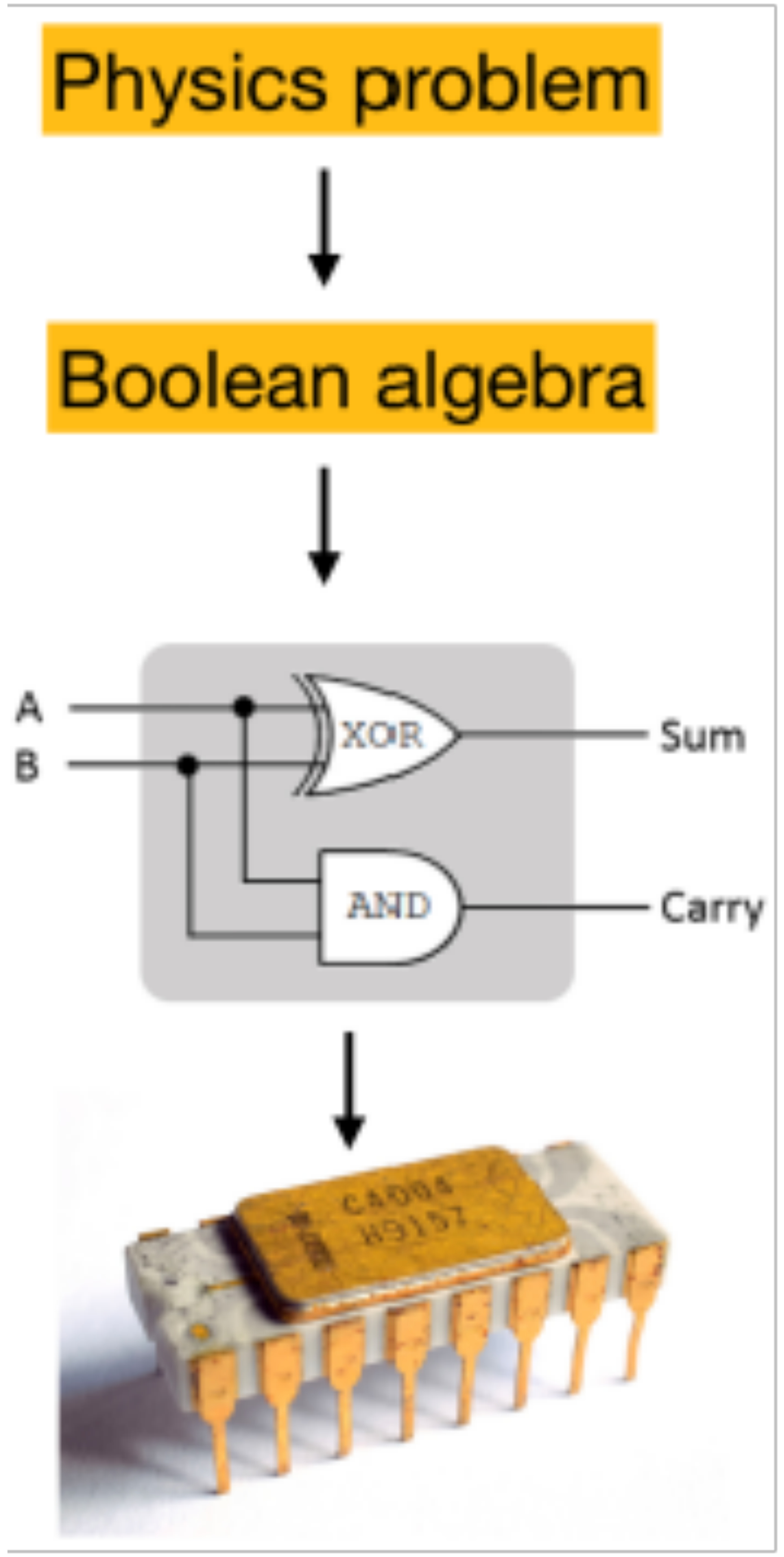
中国科学院高能物理研究所
Institute of High Energy Physics, Chinese Academy of Sciences

Introduction to Quantum Simulation

Wei Sun (孙玮, sunwei@ihep.ac.cn)

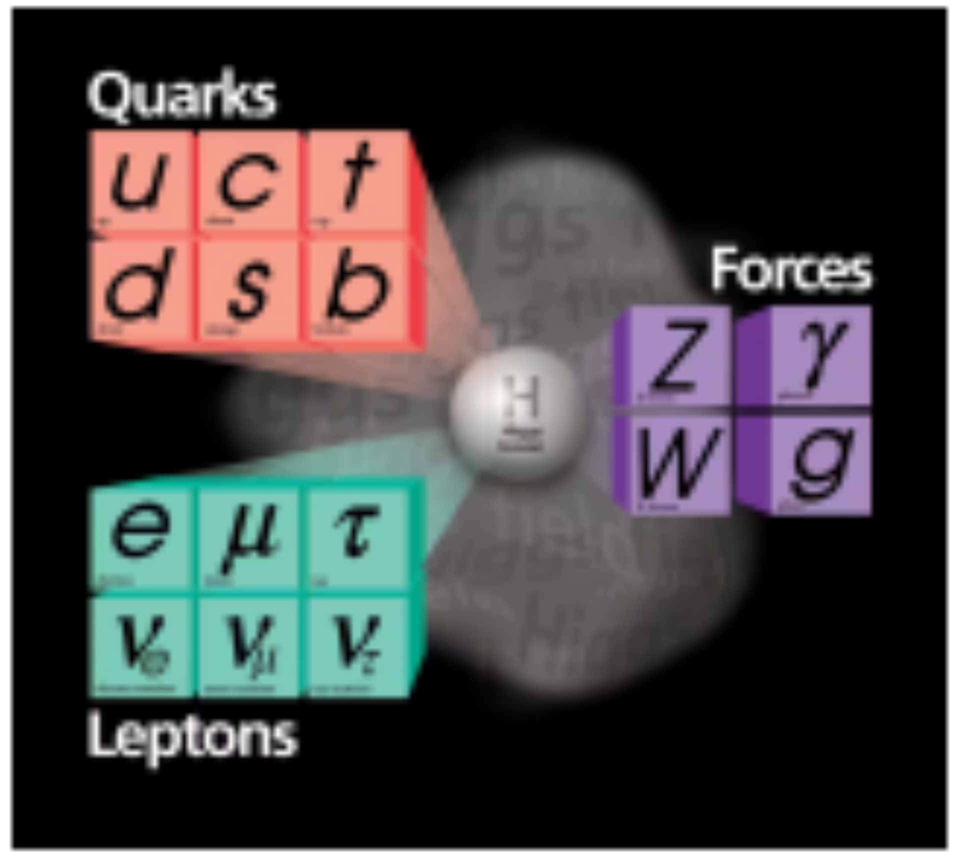
Institute of High Energy Physics, CAS

1st Machine Learning and Quantum Computing Winter School
1.12-18, 2025

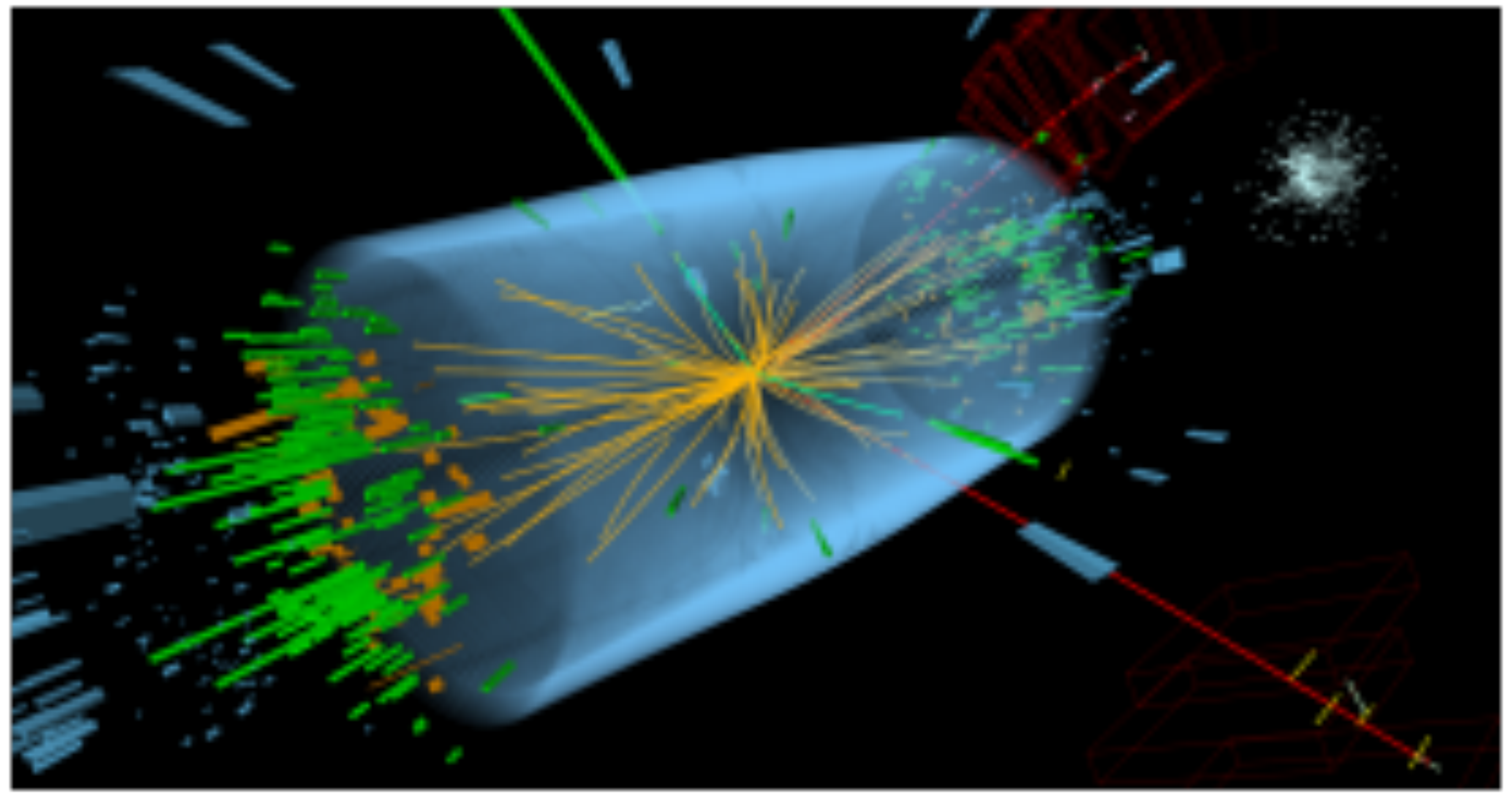


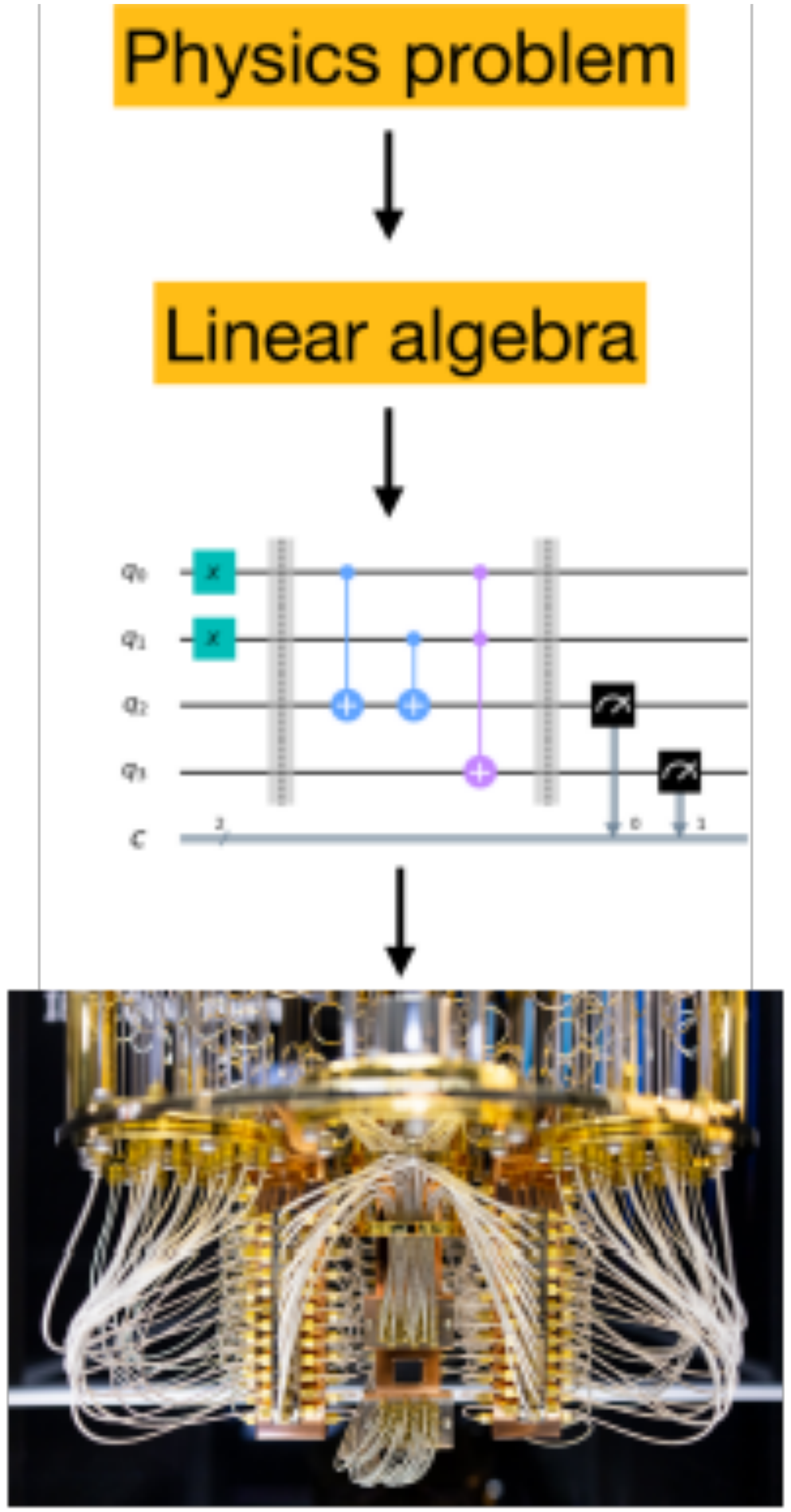
classical

Theory



Experiment



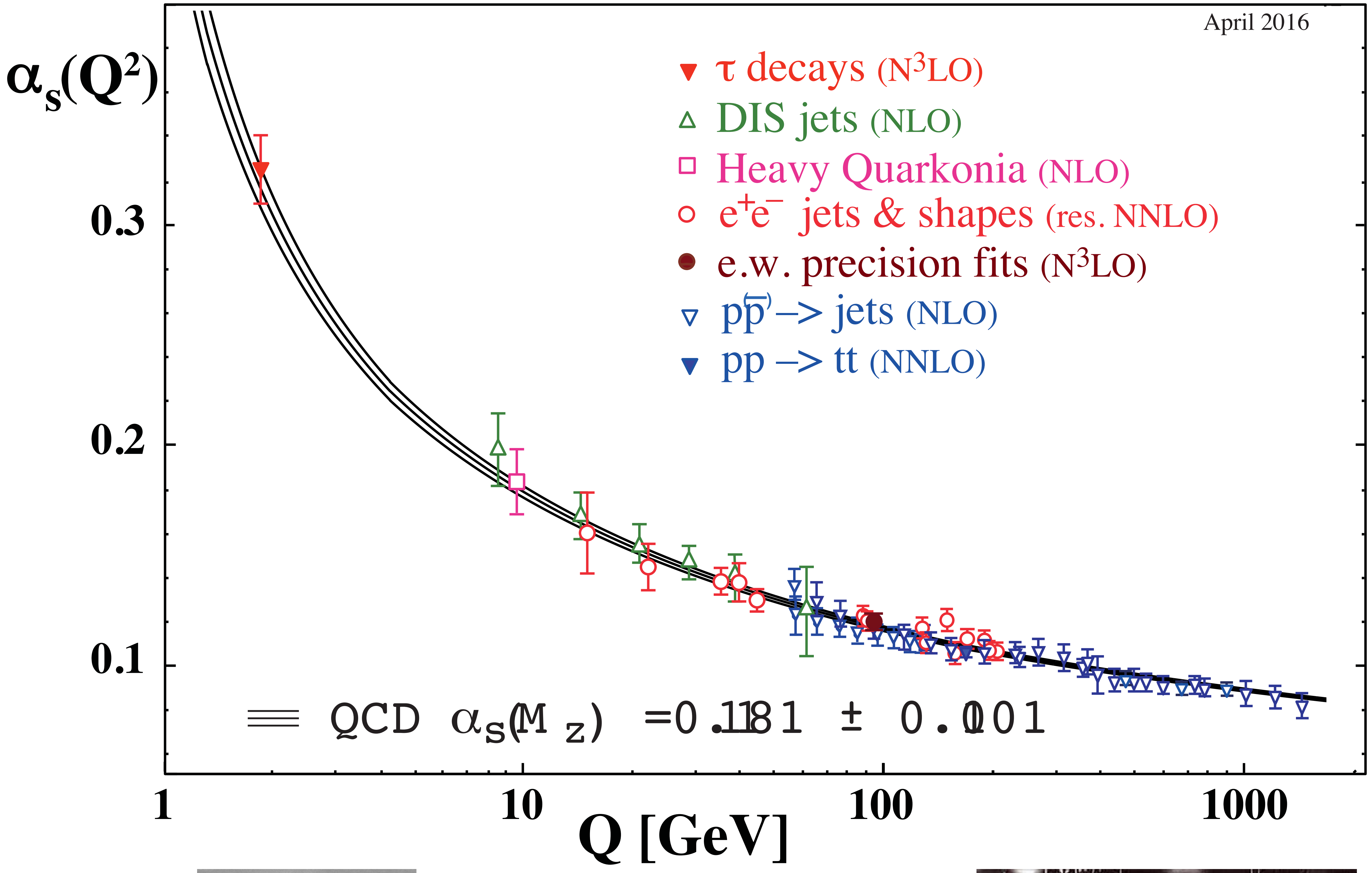


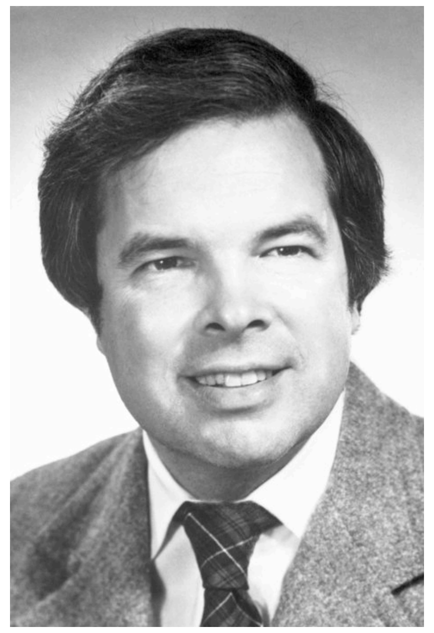
quantum

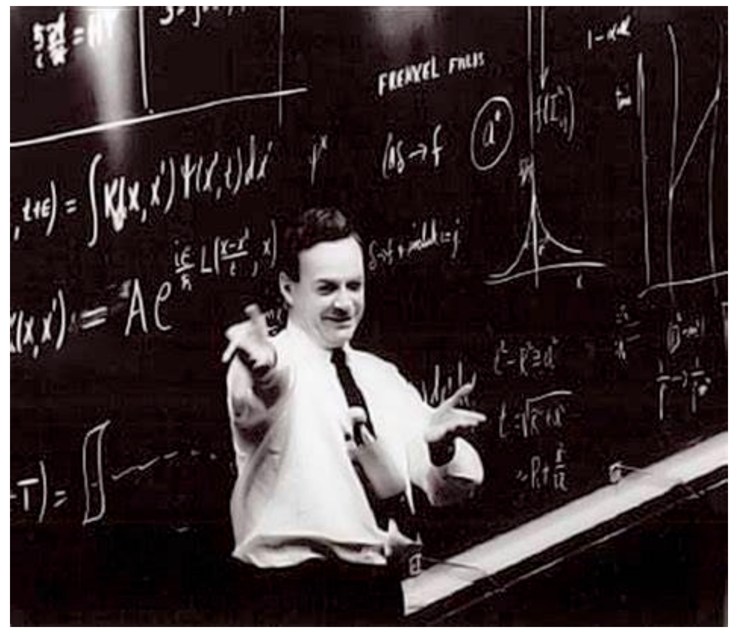
Outline

- Classical simulation
- What is a qubit
- How does quantum computers look like
- How to program a quantum computer
- Quantum simulation
- Summary and further reading

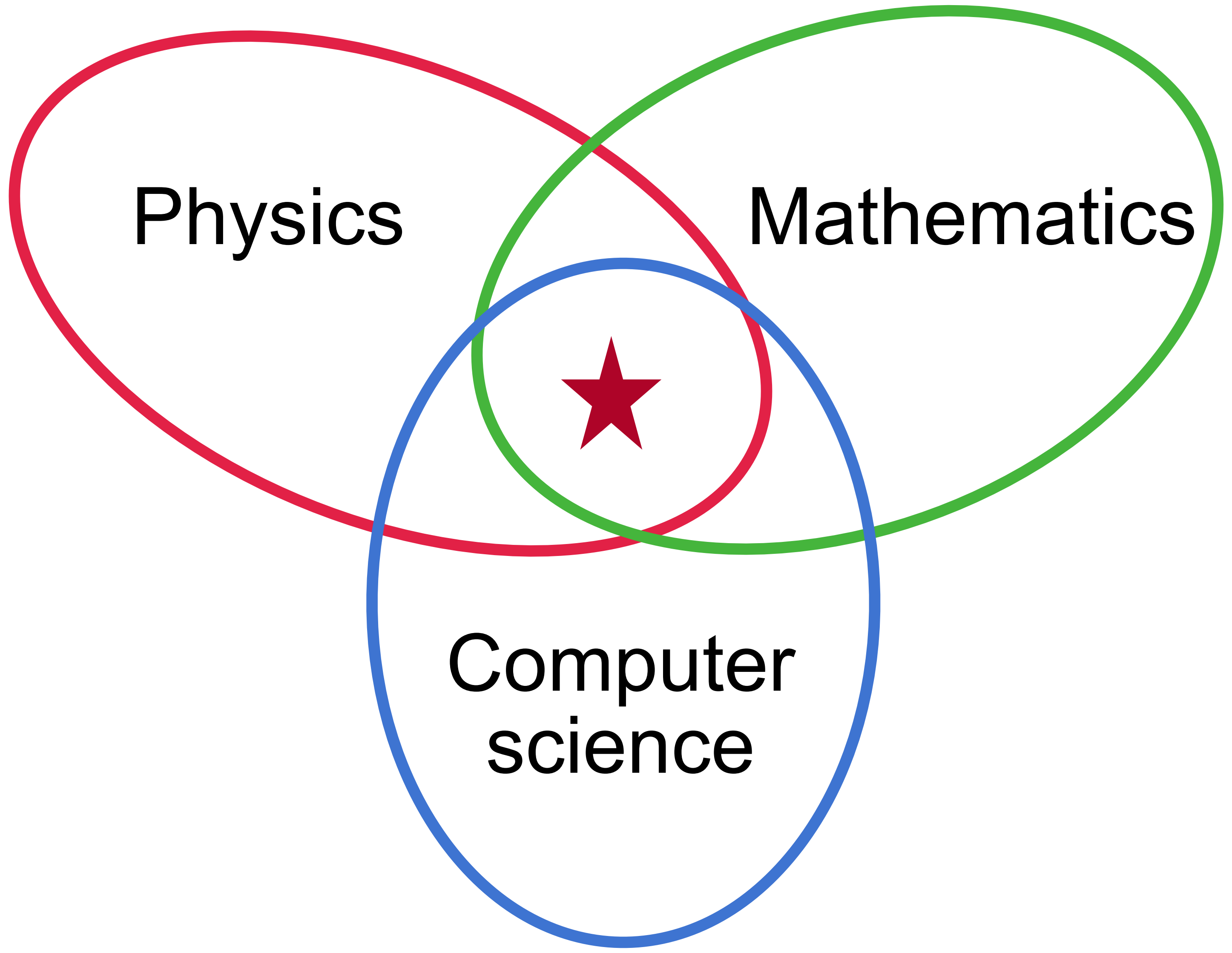
Analytical v.s. Numerical





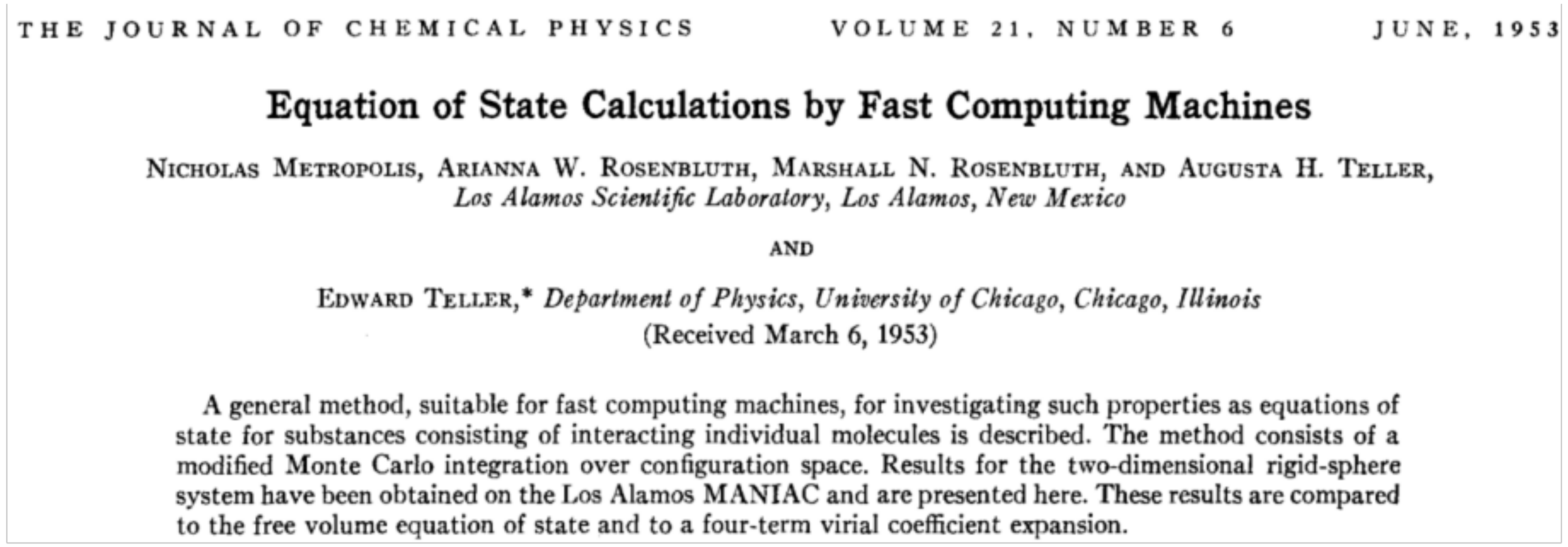


- analytical method at high energy
- numerical **Monte Carlo** method at low energy



Monte Carlo method

Integration -> **Statistical Average**


THE JOURNAL OF CHEMICAL PHYSICS VOLUME 21, NUMBER 6 JUNE, 1953

Equation of State Calculations by Fast Computing Machines

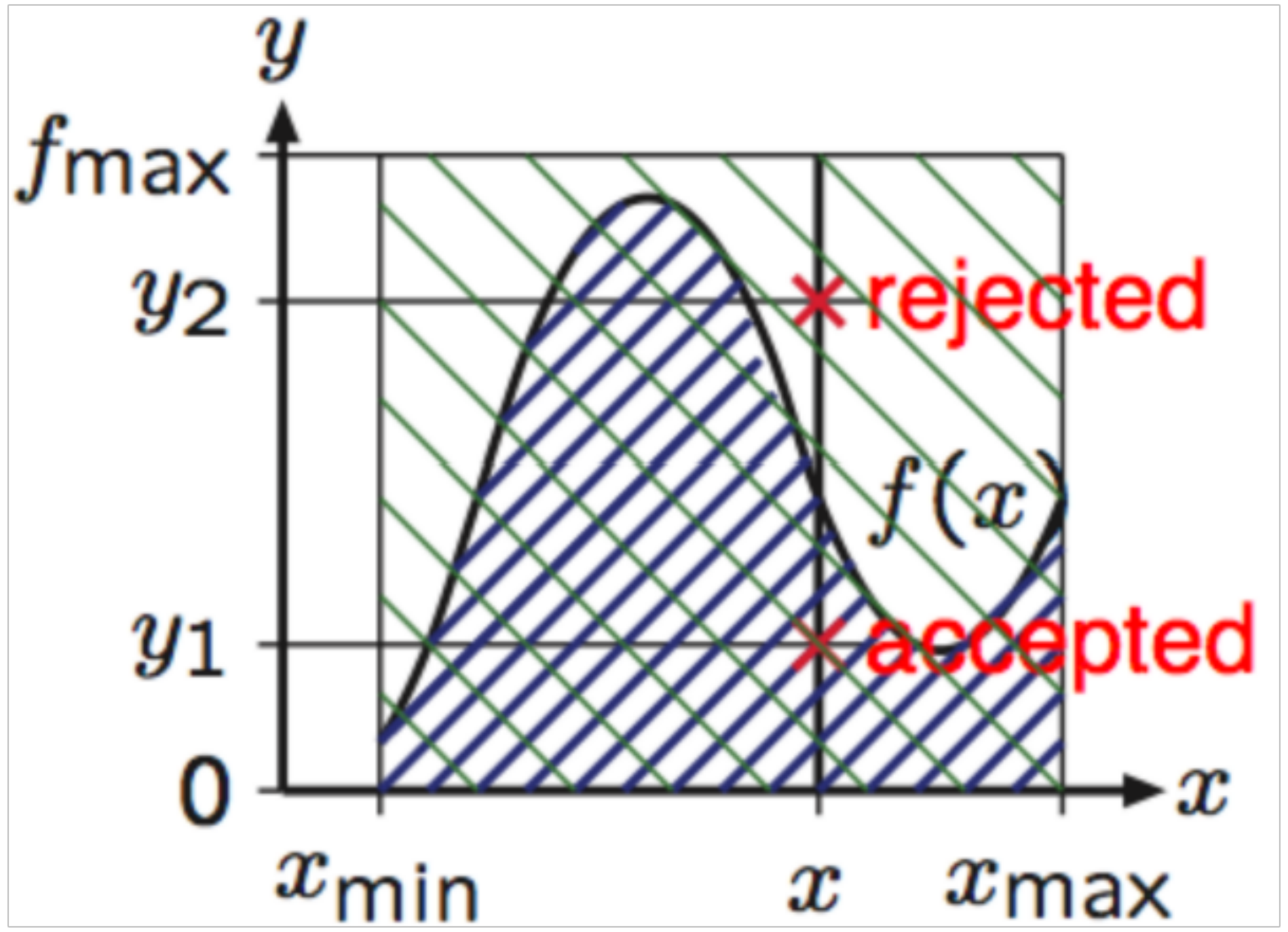
NICHOLAS METROPOLIS, ARIANNA W. ROSENBLUTH, MARSHALL N. ROSENBLUTH, AND AUGUSTA H. TELLER,
Los Alamos Scientific Laboratory, Los Alamos, New Mexico

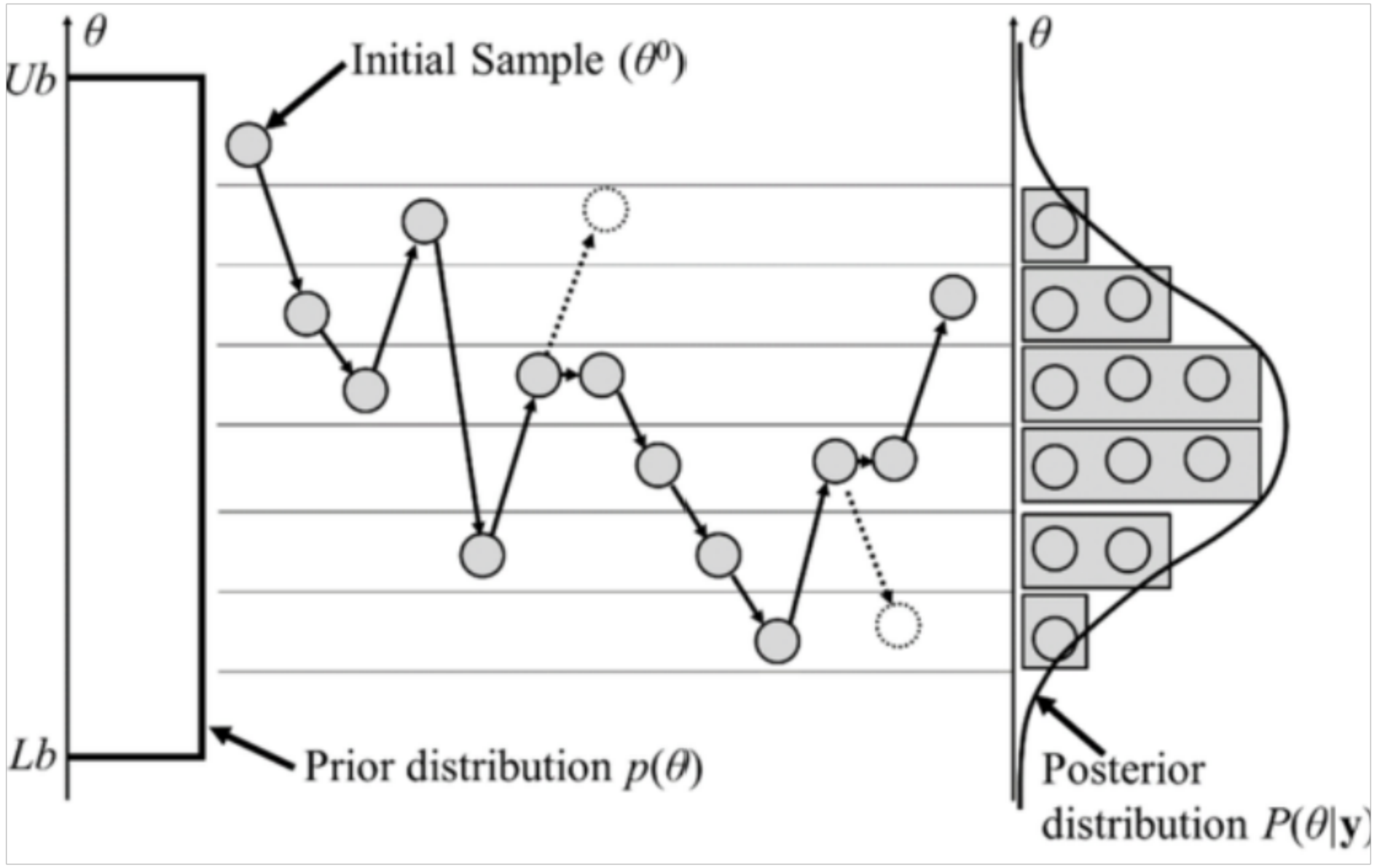
AND

EDWARD TELLER,* *Department of Physics, University of Chicago, Chicago, Illinois*
(Received March 6, 1953)

A general method, suitable for fast computing machines, for investigating such properties as equations of state for substances consisting of interacting individual molecules is described. The method consists of a modified Monte Carlo integration over configuration space. Results for the two-dimensional rigid-sphere system have been obtained on the Los Alamos MANIAC and are presented here. These results are compared to the free volume equation of state and to a four-term virial coefficient expansion.

$$I = \int_{x_1}^{x_2} f(x)dx \approx (x_2 - x_1)\frac{1}{N}\sum_{i=1}^{N} f(x_i)$$





The Top Ten Algorithms from the 20th Century

Resource requirements with classical methods

Computational Task	Current Usage	2025 Usage	Current Storage (Disk)	2025 Storage (Disk)	2025 Network Requirements (WAN)
Accelerator Modeling	$\sim$ 10M – 100M core-hrs/yr	$\sim$ 10G – 100G core-hrs/yr			
Computational Cosmology	$\sim$ 100M – 1G core-hrs/yr	$\sim$ 100G – 1000G core-hrs/yr	~10PB	>100PB	300Gb/s (burst)
Lattice QCD	~1G core-hrs/yr	$\sim$ 100G – 1000G core-hrs/yr	~1PB	>10PB	
Theory	$\sim$ 1M – 10M core-hrs/yr	$\sim$ 100M – 1G core-hrs/yr			
Cosmic Frontier Experiments	$\sim$ 10M – 100M core-hrs/yr	$\sim$ 1G – 10G core-hrs/yr	~1PB	10 – 100PB	
Energy Frontier Experiments	$\sim$ 100M core-hrs/yr	$\sim$ 10G – 100G core-hrs/yr	~1PB	>100PB	300Gb/s
Intensity Frontier Experiments	$\sim$ 10M core-hrs/yr	$\sim$ 100M – 1G core-hrs/yr	~1PB	10 – 100PB	300Gb/s

ASCR/HEP Exascale Report [arXiv:1603.09303]

Classical supercomputers

https://top500.org/lists/top500/2024/06/

Rank	System	Cores	Rmax (PFlop/s)	Rpeak (PFlop/s)	Power (kW)
1	**Frontier** - HPE Cray EX235a, AMD Optimized 3rd Generation EPYC 64C 2GHz, AMD Instinct MI250X, Slingshot-11, HPE DOE/SC/Oak Ridge National Laboratory United States	8,699,904	1,206.00	1,714.81	22,786
2	**Aurora** - HPE Cray EX - Intel Exascale Compute Blade, Xeon CPU Max 9470 52C 2.4GHz, Intel Data Center GPU Max, Slingshot-11, Intel DOE/SC/Argonne National Laboratory United States	9,264,128	1,012.00	1,980.01	38,698
3	**Eagle** - Microsoft NDv5, Xeon Platinum 8480C 48C 2GHz, NVIDIA H100, NVIDIA Infiniband NDR, Microsoft Azure Microsoft Azure United States	2,073,600	561.20	846.84	
4	**Supercomputer Fugaku** - Supercomputer Fugaku, A64FX 48C 2.2GHz, Tofu interconnect D, Fujitsu RIKEN Center for Computational Science Japan	7,630,848	442.01	537.21	29,899
5	**LUMI** - HPE Cray EX235a, AMD Optimized 3rd Generation EPYC 64C 2GHz, AMD Instinct MI250X, Slingshot-11, HPE EuroHPC/CSC Finland	2,752,704	379.70	531.51	7,107
6	**Alps** - HPE Cray EX254n, NVIDIA Grace 72C 3.1GHz, NVIDIA GH200 Superchip, Slingshot-11, HPE Swiss National Supercomputing Centre (CSCS) Switzerland	1,305,600	270.00	353.75	5,194
7	**Leonardo** - BullSequana XH2000, Xeon Platinum 8358 32C 2.6GHz, NVIDIA A100 SXM4 64 GB, Quad-rail NVIDIA HDR100 Infiniband, EVIDEN EuroHPC/CINECA Italy	1,824,768	241.20	306.31	7,494
8	**MareNostrum 5 ACC** - BullSequana XH3000, Xeon Platinum 8460Y+ 32C 2.3GHz, NVIDIA H100 64GB, Infiniband NDR, EVIDEN EuroHPC/BSC Spain	663,040	175.30	249.44	4,159
9	**Summit** - IBM Power System AC922, IBM POWER9 22C 3.07GHz, NVIDIA Volta GV100, Dual-rail Mellanox EDR Infiniband, IBM DOE/SC/Oak Ridge National Laboratory United States	2,414,592	148.60	200.79	10,096
10	**Eos NVIDIA DGX SuperPOD** - NVIDIA DGX H100, Xeon Platinum 8480C 56C 3.8GHz, NVIDIA H100, Infiniband NDR400, Nvidia NVIDIA Corporation United States	485,888	121.40	188.65	

Simulating Physics with Computers

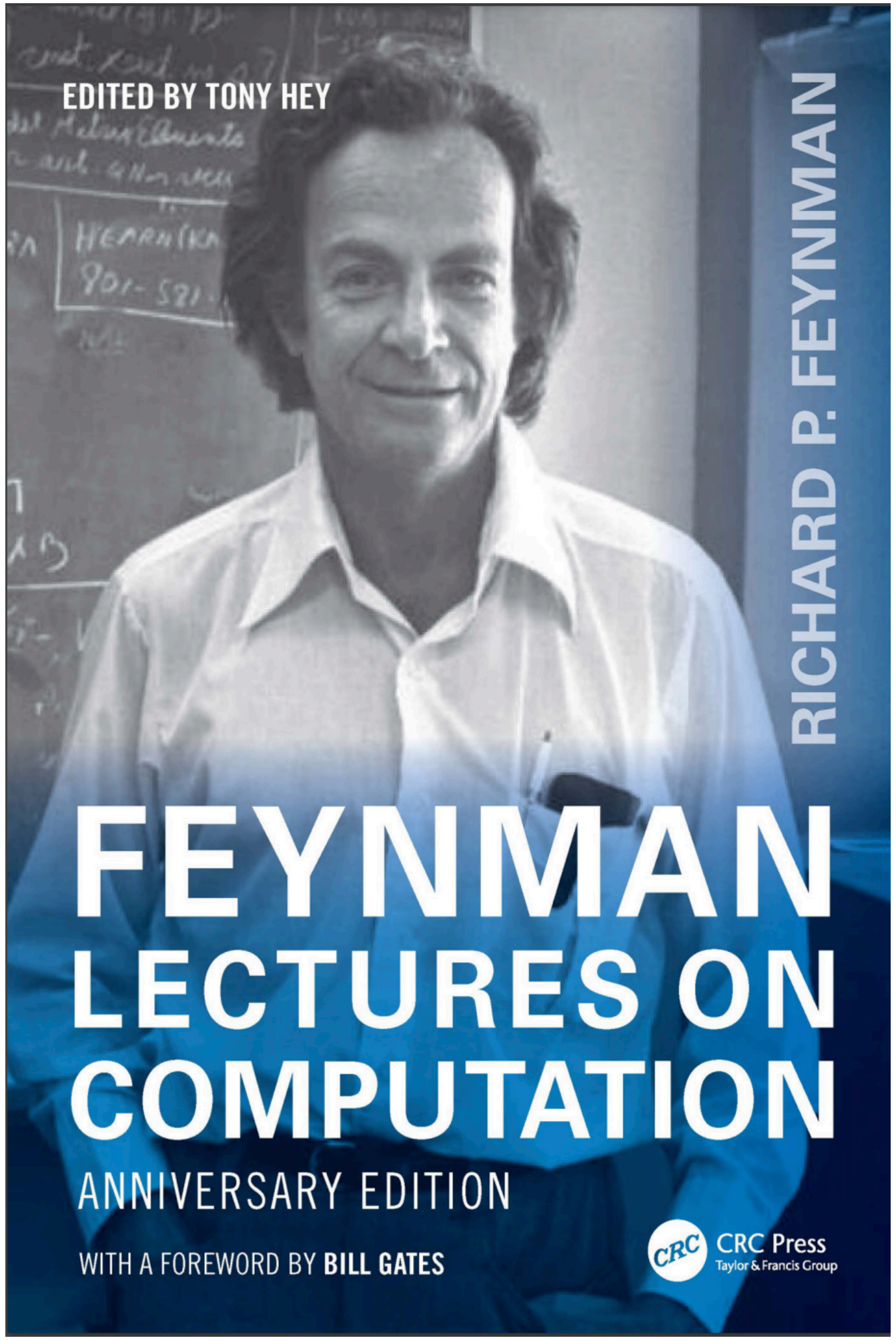
Richard P. Feynman

Department of Physics, California Institute of Technology, Pasadena, California 91107

Received May 7, 1981

Nature isn't classical, dammit, and if you want to make a simulation of nature, you'd better make it quantum mechanical... Can you do it with a new kind of computer--a quantum computer? It's not a Turing machine, but a machine of a different kind.

R. P. Feynman 1981



33. G. Felsenfeld *et al.*, *J. Am. Chem. Soc.* **79**, 2023 (1957); A. G. Letai *et al.*, *Biochemistry* **27**, 9108 (1988).
34. M. Riley, *Microbiol. Rev.* **57**, 862 (1993).
35. Supported in part by Department of Energy Cooperative Agreements DE-FC02-95ER61962 (J.C.V.) and DEFC02-95ER61963 (C.R.W. and G.J.O), NASA grant NAGW 2554 (C.R.W.), and a core grant to TIGR from Human Genome Sciences. G.J.O. is the recipient of the National Science Foundation Presidential Young Investigator Award (DIR 89-57026). M.B. is supported by National Institutes of Health grant GM00783. We thank M. Heaney, C. Gnehm, R. Shirley, J. Slagel, and W. Hayes for software and database support; T. Dixon and V. Sapiro for computer system support; K. Hong and B. Stader for laboratory assistance; and B. Mukhopadhyay for helpful discussions. The *M. jannaschii* source accession number is DSM 2661, and the cells were a gift from P. Haney (Department of Microbiology, University of Illinois).

RESEARCH ARTICLES

Universal Quantum Simulators

Seth Lloyd

Feynman's 1982 conjecture, that quantum computers can be programmed to simulate any local quantum system, is shown to be correct.

Over the past half century, the logical devices by which computers store and process information have shrunk by a factor of 2 every 2 years. A quantum computer is the end point of this process of miniaturization—when devices become sufficiently small, their behavior is governed by quantum mechanics. Information in conventional digital computers is stored on capacitors. An uncharged capacitor registers a 0 and a charged capacitor registers a 1. Information in a quantum computer is stored on individual spins, photons, or atoms. An atom can itself be thought of as a tiny capacitor. An atom in its ground state is analogous to an uncharged capacitor and can be taken to register a 0, whereas an atom in an excited state is analogous to a charged capacitor and can be taken to register a 1.

So far, quantum computers sound very much like classical computers; the only use of quantum mechanics has been to make a correspondence between the discrete quantum states of spins, photons, or atoms and the discrete logical states of a digital computer. Quantum systems, however, exhibit behavior that has no classical analog. In particular, unlike classical systems, quantum systems can exist in superpositions of different discrete states. An ordinary capacitor can be either charged or uncharged, but not both: A classical bit is either 0 or 1. In contrast, an atom in a quantum superposition of its ground and excited state is a quantum bit that in some sense registers both 0 and 1 at the same time. As a result, quantum computers can do things that classical computers cannot.

Classical computers solve problems by using nonlinear devices such as transistors to perform elementary logical operations on the bits stored on capacitors. Quantum computers can also solve problems in a similar fashion; nonlinear interactions between quantum variables can be exploited to perform elementary quantum logical operations. However, in addition to ordinary classical logical operations such as AND, NOT, and COPY, quantum logic includes operations that put quantum bits in superpositions of 0 and 1. Because quantum computers can perform ordinary digital logic as well as exotic quantum logic, they are in principle at least as powerful as classical computers. Just what problems quantum computers can solve more efficiently than classical computers is an open question.

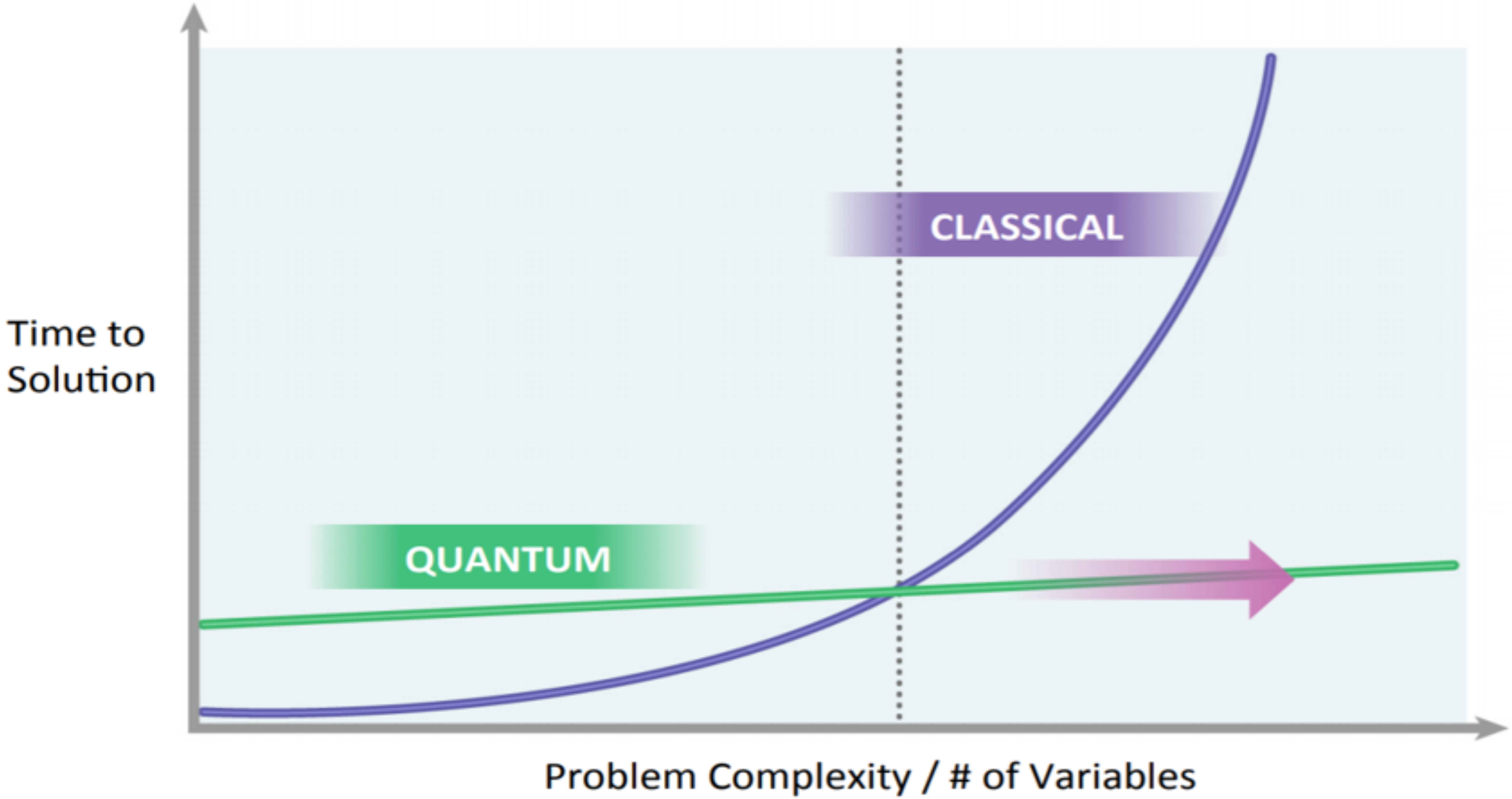
Since their introduction in 1980 (*1*) quantum computers have been investigated extensively (*2*–*29*). A comprehensive review can be found in (*15*). The best known problem that quantum computers can in principle solve more efficiently than classical computers is factoring (*14*). In this article I present another type of problem that in principle quantum computers could solve more efficiently than a classical computer—that of simulating other quantum systems. In 1982, Feynman conjectured that quantum computers might be able to simulate other quantum systems more efficiently than classical computers (*2*). Quantum simulation is thus the first classically difficult problem posed for quantum computers. Here I show that a quantum computer can in fact simulate quantum systems efficiently as long as they evolve according to local interactions.

Feynman noted that simulating quantum systems on classical computers is hard. Over the past 50 years, a considerable amount of effort has been devoted to such simulation. Much information about a quantum system's dynamics can be extracted from semiclassical approximations (when classical solutions are known), and ground state properties and correlation functions can be extracted with Monte Carlo methods (*30*–*32*). Such methods use amounts of computer time and memory space that grow as polynomial functions of the size of the quantum system of interest (where size is measured by the number of variables—particles or lattice sites, for example—required to characterize the system). Problems that can be solved by methods that use polynomial amounts of computational resources are commonly called tractable; problems that can only be solved by methods that use exponential amounts of resources are commonly called intractable. Feynman pointed out that the problem of simulating the full time evolution of arbitrary quantum systems on a classical computer is intractable: The states of a quantum system are wave functions that lie in a vector space whose dimension grows exponentially with the size of the system. As a result, it is an exponentially difficult problem merely to record the state of a quantum system, let alone integrate its equations of motion. For example, to record the state of 40 spin-½ particles in a classical computer's memory requires $2^{40} \approx 10^{12}$ numbers, whereas to calculate their time evolution requires the exponentiation of a $2^{40} \times 2^{40}$ matrix with $\approx 10^{24}$ entries. Feynman asked whether it might be possible to bypass this exponential explosion by having one quantum system simulate another directly, so that the states of the simulator obey the same equations of motion as the states of the simulated system. Feynman gave simple examples of one quantum system simulating another and conjectured that there existed a class of universal quantum simulators capable of simulating any quantum system that evolved according to local interactions.

The answer to Feynman's question is, yes. I will show that a variety of quantum systems, including quantum computers, can be "programmed" to simulate the behavior of arbitrary quantum systems whose dynamics are determined by local interactions. The programming is accomplished by inducing interactions between the variables of the simulator that imitate the interactions between the variables of the system to be simulated. In effect, the dynamics of the properly programmed simulator and the dynamics of the system to be simulated are one and the same to within any desired accuracy. So, to simulate the time evolution of 40 spin-½ particles over time t requires a simulator with 40 quantum bits evolving

The author is at the D'Arbeloff Laboratory for Information Systems and Technology, Department of Mechanical Engineering, Massachusetts Institute of Technology, Cambridge, MA 02139, USA. E-mail: slloyd@mit.edu

Seeking for quantum advantage



CERN Quantum Technology Initiative

1

Formalise and extend the existing catalogue of use cases and examples of possible applications of quantum computing to HEP workflows and algorithms

2

Collect and share information about existing resources, tools, libraries, and collaborators across the community. Set up R&D projects to adapt or design algorithms for quantum platforms and benchmark their current and potential performance.

3

Identify and coordinate access to computing resources in collaboration with other institutes or companies. Design and deploy a distributed infrastructure for quantum computing and simulation, based on existing distributed computing expertise.

4

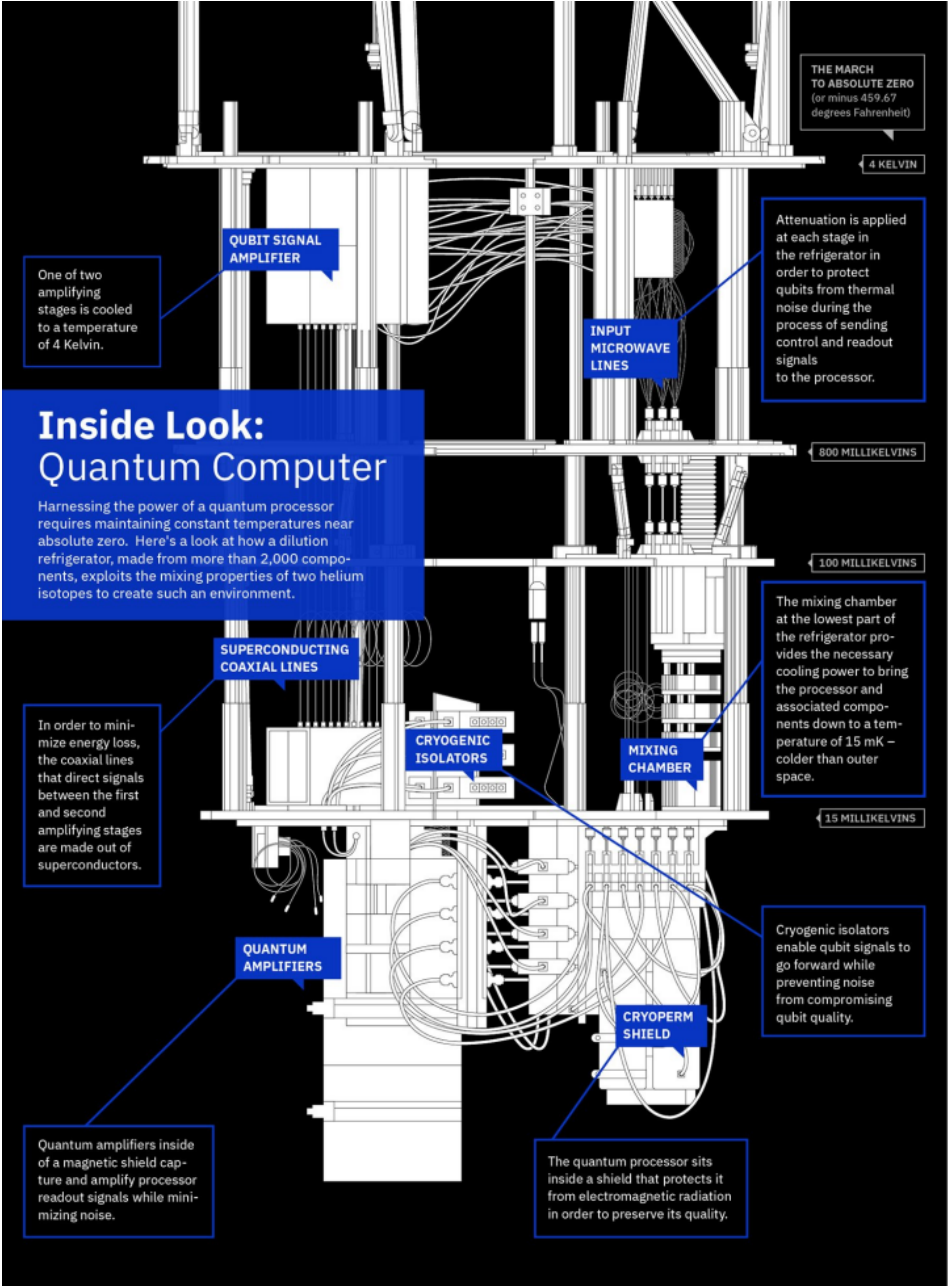
Simulation of high-dimensional classical and quantum systems on hybrid infrastructure.

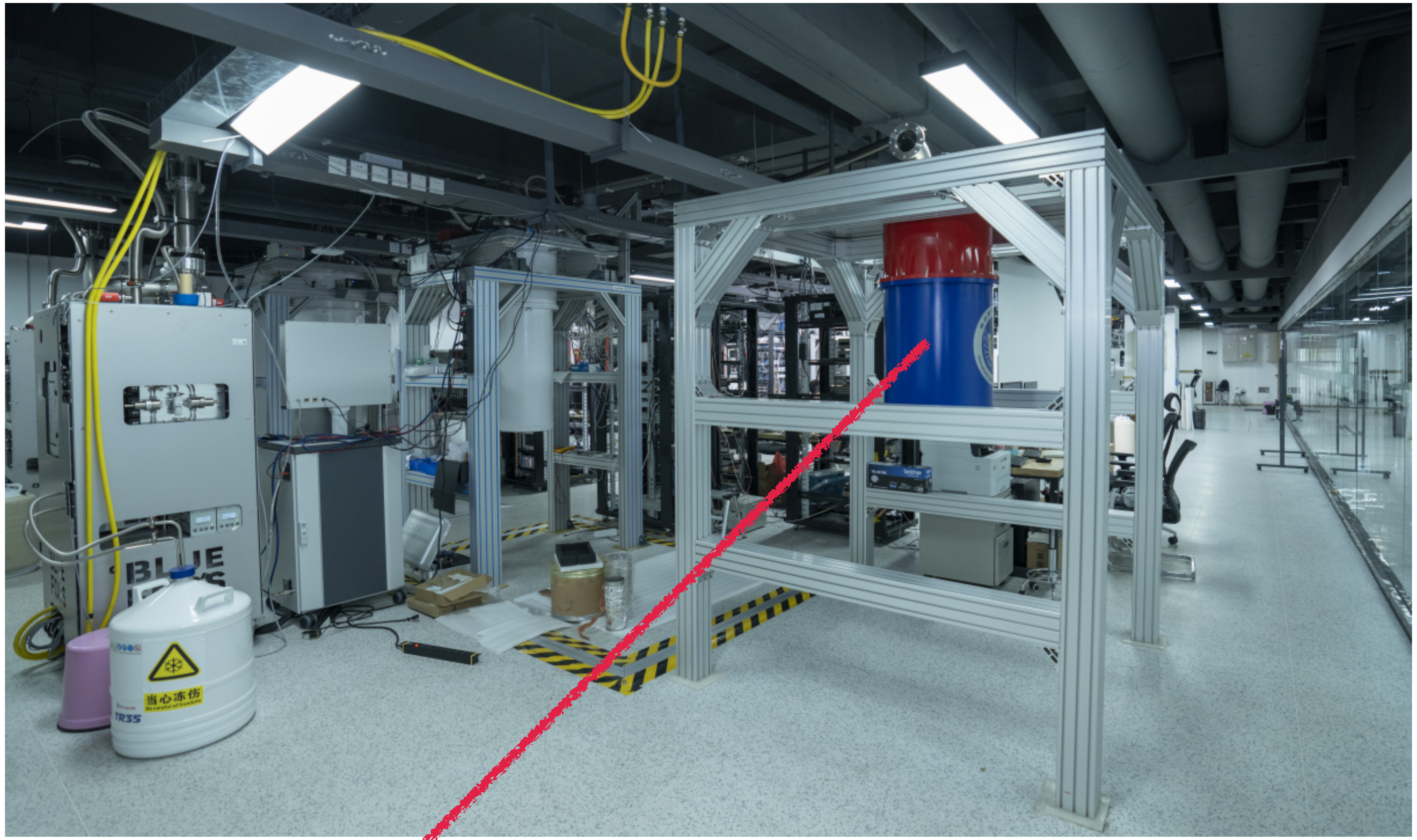
5

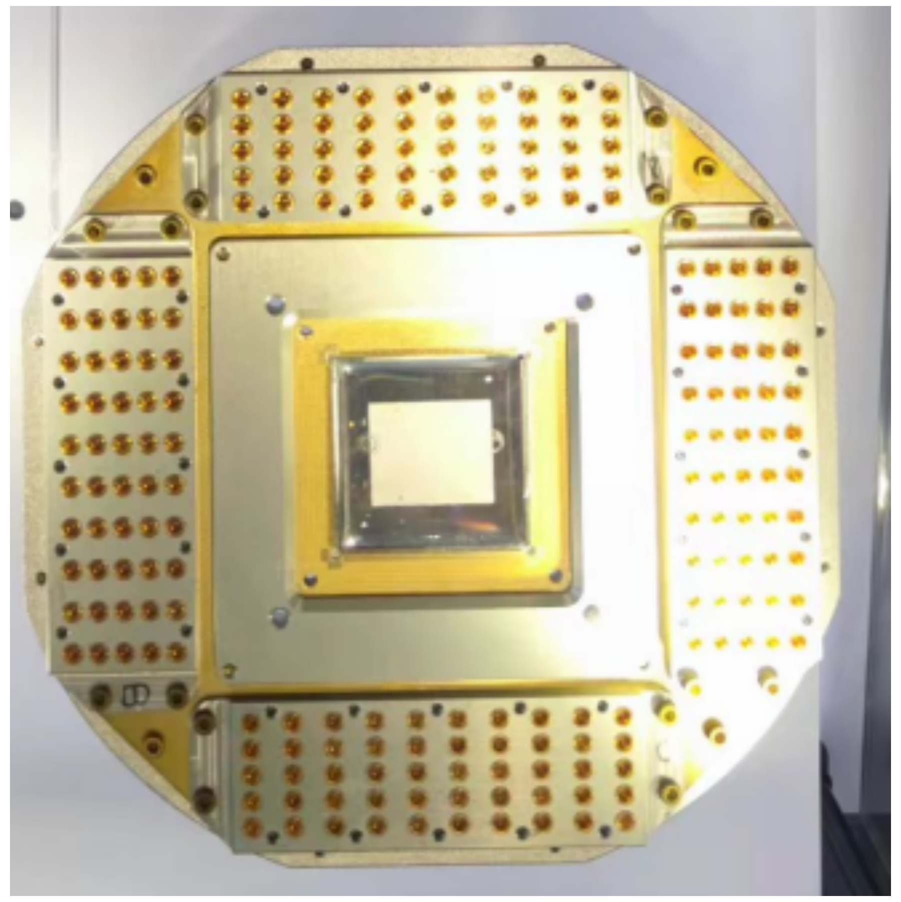
Advance the theoretical foundations of quantum machine learning algorithms.

6

Quantum algorithms as a tool to solve fundamental problems in theoretical physics.







- **A quantum computer is a machine that performs computation based on quantum mechanics**
- **The data is represented by qubits, a two level system**
- **The operations on qubits are unitary quantum gates**

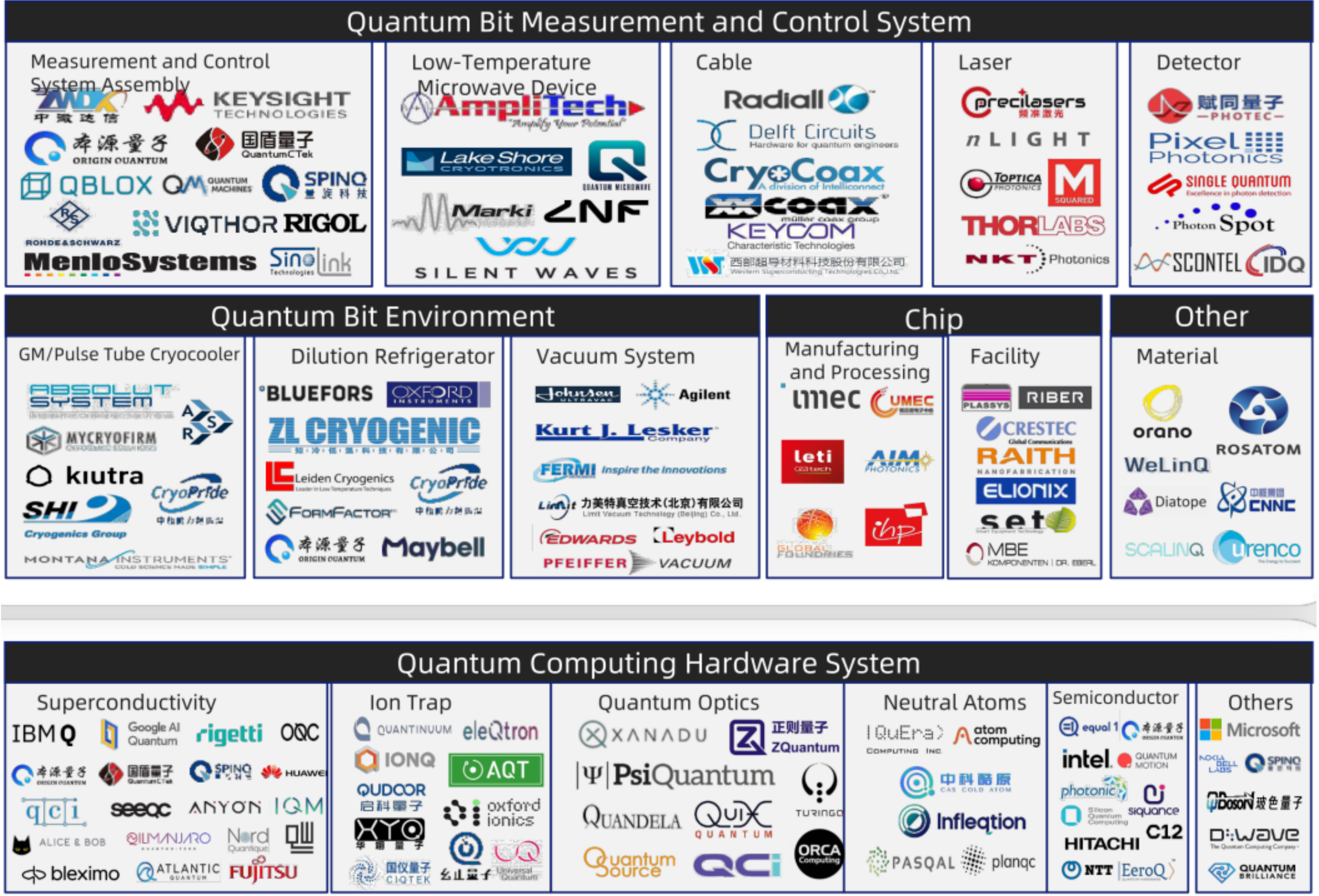
DiVincenzo's criteria

D. DiVincenzo, arXiv: quant-ph/0002077

1. A **scalable** physical system with well characterized qubits
2. The ability to **initialize** the state of the qubits to a simple fiducial state, such as $|000...000\rangle$
3. **Long relevant decoherence times**, much longer than the gate operation time
4. A **universal** set of quantum gates
5. A qubit-specific **measurement** capability

Brief history of quantum computing

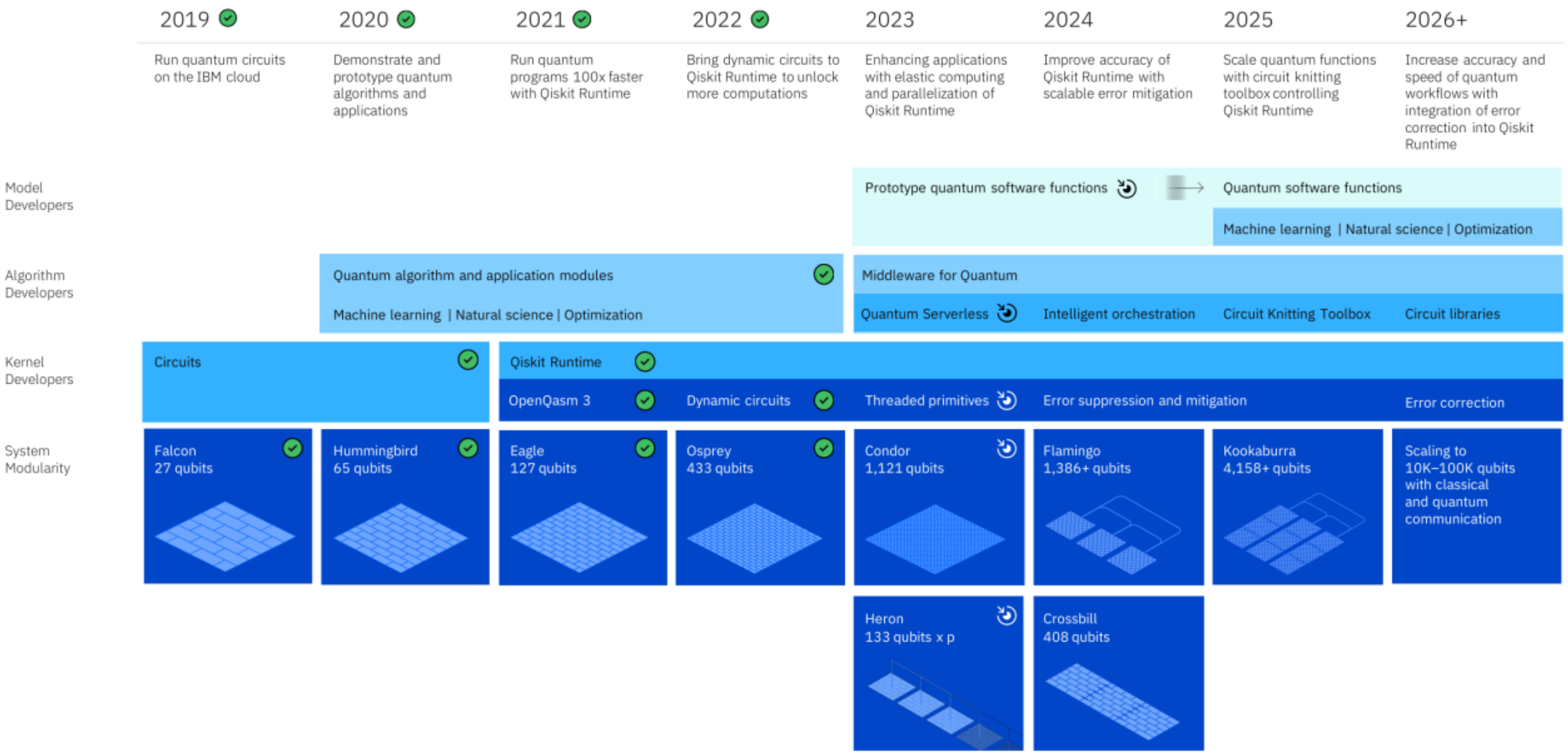
Credit: Quantumpedia



ICV TA&K and QUANTUMCHINA

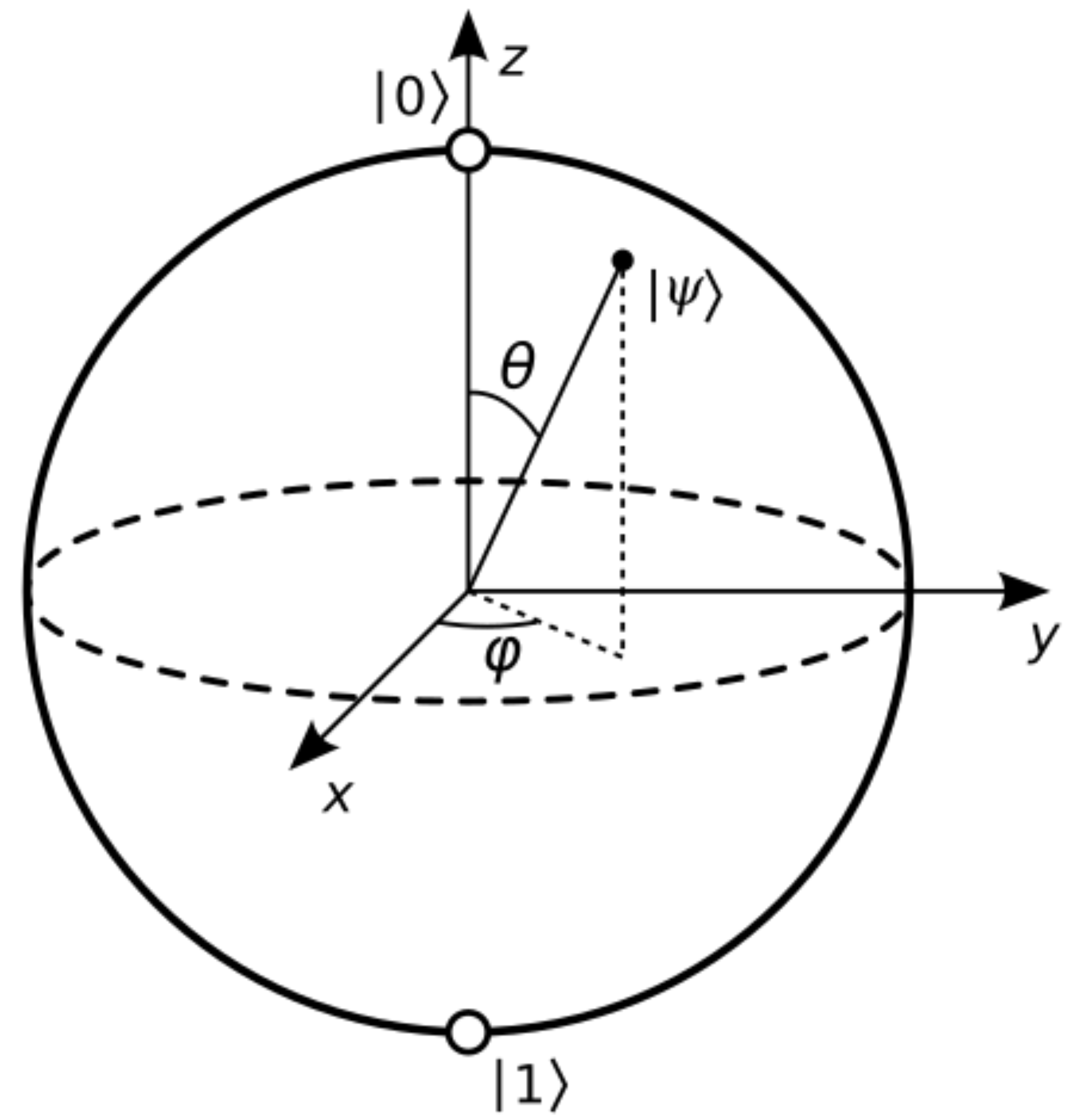
Development Roadmap

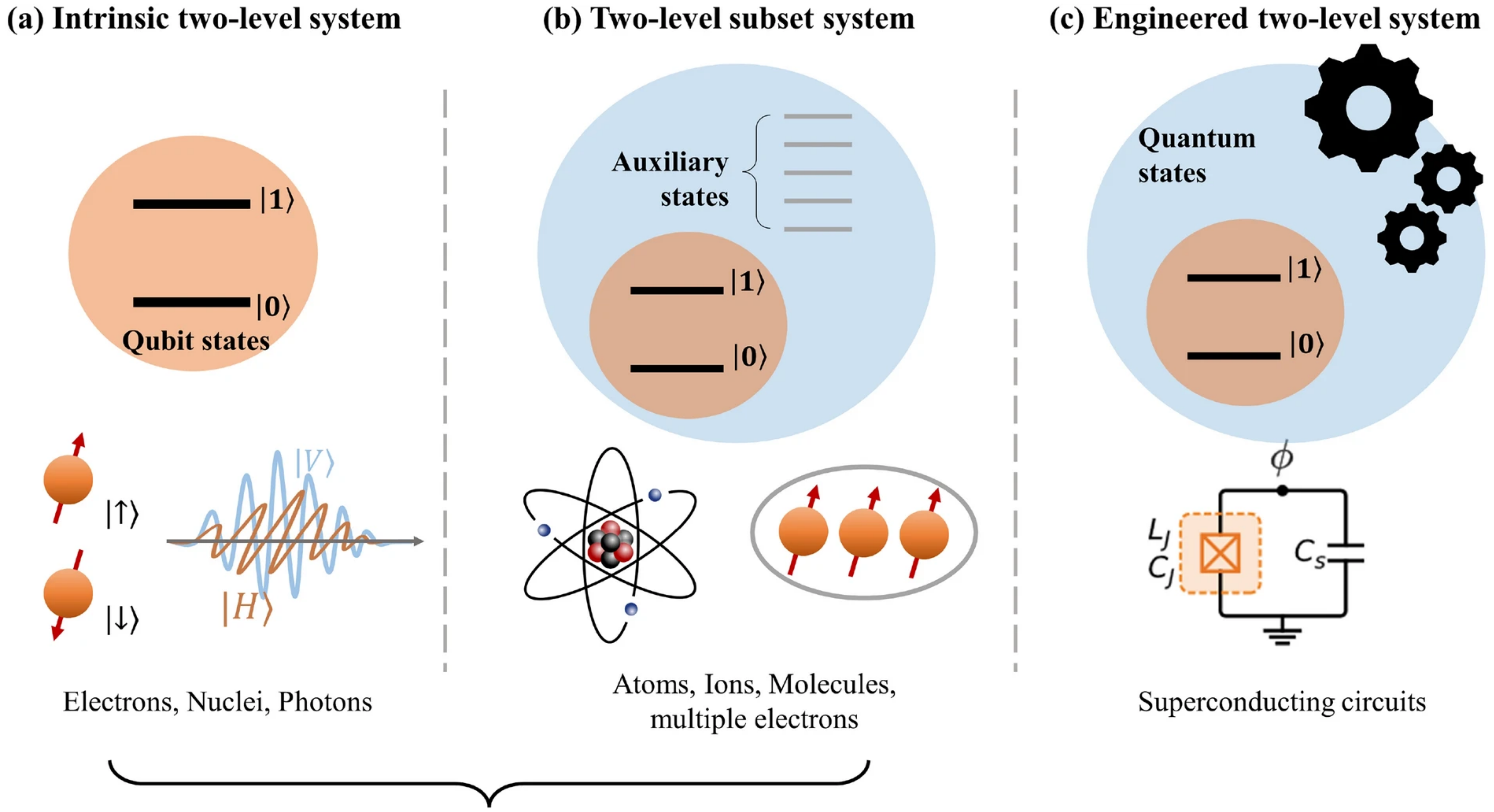
Executed by IBM ✓
On target



What is a qubit

- A qubit is a quantum state of a two-level quantum system
- Orthonormal basis states denoted as $|0\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix}, |1\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$
- A general qubit can be represented by a linear superposition of basis states, $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle, \alpha, \beta \in C, |\alpha|^2 + |\beta|^2 = 1$





Natural Qubits [*Nano Convergence* **11**, 11 (2024)] **Synthetic Qubits**

$$|+\rangle = \frac{|0\rangle + |1\rangle}{\sqrt{2}} \qquad |-\rangle = \frac{|0\rangle - |1\rangle}{\sqrt{2}}$$

$$|\Phi^+\rangle = \frac{|00\rangle + |11\rangle}{\sqrt{2}} \qquad |\Phi^+\rangle = \frac{|00\rangle - |11\rangle}{\sqrt{2}}$$

$$|\Psi^+\rangle = \frac{|01\rangle + |10\rangle}{\sqrt{2}} \qquad |\Psi^-\rangle = \frac{|01\rangle - |10\rangle}{\sqrt{2}}$$

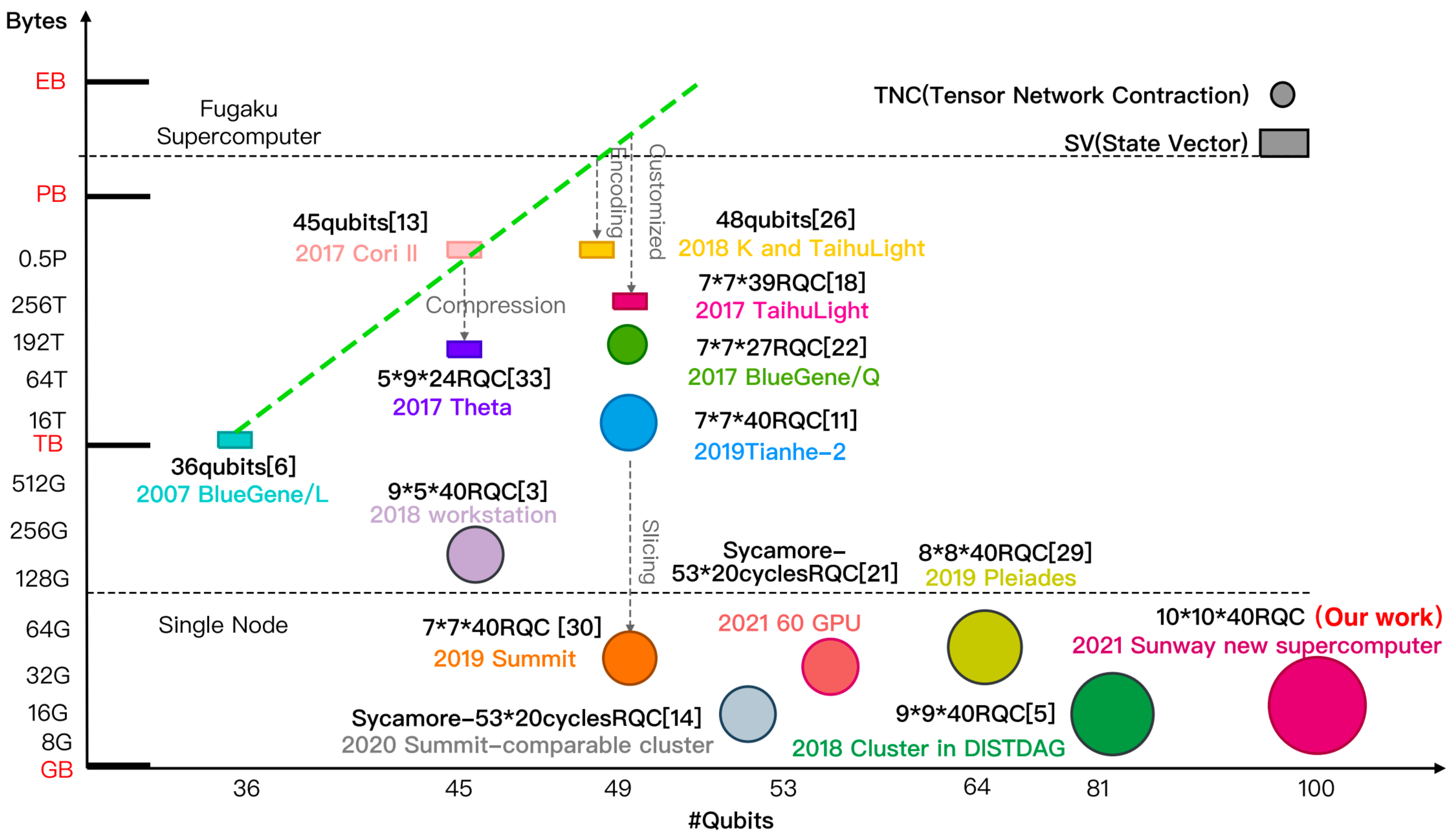
Bell states

Quantum gates

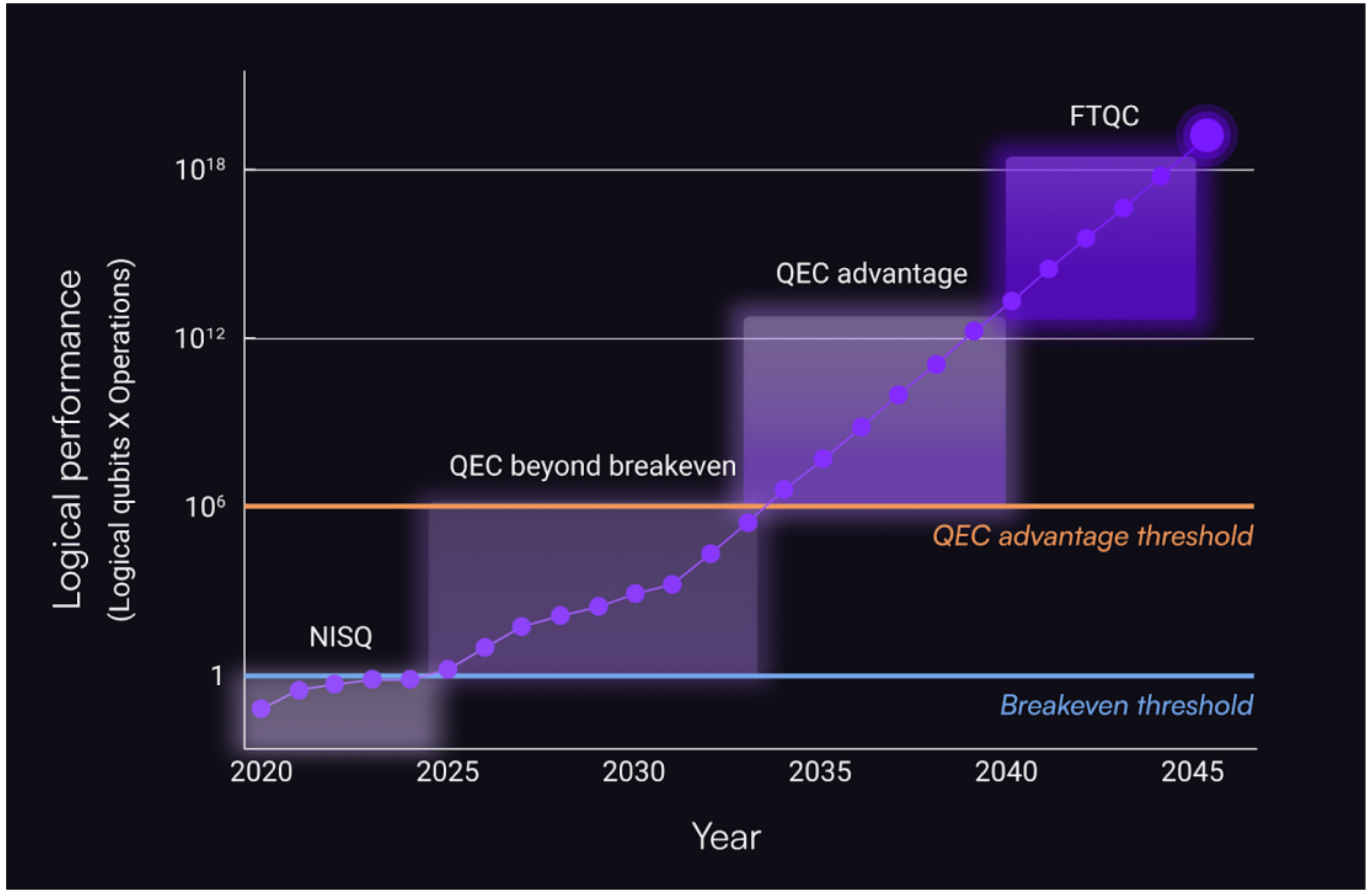
- Quantum gates are represented by unitary matrix and operated on qubits
- **Single qubit gates**: X, Y, Z, H, P, T, …
- **Two qubit gates**: CNOT, CZ, …
- Universal quantum gate sets: **approximate any unitary gate by any precision**
- Choose one of the possible universal gates set (Solovay-Kitaev theorem)
    - **{CNOT, H, T}**
    - **{CNOT, all single qubit gates}**
    - **{Toffoli, H}**
- $X|0\rangle = |1\rangle, |+\rangle = H|0\rangle$
- $\text{CNOT}|01\rangle = |01\rangle, \text{CNOT}|11\rangle = |10\rangle$

Operator	Gate(s)	Matrix
Pauli-X (X)	X, ⊕	$\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$
Pauli-Y (Y)	Y	$\begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix}$
Pauli-Z (Z)	Z	$\begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$
Hadamard (H)	H	$\frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$
Phase (S, P)	S	$\begin{bmatrix} 1 & 0 \\ 0 & i \end{bmatrix}$
$\pi/8$ (T)	T	$\begin{bmatrix} 1 & 0 \\ 0 & e^{i\pi/4} \end{bmatrix}$
Controlled Not (CNOT, CX)		$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$
Controlled Z (CZ)	Z	$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 \end{bmatrix}$
SWAP		$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$
Toffoli (CCNOT, CCX, TOFF)		$\begin{bmatrix} 1&0&0&0&0&0&0&0 \\ 0&1&0&0&0&0&0&0 \\ 0&0&1&0&0&0&0&0 \\ 0&0&0&1&0&0&0&0 \\ 0&0&0&0&1&0&0&0 \\ 0&0&0&0&0&1&0&0 \\ 0&0&0&0&0&0&0&1 \\ 0&0&0&0&0&0&1&0 \end{bmatrix}$

Simulation of quantum computer on classical computer needs exponential resource



Y. Liu et.al. SC’ 21



Current quantum computers are in the NISQ (Noisy Intermediate-Scale Quantum) era

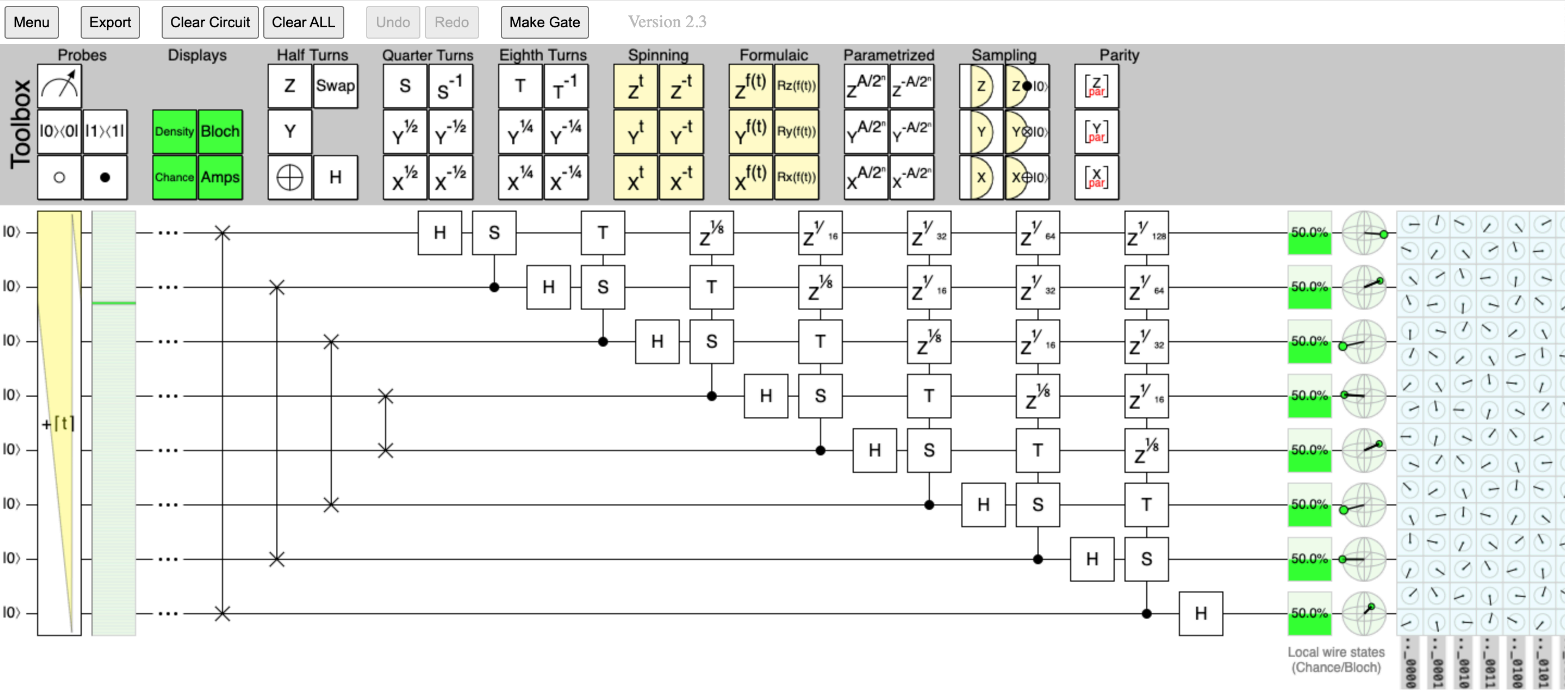
- **Real hardwares are very noisy**
- **Error mitigation / correction is essential**
- **Need classical simulator to verify the quantum algorithms, while need $O(2^N)$ memory to simulate the quantum circuits**

Quantum programming softwares

- Many high quality quantum computing softwares available
- Curated list of open-source quantum software projects
  - Most based on Python interfaced with C++
  - https://github.com/qosf/awesome-quantum-software
- Drag and drop playing with quantum circuits (https://qc.ihep.ac.cn)
- If you want to try the high performance GPU simulator, please contact me

Drag and drop playing with **Quirk**

Online web based simulator, interactive and quiet interesting



Quantum algorithms

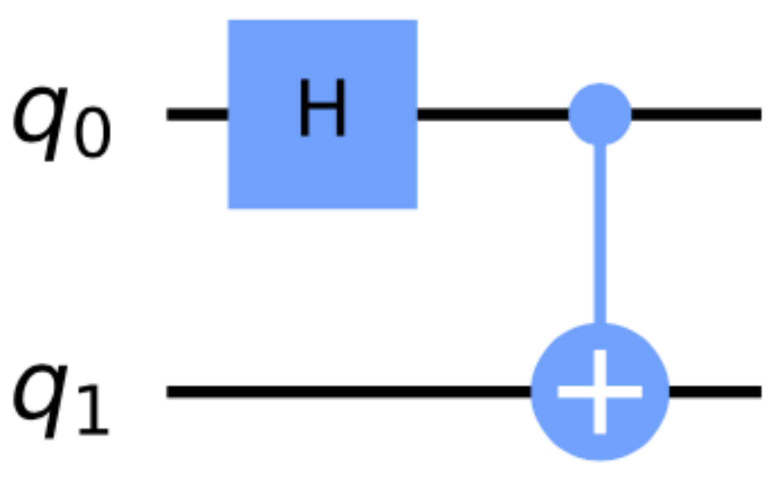
Quantum Algorithm Zoo

https://quantumalgorithmzoo.org/

- Compare the time complexity
- Try to implement the algorithms with popular qiskit package
    - pip install qiskit, play with jupyter notebook

Algorithms	Classical steps	quantum logic steps
Fourier transform e.g.: - Shor's prime factorization - discrete logarithm problem - Deutsch Jozsa algorithm	$N\log(N) = n2^n$ $N = 2^n$ - n qubits - N numbers	$\log^2(N) = n^2$ - hidden information! - Wave function collapse prevents us from directly accessing the information
Search Algorithms	N	$\sqrt{N}$
Quantum Simulation	c^N bits	kn qubits

```python
from qiskit import QuantumCircuit

# Create a new circuit with two qubits
qc = QuantumCircuit(2)

# Add a Hadamard gate to qubit 0
qc.h(0)

# Perform a CNOT gate on qubit 1, controlled by qubit 0
qc.cx(0, 1)

# draw the circuit
qc.draw("mpl")
```

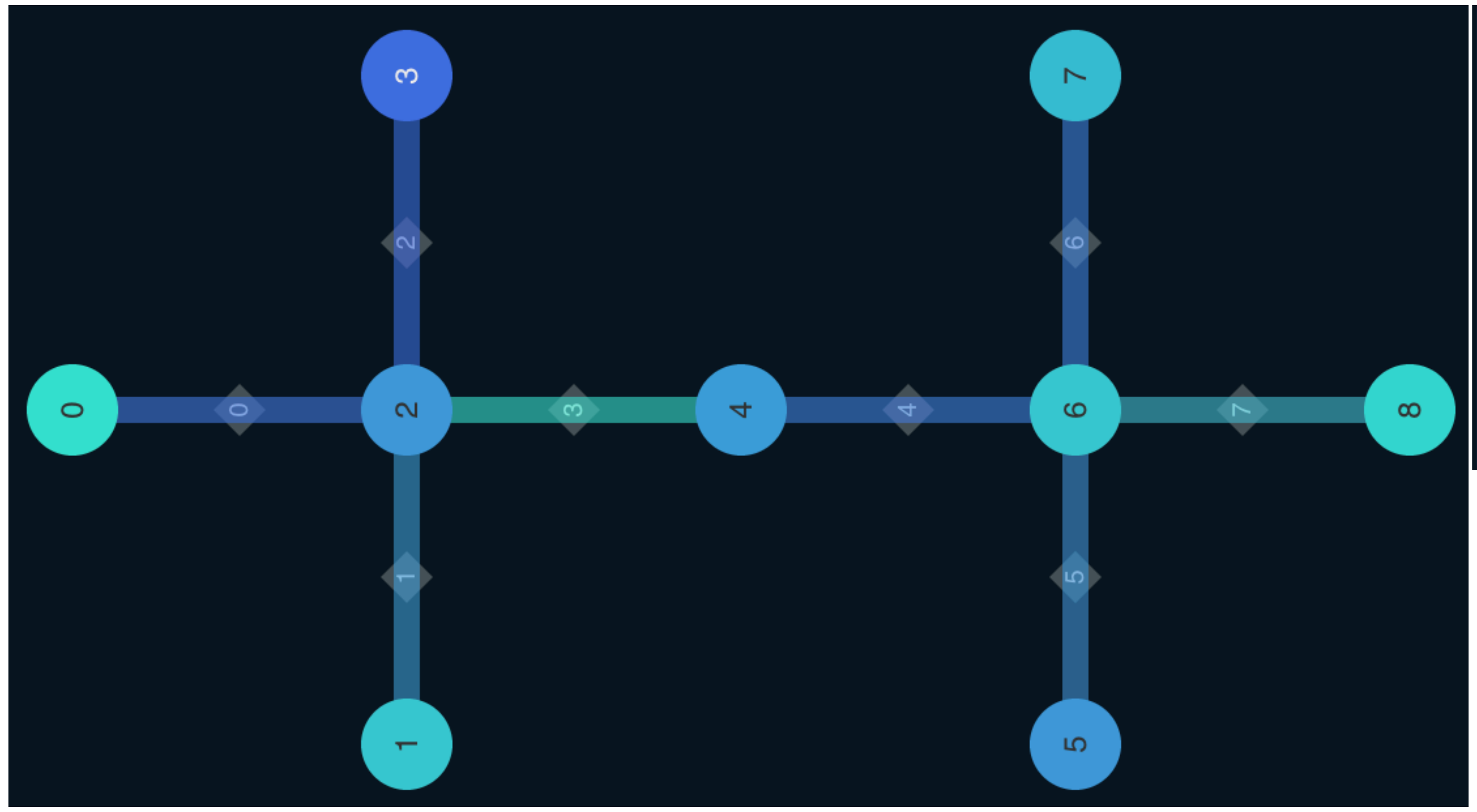


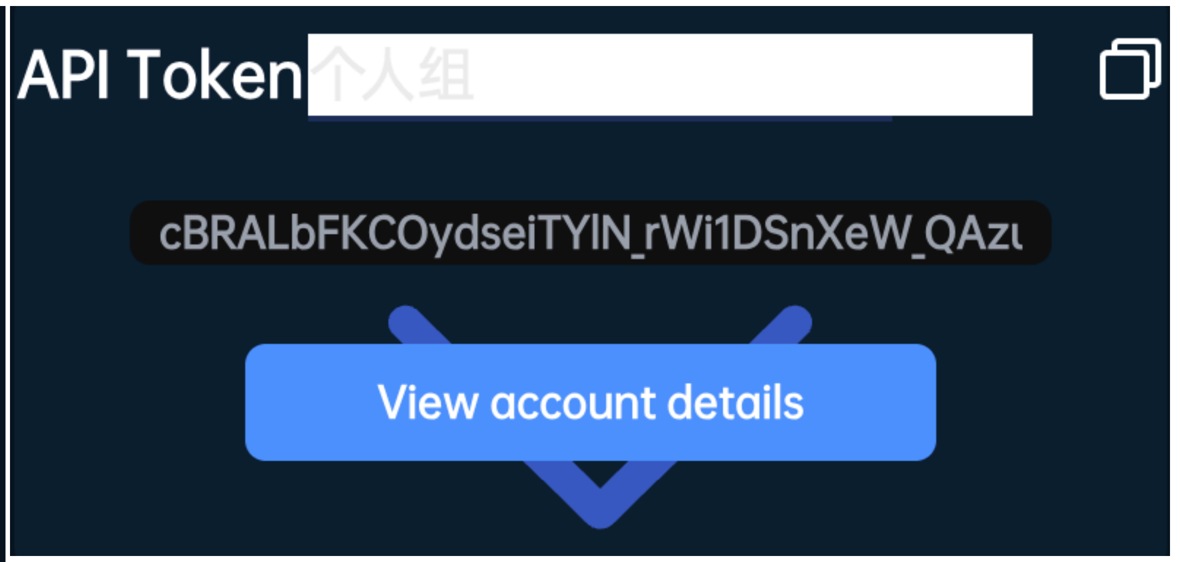
$$|\Phi^+\rangle = \frac{|00\rangle + |11\rangle}{\sqrt{2}}$$

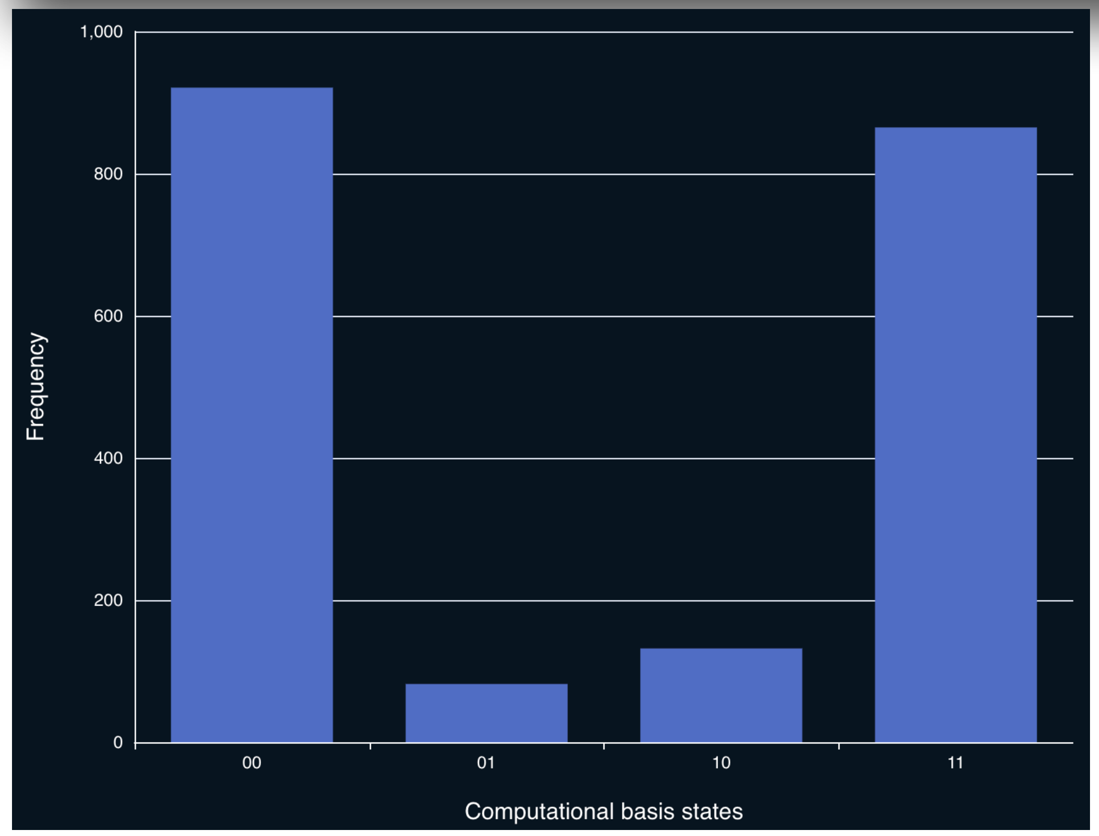
Running on real hardwares

- Quafu from Beijing Academy of Quantum Information Sciences [https://quafu.baqis.ac.cn]
- `pip install pyquafu` Some other quantum cloud platform: OriginQ (not free)

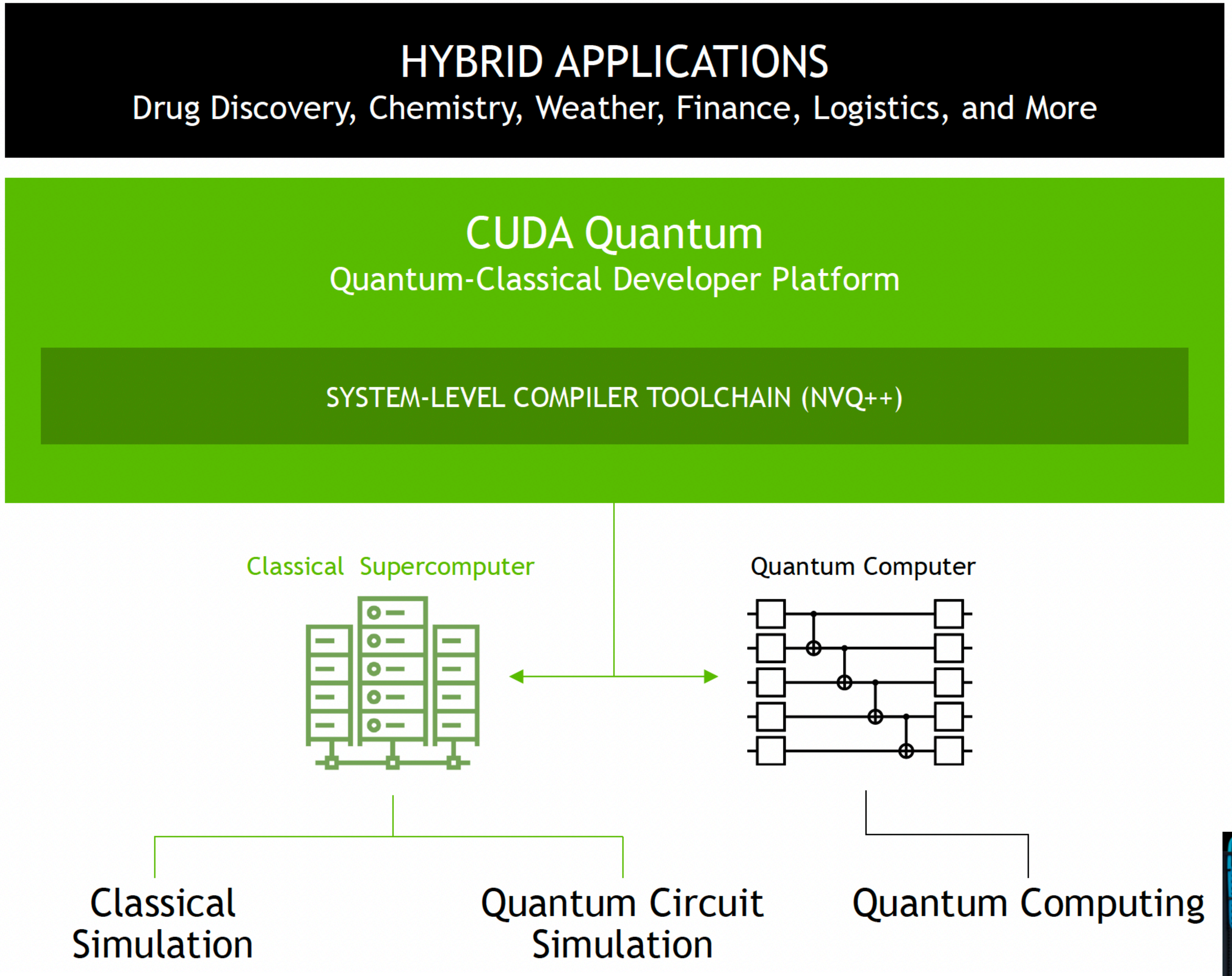

```python
from quafu import User
from quafu import QuantumCircuit
from quafu import Task

#user = User("<your API token>")
user = User("cBRALbFKCOydseiTYlN_rWi1DSnXeW_QAzu9-w3F9Da.
user.save_apitoken()
print(user.get_available_backends())

qc = QuantumCircuit(2)
qc.h(0)
qc.cnot(0,1)
qc.measure()

task = Task()
task.config(backend="Dongling", shots=2000, compile=True)

# submit job asynchronously
res = task.send(qc, wait=False)
# retrieve results after the job is done
#res = task.retrieve("<Your Task ID>")
print(res.counts) #counts
print(res.probabilities) #probabilities
```





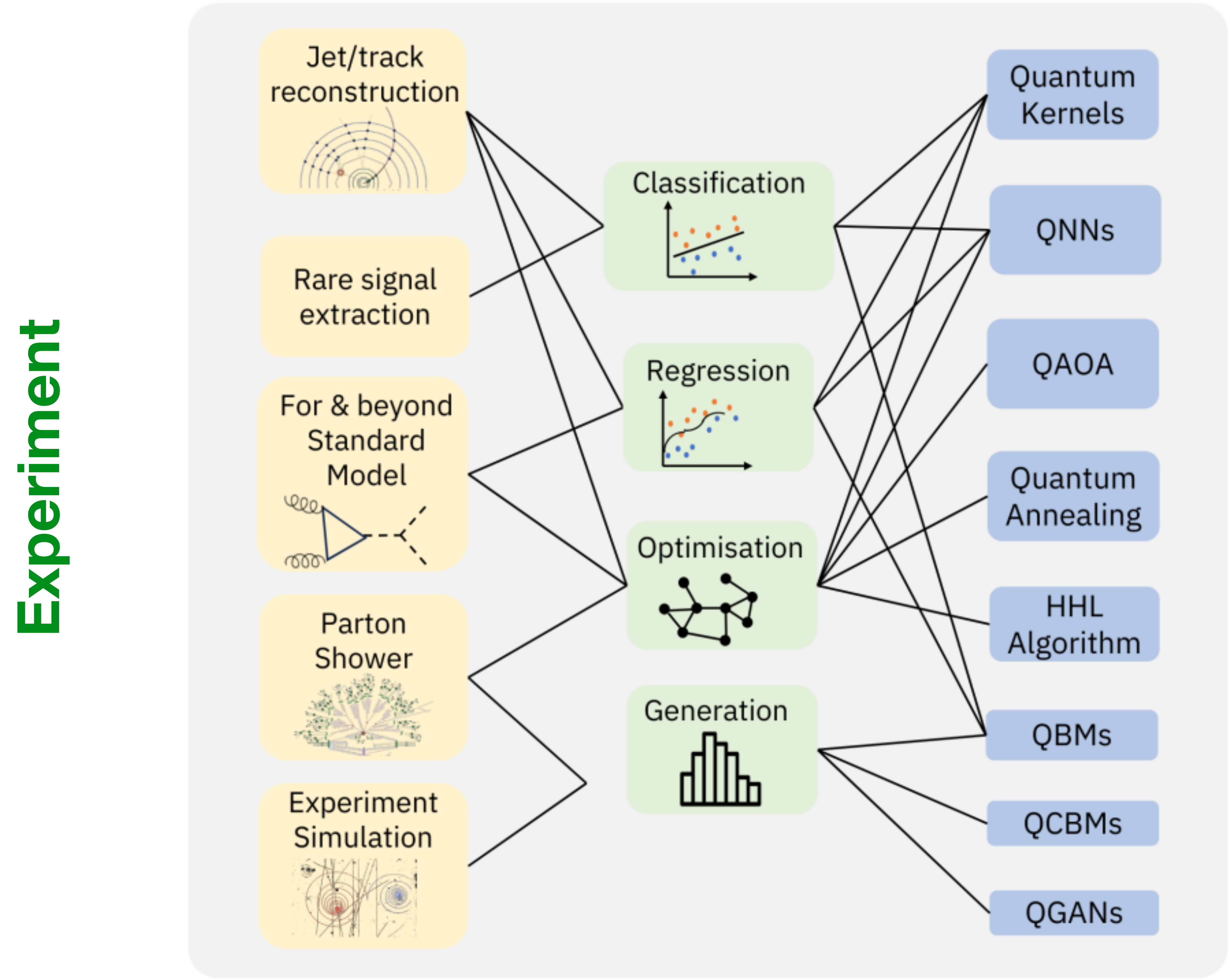


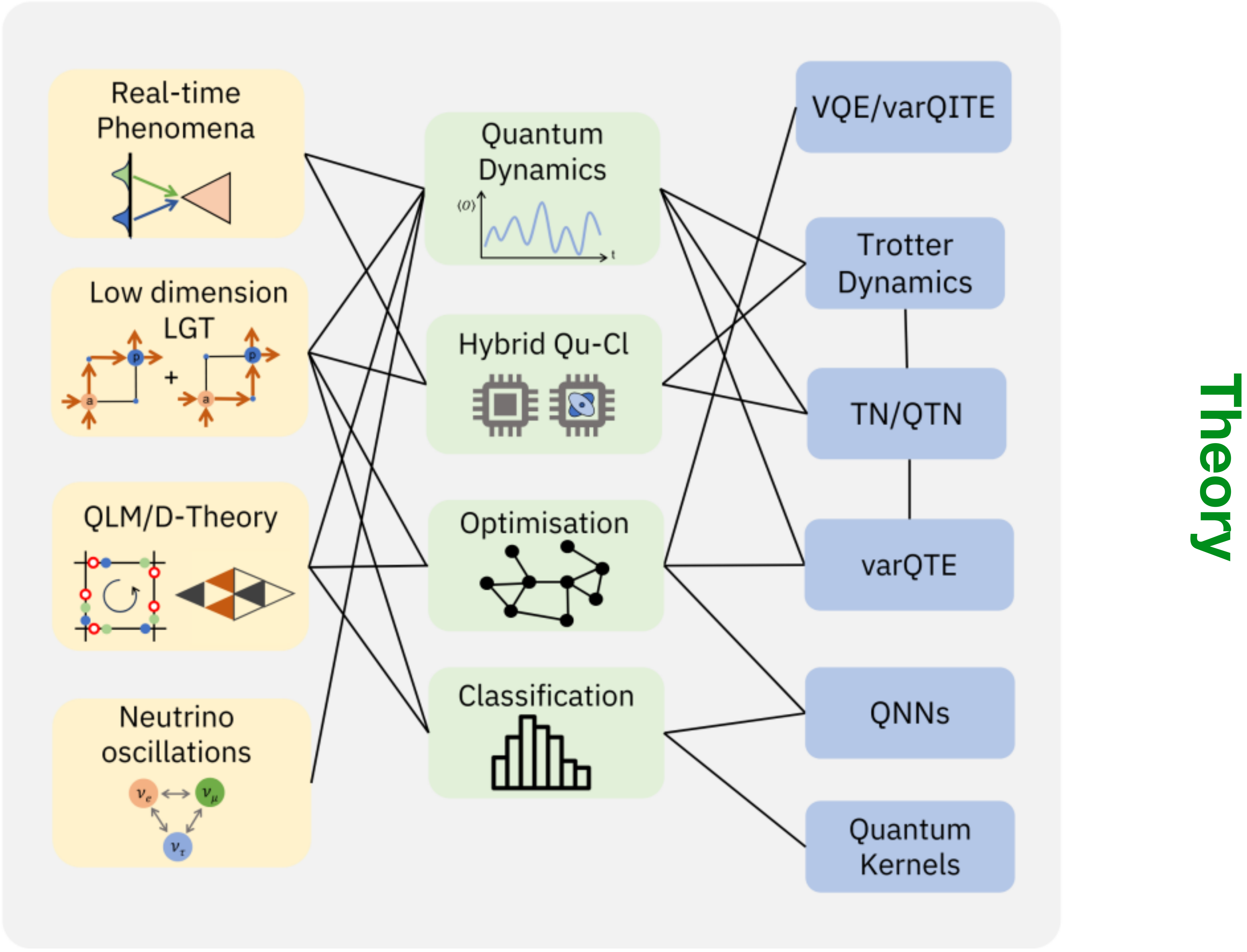
The future - hybrid quantum classical computing




```
#include <cudaq.h>

int main() {
      // Define the CUDA Quantum kernel as a C++ lambda
        auto ghz =[](int numQubits) __qpu__ {
           // Allocate a vector of qubits
                cudaq::qvector q(numQubits);

           // Prepare the GHZ state, leverage standard
           // control flow, specify the x operation
           // is controlled.
                h(q[0]);
                for (int i = 0; i < numQubits - 1; ++i)
                        x<cudaq::ctrl>(q[i], q[i + 1]);
        };

     // Sample the final state generated by the kernel
auto results = cudaq::sample(ghz, 15);
results.dump();

return 0;
}
```

Application of quantum computing in HEP





Quantum machine learning for HEP experiments

- **Classification of particle collision events**
- **Particle track reconstruction**
- W. Guan et al, Mach. Learn.: Sci. Technol. 2021

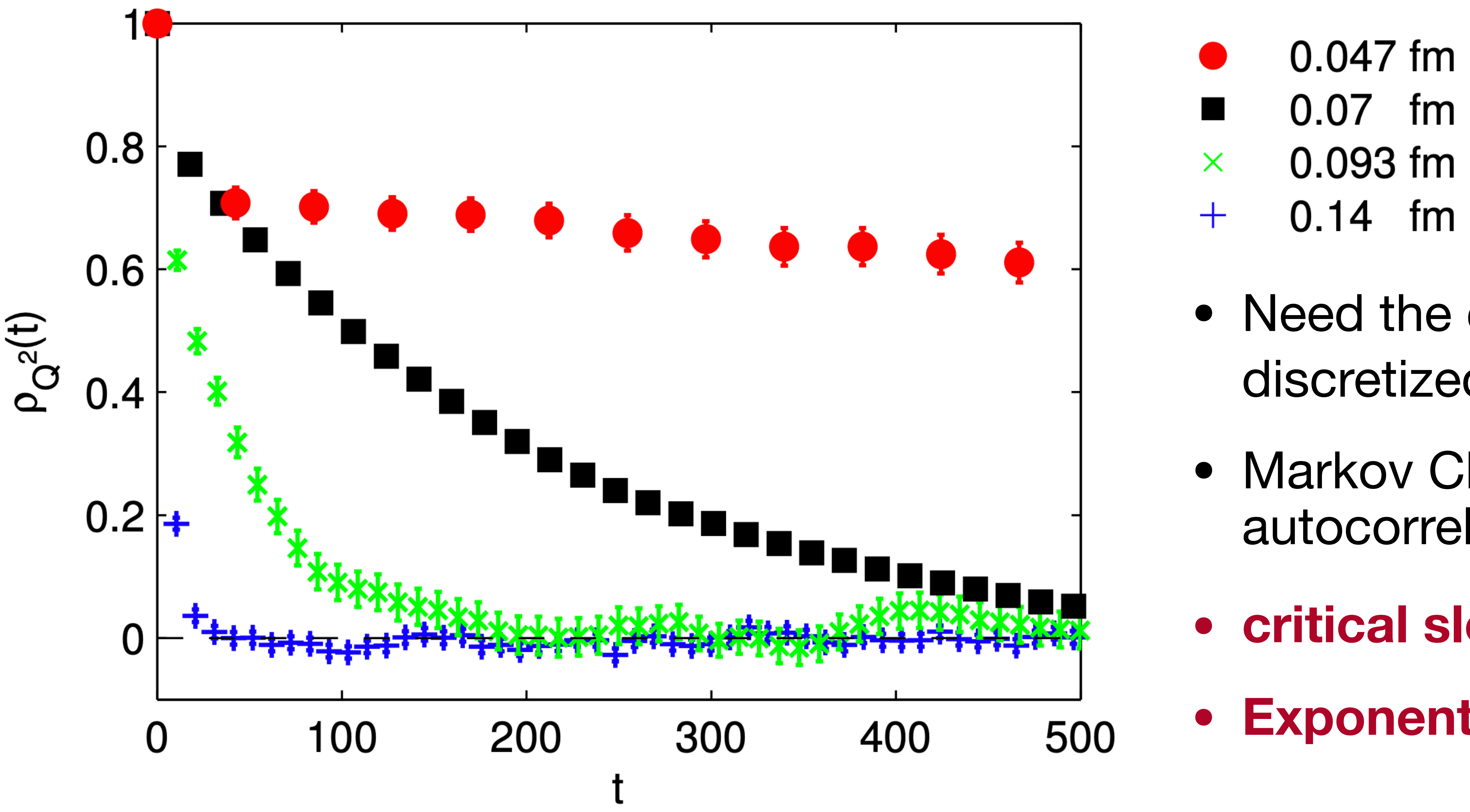
Quantum simulation of quantum field theories

- **1+1 dimensional model on atomic, optical, trapped ion, superconducting qubits**
- C. Bauer et al., PRX Quantum 4, 027001, 2023

Summary of the QC4HEP Working Group [arXiv: 2307.03236]

Why quantum simulation of quantum field theory

- **critical slowing down problem**



ALPHA Collaboration, *Nucl.Phys.B* 845 (2011) 93-119

- Need the continuum limit for lattice discretized field theories $(a \to 0)$
- Markov Chain Monte Carlo method exhibit autocorrelation inherently
- **critical slowing down**: $a \to 0,\ \tau_{\text{int}} \to \infty$
- **Exponential growth in computing time**

Why quantum simulation of quantum field theory

- **sign problem**

$$\langle O \rangle = \int DU\ O(U)\ e^{-S}$$

S is real → **Monte Carlo method**

- ☑ Hadron spectroscopy
- ☑ Hadron structure
- ☑ Finite temperature phase transition
- ☑ Standard model precision test
- ☑ …

S is complex → **Sign problem**

- ✗ Strong CP violation
- ✗ Quark gluon plasma
- ✗ Finite density QCD phase transition
- ✗ Properties of neutron star
- ✗ …

Lagrangian v.s. Hamiltonian formulation

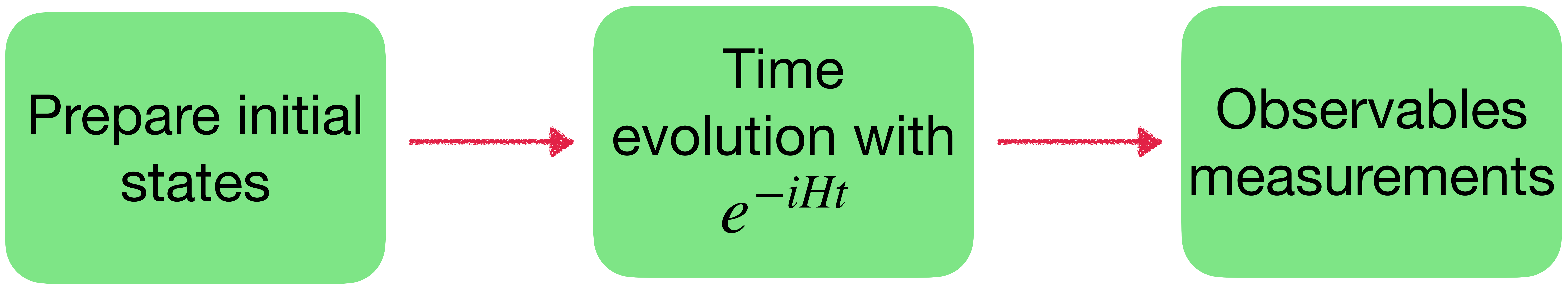
	Path integral (Lagrangian)	Hamiltonian
Degrees of freedom	Fields and their derivatives	Fields and their conjugate variables
Spacetime signature	Often Euclidean	Minkowski
Starting point	$\mathcal{L}[\varphi, \partial\varphi]$	$\hat{H}[\hat{\varphi}, \hat{\pi}]$
Hilbert space	Not explicitly constructed/relevant	Built out of $O^\dagger \lvert \text{vac.} \rangle^*$ $^*\lvert \text{vac.} \rangle = \lvert \text{empty state} \rangle$
Expectation values	$\frac{1}{\mathcal{Z}} \int \mathcal{D}\varphi \; e^{-S} O$	$\langle \psi \lvert \hat{O} \rvert \psi \rangle$
Dynamical quantities	Sometimes accessible with indirect methods, e.g., Luescher method.	In principle accessible: $\langle \psi \lvert e^{i\hat{H}t} \hat{O} e^{-i\hat{H}t} \rvert \psi \rangle$
Computational methods	Monte Carlo, etc.	Classical Hamiltonian methods like exact diag., tensor networks/ quantum simulation
Computational challenge	Sign and signal-to-noise problem for real-time quantities and finite-density systems.	Exponential scaling of the Hilbert space with the number of DOF.

Figure from Zohreh Davoudi's CERN-NORDIC school lecture

Properties of Hamiltonian

- The central equation of QM is Schrodinger equation $|\psi(t)\rangle = e^{-iHt}|\psi(0)\rangle$
  - H is Hermitian, with all real eigenvalues
  - e^{-iHt} is unitary
- Exact diagonalization to find energy eigenstates
  - Definition of some matrix function $e^A = \sum_{i=0}^{\infty} \frac{A^i}{i!}$
  - For example, $e^{\begin{bmatrix} \lambda_1 & & \\ & \cdots & \\ & & \lambda_n \end{bmatrix}} = \begin{bmatrix} e_1^{\lambda} & & \\ & \cdots & \\ & & e_n^{\lambda} \end{bmatrix}$
  - If $A = UDU^{-1}$ where D is diagonal, then $e^A = Ue^DU^{-1}$
  - With $H = UDU^{\dagger}$, U unitary, D diagonal, $e^{-iHt} = Ue^{-iDt}U^{\dagger}$
  - Exponential growth in dimension of H

Quantum simulation in general



Trotter formula

$$e^{-i(H_1+H_2+...H_n)t} = (e^{-iH_1\delta_t}e^{-iH_2\delta_t}\dots e^{-iH_n\delta_t})^{t/\delta_t} + O((\delta_t)^2)$$

first order

$$e^{-i(H_1+H_2+...H_n)t} = \left((e^{-iH_1\delta_t/2}e^{-iH_2\delta_t/2}\dots e^{-iH_n\delta_t/2})(e^{-iH_n\delta_t/2}\dots e^{-iH_2\delta_t/2}e^{-iH_1\delta_t/2})\right)^{t/\delta_t} + O((\delta_t)^3)$$

second order

Jordon-Wigner transformation

Jordan–Wigner transform fermionic operators in terms of the Pauli operators $\{I,\ \sigma_x,\ \sigma_y, \sigma_z\}$

$$a_j \Leftrightarrow \mathbf{1}^{\otimes j-1} \otimes \sigma^+ \otimes \sigma_z^{\otimes N-j-1} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}^{\otimes j-1} \otimes \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \otimes \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}^{\otimes N-j-1}$$

$$a_j^\dagger \Leftrightarrow \mathbf{1}^{\otimes j-1} \otimes \sigma^- \otimes \sigma_z^{\otimes N-j-1} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}^{\otimes j-1} \otimes \begin{bmatrix} 0 & 0 \\ 1 & 0 \end{bmatrix} \otimes \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}^{\otimes N-j-1}$$

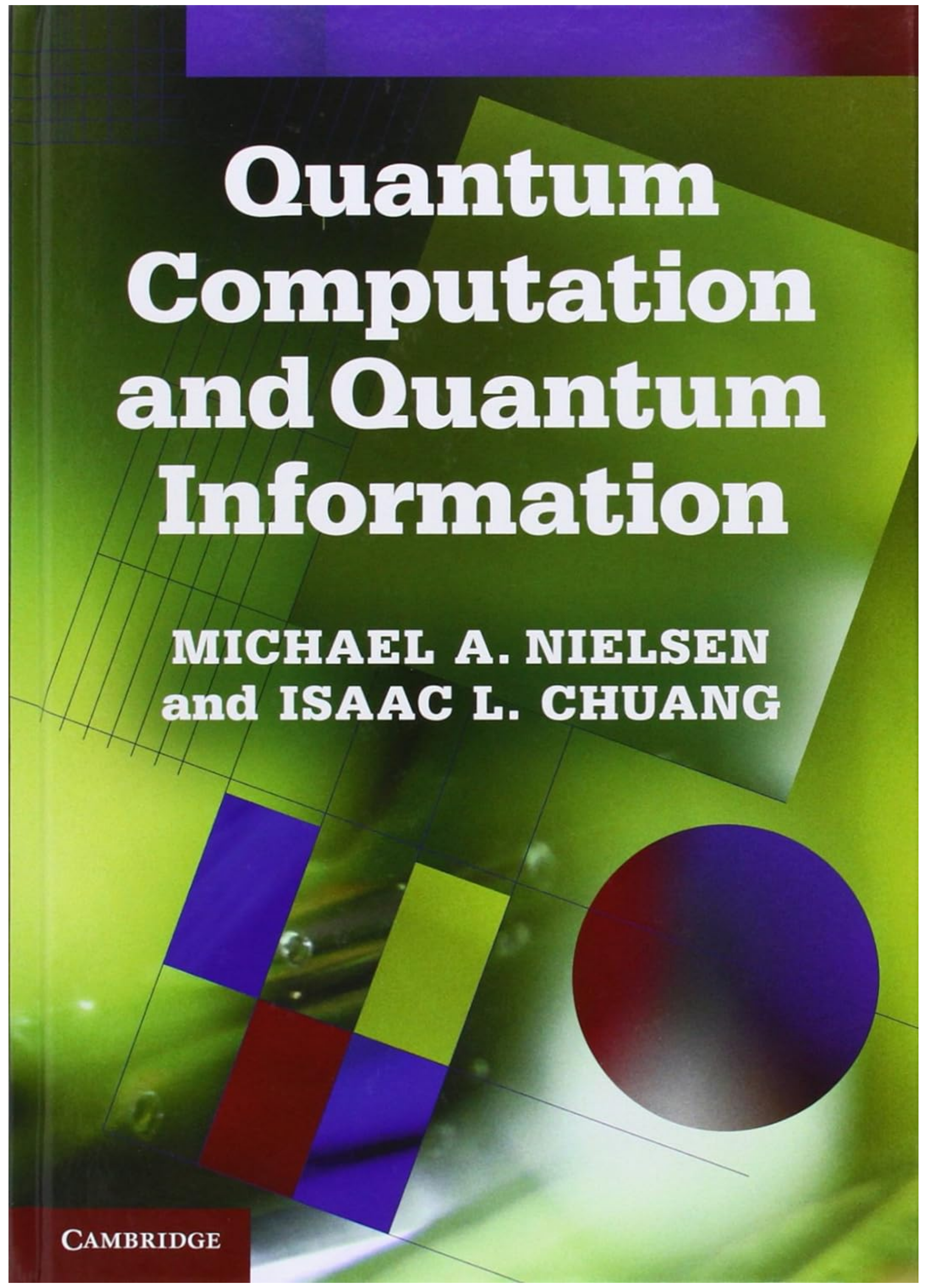
where $\sigma^+ \equiv \left(\sigma_x + \mathrm{i}\sigma_y\right)/2$ and $\sigma^- \equiv \left(\sigma_x - \mathrm{i}\sigma_y\right)/2$

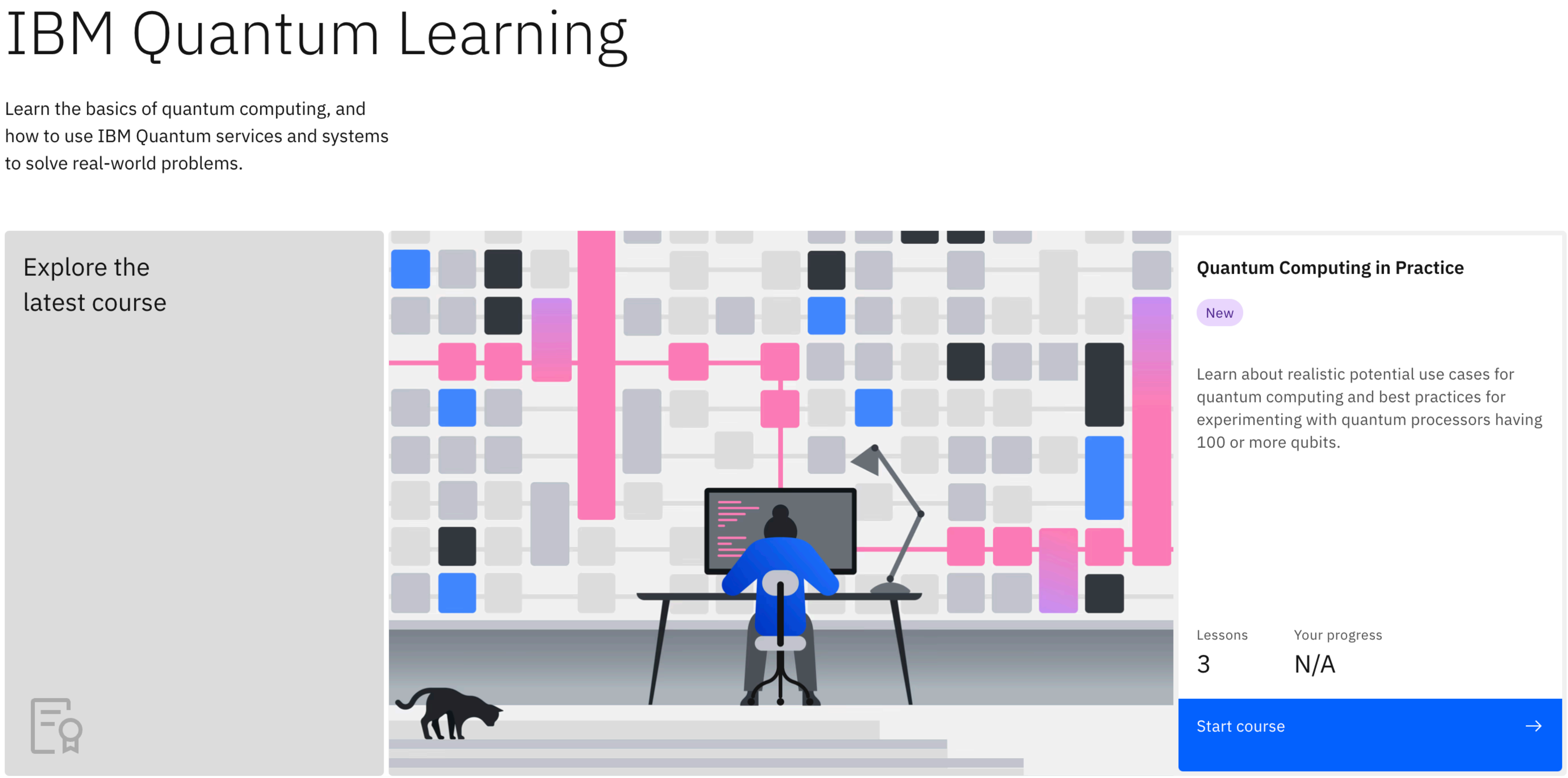
Then the fermionic anti-commutation relations are satisfied $\left[a_p, a_q\right]_+ = 0, \quad \left[a_p, a_q^\dagger\right]_+ = \delta_{pq}\mathbf{1}$

Note we'll not cover the qubit representation of **gauge** fields, which is a crucial aspect for the quantum simulation of standard model, but much hard than fermionic part

Summary and further reading

- Covered the very basics of quantum computing, quantum simulation, quantum programming softwares and running jobs on real hardware.
- Further reading: plenty of useful online resources





A practical introduction to quantum computing(CERN): https://indico.cern.ch/event/970903/

If you are interested and want more in-depth discussion on quantum computing and quantum simulation in HEP, please contact me (sunwei@ihep.ac.cn)