Representations of Quantum States

Variational Quantum Algorithms · Module 2 — Parameterized Circuits

Node 2.6

Position in Knowledge Graph

System: Variational Quantum Algorithms **Structural Domain:** Parameterized Circuits **Node:** 2.6

When you simulate, debug, or analyze a VQA before running it on real hardware, you have to choose a **representation** for the quantum state. The same state has many possible mathematical descriptions, and each representation has a different cost-vs-fidelity tradeoff.

This node walks through the four representations that matter for VQA work:

1. **Statevector** — full amplitude vector
2. **Product state** — restricted to factorizable states
3. **Density matrix** — for mixed states and noise
4. **Matrix product state (MPS)** — for low-entanglement states

A working VQA practitioner uses all four at different points in the pipeline. Knowing when to use which is one of the most practically useful skills in the field, and it is rarely taught explicitly.

---

1. Statevector

The most direct representation: a complex vector

$$|\psi\rangle = \sum_{x\in\{0,1\}^n} \psi_x |x\rangle, \quad \psi_x \in \mathbb{C}, \quad \sum_x |\psi_x|^2 = 1.$$

For n qubits, this is a vector of 2^n complex numbers. Memory cost: $2^n \times 16$ bytes (double-precision complex) $= 16$ GB at $n = 30$, 16 TB at $n = 40$. Hard ceiling for personal computers around $n = 28 - 30$; for serious clusters, around $n = 40 - 45$.

Operations are matrix-vector multiplies. Applying a single-qubit gate is $O(2^n)$. Applying an n-qubit unitary directly is $O(4^n)$, though structured unitaries (a circuit of single- and two-qubit gates) can be applied gate-by-gate at $O(2^n)$ per gate.

Use for: debugging small instances, computing exact expectation values, sanity-checking optimizer behavior on toy problems where the ground truth is known.

Don't use for: anything past $n = 30$, or for simulating noise (no room for noise channels in a pure state).

The thesis's experimental setup uses 8-qubit statevector simulation for the VQE benchmarks. At 8 qubits, the statevector is 4 KB and operations are essentially free.

---

2. Product State (Tensor Product of Single-Qubit States)

Restricted to states of the form

$$|\psi\rangle = |\psi_1\rangle \otimes |\psi_2\rangle \otimes \cdots \otimes |\psi_n\rangle.$$

Memory cost: $2n$ complex numbers (2 amplitudes per qubit). Linear in qubit count.

This representation is exact only for **non-entangled** states. For entangled states it is a strict approximation.

Use for: Hartree-Fock-style starting states (which *are* product states in the right basis), mean-field approximations, sanity-checking that an ansatz can break out of the product subspace.

Don't use for: any state past the first entangling layer of an ansatz. As soon as a CNOT runs, the state generally leaves the product subspace.

The product-state representation matters for VQAs mostly as a **starting point** or **classical baseline**, not as a working representation.

---

3. Density Matrix

For mixed states (statistical ensembles or open-system states), the right representation is

$$\rho = \sum_k p_k |\psi_k\rangle\langle\psi_k|, \quad p_k \geq 0, \quad \sum_k p_k = 1.$$

Memory cost: $2^n \times 2^n$ complex numbers $= 4^n \times 16$ bytes $= 16$ GB at $n = 15$, 16 TB at $n = 20$. Half the qubit ceiling of statevector, because the matrix has square as many entries.

Operations: unitary application is $\rho \to U\rho U^\dagger$, an $O(4^n)$ operation per gate. Noise channels are applied as Kraus sums, which is more general than statevector simulation can express.

Use for: noise simulation (mixed states are the natural language of decoherence), modeling realistic hardware behavior, computing fidelity to a target state.

Don't use for: problems past $n = 18$ or so, or when a pure-state representation suffices.

For the thesis's noise-aware simulations, density matrix representation is the workhorse.

---

4. Matrix Product State (MPS)

A clever approximation: write the amplitude tensor as a product of small matrices.

$$\psi_{x_1 x_2 \ldots x_n} = A^{(1)}_{x_1} A^{(2)}_{x_2} \cdots A^{(n)}_{x_n},$$

where each $A^{(i)}_{x_i}$ is a matrix of size $\chi_{i-1} \times \chi_i$ and χ_i is the **bond dimension** at site i.

For an exact representation of an arbitrary n-qubit state, you need bond dimension up to $2^{n/2}$ — exponential. But for **low-entanglement states**, a small bond dimension ($\chi = 16, 64, 256$) gives an excellent approximation.

The maximum half-chain entanglement entropy that an MPS with bond dimension χ can capture is $\log_2 \chi$. States obeying the **area law** for entanglement (most ground states of local Hamiltonians) can be efficiently represented this way; states with **volume-law entanglement** (random circuits past a certain depth, scrambled states) cannot.

Memory cost: $O(n\chi^2)$ — linear in qubit count, polynomial in bond dimension. Operations: $O(n\chi^3)$ per gate. Tractable for n up to many hundreds when χ is small.

Use for: simulating large 1D-like quantum systems, classical pre-optimization of ansätze, studying how entanglement grows with depth.

Don't use for: problems with strong long-range entanglement, deep random circuits, or states that genuinely live near the volume-law regime.

MPS simulation is what makes "100-qubit VQE" classically tractable for shallow ansätze on chemistry-style problems — and what makes "100-qubit barren plateau study" intractable.

How To Choose

A practical decision tree:

- **Are you debugging a small instance?** → Statevector.
- **Are you studying noise effects?** → Density matrix.
- **Is the state shallow / low-entanglement?** → MPS.
- **Are you computing a Hartree-Fock baseline?** → Product state.
- **Do you need exact answers and n is small enough?** → Statevector.
- **Is n large but the ansatz is shallow?** → MPS, then validate with statevector on a smaller instance.
- **Is the state highly entangled and n is large?** → You're in trouble. This is the regime where classical simulation fails and you actually need the quantum hardware.

The last point is the **quantum advantage frontier**: VQAs are interesting precisely when none of the classical representations can handle the state cheaply.

What The Hardware Returns

None of these representations is what the hardware itself stores. The hardware stores a physical quantum state, and you can only query it through measurement (node 2.7). You **never see** the statevector or density matrix from a real device — you see samples from a distribution induced by it.

This means:

- Debugging on a real device is much harder than debugging in simulation, because you can't inspect intermediate states.
- Comparing simulation to hardware requires statistically reconciling measurement outcomes with simulated probabilities.
- Process tomography (reconstructing the full state) is expensive ($O(4^n)$ measurements) and only practical for very small n.

In practice, VQA development happens in three stages:

1. **Statevector simulation** — verify the ansatz is correct, the optimizer works, the cost converges.
2. **Density matrix simulation** — add a noise model, see how noise affects convergence.
3. **Real hardware** — submit to actual device, deal with the gap between modeled and real noise.

Skipping any stage is asking for trouble.

Structural Role

This node is the **practitioner's toolbox** node of Module 2. It is mostly free of authored opinions and consists of standard background that any VQA worker needs. Forwards to 2.7 (measurement) which describes

how the hardware exposes its state, and to 2.9 (ansatz as manifold) which uses the statevector representation to define the geometric picture.

It also feeds Module 4 (barren plateaus): the proofs of the barren plateau theorem use random unitary averages over the statevector representation, and being able to compute these averages depends on being able to write down the state.

Relations to Other Concepts

- **Tensor network states** — the broader family that MPS belongs to (PEPS for 2D, MERA for hierarchical, tree tensor networks for tree-like).
- **Stabilizer formalism** — a different polynomial classical representation, exact for Clifford circuits but unable to represent general states.
- **Neural network quantum states** — encode amplitudes as a neural network; can sometimes outperform MPS for non-1D structure.
- **Decoherence-free subspaces** — restricted state subspaces immune to particular noise types; useful for representation choice when noise structure is known.

Worked Example — Memory For 30 Qubits

How much memory does a 30-qubit state cost in each representation?

- **Statevector:** $2^{30} \times 16$ bytes = **16 GB**. Fits in a high-memory workstation.
- **Density matrix:** $2^{60} \times 16$ bytes = **16 EB**. Completely infeasible.
- **Product state:** 60 complex numbers = **480 bytes**. Trivial, but only correct for unentangled states.
- **MPS, bond dimension 64:** $30 \times 64^2 \times 16$ bytes $\approx$ **2 MB**. Trivial. But only correct if half-chain entanglement entropy is ≤ 6.
- **MPS, bond dimension 1024:** $30 \times 1024^2 \times 16$ bytes $\approx$ **500 MB**. Workable. Captures entanglement entropy up to 10.

Lesson: the order-of-magnitude difference between representations is larger than the order-of-magnitude difference between problem sizes. **Choosing the right representation is more important than buying more RAM.**

Visualization

Pending. A 4-quadrant plot. X-axis: qubit count. Y-axis: log memory cost. Four lines, one per representation, with characteristic crossovers and ceilings. Annotations on each line indicating "use this region."

Exercises

1. Show that a Bell state $\frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$ cannot be written as a product state $|\psi_1\rangle \otimes |\psi_2\rangle$ for any choice of $|\psi_1\rangle, |\psi_2\rangle$.
2. What is the bond dimension required to represent the Bell state as an MPS? What about the GHZ state $\frac{1}{\sqrt{2}}(|00\ldots0\rangle + |11\ldots1\rangle)$ on n qubits?
3. (VQA) For a hardware-efficient ansatz with L layers on n qubits, give a heuristic argument for the minimum bond dimension needed to represent the resulting state to fixed accuracy as a function of L and n.

Research Extensions

- **Tree tensor networks for VQA** — alternative to MPS, sometimes better for branched problem structures.
- **Neural quantum states for VQA simulation** — using NQS as a classical baseline against quantum hardware.
- **Approximate density matrix methods** — low-rank density matrix representations for noisy state simulation at higher qubit count.

Cross-links to Other Courses

- **Quantum Information** — pure vs mixed states, density matrix formalism, partial traces.
- **Tensor Networks (separate course)** — MPS, PEPS, MERA, contractions.
- **Numerical Methods** — large-vector arithmetic, sparse matrix tricks.
- **Statistics Module 7** — high-dimensional state representations and their information geometry.

Variational Quantum Algorithms · Module 2 — Parameterized Circuits · Node 2.6
Concepts: hilbert-space, density-matrix, dimensionality, vector-space
Prerequisites: 2.1
Enables: 2.7, 2.9

hbar.university — own.your.cognition