], Kevin J. Sung[1,13], Matthew D. Trevithick[1], Amit Vainsencher[1], Benjamin Villalonga[1,14], Theodore White[1], Z. Jamie Yao[1], Ping Yeh[1], Adam Zalcman[1], Hartmut Neven[1] & John M. Martinis[1,5*]

J. Preskill, Quantum 2, 79 (2018)
Image taken from https://ai.googleblog.com/2019/10/quantum-supremacy-using-programmable.html
Arute et al., Nature 574, 505 (2019)

Motivation

On the verge of the NISQ era

- Noisy Intermediate-Scale Quantum (NISQ) technology is available
- Noise significantly limits the circuit depths that can be executed reliably
- Current NISQ devices have already outperformed classical devices
- Commercially/openly available devices
  - D-Wave
  - IBM Quantum Experience
  - Rigetti Computing
  - IONQ
  - ...



Article

Quantum supremacy using a programmable superconducting processor

https://doi.org/10.1038/s41586-019-1666-5
Received: 22 July 2019
Accepted: 20 September 2019
Published online: 23 October 2019

Frank Arute[1], Kunal Arya[1], Ryan Babbush[1], Dave Bacon[1], Joseph C. Bardin[1,2], Rami Barends[1], Rupak Biswas[3], Sergio Boixo[1], Fernando G. S. L. Brandao[1,4], David A. Buell[1], Brian Burkett[1], Yu Chen[1], Zijun Chen[1], Ben Chiaro[5], Roberto Collins[1], William Courtney[1], Andrew Dunsworth[1], Edward Farhi[1], Brooks Foxen[1,5], Austin Fowler[1], Craig Gidney[1], Marissa Giustina[1], Rob Graff[1], Keith Guerin[1], Steve Habegger[1], Matthew P. Harrigan[1], Michael J. Hartmann[1,6], Alan Ho[1], Markus Hoffmann[1], Trent Huang[1], Travis S. Humble[7], Sergei V. Isakov[1], Evan Jeffrey[1], Zhang Jiang[1], Dvir Kafri[1], Kostyantyn Kechedzhi[1], Julian Kelly[1], Paul V. Klimov[1], Sergey Knysh[1], Alexander Korotkov[1,8], Fedor Kostritsa[1], David Landhuis[1], Mike Lindmark[1], Erik Lucero[1], Dmitry Lyakh[9], Salvatore Mandrà[3,10], Jarrod R. McClean[1], Matthew McEwen[5], Anthony Megrant[1], Xiao Mi[1], Kristel Michielsen[11,12], Masoud Mohseni[1], Josh Mutus[1], Ofer Naaman[1], Matthew Neeley[1], Charles Neill[1], Murphy Yuezhen Niu[1], Eric Ostby[1], Andre Petukhov[1], John C. Platt[1], Chris Quintana[1], Eleanor G. Rieffel[3], Pedram Roushan[1], Nicholas C. Rubin[1], Daniel Sank[1], Kevin J. Satzinger[1], Vadim Smelyanskiy[1], Kevin J. Sung[1,13], Matthew D. Trevithick[1], Amit Vainsencher[1], Benjamin Villalonga[1,14], Theodore White[1], Z. Jamie Yao[1], Ping Yeh[1], Adam Zalcman[1], Hartmut Neven[1] & John M. Martinis[1,5*]

J. Preskill, Quantum 2, 79 (2018)
Image taken from https://ai.googleblog.com/2019/10/quantum-supremacy-using-programmable.html
Arute et al., Nature 574, 505 (2019)

Outline

1. Motivation
2. Basics of the circuit model of quantum computing
3. The Qiskit SDK
4. Hands-on Exercises
5. Further reading & Outlook

Basics of the circuit model of quantum computing

Quantum bits

- Qubit: **two-dimensional quantum system**
- Hilbert space $\mathcal{H}$ with basis $\{|0\rangle, |1\rangle\}$
- Contrary to classical bits, it can be in a **superposition**

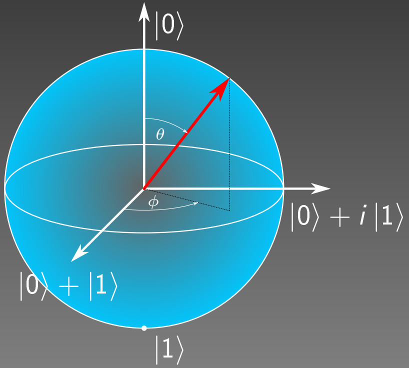
$$|\psi\rangle = \cos\left(\frac{\theta}{2}\right)|0\rangle + e^{i\phi}\sin\left(\frac{\theta}{2}\right)|1\rangle$$

Basics of the circuit model of quantum computing

Quantum bits

- Qubit: **two-dimensional quantum system**
- Hilbert space $\mathcal{H}$ with basis $\{|0\rangle, |1\rangle\}$
- Contrary to classical bits, it can be in a **superposition**

$$|\psi\rangle = \cos\left(\frac{\theta}{2}\right)|0\rangle + e^{i\phi}\sin\left(\frac{\theta}{2}\right)|1\rangle$$



$$\vec{r} = \begin{pmatrix} \sin(\theta)\cos(\phi) \\ \sin(\theta)\sin(\phi) \\ \cos(\theta) \end{pmatrix}$$

Basics of the circuit model of quantum computing

Quantum bits

- n qubits: Hilbert space is the tensor product $\underbrace{\mathcal{H} \otimes \cdots \otimes \mathcal{H}}_{n \text{ times}}$
- Most general state in the computational basis

$$|\psi\rangle = \sum_{i_1,\ldots,i_n=0}^{1} c_{i_1\ldots i_n} |i_1\rangle \otimes \cdots \otimes |i_N\rangle$$

- A quantum state that cannot be factored as a tensor product of states of its local constituents is called **entangled**
  - $|\psi_1\rangle = \frac{1}{2}(|0\rangle \otimes |0\rangle + |0\rangle \otimes |1\rangle + |1\rangle \otimes |0\rangle + |1\rangle \otimes |1\rangle)$

Basics of the circuit model of quantum computing

Quantum bits

- n qubits: Hilbert space is the tensor product $\underbrace{\mathcal{H} \otimes \cdots \otimes \mathcal{H}}_{n \text{ times}}$
- Most general state in the computational basis

$$|\psi\rangle = \sum_{i_1,\ldots,i_n=0}^{1} c_{i_1\ldots i_n} |i_1\rangle \otimes \cdots \otimes |i_N\rangle$$

- A quantum state that cannot be factored as a tensor product of states of its local constituents is called **entangled**
  - $|\psi_1\rangle = \frac{1}{2}(|0\rangle \otimes |0\rangle + |0\rangle \otimes |1\rangle + |1\rangle \otimes |0\rangle + |1\rangle \otimes |1\rangle)$
    $= \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \otimes \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$
    $\Rightarrow$ product state
  - $|\psi_2\rangle = \frac{1}{\sqrt{2}}(|0\rangle \otimes |0\rangle + |1\rangle \otimes |1\rangle)$

Basics of the circuit model of quantum computing

Quantum bits

- n qubits: Hilbert space is the tensor product $\underbrace{\mathcal{H} \otimes \cdots \otimes \mathcal{H}}_{n \text{ times}}$
- Most general state in the computational basis

$$|\psi\rangle = \sum_{i_1,\ldots,i_n=0}^{1} c_{i_1\ldots i_n} |i_1\rangle \otimes \cdots \otimes |i_N\rangle$$

- A quantum state that cannot be factored as a tensor product of states of its local constituents is called **entangled**
  - $|\psi_1\rangle = \frac{1}{2}(|0\rangle \otimes |0\rangle + |0\rangle \otimes |1\rangle + |1\rangle \otimes |0\rangle + |1\rangle \otimes |1\rangle)$
    $= \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \otimes \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$
    $\Rightarrow$ product state
  - $|\psi_2\rangle = \frac{1}{\sqrt{2}}(|0\rangle \otimes |0\rangle + |1\rangle \otimes |1\rangle)$
    $\Rightarrow$ entangled state (Bell state)
- In the following $\otimes$ often suppressed: $|0\rangle \otimes |0\rangle \rightarrow |0\rangle\,|0\rangle\,,\ |00\rangle$

Basics of the circuit model of quantum computing

Quantum gates

- Quantum mechanics is reversible, $|\psi\rangle$ undergoes unitary evolution under some (time-dependent) Hamiltonian $H(t)$

$$|\psi(t)\rangle = T \exp\left(-i \int_0^t ds\, H(s)\right) |\psi_0\rangle$$

- **Quantum gates** are represented by **unitary matrices**

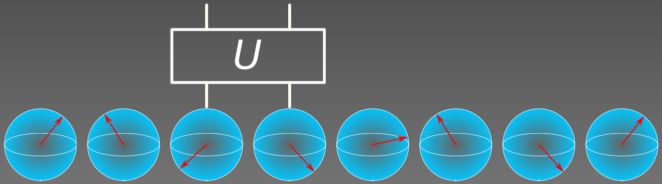
Basics of the circuit model of quantum computing

Quantum gates

- Quantum mechanics is reversible, $|\psi\rangle$ undergoes unitary evolution under some (time-dependent) Hamiltonian $H(t)$

$$|\psi(t)\rangle = T\exp\left(-i\int_0^t ds\, H(s)\right)|\psi_0\rangle$$

- **Quantum gates** are represented by **unitary matrices**
- Typically gates only act on a few qubits in a nontrivial way



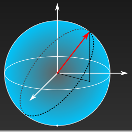
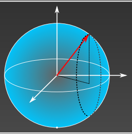
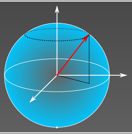
Basics of the circuit model of quantum computing

Common single-qubit quantum gates

X	—[X]—	$X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$	$\lvert 0\rangle \to \lvert 1\rangle$ $\lvert 1\rangle \to \lvert 0\rangle$
Y	—[Y]—	$Y = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}$	$\lvert 0\rangle \to -i\lvert 1\rangle$ $\lvert 1\rangle \to i\lvert 0\rangle$
Z	—[Z]—	$Z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$	$\lvert 0\rangle \to \lvert 0\rangle$ $\lvert 1\rangle \to -\lvert 1\rangle$

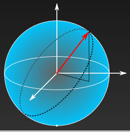
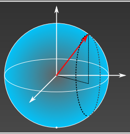
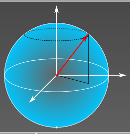
Basics of the circuit model of quantum computing

Common single-qubit quantum gates

$R_x(\theta)$	—[$R_x(\theta)$]—	$R_x = \exp\left(-i\frac{\theta}{2}X\right)$	
$R_y(\theta)$	—[$R_y(\theta)$]—	$R_y = \exp\left(-i\frac{\theta}{2}Y\right)$	
$R_z(\theta)$	—[$R_z(\theta)$]—	$R_z(\theta) = \exp\left(-i\frac{\theta}{2}Z\right)$	

Basics of the circuit model of quantum computing

Common single-qubit quantum gates

$R_x(\theta)$	$-\boxed{R_x(\theta)}-$	$R_x = \exp\left(-i\frac{\theta}{2}X\right)$	
$R_y(\theta)$	$-\boxed{R_y(\theta)}-$	$R_y = \exp\left(-i\frac{\theta}{2}Y\right)$	
$R_z(\theta)$	$-\boxed{R_z(\theta)}-$	$R_z(\theta) = \exp\left(-i\frac{\theta}{2}Z\right)$	
Hadamard	$-\boxed{H}-$	$H = \begin{pmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \end{pmatrix}$	$\lvert 0\rangle \to \frac{1}{\sqrt{2}}(\lvert 0\rangle + \lvert 1\rangle)$ $\lvert 1\rangle \to \frac{1}{\sqrt{2}}(\lvert 0\rangle - \lvert 1\rangle)$

Basics of the circuit model of quantum computing

Common multi-qubit quantum gates

CNOT		$\text{CNOT} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}$	$\lvert 00\rangle \to \lvert 00\rangle$ $\lvert 01\rangle \to \lvert 01\rangle$ $\lvert 10\rangle \to \lvert 11\rangle$ $\lvert 11\rangle \to \lvert 10\rangle$

Basics of the circuit model of quantum computing

Common multi-qubit quantum gates

CNOT		$\mathrm{CNOT} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}$	$\|00\rangle \to \|00\rangle$ $\|01\rangle \to \|01\rangle$ $\|10\rangle \to \|11\rangle$ $\|11\rangle \to \|10\rangle$
$R_{xx}(\theta)$	$R_{xx}(\theta)$	$R_{xx}(\theta) = \exp\left(-i\frac{\theta}{2} X \otimes X\right)$	
$R_{yy}(\theta)$	$R_{yy}(\theta)$	$R_{yy}(\theta) = \exp\left(-i\frac{\theta}{2} Y \otimes Y\right)$	
$R_{zz}(\theta)$	$R_{zz}(\theta)$	$R_{zz}(\theta) = \exp\left(-i\frac{\theta}{2} Z \otimes Z\right)$	

Basics of the circuit model of quantum computing

Quantum gates

- Since quantum mechanics is linear, we can apply gates to superpositions of basis states

$$\begin{aligned}\mathsf{CNOT}\big(&\alpha\,|00\rangle + \beta\,|01\rangle + \gamma\,|10\rangle + \delta\,|11\rangle\big)\\ =&\alpha\,|00\rangle + \beta\,|01\rangle + \gamma\,|11\rangle + \delta\,|10\rangle\end{aligned}$$

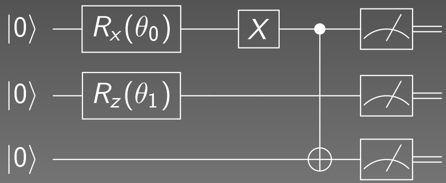
Basics of the circuit model of quantum computing

Quantum gates

- Since quantum mechanics is linear, we can apply gates to superpositions of basis states

$$\begin{aligned}\mathrm{CNOT}\big(\alpha\,|00\rangle + \beta\,|01\rangle + \gamma\,|10\rangle + \delta\,|11\rangle\big) \\ = \alpha\,|00\rangle + \beta\,|01\rangle + \gamma\,|11\rangle + \delta\,|10\rangle\end{aligned}$$

- Combining multiple gates we can build **quantum circuits**



3.

1. Motivation
2. Basics of the circuit model of quantum computing
3. The Qiskit SDK
4. Hands-on Exercises
5. Further reading & Outlook

The Qiskit SDK

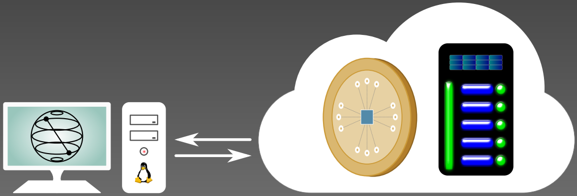
The Qiskit SDK

- Open source Python SDK for developing and testing quantum programs
- Based on the circuit model of quantum computation

The Qiskit SDK

The Qiskit SDK

- Open source Python SDK for developing and testing quantum programs
- Based on the circuit model of quantum computation
- Allows for seamlessly running quantum programs on IBM's quantum devices



https://qiskit.org
https://github.com/Qiskit

The Qiskit SDK

The Qiskit Elements

Terra 🌍

- Quantum circuits
- Pulse schedule
- Transpiler
- Providers allowing to access backends
- Basic quantum information tasks
- Visualization

The Qiskit SDK

The Qiskit Elements

Ignis 🔥

- Dealing with noise and errors
- Characterizing errors
- Error mitigation

Aqua 💦

- Algorithms for
  - Chemistry
  - Finance
  - Machine Learning
  - Optimization

Aer 💨

- Various classical simulators for quantum circuits
  - Qasm
  - State vector
  - Unitary

Terra 🌍

- Quantum circuits
- Pulse schedule
- Transpiler
- Providers allowing to access backends
- Basic quantum information tasks
- Visualization

The Qiskit SDK

The Qiskit Elements

Ignis 🔥

- Dealing with noise and errors
- Characterizing errors
- Error mitigation

Aqua 💦

- Algorithms for
  - Chemistry
  - Finance
  - Machine Learning
  - Optimization

Aer 💨

- Various classical simulators for quantum circuits
  - Qasm
  - State vector
  - Unitary

Terra 🌍

- Quantum circuits
- Pulse schedule
- Transpiler
- Providers allowing to access backends
- Basic quantum information tasks
- Visualization

The Qiskit SDK

Example

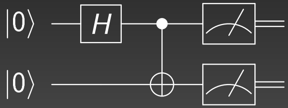
```
# Importing standard Qiskit libraries
from qiskit import QuantumCircuit, execute, Aer, IBMQ
from qiskit.visualization import *
from qiskit.quantum_info import state_fidelity

# Create a quantum circuit for 3 qubits
qc = QuantumCircuit(3)
# Add gates
qc.cnot(0,1)
qc.rz(np.pi/8, 2)
# Add a measurement to all qubits
qc.measure_all()

# Simulate the experiment
qasm_simulator = Aer.get_backend("qasm_simulator")
job = execute(qc, qasm_simulator, shots=500)
result = job.result()
counts = result.get_counts()
print("Counts for the basis states:",counts)
```

Exercise 1: Superposition and Entanglement

Exercise 1: Superposition and Entanglement

The Bell state

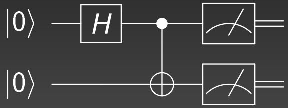
- Simple circuit preparing an entangled state (Bell state)



Exercise 1: Superposition and Entanglement

The Bell state

- Simple circuit preparing an entangled state (Bell state)

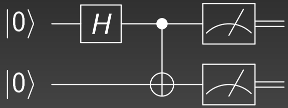


1. $|0\rangle \otimes |0\rangle \xrightarrow{H} \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \otimes |0\rangle = \frac{1}{\sqrt{2}}(|0\rangle \otimes |0\rangle + |1\rangle \otimes |0\rangle)$

Exercise 1: Superposition and Entanglement

The Bell state

- Simple circuit preparing an entangled state (Bell state)

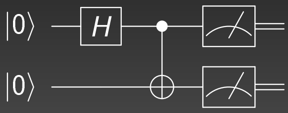


1. $|0\rangle \otimes |0\rangle \xrightarrow{H} \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \otimes |0\rangle = \frac{1}{\sqrt{2}}(|0\rangle \otimes |0\rangle + |1\rangle \otimes |0\rangle)$
2. $\frac{1}{\sqrt{2}}(|0\rangle \otimes |0\rangle + |1\rangle \otimes |0\rangle) \xrightarrow{\text{CNOT}} \frac{1}{\sqrt{2}}(|0\rangle \otimes |0\rangle + |1\rangle \otimes |1\rangle)$

Exercise 1: Superposition and Entanglement

The Bell state

- Simple circuit preparing an entangled state (Bell state)



1. $|0\rangle \otimes |0\rangle \xrightarrow{H} \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \otimes |0\rangle = \frac{1}{\sqrt{2}}(|0\rangle \otimes |0\rangle + |1\rangle \otimes |0\rangle)$
2. $\frac{1}{\sqrt{2}}(|0\rangle \otimes |0\rangle + |1\rangle \otimes |0\rangle) \xrightarrow{\text{CNOT}} \frac{1}{\sqrt{2}}(|0\rangle \otimes |0\rangle + |1\rangle \otimes |1\rangle)$
3. Measurement in computational basis yields:
   $P(|0\rangle \otimes |0\rangle) = 1/2$, $P(|1\rangle \otimes |1\rangle) = 1/2$

Exercise 1: Superposition and Entanglement

- Study the Hadamard gate

Tasks

1. Implement the circuit above.
2. Visualize the circuit and make sure it is correct.
3. Measure the results with 100, 500, 1000 and 10000 shots and visualize the result. What do you observe?

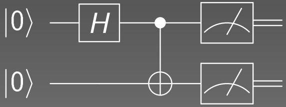
Exercise 1: Superposition and Entanglement

- Study the Hadamard gate

Tasks

1. Implement the circuit above.
2. Visualize the circuit and make sure it is correct.
3. Measure the results with 100, 500, 1000 and 10000 shots and visualize the result. What do you observe?

- Create a new circuit generating the Bell state



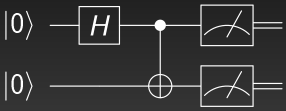
Tasks

Repeat the same tasks for this circuit.

Exercise 1: Superposition and Entanglement

The Bell states

- The following circuit yields $|\Phi^+\rangle = \frac{1}{\sqrt{2}}(|0\rangle \otimes |0\rangle + |1\rangle \otimes |1\rangle)$



Optional tasks

1. Can you find circuits preparing the other three Bell states?

$$|\Phi^-\rangle = \frac{1}{\sqrt{2}}(|0\rangle \otimes |0\rangle - |1\rangle \otimes |1\rangle)$$
$$|\Psi^+\rangle = \frac{1}{\sqrt{2}}(|0\rangle \otimes |1\rangle + |1\rangle \otimes |0\rangle)$$
$$|\Psi^-\rangle = \frac{1}{\sqrt{2}}(|0\rangle \otimes |1\rangle - |1\rangle \otimes |0\rangle)$$

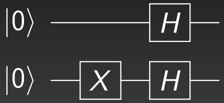
2. Convince yourself that the Bell states are orthonormal.

Exercise 2: Phase kickback

Exercise 2: Phase kickback

Simple phase kickback

- Consider the following simple circuit



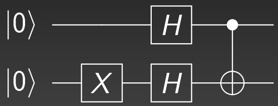
Tasks:

1. Implement the circuit and visualize it to make sure it is correct.
2. Obtain the final state using the state vector simulator and visualize the state of the individual qubits on the Bloch sphere. Which state did you obtain?
3. Add an additional CNOT gate to the circuit.
4. Execute it again the state vector simulator and the state of the individual qubits on the Bloch sphere. What do you observe?

Exercise 2: Phase kickback

Simple phase kickback

- Our final circuit looks like



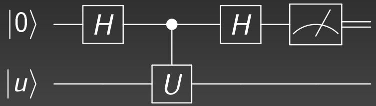
Optional tasks:

1. Replace the CNOT gate in the circuit above with a controlled R_x rotation.
2. Simulate the circuit for various angles of the rotation gate and visualize the results on the Bloch sphere. What do you observe? Can you explain the effect?

Exercise 2: Phase kickback

Remarks

- These two circuits are special cases of **phase kickback**
- Phase kickback is a fundamental building block of many quantum algorithms
- General form

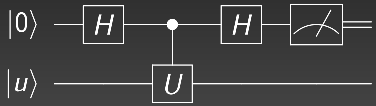


- Choosing $|u\rangle$ as an eigenstate of U with eigenvalue $\exp(i\phi)$
  1. $|0\rangle \otimes |u\rangle \xrightarrow{H} \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \otimes |u\rangle$
  2. $\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \otimes |u\rangle \xrightarrow{\text{cU}} \frac{1}{\sqrt{2}}(|0\rangle \otimes |u\rangle + e^{i\phi}|1\rangle \otimes |u\rangle)$
     $\Rightarrow$ **Phase** has been **kicked back into the control qubit**
  3. $\frac{1}{\sqrt{2}}(|0\rangle \otimes |u\rangle + e^{i\phi}|1\rangle \otimes |u\rangle) \xrightarrow{H} \left(\cos\frac{\phi}{2}|0\rangle + i\sin\frac{\phi}{2}|1\rangle\right) \otimes |u\rangle$

Exercise 2: Phase kickback

Remarks

- These two circuits are special cases of **phase kickback**
- Phase kickback is a fundamental building block of many quantum algorithms
- General form



- Choosing $|u\rangle$ as an eigenstate of U with eigenvalue $\exp(i\phi)$
  1. $|0\rangle \otimes |u\rangle \xrightarrow{H} \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \otimes |u\rangle$
  2. $\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \otimes |u\rangle \xrightarrow{\text{cU}} \frac{1}{\sqrt{2}}(|0\rangle \otimes |u\rangle + e^{i\phi}|1\rangle \otimes |u\rangle)$
     $\Rightarrow$ **Phase** has been **kicked back into the control qubit**
  3. $\frac{1}{\sqrt{2}}(|0\rangle \otimes |u\rangle + e^{i\phi}|1\rangle \otimes |u\rangle) \xrightarrow{H} \left(\cos\frac{\phi}{2}|0\rangle + i\sin\frac{\phi}{2}|1\rangle\right) \otimes |u\rangle$
- Unitaries $\bar{U}$ computing $f : \{0,1\}^n \rightarrow \{0,1\}^m$

$$\bar{U}|x\rangle|y\rangle = |x\rangle|y \oplus f(x)\rangle$$

can be shown to be of the controlled-U type.

Exercise 3: Real-time evolution of the Ising model

Exercise 3: Real-time evolution of the Ising model

The Ising model

- Ising Hamiltonian with open boundary conditions

$$H = \sum_{i=0}^{N-2} Z_i Z_{i+1} + h \sum_{i=0}^{N-1} X_i$$

- Evolution of the wave function $|\psi_0\rangle$ under the Hamiltonian

$$|\psi(t)\rangle = \exp\left(-iHt\right)|\psi_0\rangle$$

Exercise 3: Real-time evolution of the Ising model

The Ising model

- Ising Hamiltonian with open boundary conditions

$$H = \sum_{i=0}^{N-2} Z_i Z_{i+1} + h \sum_{i=0}^{N-1} X_i$$

- Evolution of the wave function $|\psi_0\rangle$ under the Hamiltonian

$$|\psi(t)\rangle = \exp\left(-iHt\right)|\psi_0\rangle \; = \left[\exp\left(-iH\frac{t}{n}\right)\right]^n |\psi_0\rangle$$

Exercise 3: Real-time evolution of the Ising model

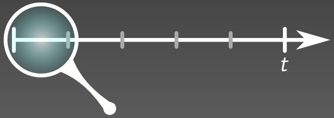
The Ising model

- Ising Hamiltonian with open boundary conditions

$$H = \sum_{i=0}^{N-2} Z_i Z_{i+1} + h \sum_{i=0}^{N-1} X_i$$

- Evolution of the wave function $|\psi_0\rangle$ under the Hamiltonian

$$|\psi(t)\rangle = \exp\left(-iHt\right)|\psi_0\rangle = \left[\exp\left(-iH\frac{t}{n}\right)\right]^n |\psi_0\rangle$$



- Using a Suszuki-Trotter decomposition with $\Delta t = t/n$ we can approximate

$$\exp\left(-iH\Delta t\right) \approx \prod_k \exp\left(-iZ_k Z_{k+1}\Delta t\right) \prod_k \exp\left(-ihX_k\Delta t\right) + \mathcal{O}\left((\Delta t)^2\right)$$

Exercise 3: Real-time evolution of the Ising model

The Ising model

- Trotterized time evolution operator for a small step Δt

$$U \approx \prod_k \exp\left(-iZ_k Z_{k+1} \Delta t\right) \prod_k \exp\left(-ihX_k \Delta t\right)$$

Tasks

1. Complete the function preparing the initial state $|0010\rangle$.
2. Complete the function that implements the Trotter evolution of the Ising model.
3. Run the quantum circuit and compute the time evolution.
4. Visualize the expectation value of the total magnetization $M = \langle \sum_i Z_i \rangle$ and site resolved expectation $\langle Z_i \rangle$ as a function of time and compare to the exact solution.

Exercise 3: Real-time evolution of the Ising model

The Ising model

- Trotterized time evolution operator for a small step Δt

$$U \approx \prod_k \exp\left(-iZ_k Z_{k+1}\Delta t\right) \prod_k \exp\left(-ihX_k \Delta t\right)$$

Optional tasks

1. Create a parameterized version of your previous circuit with parameters Δt and h. Instructions for creating parameterized circuits can be found here: `https://qiskit.org/documentation/tutorials/circuits_advanced/1_advanced_circuits.html#Parameterized-circuits`

Further reading & Outlook

Further reading

- Tutorials provided by Qiskit can be found here
  https://github.com/Qiskit/qiskit-tutorials
- Qiskit textbook
  https://qiskit.org/textbook/preface.html

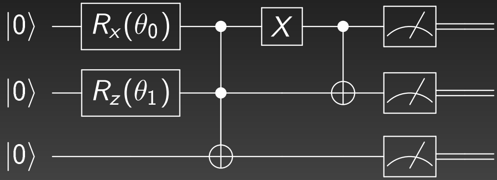
Next steps

- Register for an IBM ID to access cloud services
  https://quantum-computing.ibm.com/
  - Access IBM's small scale quantum devices
  - Powerful Qasm simulator for up 32 qubits
  - Interactive circuit composer
- Qiskit documentation
  https://qiskit.org/documentation/index.html

Appendix A. Projective measurements

Setting

- We assume the quantum device prepares a pure state $|\psi\rangle$



- $|\psi\rangle$ is measured in the computational basis $\{|0\rangle, |1\rangle\}^{\otimes N}$

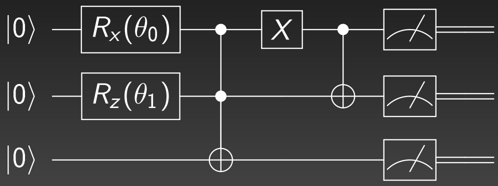
$$|\psi\rangle = \sum_{i=0}^{2^N-1} c_i \, |\text{binary}(i)\rangle$$

$\Rightarrow$ with probability $|c_i|^2$ we record the bit string $|\text{binary}(i)\rangle$

Appendix A. Projective measurements

Setting

- We assume the quantum device prepares a pure state $|\psi\rangle$



- $|\psi\rangle$ is measured in the computational basis $\{|0\rangle, |1\rangle\}^{\otimes N}$

$$|\psi\rangle = \sum_{i=0}^{2^N-1} c_i \, |\mathrm{binary}(i)\rangle$$

$\Rightarrow$ with probability $|c_i|^2$ we record the bit string $|\mathrm{binary}(i)\rangle$

- In practice we have to repeat the experiment many times to get enough samples from the probability distribution
$\Rightarrow$ "Number of shots" s

Appendix A. Projective measurements

Measuring observables

- Given an observable O we want to compute $\langle\psi| O |\psi\rangle$
- State can only be measured in the computational basis

$$\begin{aligned}
\langle\psi| O |\psi\rangle &= \langle\psi| U^\dagger U O U^\dagger U |\psi\rangle \\
&= \langle\psi'| U O U^\dagger |\psi'\rangle \\
&= \langle\psi'| D |\psi'\rangle
\end{aligned}$$

Appendix A. Projective measurements

Measuring observables

- Given an observable O we want to compute $\langle\psi| O |\psi\rangle$
- State can only be measured in the computational basis

$$\begin{aligned}\langle\psi| O |\psi\rangle &= \langle\psi| U^\dagger U O U^\dagger U |\psi\rangle \\ &= \langle\psi'| U O U^\dagger |\psi'\rangle \\ &= \langle\psi'| D |\psi'\rangle\end{aligned}$$

- Choose U such that $D = UOU^\dagger = \mathrm{diag}(\lambda_0, \ldots, \lambda_{2^N-1})$ in the computational basis

$$\langle\psi| O |\psi\rangle = \sum_{i=0}^{2^N-1} |c_i'|^2 \lambda_i$$

- U is often called **post rotation**
- Instead of $|\psi\rangle$ we prepare $|\psi'\rangle$ and measure the probability distribution $|c_i'|^2$

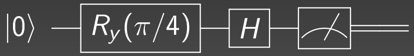
Appendix A. Projective measurements

Example

- State $|\psi\rangle = R_y(\pi/4)\,|0\rangle$
- Observable we want to measure $O = X$

$$D = UOU^\dagger = \frac{1}{\sqrt{2}}\begin{pmatrix}1 & 1\\ 1 & -1\end{pmatrix}\; X \;\frac{1}{\sqrt{2}}\begin{pmatrix}1 & 1\\ 1 & -1\end{pmatrix} = HXH = Z$$

- Circuit to prepare and measure $|\psi'\rangle = U\,|\psi\rangle = HR_y(\pi/4)\,|0\rangle$



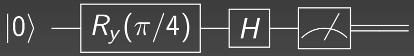
Appendix A. Projective measurements

Example

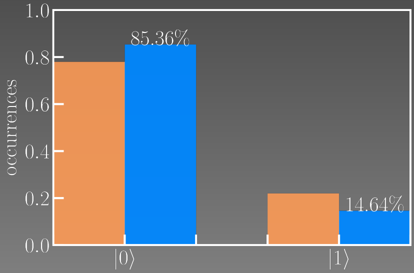
- State $|\psi\rangle = R_y(\pi/4)\,|0\rangle$
- Observable we want to measure $O = X$

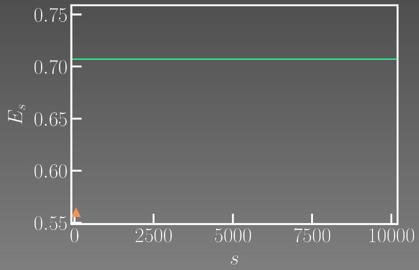
$$D = UOU^{\dagger} = \frac{1}{\sqrt{2}}\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} X \frac{1}{\sqrt{2}}\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} = HXH = Z$$

- Circuit to prepare and measure $|\psi'\rangle = U\,|\psi\rangle = HR_y(\pi/4)\,|0\rangle$



- Results for Z preparing $|\psi'\rangle$





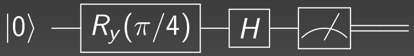
Appendix A. Projective measurements

Example

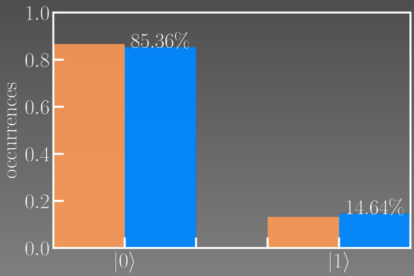
- State $|\psi\rangle = R_y(\pi/4)\,|0\rangle$
- Observable we want to measure $O = X$

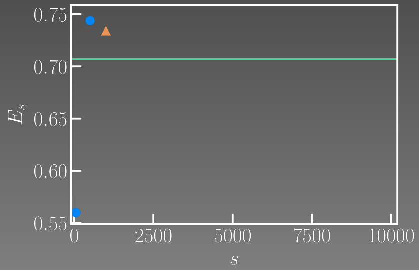
$$D = UOU^\dagger = \frac{1}{\sqrt{2}}\begin{pmatrix}1 & 1\\ 1 & -1\end{pmatrix}\ X\ \frac{1}{\sqrt{2}}\begin{pmatrix}1 & 1\\ 1 & -1\end{pmatrix} = HXH = Z$$

- Circuit to prepare and measure $|\psi'\rangle = U\,|\psi\rangle = HR_y(\pi/4)\,|0\rangle$



- Results for Z preparing $|\psi'\rangle$





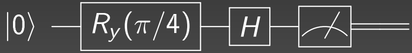
Appendix A. Projective measurements

Example

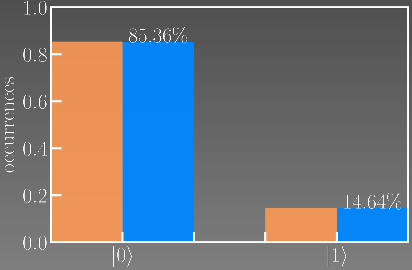
- State $|\psi\rangle = R_y(\pi/4)\,|0\rangle$
- Observable we want to measure $O = X$

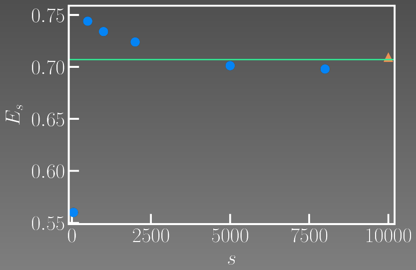
$$D = UOU^\dagger = \frac{1}{\sqrt{2}}\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} X \frac{1}{\sqrt{2}}\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} = HXH = Z$$

- Circuit to prepare and measure $|\psi'\rangle = U\,|\psi\rangle = HR_y(\pi/4)\,|0\rangle$



- Results for Z preparing $|\psi'\rangle$

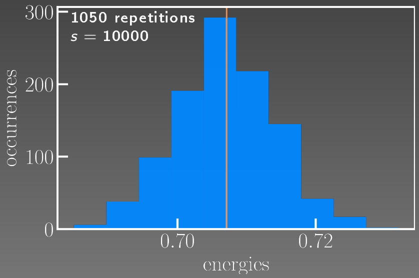




Appendix A. Projective measurements

Example

- Repeating the measurement a number of times for fixed s yields a histogram with peak around $E_0 = \langle\psi| X |\psi\rangle$



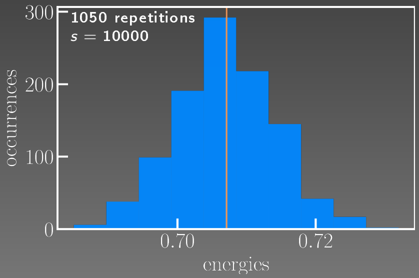
Appendix A. Projective measurements

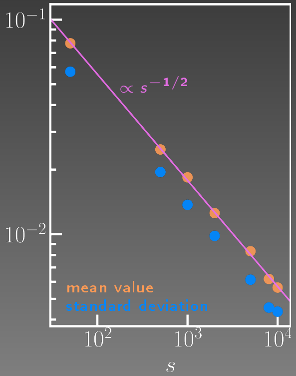
Example

- Repeating the measurement a number of times for fixed s yields a histogram with peak around $E_0 = \langle\psi| X |\psi\rangle$
- The mean and standard deviation of the error

$$\left|\langle\psi| D |\psi\rangle_{\text{measured}} - \langle\psi| D |\psi\rangle_{\text{exact}}\right|$$

decay as a power law in s





Appendix B. Universal quantum gates

Universal gate set

A set of gates is **universal** if, by composing gates from it, one can express **any unitary transformation** on any number of qubits.

- Since the n-qubit unitaries form an uncountable infinite set $U(2^n)$, this requires an infinite number of gates
- Example: $\{\mathrm{CNOT}, R_x(\theta), R_y(\theta), R_z(\theta)\}$, $\theta \in [0, 2\pi]$

Approximate universal gate set

A set of gates is **universal** if, by composing gates from it, one can **approximate any unitary transformation** on any number of qubits to **any desired precision**.

- Examples: $\{\mathrm{CNOT}, R_y(\pi/4), R_z(\pi/2)\}$, $\{\mathrm{Toffoli}, H, R_z(\pi/2)\}$
- Approximation can be done efficiently (Solovay–Kitaev theorem, $\mathcal{O}(\mathrm{polylog}(1/\varepsilon))$)