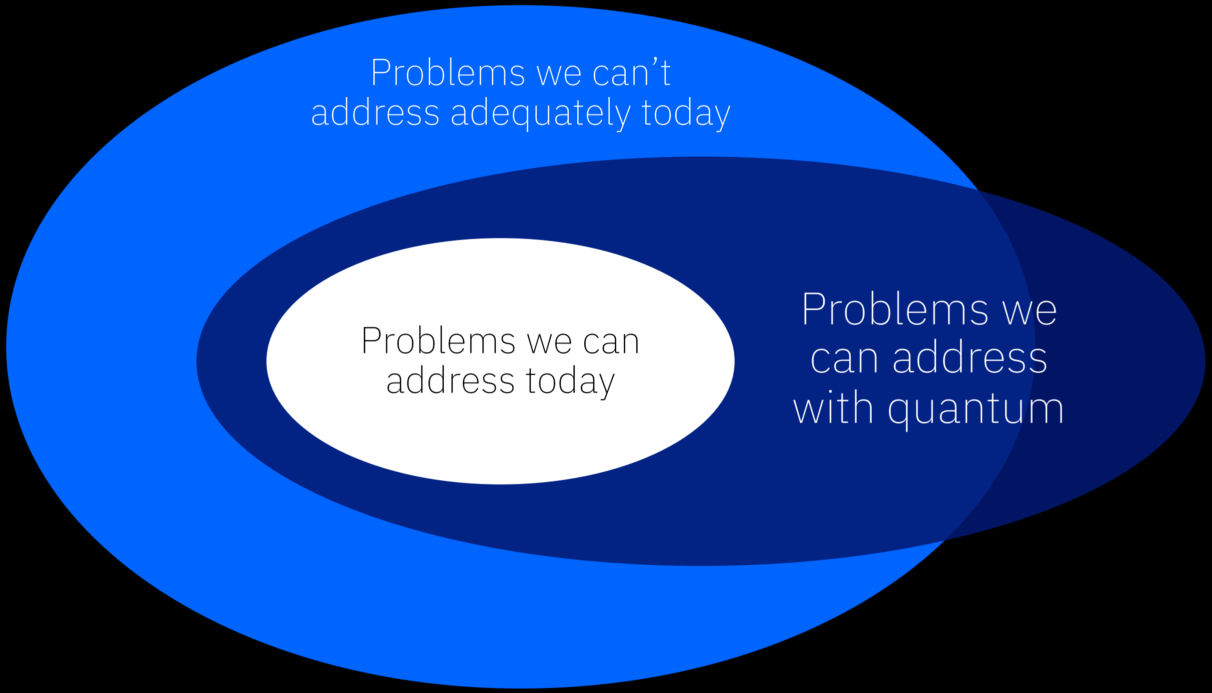
IBM Quantum: An Introduction

[Michele Grossi]
IBM **Technical** Quantum Ambassador

Why quantum?



Despite how sophisticated digital computing has become, there are many scientific and business problems for which we’ve barely scratched the surface.

Quantum applications span three general areas

Simulating Quantum Systems

Quantum chemistry
Material science
High energy physics

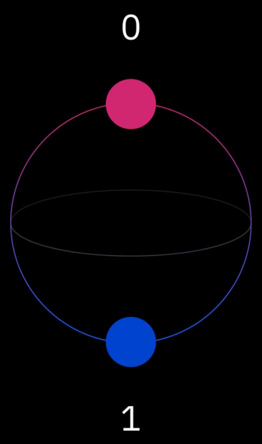
Artificial Intelligence

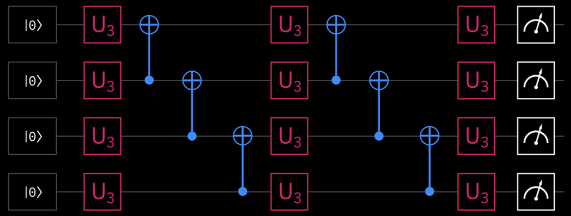
Better model training
Pattern recognition
Fraud detection

Optimization / Monte Carlo

Portfolio optimization
Risk analysis
Loans & credit scoring
Monte Carlo-like applications

Quantum bits and quantum circuits



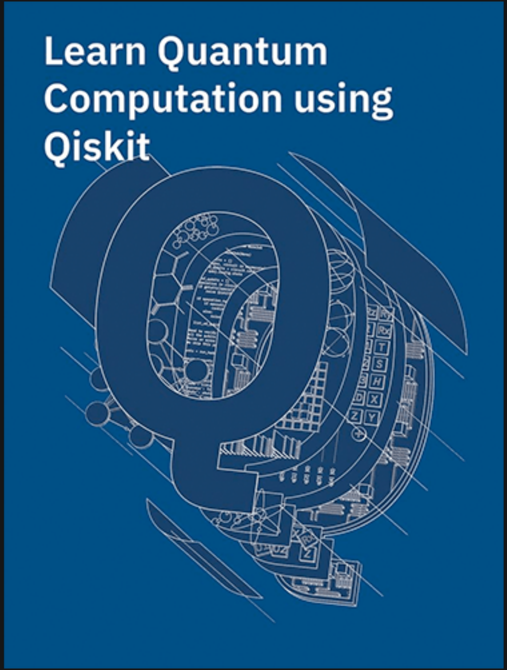


A quantum bit or **qubit** is a controllable quantum object that is the unit of information

A **quantum circuit** is a set of quantum gate operations on qubits and is the unit of computation

Open Source Textbook

community.qiskit.org/textbook



Traditional Quantum Computation Course

Linear Algebra
Quantum Mechanics → Quantum Algorithms → Quantum Hardware

Learn Quantum Computation using Qiskit Textbook

Python
Qiskit → Quantum Programming → Quantum Algorithms on Today's Hardware

Chapters:

0. Prerequisites
1. Quantum States and Qubits

2. Single Qubits and Multi-Qubit Gates
3. Quantum Algorithms
4. Quantum Algorithms for Applications

5. Investigating Quantum Hardware Using Qiskit

6. Implementations of Recent Quantum Algorithms

Our intuition about what we can compute is wrong

Are quantum computers "faster"?

$$p * q = N$$

How long does it take to multiply 2048 bit integers ?

Classical Cost of multiplication [1]:
~ 0.0025s

Quantum Cost of multiplication [2]:
~ 75.0000s

[1]: A. Emerencia,. "Multiplying huge integers using fourier transforms." (2007).

[2]: C. Gidney, Craig, and M. Ekerå. arXiv preprint arXiv:1905.09749 (2019).

Are quantum computers “faster”?

$$N = p * q$$

How long does it take to factor 2048 bit integers ?

Classical Cost of factoring [1]:
~ 4.7 billion CPU years

(largest factored number RSA-768 bit for approx. 1500 CPU years)

[1]: Kleinjung, Thorsten, et al. "Factorization of a 768-bit RSA modulus." Annual Cryptology Conference. Springer, Berlin, Heidelberg, 2010.

Quantum Cost of factoring [2]:
~ 8 hours

[2]: C. Gidney, Craig, and M. Ekerå. arXiv preprint arXiv:1905.09749 (2019).

Problems for a quantum computer

Modeling nature at the quantum level

Mathematics

Quantum chemistry,
Material science and
High energy physics

Algebraic algorithms
with provable speed-ups

- Shor’s Algorithm
- Grover’s Algorithm

Empirical heuristic
algorithms

- Optimization
- Machine learning

Quantum Computing Applications

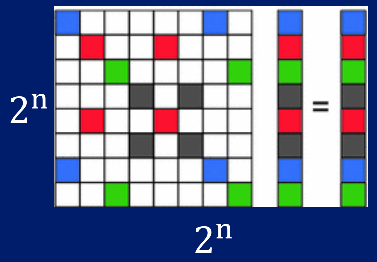
Quantum Simulations

Physics

Chemistry

Materials discovery

Linear Systems ($\mathbf{Ax} = \mathbf{b}$)



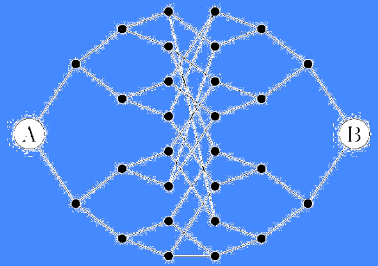
Network analysis

Differential equations

Option pricing, heat transfer

Classification (Machine Learning)

Quantum Walks



Graph properties (network flows, electrical resistance)

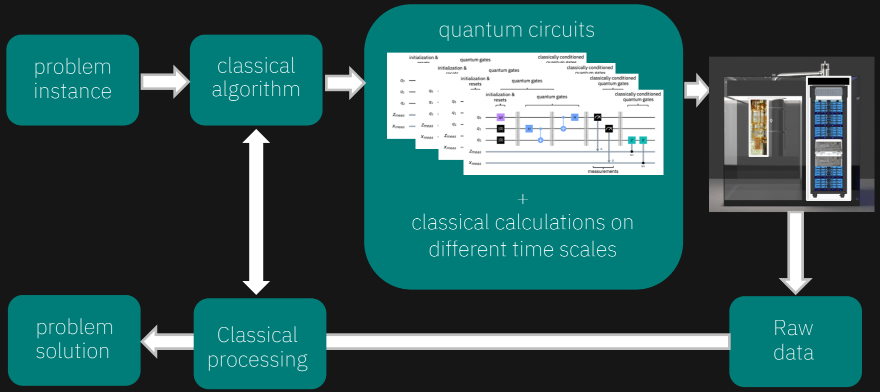
Search

Collision finding

How we program
Two critical program elements

Program modules

Reusable building blocks to construct programs that manage quantum and classical resources at runtime.



Developer size > 100000 (Model)

Applications modules: Optimization,

Use a quantum computer without ever directly referring to qubits or circuit (frictionless development).

```
# describe the problem
qubo = QuadraticProgram()
qubo.binary_var('x')
qubo.binary_var('y')
qubo.binary_var('z')
qubo.minimize(linear=[1, -2, 3], quadratic={('x', 'y'): 1,
                                            ('x', 'z'): -1,
                                            ('y', 'z'): 2})
```

```
# choose a solver
qaoa = MinimumEigenOptimizer(QAOA(backend))

# solve it
result = qaoa.solve(qubo)
```

QAOA is a quantum program

Developer size > 1Million (no quantum)

qiskit 0.22.0

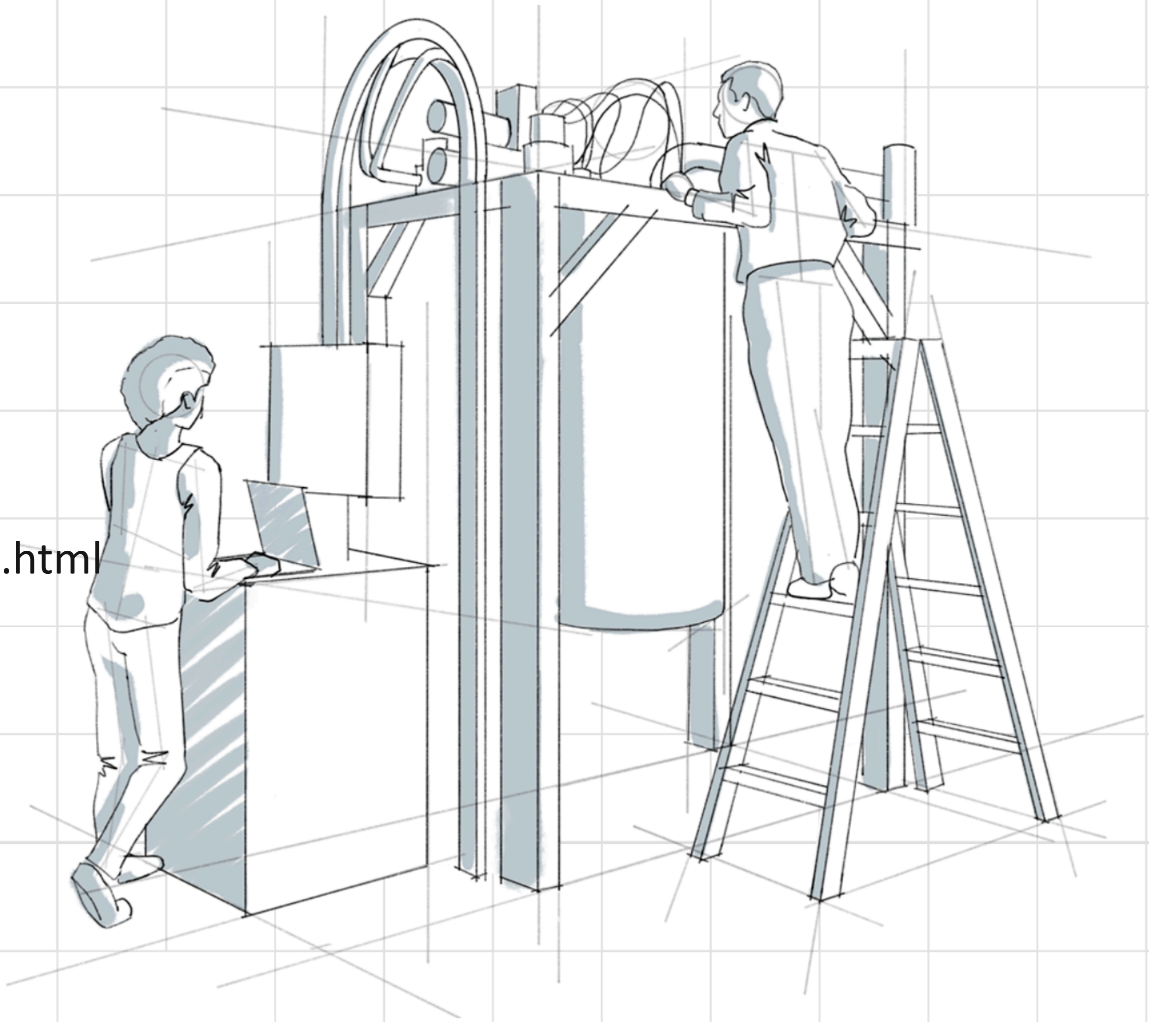
see release notes

Open-Source Quantum Development

https://qiskit.org/textbook/preface.html

Qiskit [kiss-kit] is an open source SDK for working with quantum computers at the level of pulses, circuits and algorithms.

Get started →

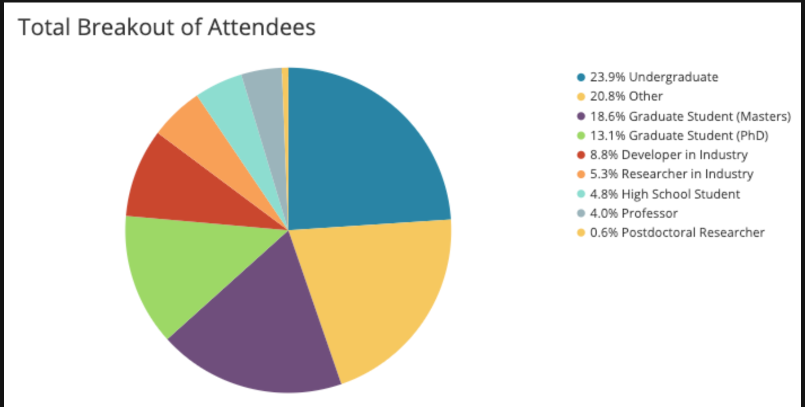


Qiskit Summer School 2020

Total registrations: 4,084

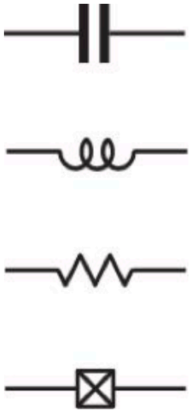
Total Lecture attendees: 3,915

Total Lab participants: 1,488



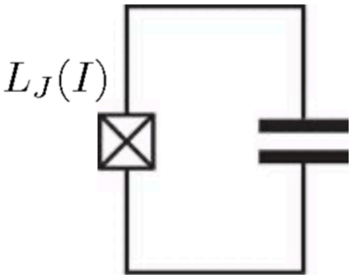
Constructing Nonlinear Quantum Electronic Circuits

Basic circuit elements

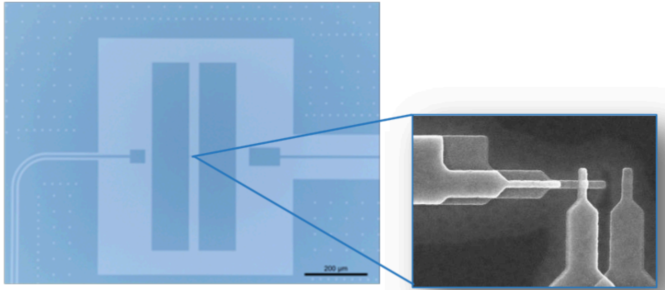


Josephson junction:
a non-dissipative
nonlinear element
(inductor)

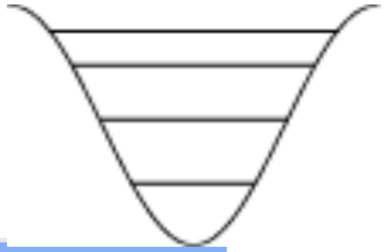
anharmonic oscillator



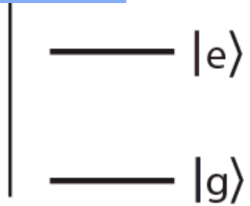
current-dependent
inductance

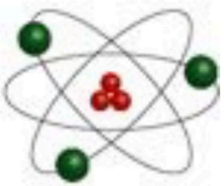


Non-linear energy level spectrum



electronic
artificial atom





M. H. Devoret, A. Wallraff and J. M. Martinis, *condmat/0411172* (2004)

Dilution refrigerator

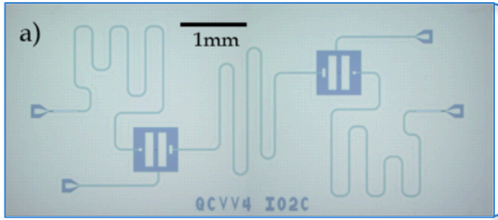
Dilution refrigeration to 10mK

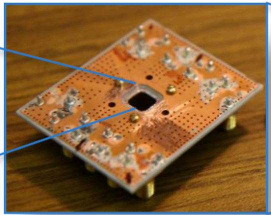
- $10\mu W$ cooling power
- $1\ nW$ microwave power for qubit control

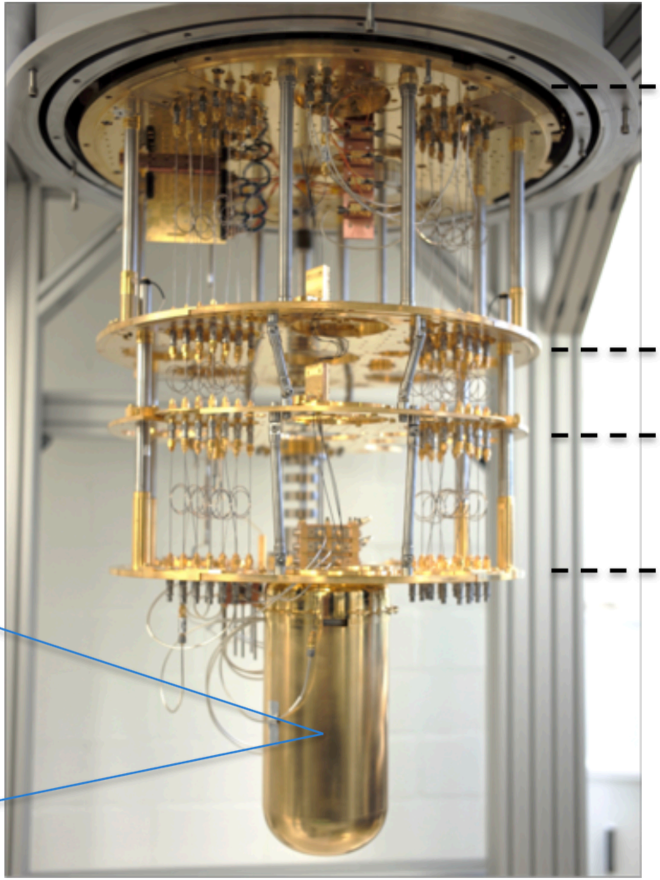
Can accommodate > 10000 qubit

Protection from environment

Earth magnetic field, cell phone signals, highway, ….

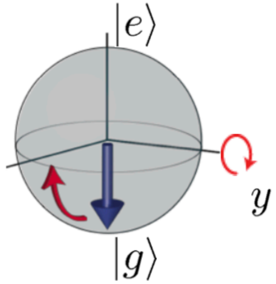


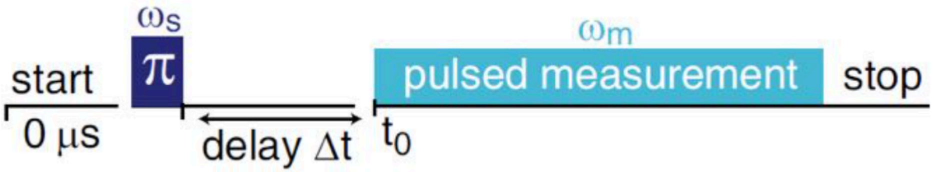




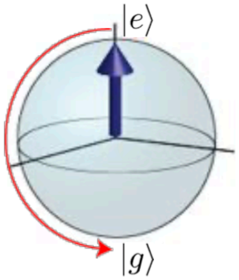
Relaxation time (T_1) measurements

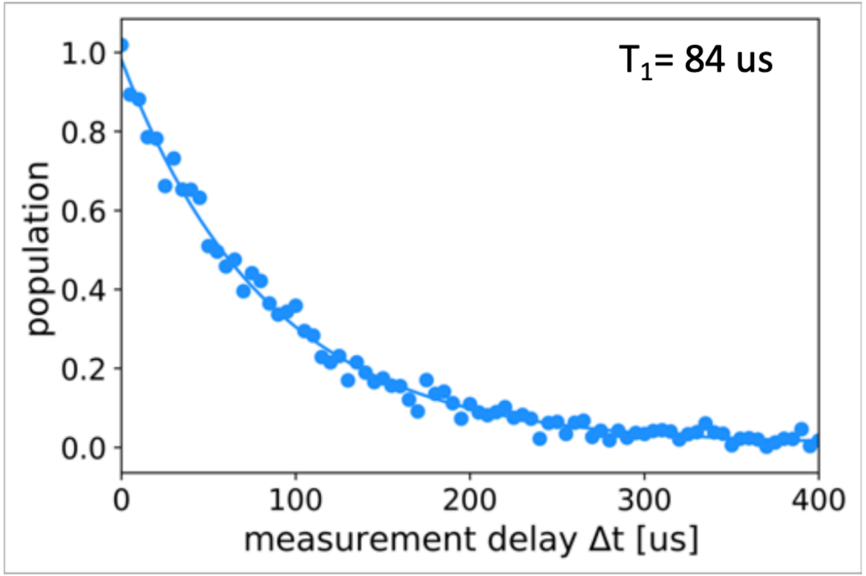
pulse scheme:





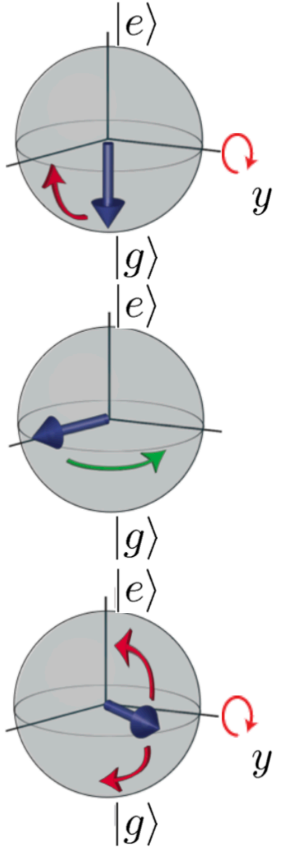
decay

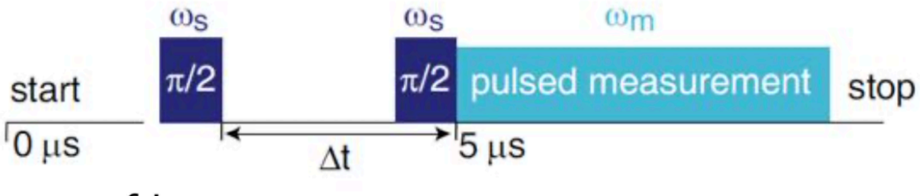




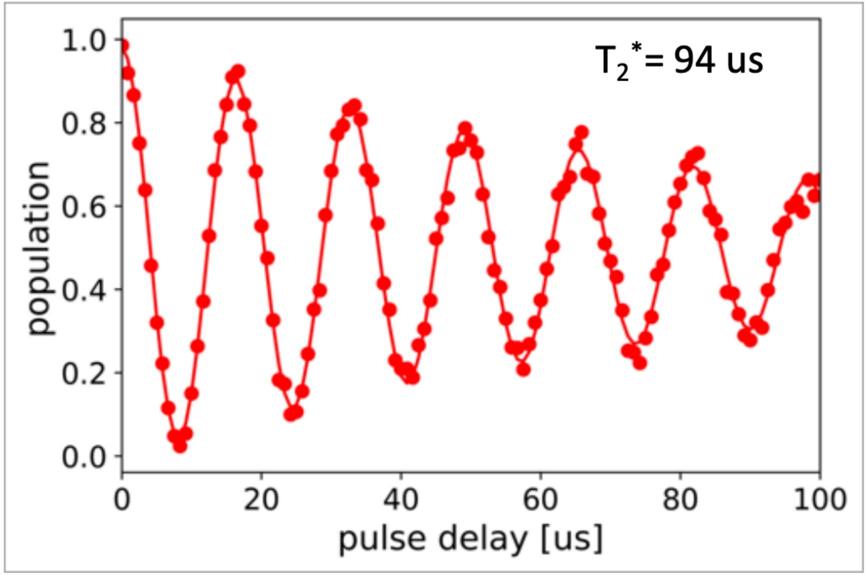
Coherence time (T_2^*) measurements

pulse scheme:





Ramsey fringes:



Time propagation algorithm

Given an Hamiltonian H, time propagations deals with the calculation of

$$|\psi(t)\rangle = e^{-iHt}|\psi(0)\rangle$$

where

- we use atomic units $\hbar = 1$.
- $|\psi(0)\rangle$ in encoded in a qubit register.
- the propagation operators e^{-iHt} is encoded as a series of gate operations
- $|\psi(t)\rangle$ is read qubit-by-qubit at the circuit end.

The Hamiltonian is assumed to be expressed as a Pauli string, meaning a tensor product of a sequence of Pauli matrices.

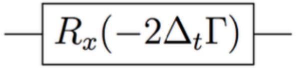
$$H = \sum_i g_i \, \sigma_{i_1} \otimes \sigma_{i_2} \otimes \ldots \sigma_{i_N}$$

with $i = \{i_1, i_2, \ldots, i_N\}$.

Time propagation algorithm

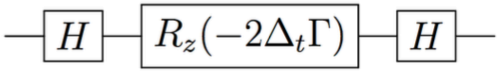
One-qubit rotations

Case 1: $e^{i\Delta_t \Gamma \sigma_x}$:



Case 2: transform it to a $e^{i\Delta_t \Gamma \sigma_z}$ rotation by applying a change of basis.

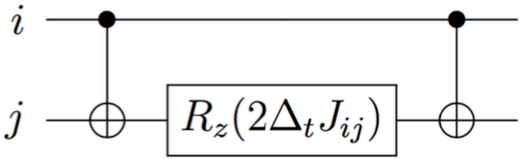
In this case we have first to make a basis transformation and then apply the rotation in z:



Two-qubit rotations

Case 3: $e^{i\delta\sigma_z\sigma_z} = e^{i\delta\sigma_z\otimes\sigma_z}$ $(\delta = \Delta_t\Gamma)$:

We will show that this can be computed using:



Time propagation algorithm

We first derive the matrix for $e^{i\delta\sigma_z\otimes\sigma_z}$

$$\sigma_z\otimes\sigma_z = \begin{pmatrix}1&0&0&0\\0&-1&0&0\\0&0&-1&0\\0&0&0&1\end{pmatrix} \quad\rightarrow\quad e^{i\delta\sigma_z\otimes\sigma_z} = \begin{pmatrix}e^{i\delta}&0&0&0\\0&e^{-i\delta}&0&0\\0&0&e^{-i\delta}&0\\0&0&0&e^{i\delta}\end{pmatrix}$$

and

$$e^{i\delta\sigma_z\otimes\sigma_z} = \begin{pmatrix}1&0&0&0\\0&1&0&0\\0&0&0&1\\0&0&1&0\end{pmatrix}\begin{pmatrix}e^{-i\delta}&0&0&0\\0&e^{i\delta}&0&0\\0&0&e^{-i\delta}&0\\0&0&0&e^{i\delta}\end{pmatrix}\begin{pmatrix}1&0&0&0\\0&1&0&0\\0&0&0&1\\0&0&1&0\end{pmatrix}$$

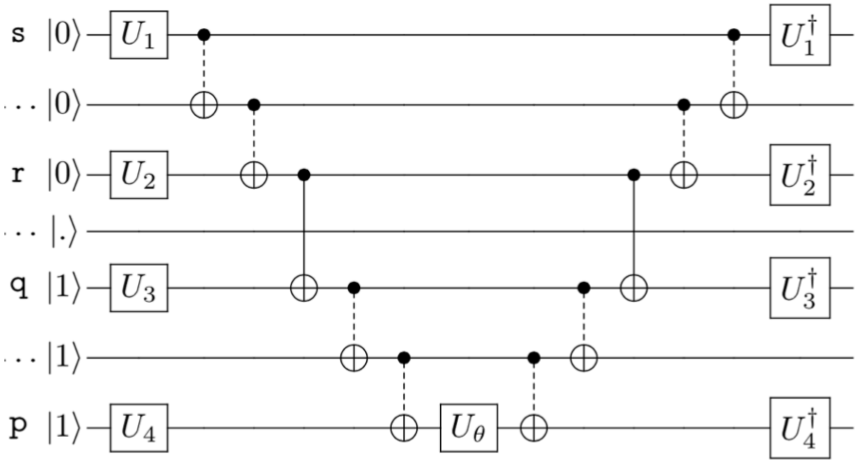
$$= \quad (\text{CNOT}) \quad . \quad (\mathbb{I}\otimes R_z(\delta)) \quad . \quad (\text{CNOT})\,.$$

Time propagation algorithm

Exercise:

Prove that $e^{i\delta\sigma_z\otimes\sigma_z}$ performs a rotation of $-\delta$ if the two qubits are in the same state, $|00\rangle, |11\rangle$ and of $+\delta$ when the spins are opposite, $|10\rangle, |01\rangle$.

This result can be generalized to any number of qubits (see lecture on quantum chemistry)



$(\mathrm{U}_1, \mathrm{U}_2, \mathrm{U}_3, \mathrm{U}_4) = \{(\mathrm{H}, \mathrm{H}, \mathrm{Y}, \mathrm{H}), (\mathrm{Y}, \mathrm{H}, \mathrm{Y}, \mathrm{Y}), (\mathrm{H}, \mathrm{Y}, \mathrm{Y}, \mathrm{Y}), (\mathrm{H}, \mathrm{H}, \mathrm{H}, \mathrm{Y}),$

Python Suggestion:

http://personalpages.to.infn.it/~maina/didattica/TIF_2020/

LEZ:6,7,8,9

https://realpython.com/learning-paths/python3-introduction/

IBM **Quantum**

michele.grossi@it.ibm.com