Quantum Coding Course

Ali Kookani
Yousef Mafi

Summer 2024

اطلس کوانتوم

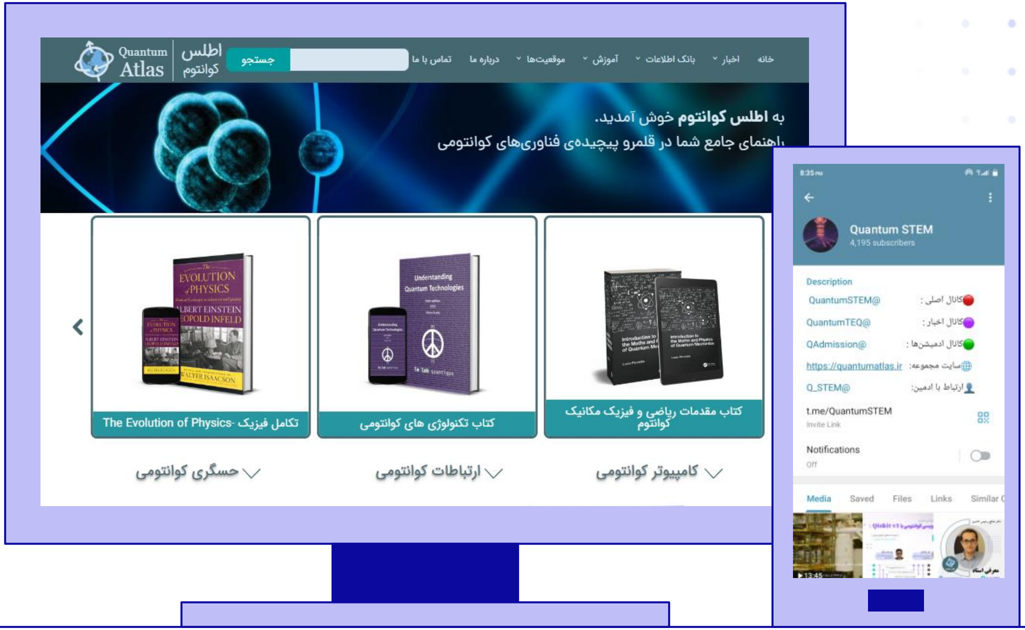
About us

Quantum Atlas is an educational group which aims to educate people in various fields of quantum, from hardware to software and quantum machine learning.

www.quantumatlas.ir

QuantumSTEM

Quantum Atlas



Syllabus

اطلس کوانتوم

Section 1	Lecture 1	Quantum Computation and Information (Theoretical lecture) – By Y. Mafi
	Lecture 2	Quantum Circuits (Coding lecture) – By A. Kookani
Section 2	Lecture 3	Quantum Simulation (Coding lecture) – By A. Kookani
	Lecture 4	IBMQ and Error Correction (Implementation and Theoretical lecture) – By Y. Mafi
Section 3	Lecture 5	Quantum Algorithm (Theoretical lecture) – By Y. Mafi
	Lecture 6	Quantum Algorithm Simulation (Coding lecture) – By A. Kookani

Coding time ;)

اطلس کوانتوم

Qiskit function list

Single-qubit gates

Names	Example	Notes
H, Hadamard	qc.h(0)	Applies H gate to qubit 0
I, Identity	qc.id(2) or qc.i(2)	Applies I gate to qubit 2
P, Phase	qc.p(math.pi/2, 0)	Applies P gate with π/2 phase rotation to qubit 0
RX	qc.rx(math.pi/4, 2)	Applies RX gate with π/4 rotation to qubit 2
RY	qc.ry(math.pi/8, 0)	Applies RY gate with π/8 rotation to qubit 0
RZ	qc.rz(math.pi/2, 1)	Applies RZ gate with π/2 rotation to qubit 1
S	qc.s(3)	Applies S gate to qubit 3. Equivalent to P gate with π/2 phase rotation
$S^\dagger$	qc.sdg(3)	Applies $S^\dagger$ gate to qubit 3. Equivalent to P gate with 3π/2 phase rotation
SX	qc.sx(2)	Applies SX (square root of X) gate to qubit 2. Equivalent to RX gate with π/2 rotation
T	qc.t(1)	Applies T gate to qubit 1. Equivalent to P gate with π/4 phase rotation

Single-qubit gates

Names	Example	Notes
$T^\dagger$	qc.tdg(1)	Applies RZ gate with π/2 rotation to qubit 1
U	qc.u(math.pi/2, 0, math.pi, 1)	Applies rotation with 3 Euler angles to qubit 1
X	qc.x(3)	Applies X gate to qubit 3
Y	qc.y([0,2,3])	Applies Y gates to qubits 0, 2, and 3
Z	qc.z(2)	Applies Z gate to qubit 2. Equivalent to P gate with π phase rotation

Multi-qubit gates

Names	Example	Notes
CCX, Toffoli	qc.ccx(0,1,2)	Applies the X gate to quantum wire 2, subject to the state of the control qubits on wires 0 and 1
CH	qc.ch(0,1)	Applies the H gate to quantum wire 1, subject to the state of the control qubit on wire 0
CP, Control-Phase	qc.cp(math.pi/4,0,1)	Applies the phase gate to quantum wire 1, subject to the state of the control qubit on wire 0
CRX, Control-RX	qc.crx(math.pi/2,2,3)	Applies the RX gate to quantum wire 3, subject to the state of the control qubit on wire 2
CRY, Control-RY	qc.cry(math.pi/8,2,3)	Applies the RY gate to quantum wire 3, subject to the state of the control qubit on wire 2

Multi-qubit gates

Names	Example	Notes
CRZ	qc.crz(math.pi/4,0,1)	Applies the RZ gate to quantum wire 1, subject to the state of the control qubit on wire 0
CSwap, Fredkin	qc.cswap(0,2,3) or qc.fredkin(0,2,3)	Swaps the qubit states of wires 2 and 3, subject to the state of the control qubit on wire 0
CSX	qc.csx(0,1)	Applies the SX (square root of X) gate to quantum wire 1, subject to the state of the control qubit on wire 0
CU	qc.cu(math.pi/2,0, math.pi,0,0,1)	Applies the U gate with an additional global phase argument to quantum wire 1, subject to the state of the control qubit on wire 0
CX, CNOT	qc.cx(2,3) or qc.cnot(2,3)	Applies the X gate to quantum wire 3, subject to the state of the control qubit on wire 2

اطلس کوانتوم

Multi-qubit gates

Names	Example	Notes
CY, Control-Y	qc.cy(2,3)	Applies the Y gate to quantum wire 3, subject to the state of the control qubit on wire 2
CZ, Control-Z	qc.cz(1,2)	Applies the Z gate to quantum wire 2, subject to the state of the control qubit on wire 1
DCX	qc.dcx(2,3)	Applies two CNOT gates whose control qubits are on wires 2 and 3
iSwap	qc.iswap(0,1)	Swaps the qubit states of wires 0 and 1, and changes the phase of the $\|01\rangle$ and $\|10\rangle$ amplitudes by i
MCP, Multi-control phase	qc.mcp(math.pi /4, [0,1,2],3)	Applies the phase gate to quantum wire 3, subject to the state of the control qubits on wires 0, 1, and 2

Quantum circuit Info

Names	Example	Notes
depth	qc.depth()	Returns the depth (critical path) of a circuit if directives such as barrier were removed
size	qc.size()	Returns the total number of gate in a circuit
width	qc.width()	Returns the sum of qubits wires and classical wires in a circuit

Names	Example	Notes
clbits	qc. clbits	Obtains the list of classical bits in the order that the registers were added
data	qc. data	Obtains a list of the operations (e.g., gates, barriers, and measurement operations) in the circuit
global_phase	qc. global_phase	Obtains the global phase of the circuit in radians
num_clbits	qc.num_clbits	Obtains the number of classical wires in the circuit
num_qubits	qc.num_qubits	Obtains the number of quantum wires in the circuit
qubits	qc.qubits	Obtains the list of quantum bits in the order that the registers were added

OpenQASM: A Programming Language for Quantum Computing

- Created for quantum computing algorithms and applications.
- Uses a measurement-based quantum circuit model to define quantum programs.
- Enables higher-level compilers to connect with quantum hardware.
- Supports concurrent real-time classical computations.
- Allows users to switch between high-level and pulse representations of the same program.
- OpenQASM 3.0 includes classical control flow, instructions, data types, and an external mechanism for generic classical computations.
- Essential for quantum computing research and development.
- Benefits quantum computing projects.

Difference between .to_instruction() and .to_gate()

•**.to_gate()**: Use when you want to create a new gate from a quantum circuit. Limited to unitary gates.

•**.to_instruction()**: Use when you want to create a more general instruction from a quantum circuit that might involve more complexity than a simple gate. Not limited.

Save simulator state

Method name	Description
save_state	Saves the simulator state as appropriate for the simulation method
save_density_matrix	Saves the simulator state as a density matrix
save_matrix_product_state	Saves the simulator state as a matrix product state tensor
save_stabilizer	Saves the simulator state as a Clifford stabilizer
save_statevector	Saves the simulator state as a statevector
save_superop	Saves the simulator state as a superoperator matrix of the run circuit
save_unitary	Saves the simulator state as a unitary matrix of the run circuit

QuantumRegister and ClassicalRegister attributes

Names	Example	Notes
name	qr.name	Obtains the name of the quantum register
size	qr.size	Obtains the number of qubit wires in the quantum register

Names	Example	Notes
name	cr.name	Obtains the name of the classical register
size	cr.size	Obtains the number of qubit wires in the classical register

اطلس کوانتوم

Instruction class's methods and attributes

Names	Example	Notes
copy	**inst.copy("My inst")**	Returns a copy of the instruction, giving the supplied name to the copy
repeat	**inst.repeat(2)**	Returns an instruction with this instruction repeated a given number of times
reverse_ ops	**inst.reverse_op s()**	Returns an instruction with its operations in reverse order

Gate class's methods and attributes

Names	Example	Notes
control	gate.control(1)	Given a number of control qubits, returns a controlled version of the gate
copy	gate.copy("Mygate")	Returns a copy of the gate, giving the supplied name to the copy
inverse	gate.inverse()	Returns the inverse of the gate
power	gate.power(2)	Returns the gate raised to a given floating-point power
repeat	gate.repeat(3)	Returns a gate with this gate repeated a given number of times
reverse_ ops	gate.reverse_ops()	Returns a gate with its operations in reverse order
to_matri x	gate.to_matrix()	Returns an array for the gate's unitary matrix

Names	Example	Notes
definition	gate.definition	Returns the definition in terms of basic gates
label	gate.label	Obtains the label for the instruction
params	gate.params	Obtains the parameters to the instruction

quantum_info module

Use it to:

- Find the state vector / density matrix at the output of a circuit
- Find probabilities of outcomes
- Sample from state
- Find expectation value with respect to an observable
- Plot state
- Find unitary of a circuit

Don’t use it if:

- Circuit is large (many qubits, many gates)
- Circuit has classical registers or mid-circuit measurements

Coding time ;)