Quantum Programming

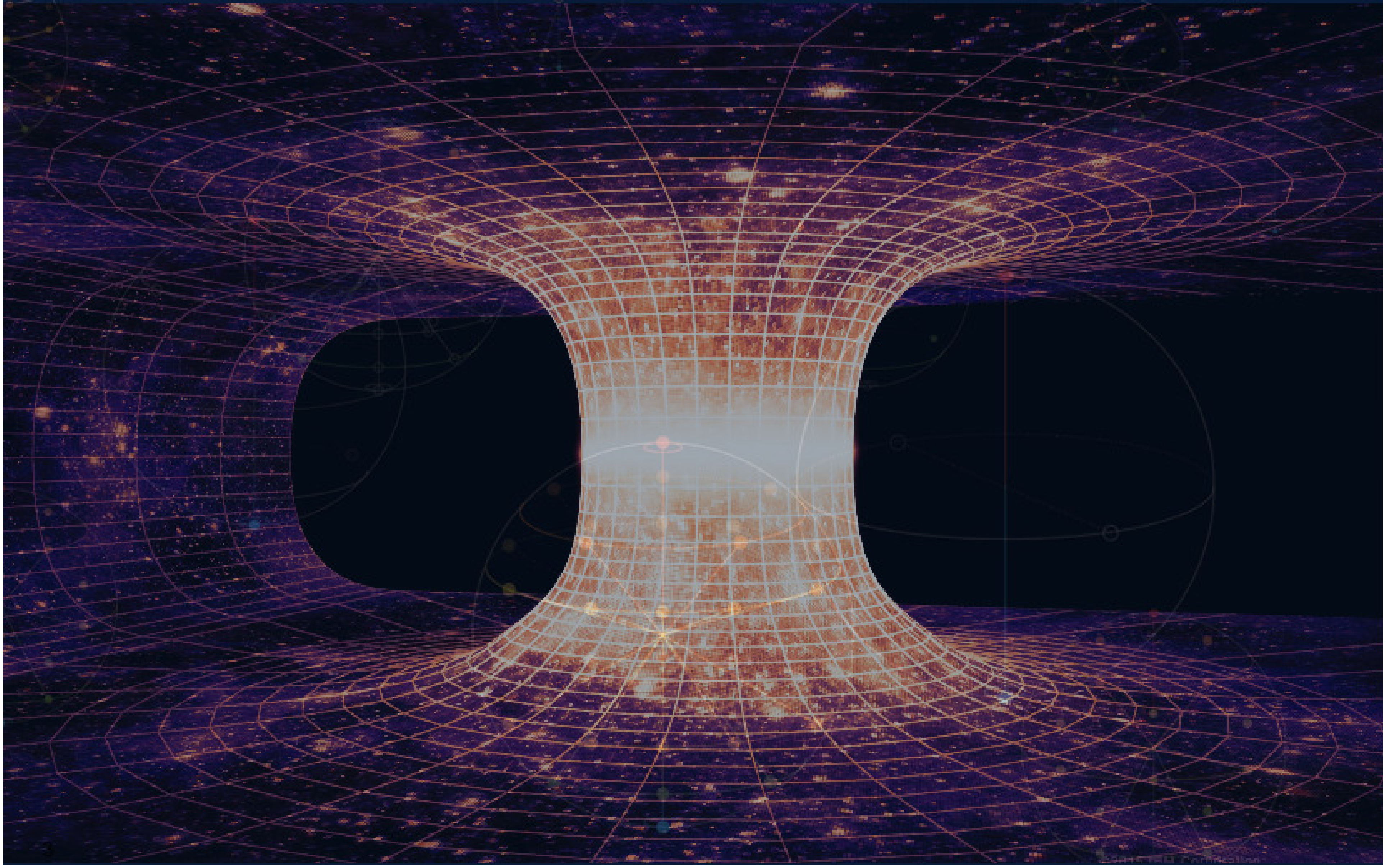
Brief introduction to quantum programming languages

Francisco J, Gálvez Ramírez
fjgramirez@es.ibm.com
IBM Technical Staff

Agenda

- Basic Concepts
- Quantum Algorithms
- Programming Languages
- IBM Quantum Experience

Basic Concepts



What is a quantum computer

- A Quantum Computer makes use of the natural laws of quantum mechanics to carry out a calculation.

- Why do we need a quantum computer
  - **Resolution of some specific problems** → Some problems cannot be treated with classical computer with full reliability.
  - **Performance** → Faster troubleshooting than a classical computer.

Basic Concepts in Quantum Mechanics

- ## Uncertainty Principle

It is impossible to carry out a measurement on a system without disturbing the system

- ## State Superposition

Every state can exists as any possible conifguration in the states space

- ## Quantum Entanglement

EPR Paradox – There's a relationship among the features of entangled particles.

- ## State Decoherence

In a coherent state all the quantum properties remain on every component identified as being part of the system. Decoherence returns the individual identity to each component and drops off the quantum characteristics

Features of a Quantum Computer

1. Use of Quantum Bits or Qubits

2. Make use of Quantum Parallelism

3. Quantum Entanglement

4. Keeps coherence

What is a Quantum bit or Qubit?

- A Qubit is the quantum concept of a bit

- It is either an element nor a device. A Qubit is a logical concept that can be implemented on a vast number of systems with a quantum behaviour.

- As a bit, the Qubit can be in two base states: 0 and 1.

However, a Qubit can work with all the possible combinations that can be built with these base states 0 and 1

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$$

Quantum Operations

Quantum Gates

- It is a basic circuit that acts on one or several qubits
- It is equivalent to logical gates in digital circuits

1. Quantum Gates are Reversible
2. Matematically are represented by unitary matrices.
3. The qubits upon they act keep their quantum nature.

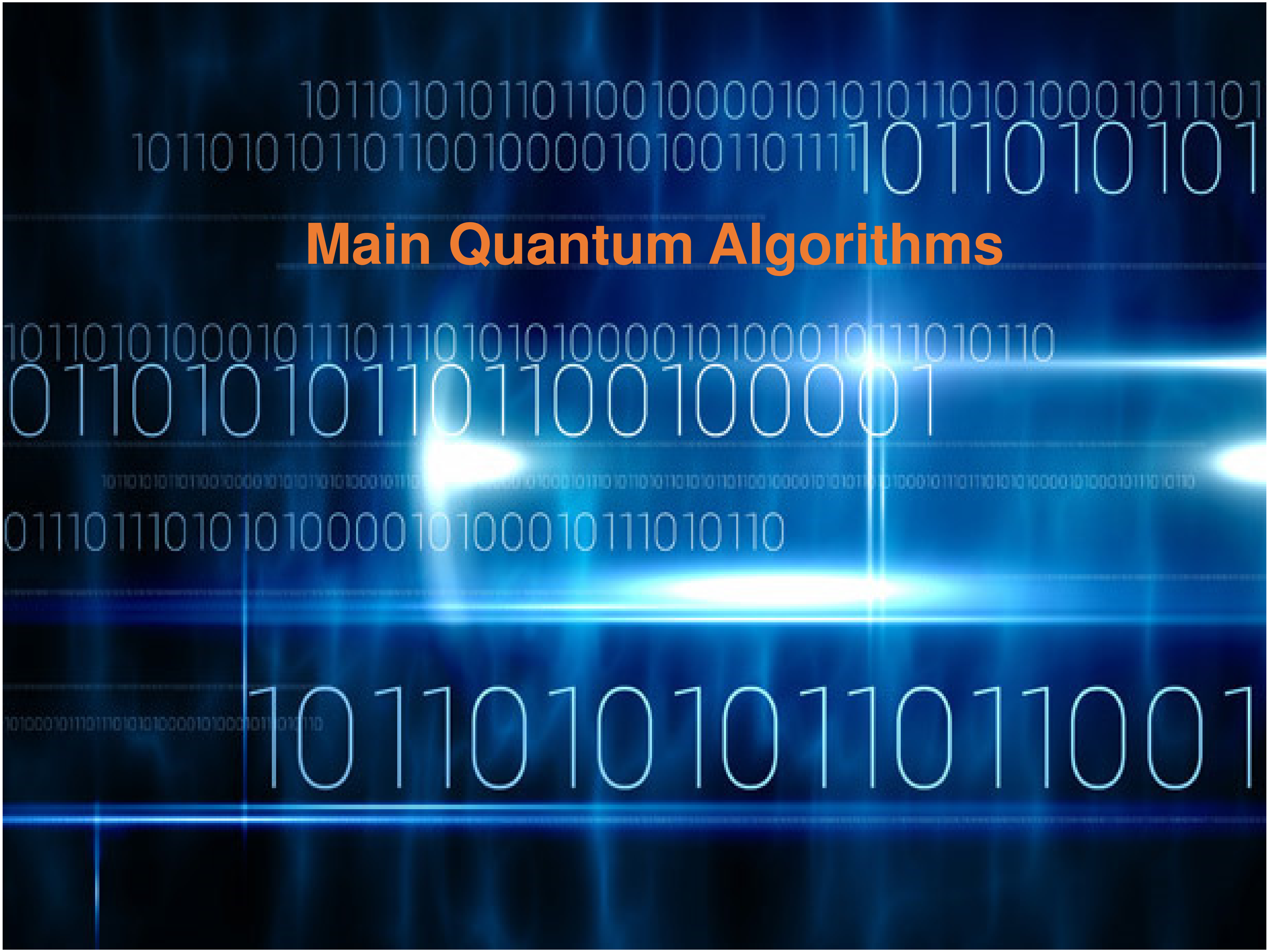
$$\boxed{H} = \frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$$

Hadamard Gate

$$\boxed{\text{CNOT}} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

CNOT Gate

Main Quantum Algorithms



Main Quantum Algorithms

- **Deusch Algorithm** – Determines whether a funcion is balanced or unbalanced
- **Shor Algorithm** – Large numbers factorization
- **Grover Algoritm** – Search in unstructured spaces

Deustch Algorithm

f_1:

$0 \longrightarrow 0$

$1 \longrightarrow 0$

f_2:

$0 \longrightarrow 1$

$1 \longrightarrow 1$

f_3:

$0 \longrightarrow 0$

$1 \longrightarrow 1$

f_4:

$0 \longrightarrow 1$

$1 \longrightarrow 0$

Deustch-Josza's Algorithm → Extend of the Deustch's algorithm for records with n values

Shor Algorithm

- Number of steps that a classic computer needs to run in order to find the prime factors of a number N of x digits

It grows exponencially with x

- The Shor algorithm is made up of two parts:

   1. One Classical part - Which focuses on finding the period of a function

   2. A quantum part based on QFT technics

In 2001, IBM and Stanford University, executed for the first time the Shor algorithm in the first quantum computer of 7 qubits developed in Los Álamos.

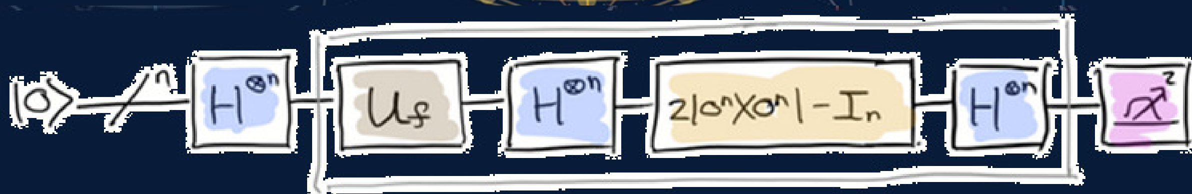
https://www-03.ibm.com/press/us/en/pressrelease/965.wss

Grover Algorithm

- How many attempts need a data search in an unordered N-element database to locate a particular element??

An average of N/2 attemps are needed)

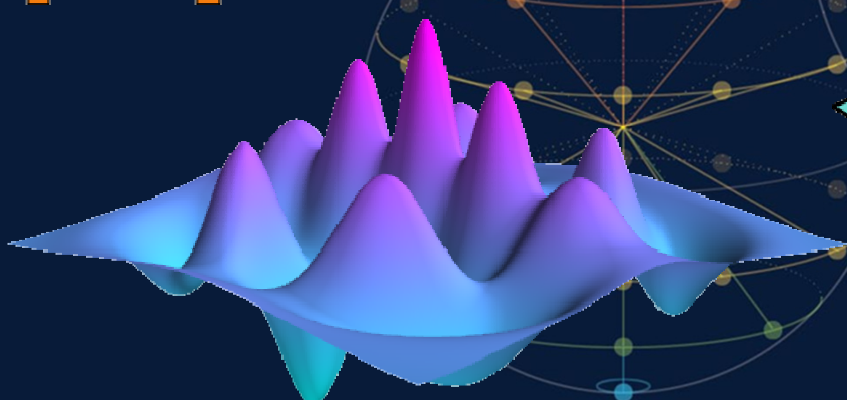
A quantum computing executing the Grover algorithm would run $\sqrt{N}$ attemps



http://www.dma.eui.upm.es/MatDis/Seminario4/AlgoritmoGrover.pdf

Programming Languages

C++



Quantum Pseudocode

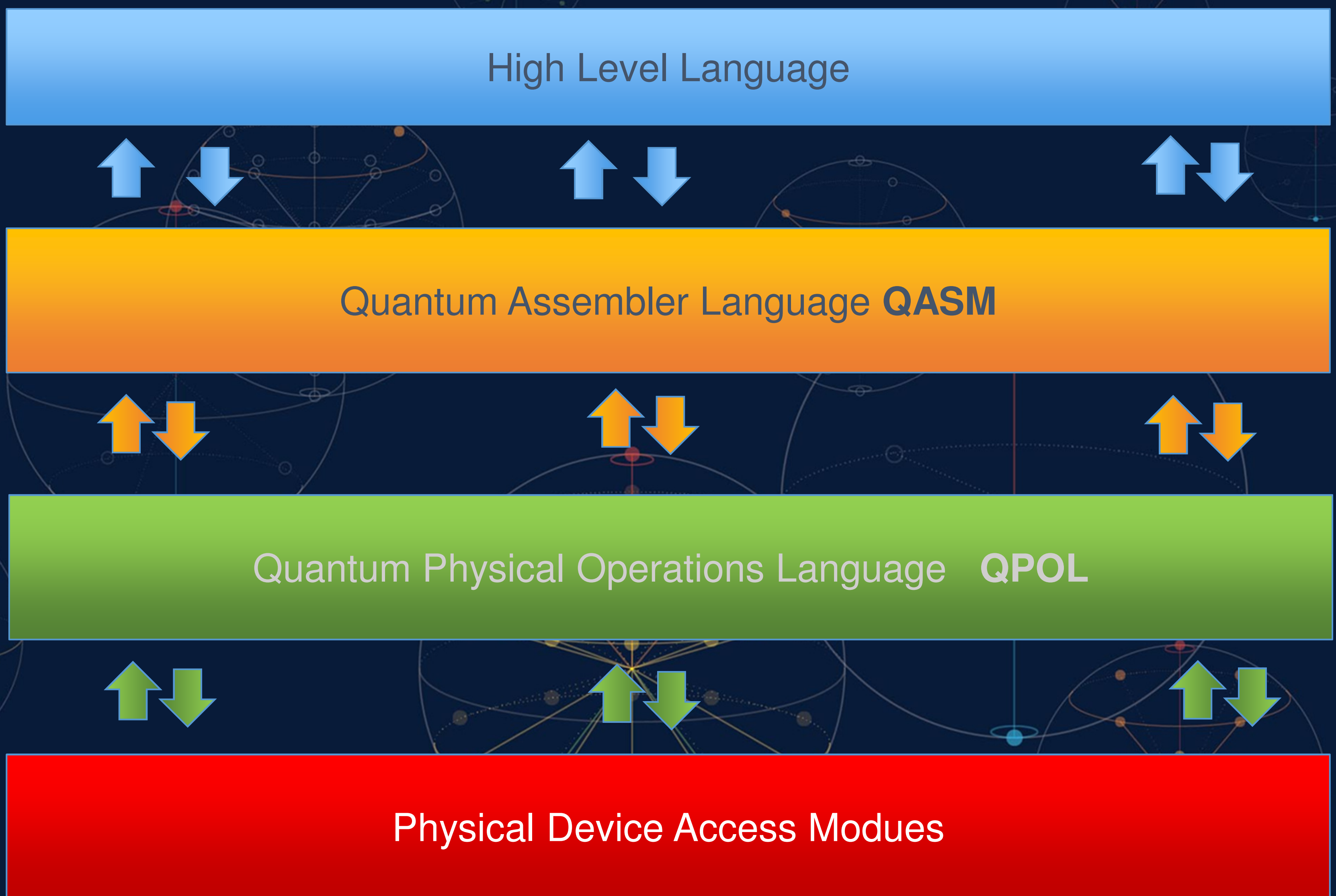
- The quantum pseudocode was initially proposed by E. Knill was the first forma language for quantum algorithms description.

- It is related with a quantum machine model know as QRAM (Quantum Random Access Machine)

- Source : https://www.researchgate.net/publication/51394884_Quantum_Random_Access_Memory

Four Layer Architectural Proposal

- **Layer's architecture**



Quipper – A Haskel Library

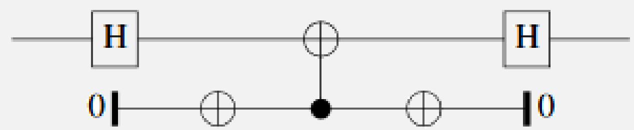
- Published in 2013.
- It is an embeded language based on Haskel, developed as part of the IARPA ‘s QCS project
- The quantum programs are written in Haskel adding the appropiate libraries
- Quipper is a circuit description language
- Example:

```
import Quipper
spos :: Bool -> Circ Qubit
spos b = do
        q <- qinit b
        r <- hadamard q
        return r
```

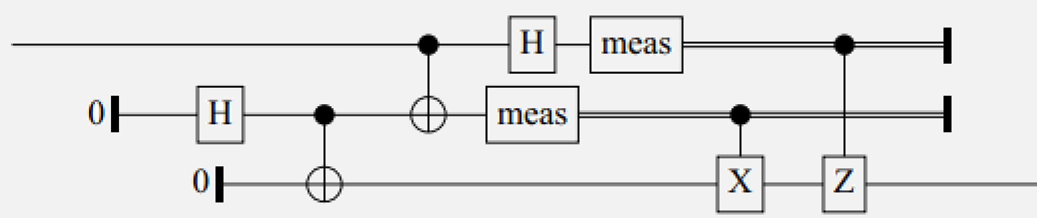
Quipper – A Haskel Library

Examples of circuit generation code for Quipper

```
circ :: Qubit -> Circ Qubit
circ x = do
    hadamard_at x
    with_ancilla $ \y -> do
        qnot_at y
        qnot x `controlled` y
        qnot_at y
    hadamard_at x
    return x
```




```
teleport :: Qubit -> Circ Qubit
teleport q = do
    (a,b) <- bell00
    (x,y) <- alice q a
    b <- bob b (x,y)
    return b
```



Introduction to Quipper: https://arxiv.org/pdf/1304.5485v1.pdf

Quipper – A Haskel Library

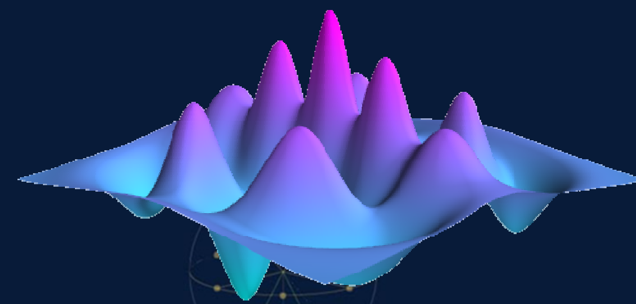
Algorithms used in Quipper development

- **BWT – Binary Welded Tree**. How to find out an identified node in a graph.
- **BF – Boolean Formula**. Evaluates NAND expresion.
- **CL – Class Number**. Aproximation of a group class to a real cuadratic number.
- **GSE – Ground State Estimation**. Calculate the lowers energy level of a particular molecule.
- **QLS – Quantum Linear System**. Solving of a linear system of equations.
- **USV – Unique Shortest Vector**. Finding out the shortest vector in a given set of vectors
- **TF – Triangle Finding**. Drawing of a triangle in a dense graph

Quantum Primitives:

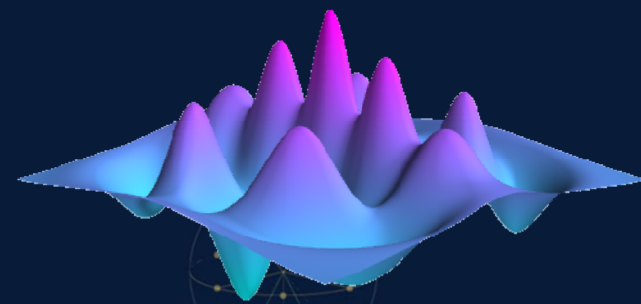
- Quantum Fourier Transform
- Amplitude Amplification
- Quantum Walk
- ...

The Python Quantum Toolbox

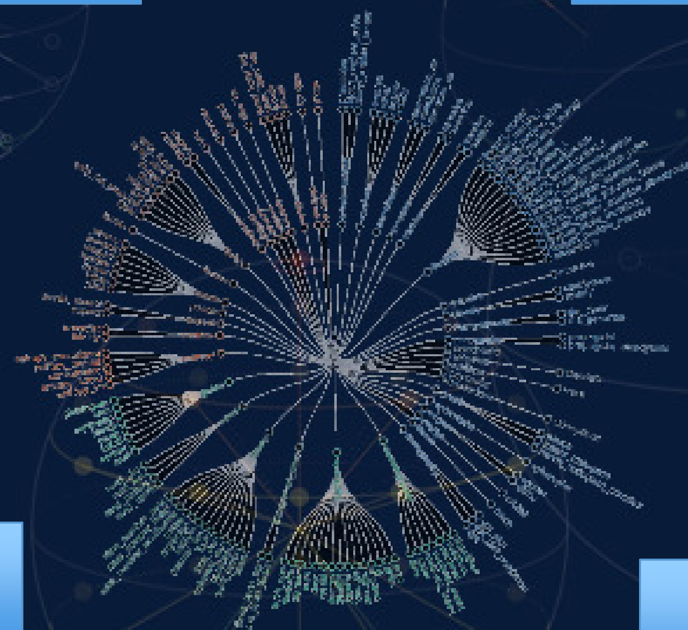


Authors:	Paul Nation and Robert Johansson
Web site:	http://qutip.googlecode.com
Discussion:	Google group “qutip”
Blog:	http://qutip.blogspot.com
Platforms:	Linux and Mac
License:	GPLv3
Download:	http://code.google.com/p/qutip/downloads
Repository:	http://github.com/qutip
Publication:	Comp. Phys. Comm. **183**, 1760 (2012) arXiv:1211.6518 (2012)

The Python Quantum Toolbox



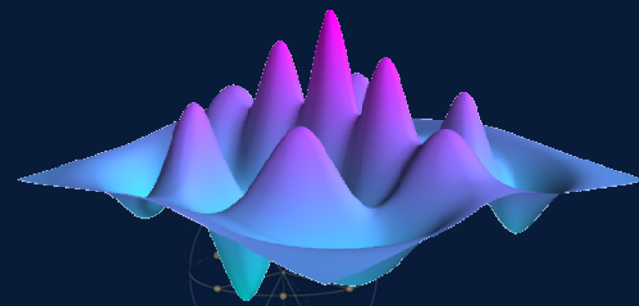
QuTiP – Is an object oriented open source framework for open quantum systems calculus



- States
- Temporal Evolution
- Core Functions
- Visualization
- Operators
- Logic Gates

QuTiP Documentation: http://qutip.org/downloads/3.1.0/qutip-doc-3.1.0.pdf

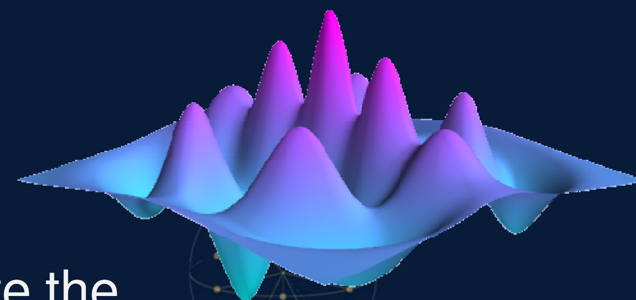
The Python Quantum Toolbox



- **Relationship between quantum concepts and their represetation in QuTiP**

Quantum Concept	QuTiP Represenation
Quantum State or Wave Function Amplitude of probability that describes the state of the quantum system.	**Vectors and Matrices** Complex elements
Operators The Hamiltonian operator es the function that represents the total energy of a system and describes the energy of every possible state of the system. The operator represents physical observables	**Matrices** The operators are represented as matrices.
Equation of motion Describes how the states of the quantum system evolves in time.	**Diferential equations** Systems of coupled differential equations.
Observables and expected values Physical obsevables are quantities that correspond to operators	**Internal Product** The results are calculated as internal product between state vectors and matrices that represent operators, yielding as a result real numbers for physical observables

The Python Quantum Toolbox



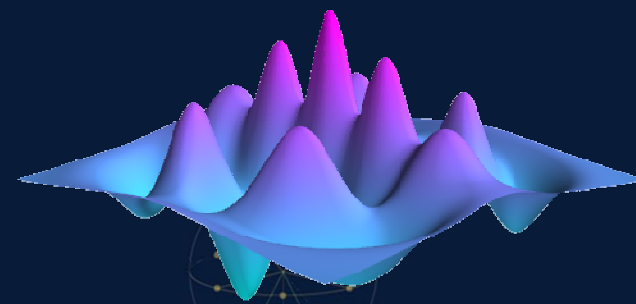
Objetcs and datatypes

Qobj provides the necessary structure to encapsulate the quantum operators and the vectors <bra| and |ket>

```
In [3]: Qobj()
Out[3]:
Quantum object: dims = [[1], [1]], shape = [1, 1], type = oper, isherm = True
Qobj data =
[[ 0.]]
```

Data	Q.data
Dimensions	Q.dims
Shape	Q.shape
is Hermitian?	Q.ishermIs
Type	Q.type

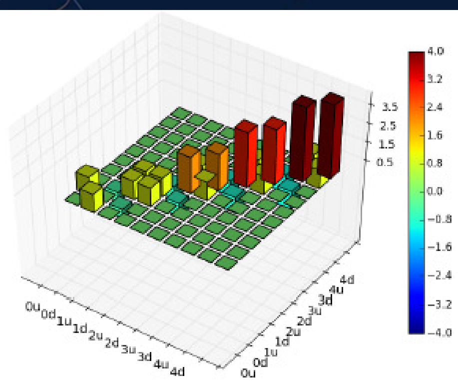
The Python Quantum Toolbox

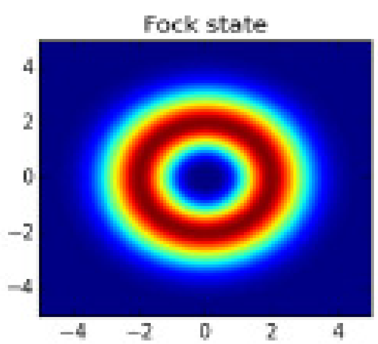


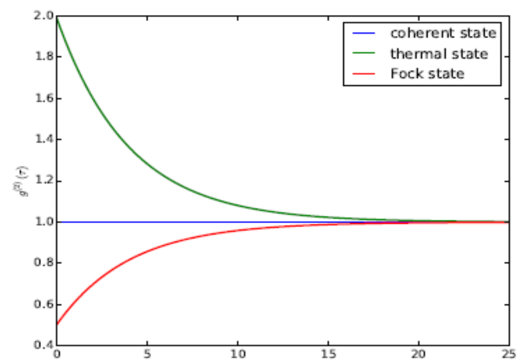
- Visualization capabilities

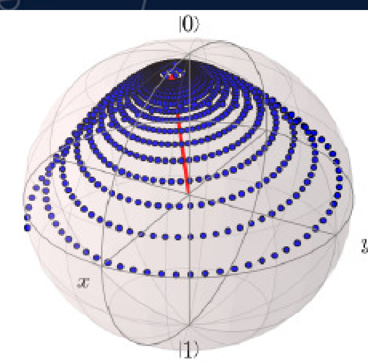
- Function for Distribution of Probability
- Operators Visualization
- Quantum Process Tomography

- 2D & 3D histograms
- Color Maps
- Lineal Graphs
- Bloch Sphere representation









Mathematica

List of Mathematica packages for Quantum Calculus

- **QDENSITY** - Quantum Computer Simulation. Density Matrices.

  http://www.pitt.edu/~tabakin/QDENSITY/UPDATE14.pdf

- **qmatrix** – Package for quantum information computation.

  http://library.wolfram.com/infocenter/MathSource/1893/

- **Quantum** Add-On that performa a vast quantity of computations and simulations in quantum mechanics. (University of Monterrey)

  http://homepage.cem.itesm.mx/lgomez/quantum/index.htm

- **CMU: Quantum Information Programs in Mathematica** – Library of functions and objetcs: <bra|ket> notation, operators, etc. (Carnegie-Mellon University)

  http://quantum.phys.cmu.edu/QPM/

- **Quantum Turing Machine Simulator** Oriented to Turing Quantum Machines

  http://www.mathematica-journal.com/issue/v8i3/features/hertel/contents/html/index.html

- **QI** – Package for quantum computation, mainliy focused on geometrical aspects of the quantum information theory.

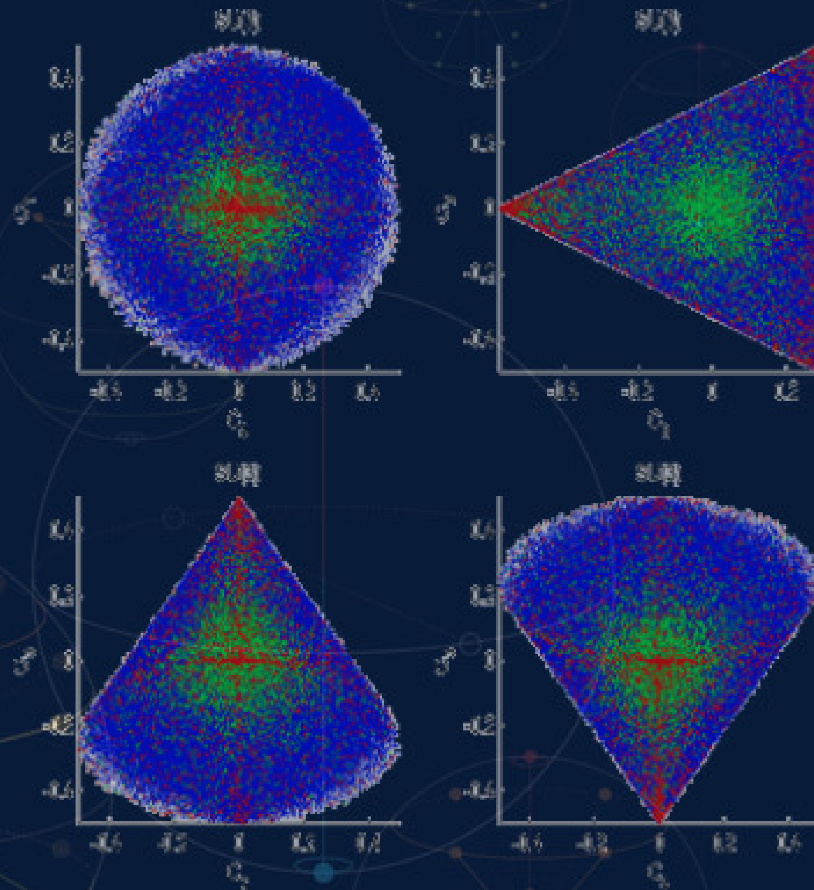
  https://zksi.iitis.pl/wiki/projects:mathematica-qi

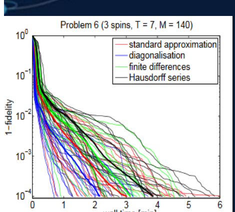
MatLab QLib

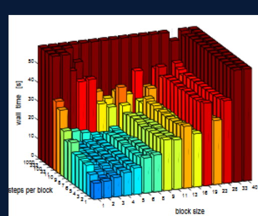
Matlab Library for Quantum Computation

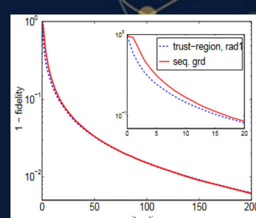
- Probability Distribution(clasical)
- Pure States
- Density Matrices
- Hermítian Matrices

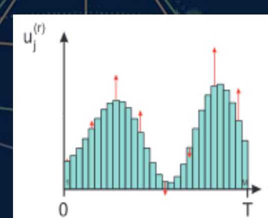
- Several entanglement test
- Lineal Entroy, de Von Neumann Entropy ...
- Distance Measurement (Trace, fidelity, Hilbert..)
- Schmidt Decomposition
- Observables Measurements (POVM)



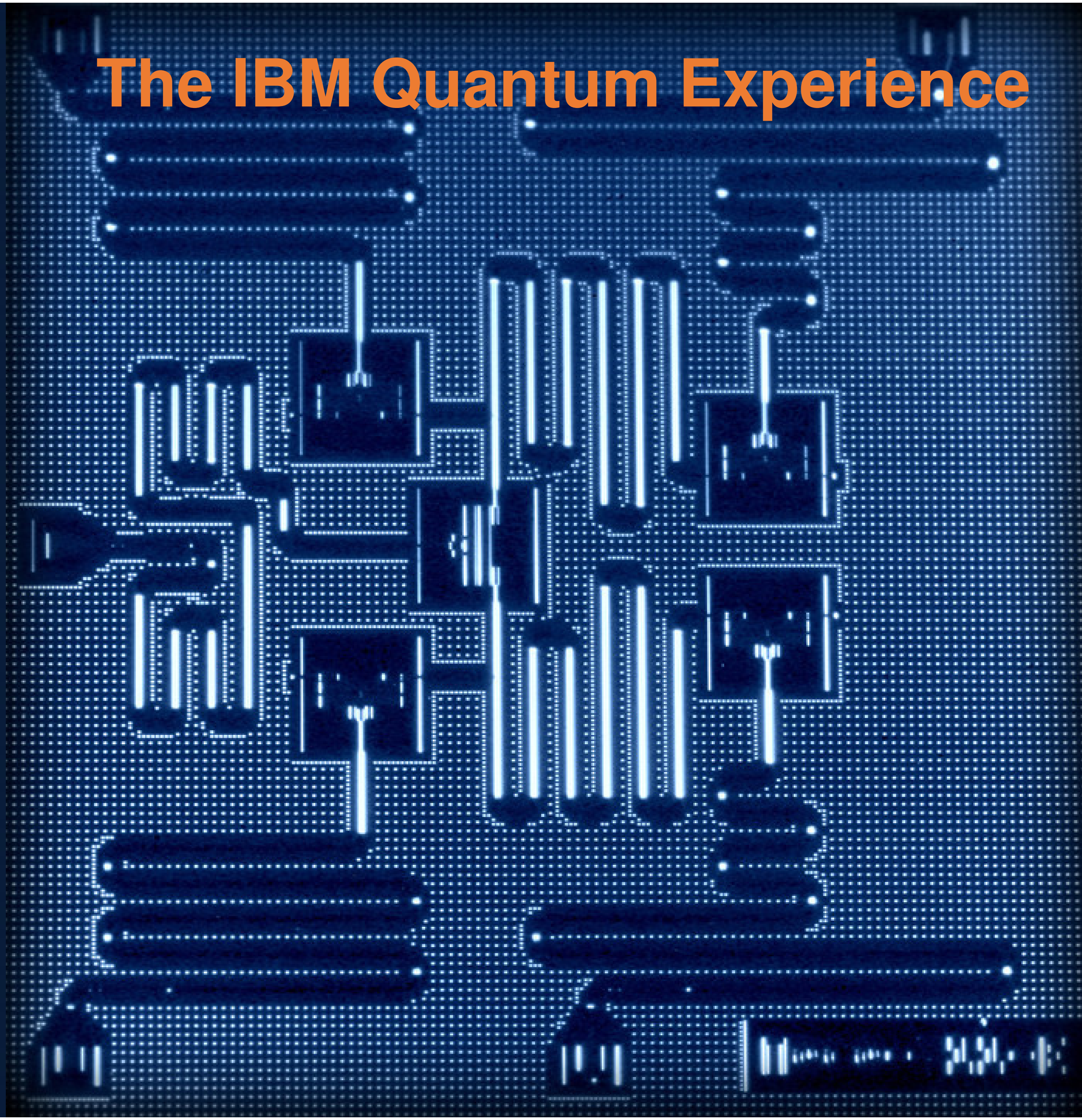








The IBM Quantum Experience



En que consiste IBM Quantum Experience

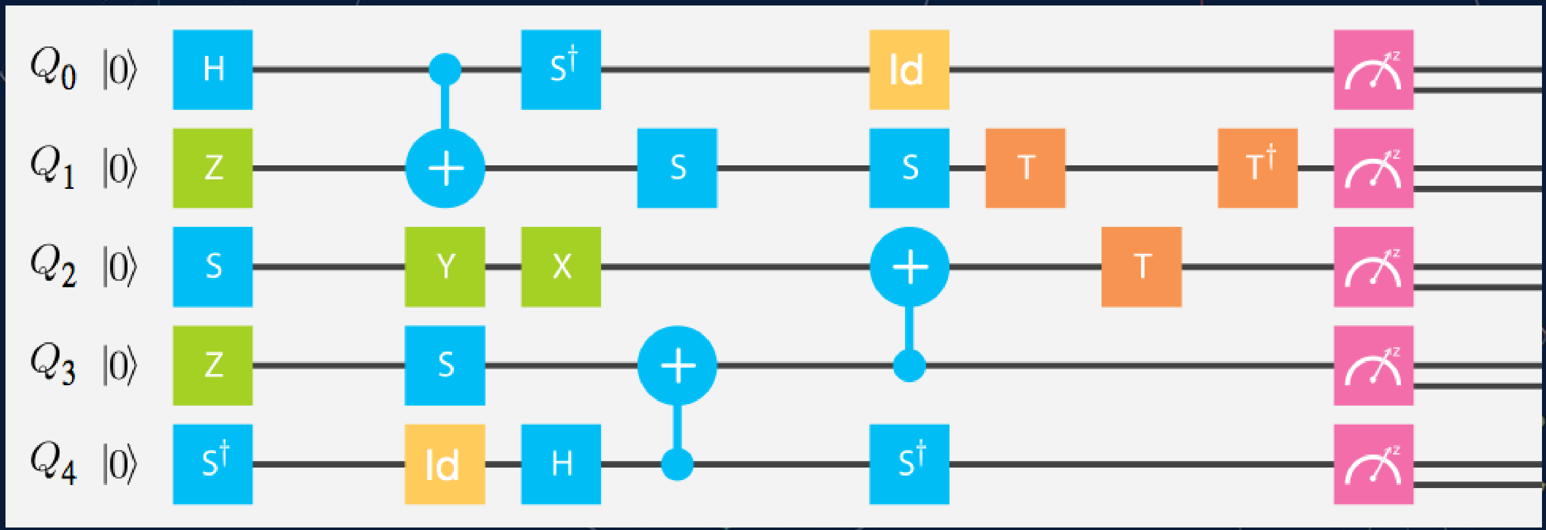
- A set of tutorials which is guide to understand all the quantum experiments.
- The ***quantum Composer,*** is a graphical interface where a quantum circuit can be designed.
- **A simulator** used to execute the quantum circuits designed in the composer.
- **Access to a real Quantum Processor** which is physcally located and working at Quantum Computing IBM Lab
- Under construction: **A Quantum Community**

Intoruducing the IBM Quantum Experience

- The standar user has total access to:
  - A real Quantum Processor
  - Simulation Capabilities
  - Memory cache with results from previous experiments.
- There’s a unique quantum processor available in the cloud.
- When the user consumes the computing units, new computing units can be acquired from the “Account Page”.

The Quantum Composer

- Graphical Interface used to create programs for the quantum processor
- It allows to create quantum circuits using a logical gates library and well defined points of measurement



La librerias de Operaciones Cuánticas

Id

Yellow Blocks. Are empty operation on a qubit for a unit of time which equals the duration of a gate for one qubit.

X

Green Blocks. These are the Operators the the Pauli Group (X, Y, Z).

H

Blue Blocks. *Clifford Operators*. These gates are H, S and S† and they are used to generate quantum superposition

T

Orange Blocks. These are the gates necessary for universal computation (Non-Cliford gates).

The Full Library ... (... at versión 1 ...)

Id	The identity gate performs an idle operation on the qubit for a time equal to the single-qubit gate duration.	X	The Pauli X gate is a π-rotation around the X axis and has the property that $X \rightarrow X$, $Z \rightarrow -Z$. Also referred to as a bit-flip.	Z	The Pauli Z gate is a π-rotation around the Z axis and has the property that $X \rightarrow -X$, $Z \rightarrow Z$. Also referred to as a phase-flip.	Y	The Pauli Y gate is a π-rotation around the Y axis and has the property that $X \rightarrow -X$, $Z \rightarrow -Z$. This is both a bit-flip and a phase-flip, and satisfies $Y = XZ$.
H	The Hadamard gate has the property that it maps $X \rightarrow Z$, and $Z \rightarrow X$. This gate is required to make superpositions.	S	The Phase gate that is $\sqrt{Z}$ and has the property that it maps $X \rightarrow Y$ and $Z \rightarrow Z$. This gate extends H to make complex superpositions.	$S^{\dagger}$	The Phase gate that is the transposed conjugate of S and has the property that it maps $X \rightarrow -Y$, and $Z \rightarrow Z$.	+	Controlled-NOT gate: a two-qubit gate that flips the target qubit (i.e. applies Pauli X) if the control is in state 1. This gate is required to generate entanglement.
T	The Phase gate that is $\sqrt{S}$, which is a $\pi/4$ rotation around the Z axis. This gate is required for universal control.	$T^{\dagger}$	The Phase gate that is the transposed conjugate of T.		Measurement in the computational (standard) basis (Z).		Bloch measurement: Tomography of the individual qubits.

Working with the Composer
(Demo)