Module 9 — Hardware Noise Models

Variational Quantum Algorithms

hbar.university

Quantum Noise Channels

Position in Knowledge Graph

System: Variational Quantum Algorithms **Structural Domain:** Hardware Noise Models **Node:** 9.1

This is the **opening node** of Module 9. Modules 1-8 have mostly treated VQAs as if they ran on ideal quantum computers: unitary evolution, exact measurements, no decoherence. That picture is useful for theory but wrong in every practical sense. **Real VQAs run on noisy hardware**, and noise changes the analysis of essentially every module:

- It shifts the cost landscape (Modules 5, 6).
- It creates new barren plateaus (Module 4.5).
- It modifies the expressivity-trainability tradeoff (Module 8.5).
- It introduces drift that adaptive controllers must reject (Module 7.6, Module 10).

Module 9 is the module that takes noise seriously. It builds up the noise framework from first principles — quantum channels, specific noise models, gate error budgets, mitigation techniques — and by the end of the module, you can read any experimental VQA paper and understand what noise assumptions are in play.

This node introduces the mathematical framework: **quantum channels**, the operator-sum (Kraus) representation, and the Stinespring dilation. Noise in VQAs is modeled as a quantum channel applied after each gate (or after the whole circuit), and everything downstream builds on this foundation.

What A Quantum Channel Is

A **quantum channel** $\mathcal{E}$ is a linear map from density matrices to density matrices:

$$\mathcal{E} : \rho \mapsto \mathcal{E}(\rho),$$

satisfying three properties:

1. **Linear:** $\mathcal{E}(a\rho_1 + b\rho_2) = a\mathcal{E}(\rho_1) + b\mathcal{E}(\rho_2)$.
2. **Trace-preserving:** $\mathrm{Tr}(\mathcal{E}(\rho)) = \mathrm{Tr}(\rho)$ for all ρ.
3. **Completely positive:** $(\mathcal{E} \otimes I_n)(\rho)$ is positive-semidefinite for all ρ and all $n \geq 0$.

Together these are called **CPTP maps** (completely positive trace-preserving). A quantum channel is the most general physically allowed transformation of a quantum state, and every noise process can be modeled as one.

Examples:

- **Unitary evolution:** $\mathcal{E}(\rho) = U\rho U^\dagger$. A noise-free channel.

- **Measurement + discard outcome:** $\mathcal{E}(\rho) = \sum_k P_k \rho P_k$ where $\{P_k\}$ are projectors. The post-measurement state with the outcome forgotten.
- **Depolarizing channel:** $\mathcal{E}(\rho) = (1-p)\rho + p \cdot I/d$. Section 9.2.
- **Amplitude damping:** models energy loss $|1\rangle \to |0\rangle$. Section 9.3.

The Kraus Representation

Every quantum channel can be written as

$$\mathcal{E}(\rho) = \sum_k K_k \rho K_k^\dagger, \quad \sum_k K_k^\dagger K_k = I,$$

where the $\{K_k\}$ are **Kraus operators** (or "operation elements"). The Kraus representation is not unique — a channel has many equivalent Kraus decompositions — but any one of them fully specifies the channel.

Properties:

1. **A channel with one Kraus operator is a unitary** (up to global phase): $K = U$, so $\mathcal{E}(\rho) = U\rho U^\dagger$.
2. **A channel with two Kraus operators is a "one-error" channel**: probability p of applying an error, probability $1-p$ of the identity. The depolarizing channel can be written with 4 Kraus operators (I, X, Y, Z).
3. **The number of Kraus operators is at most** d^2 for a d-dimensional system. So a single-qubit channel has at most 4 Kraus operators; a two-qubit channel has at most 16.
4. **The Kraus representation is a kind of "unraveling"** — it decomposes the channel into probabilistic branches, each branch being a distinct way the noise could act. This is the operational intuition.

Canonical Noise Channels

A handful of channels appear everywhere in quantum hardware modeling:

Bit flip channel

$$\mathcal{E}(\rho) = (1-p)\rho + p \cdot X\rho X.$$

Flips $|0\rangle \leftrightarrow |1\rangle$ with probability p. Classical bit error analogue.

Phase flip channel

$$\mathcal{E}(\rho) = (1-p)\rho + p \cdot Z\rho Z.$$

Flips the relative phase $|+\rangle \leftrightarrow |-\rangle$ with probability p. Purely quantum — no classical analogue.

Depolarizing channel

$$\mathcal{E}(\rho) = (1-p)\rho + p/3(X\rho X + Y\rho Y + Z\rho Z).$$

Applies a uniformly random Pauli error with probability p. The "worst-case" unbiased error model. Section 9.2.

Amplitude damping channel

$$\mathcal{E}(\rho) = E_0 \rho E_0^\dagger + E_1 \rho E_1^\dagger,$$

with $E_0 = \begin{pmatrix} 1 & 0 \\ 0 & \sqrt{1-\gamma} \end{pmatrix}, E_1 = \begin{pmatrix} 0 & \sqrt{\gamma} \\ 0 & 0 \end{pmatrix}$.

Models energy relaxation: $|1\rangle \to |0\rangle$ with probability γ. The dominant error in superconducting qubits. Section 9.3.

Dephasing channel

$$\mathcal{E}(\rho) = \begin{pmatrix} \rho_{00} & \sqrt{1-\lambda}\rho_{01} \\ \sqrt{1-\lambda}\rho_{10} & \rho_{11} \end{pmatrix}.$$

Destroys coherence between $|0\rangle$ and $|1\rangle$ without changing populations. Closely related to the phase flip but continuous rather than probabilistic.

These five channels cover 95% of the noise models you'll encounter in practice. Real hardware is some weighted combination of these, with parameters (p, γ, λ) determined by calibration.

Stinespring Dilation

A deeper view: every quantum channel can be written as a unitary on a larger system.

Theorem (Stinespring 1955). For any CPTP channel $\mathcal{E}$ on a d-dimensional system, there exists an ancilla system of dimension at most d^2, a pure initial state $|0\rangle_{\text{anc}}$, and a unitary U on the joint system such that

$$\mathcal{E}(\rho) = \text{Tr}_{\text{anc}}\left[U(\rho \otimes |0\rangle\langle 0|_{\text{anc}})U^\dagger\right].$$

Interpretation. Every noise channel is equivalent to: add an ancilla in a known state, apply a joint unitary, then trace out (i.e., forget) the ancilla. The noise comes from the information lost when you discard the ancilla.

Why this matters. It means noise is not fundamentally mysterious — it is always "unitary evolution with part of the system hidden." This is important because:

1. It explains why quantum error correction works: if you can track the ancilla, the noise is reversible.
2. It unifies different noise models: they are all different choices of ancilla and joint unitary.
3. It connects to the fluctuation-dissipation theorem of statistical mechanics.

The Pauli Transfer Matrix

A useful computational tool: the **Pauli transfer matrix (PTM)** of a channel.

For a single-qubit channel, write $\rho = (I + r_x X + r_y Y + r_z Z)/2$ (Bloch vector representation). The channel acts linearly on $\mathbf{r} = (r_x, r_y, r_z)$:

$$\mathbf{r}' = T\mathbf{r} + \mathbf{t},$$

where T is a 3×3 matrix and $\mathbf{t}$ is a translation vector. Together $(T, \mathbf{t})$ form the **affine map** representation of the channel.

For the standard channels:

- **Identity:** $T = I$, $\mathbf{t} = 0$.
- **Bit flip with prob p:** $T = \mathrm{diag}(1, 1-2p, 1-2p)$, $\mathbf{t} = 0$.
- **Depolarizing with prob p:** $T = (1-p)I$, $\mathbf{t} = 0$.
- **Amplitude damping with prob γ:** $T = \mathrm{diag}(\sqrt{1-\gamma}, \sqrt{1-\gamma}, 1-\gamma)$, $\mathbf{t} = (0, 0, \gamma)$.

The PTM is convenient for computations — composition of channels is just matrix multiplication of PTMs, and expectation values of Pauli observables can be read off directly.

Composing Channels

Real circuits have many noise events. How do they compose?

Sequential composition. If you apply channels $\mathcal{E}_1, \mathcal{E}_2, \ldots, \mathcal{E}_L$ in sequence, the total channel is $\mathcal{E}_{\text{total}} = \mathcal{E}_L \circ \cdots \circ \mathcal{E}_2 \circ \mathcal{E}_1$.

In Kraus form: if $\mathcal{E}_i(\rho) = \sum_k K_{i,k} \rho K_{i,k}^\dagger$, then $\mathcal{E}_{\text{total}}$ has Kraus operators $K_{L,k_L} \cdots K_{2,k_2} K_{1,k_1}$ over all index sequences $(k_1, \ldots, k_L)$. The number of Kraus operators can grow exponentially — for an L-gate circuit with 2-Kraus-operator channels, up to 2^L terms.

In PTM form: $T_{\text{total}} = T_L \cdots T_2 T_1$. A single matrix product, regardless of circuit depth. This is much cheaper computationally.

For a depth-L circuit with uniform depolarizing channel $\mathcal{E}_p$ at each gate, the composed channel is $\mathcal{E}_{\text{total}}(\rho) = (1-p)^L \rho + (1 - (1-p)^L) \cdot I/d$. The **fidelity decays exponentially in L** — this is the root of all hardware-imposed depth limits.

Noise Rate Vs Fidelity

Two common ways of parameterizing noise:

1. Error probability p. The probability that a gate application produces an error. Used for discrete noise models (Pauli channels).

2. Process fidelity F. The fidelity of the noisy channel to the ideal channel. Related to p by (for depolarizing) $F = 1 - p(d^2-1)/d^2 \approx 1 - p$ for small p.

3. Average gate fidelity $\bar{F}$. The fidelity averaged over input states (Haar average). Related to process fidelity by $\bar{F} = (dF + 1)/(d+1)$.

For practical purposes, **gate fidelity** is the number you'll see on hardware spec sheets: "single-qubit fidelity 99.9%, two-qubit fidelity 99.5%". The corresponding error probabilities are 10^{-3} and 5×10^{-3} respectively.

In 2026, superconducting qubits typically achieve $\sim 99.9\%$ single-qubit and $\sim 99.5 - 99.8\%$ two-qubit fidelity. Trapped ions reach $\sim 99.99\%$ single-qubit and $\sim 99.9\%$ two-qubit. Neutral atoms are somewhere in between.

For a VQA circuit with hundreds of gates, these small per-gate errors compound: a 500-gate circuit at 99.5% two-qubit fidelity has total circuit fidelity $0.995^{500} \approx 0.08$ — only 8% of shots give correct results, 92% are pure noise.

Noise In The Heisenberg Picture

An alternative and often cleaner view: push the noise to the **observables**, not the states.

In the Heisenberg picture, a channel $\mathcal{E}$ acts on observables via its **adjoint** $\mathcal{E}^\dagger$:

$$\mathrm{Tr}[O\mathcal{E}(\rho)] = \mathrm{Tr}[\mathcal{E}^\dagger(O)\rho].$$

Implication for VQAs: instead of computing $\mathcal{E}(\rho)$ and then measuring O, you can compute $\mathcal{E}^\dagger(O)$ once and measure it against the noiseless state.

For depolarizing noise: $\mathcal{E}_p^\dagger(O) = (1-p)O + p \cdot \mathrm{Tr}(O)/d$. The observable's coefficient shrinks by $(1-p)$ while the identity component is unaffected. For Pauli observables (traceless), this is just $O \to (1-p)O$ — the expectation value is scaled down by $(1-p)$.

Compound effect: after L noisy gates with depolarizing rate p, the effective observable is $(1-p)^L O$. Measuring this is the same as measuring O on the noiseless state but with the result scaled by $(1-p)^L$ — which is exactly the expected fidelity decay.

This Heisenberg-picture view is the foundation of **zero-noise extrapolation** (node 9.5) and other mitigation techniques — you're essentially rescaling the measurement to compensate for the noise's effect on observables.

How VQAs See Noise

When running a VQA on noisy hardware, noise affects several things:

1. The cost function becomes biased. $C_{\text{noisy}}(\boldsymbol{\theta}) = \mathrm{Tr}[H\mathcal{E}(|\psi(\boldsymbol{\theta})\rangle\langle\psi(\boldsymbol{\theta})|)] \neq C_{\text{ideal}}(\boldsymbol{\theta})$. The optimizer is optimizing the wrong function.

2. The bias depends on $\boldsymbol{\theta}$. Different parameter values give different noisy costs, not all shifted by the same amount. So the optimizer can be actively misled.

3. The gradient becomes biased. $\nabla C_{\text{noisy}} \neq \nabla C_{\text{ideal}}$. Parameter-shift gradients are still *exact for the noisy cost*, but that's not the cost you want.

4. The landscape changes shape. Local minima, saddle points, and plateaus can all shift under noise. Module 4.5 covered the noise-induced plateau as an extreme example.

5. Shot noise adds on top. Even the noisy cost has to be estimated from finite shots, so there's an additional stochastic layer.

Practical upshot: the optimizer trained on a noisy VQA converges to the wrong minimum, and the wrongness grows with circuit depth and noise rate. This is the core practical problem that Modules 9-10 address.

Structural Role

This node is the **mathematical foundation** of Module 9. It introduces quantum channels, the Kraus and Stinespring representations, the Pauli transfer matrix, the composition rules, and the impact of noise on VQA cost functions.

It is the prerequisite for every other node in Module 9 — subsequent nodes (9.2 depolarizing, 9.3 amplitude damping, 9.4 mitigation, 9.5 ZNE, etc.) all build on the quantum-channel framework established here.

Relations to Other Concepts

- **Kraus (1971)** — the operator-sum representation.
- **Stinespring (1955)** — the dilation theorem.
- **Nielsen-Chuang** — *Quantum Computation and Quantum Information*, canonical reference.
- **Module 4.5 (Noise-induced plateaus)** — the qualitative consequence of these models.

- **Module 7.6 (Feedback control)** — treating noise as a disturbance to be rejected.

Worked Example — Depolarizing Channel On A 1-Qubit State

Take $\rho = |+\rangle\langle+| = (I + X)/2$ and apply the single-qubit depolarizing channel at rate $p = 0.1$:

$$\mathcal{E}_{0.1}(\rho) = 0.9 \cdot \rho + 0.1 \cdot I/2 = 0.9 \cdot (I + X)/2 + 0.1 \cdot I/2 = (I + 0.9X)/2.$$

The Bloch vector has shrunk from $(1, 0, 0)$ to $(0.9, 0, 0)$ — the state has moved toward the maximally mixed state.

Expectation value of X: $\mathrm{Tr}[X\rho] = 1$ (ideal) vs $\mathrm{Tr}[X\mathcal{E}(\rho)] = 0.9$ (noisy). The noise has decreased the expectation by a factor of $0.9 = 1 - p$.

Effect on a VQA cost function. If the cost is $C = \langle X \rangle$, the VQA sees $C_{\text{noisy}} = 0.9 C_{\text{ideal}}$. The optimizer still finds the right minimum (scaling doesn't change the location of extrema), just with a weaker signal. Signal-to-noise ratio is decreased, but convergence location is preserved.

Multi-layer circuit. Apply L layers of depolarizing noise. The expectation becomes $0.9^L \langle X \rangle$. For $L = 20$, this is $0.9^{20} \approx 0.12$ — only 12% of the signal remains. Below some threshold, the signal is indistinguishable from shot noise and the optimizer stalls.

This is the quantitative mechanism behind the depth limit on noisy hardware.

Visualization

Pending. A Bloch sphere showing a pure state and its trajectory under depolarizing noise as p increases from 0 to 1. The Bloch vector shrinks toward the center (maximally mixed). Caption: "Noise contracts the Bloch sphere. Everything collapses to $I/2$."

Exercises

1. Write down the Kraus operators for a bit flip channel with probability p. Verify they satisfy $\sum K^\dagger K = I$.
2. For a two-qubit system with single-qubit depolarizing noise (rate p) on each qubit independently, write down the joint channel as a Kraus decomposition. How many Kraus operators does it have?
3. (VQA) A VQA circuit has 200 gates with average two-qubit error $p = 0.005$. Estimate the total circuit fidelity. Is this sufficient for a useful result?

Research Extensions

- **Non-Markovian noise models** — correlations between errors that violate the CPTP assumption.
- **Learned noise models** — data-driven noise characterization from hardware calibration.
- **Structured noise for VQA design** — ansätze designed to exploit specific noise channels.
- **Coherent noise** — over-rotations and miscalibrations that don't fit the standard Pauli framework.

Cross-links to Other Courses

- **Open Quantum Systems** — Lindblad equation, master equations, decoherence theory.
- **Quantum Error Correction** — stabilizer codes, syndromes, logical operations.
- **Module 4.5 (Noise-induced plateaus)** — the qualitative effect on trainability.
- **Module 9.2 (Depolarizing Noise)** — the canonical example.

Depolarizing Noise

Position in Knowledge Graph

System: Variational Quantum Algorithms **Structural Domain:** Hardware Noise Models **Node:** 9.2

Depolarizing noise is the **workhorse noise model** of VQA theory. It is not the most physically realistic — real hardware has a mix of amplitude damping, dephasing, coherent errors, and crosstalk — but it is the most **mathematically tractable**, the most **symmetric**, and the most **commonly assumed** in theoretical analyses.

Understanding depolarizing noise deeply is worthwhile for several reasons:

1. It gives a **clean upper bound** on the effects of noise — if your VQA can handle depolarizing at rate p, it can usually handle more structured noise at similar rates.
2. It is the **simplest to simulate** — a single rate parameter controls everything.
3. It **reveals the depth-fidelity tradeoff** in closed form.
4. It is the **canonical example** for barren plateau theory under noise (Module 4.5, Wang et al. 2021).

This node derives the single-qubit depolarizing channel carefully, covers the multi-qubit generalizations, works through the composition rules for deep circuits, and explains why depolarizing noise is the "uniform" benchmark against which other noise models are measured.

The Single-Qubit Depolarizing Channel

The single-qubit depolarizing channel at rate p is defined as

$$\mathcal{D}_p(\rho) = (1-p)\rho + \frac{p}{3}\left(X\rho X + Y\rho Y + Z\rho Z\right).$$

Interpretation: with probability $1-p$, nothing happens (identity). With probability p, a uniformly random Pauli error X, Y, Z is applied. The factor $1/3$ splits the error probability evenly among the three Pauli types.

Kraus operators:

$$K_0 = \sqrt{1-p}\, I, \quad K_X = \sqrt{p/3}\, X, \quad K_Y = \sqrt{p/3}\, Y, \quad K_Z = \sqrt{p/3}\, Z.$$

Alternative form. Using the identity $(X\rho X + Y\rho Y + Z\rho Z + \rho)/4 = I/2 \cdot \mathrm{Tr}(\rho)$, the channel can be rewritten as

$$\mathcal{D}_p(\rho) = (1 - 4p/3)\rho + (4p/3) \cdot I/2.$$

Letting $\lambda = 4p/3$, this is $(1-\lambda)\rho + \lambda \cdot I/2$ — a **convex combination of the original state and the maximally mixed state**. This is the "depolarization" intuition: the state gets mixed with noise at rate λ.

Rescaling convention. Some authors use p to mean the total error probability (p above), others use p to mean the λ parameter. Be careful with the conventions.

Effect On Observables

Because the depolarizing channel is symmetric under Pauli conjugation, its action on Pauli observables is particularly simple:

$$\mathcal{D}_p^\dagger(P) = (1 - 4p/3)P \quad \text{for any non-identity Pauli } P.$$

And $\mathcal{D}_p^\dagger(I) = I$ (identity is preserved).

Consequence. Every Pauli expectation value is *uniformly shrunk* by a factor of $1 - 4p/3$. The measurement is unchanged in direction — only in magnitude.

For VQAs: if the cost Hamiltonian is $H = \sum_i c_i P_i$ (decomposed into Paulis), then the noisy cost is

$$C_{\text{noisy}}(\boldsymbol{\theta}) = (1 - 4p/3) \cdot C_{\text{ideal}}(\boldsymbol{\theta}) + 4p/3 \cdot \text{Tr}(H)/d.$$

The noisy cost is an **affine transformation** of the ideal cost: scaled by $(1 - 4p/3)$ and shifted by a constant. **Importantly, this affine transformation preserves the *location* of the minimum** — it only changes the value at the minimum and the slope of the landscape.

This is why depolarizing noise is often called "benign" for VQA optimization: the optimizer still finds the right parameters, just with a reduced signal-to-noise ratio. More problematic noise types (amplitude damping, coherent errors) actually *move* the minimum.

Multi-Qubit Depolarizing Channels

Several distinct generalizations exist for n qubits:

1. Independent single-qubit depolarizing

Each qubit experiences depolarizing noise independently:

$$\mathcal{D}_p^{\otimes n}(\rho).$$

Properties: factors over qubits; each single-qubit reduced state is depolarized independently; entanglement is *not* directly affected by the channel structure but does decay because of the per-qubit depolarization.

This is the most physically realistic model when the dominant noise is uncorrelated single-qubit error.

2. Global depolarizing

$$\mathcal{D}_p^{\text{global}}(\rho) = (1 - p)\rho + p \cdot I/2^n.$$

With probability p, the whole n-qubit state is replaced by the maximally mixed state; with probability $1 - p$, nothing happens.

Properties: the simplest theoretical model; often used in barren plateau proofs; rarely physically accurate, but gives clean bounds.

3. Correlated two-qubit depolarizing

Applied to pairs of qubits where two-qubit gates act:

$$\mathcal{D}_p^{(2)}(\rho) = (1 - p)\rho + \frac{p}{15} \sum_{k=1}^{15} P_k \rho P_k,$$

where the sum is over the 15 non-identity two-qubit Paulis. Used to model two-qubit gate errors.

Practical hardware modeling: real hardware is usually approximated as **single-qubit depolarizing after each single-qubit gate plus two-qubit depolarizing after each two-qubit gate**, with rates determined by the respective gate fidelities.

The Depth-Fidelity Tradeoff

A central calculation: **how does fidelity decay with circuit depth under depolarizing noise?**

For a circuit with L gates, each followed by depolarizing noise at rate p, the total channel on any Pauli observable is

$$\mathcal{D}^L(P) = (1 - 4p/3)^L P.$$

The signal decays exponentially in depth. Define the **effective depth** at which the signal is below the shot noise floor:

$$(1 - 4p/3)^{L^*} = \frac{1}{\sqrt{N_{\text{shots}}}}.$$

Solving: $L^* = -\log(\sqrt{N_{\text{shots}}})/\log(1 - 4p/3) \approx (3/4p) \cdot \log(\sqrt{N_{\text{shots}}})$.

Numerical examples:

- $p = 10^{-3}$, $N_{\text{shots}} = 10^6$: $L^* \approx 750 \cdot 6.9 \approx 5200$ gates.
- $p = 10^{-2}$, $N_{\text{shots}} = 10^6$: $L^* \approx 75 \cdot 6.9 \approx 520$ gates.
- $p = 10^{-1}$, $N_{\text{shots}} = 10^6$: $L^* \approx 7.5 \cdot 6.9 \approx 52$ gates.

Implication: for current hardware ($p \sim 5 \times 10^{-3}$ for two-qubit gates), useful VQA circuits are limited to ~1500 two-qubit gates. Beyond that, the signal is lost in noise and no amount of shots recovers it.

This is the **fundamental depth barrier** for NISQ-era VQAs, and it is tight — depolarizing noise is essentially the best-case scenario.

Depolarizing Noise And Barren Plateaus

Wang et al. (2021) showed that depolarizing noise creates **noise-induced barren plateaus** in addition to the unitary plateaus of Module 4.

Theorem (Wang et al. 2021). For a VQA with depth L under single-qubit depolarizing noise at rate p per gate, the gradient variance is bounded by

$$\text{Var}[\partial_j C_{\text{noisy}}] \leq C_0 \cdot (1 - 4p/3)^{2L},$$

where C_0 is a constant depending on the cost.

Implication: the gradient variance decays *exponentially in depth times noise rate*. Even ansätze that are trainable in the noiseless case become untrainable at sufficient depth under noise.

The noise-induced plateau depth: setting the variance bound equal to the shot noise floor $1/N_{\text{shots}}$, the plateau sets in at depth

$$L_{\text{NP}} \approx \frac{1}{4p/3} \cdot \frac{1}{2} \log(N_{\text{shots}} C_0^{-1}).$$

For $p = 5 \times 10^{-3}$, $N_{\text{shots}} = 10^6$, this is $L_{\text{NP}} \approx 500 - 1000$ gates — consistent with the depth limit from the fidelity argument.

The fundamental point: the depth limit from fidelity decay and the depth limit from noise-induced plateaus are *the same limit*, approached from two different angles. Depolarizing noise caps the usable circuit depth in exactly the regime where shot noise starts to dominate.

Why Depolarizing Is The Worst Case (In Some Sense)

A common claim: depolarizing noise is the "worst case" among unital noise channels at fixed average error rate.

Precise statement: for a fixed average gate fidelity, the depolarizing channel has the **maximum entropy** (most random) distribution of error types. Any structured noise (e.g., only phase errors, or only bit flips) gives a more concentrated error distribution.

Operational consequence: mitigation techniques that work for depolarizing noise work for more structured noise. Conversely, analyses that fail under depolarizing noise are unlikely to succeed under arbitrary noise.

Caveat: this is only true for **unital** channels (channels that preserve the maximally mixed state). **Non-unital** channels like amplitude damping are fundamentally different — they have a *preferred direction* in state space, and the "worst case" framing doesn't apply. Amplitude damping is discussed in node 9.3.

So: **depolarizing noise is the worst case among unital noise models**, and it is a useful upper bound for any noise that can be approximated as unital.

The Pauli Twirling Trick

A useful technique: **Pauli twirling** converts an arbitrary single-qubit channel into a depolarizing channel of equivalent fidelity.

Procedure. Before and after the noisy gate, apply a uniformly random Pauli (sampled from $\{I, X, Y, Z\}$). Average over many shots.

Effect. The average channel becomes

$$\bar{\mathcal{E}}(\rho) = \frac{1}{4} \sum_{P} P\mathcal{E}(P\rho P)P.$$

For a general single-qubit channel, this averages out to a **depolarizing channel** with the same average gate fidelity as the original.

Why this matters: twirling converts "messy" noise into "clean" depolarizing noise. Theoretical analyses that assume depolarizing noise are then *actually valid* for the twirled system, even if the raw hardware noise is more complex.

Cost: twirling doesn't reduce the noise — it just symmetrizes it. You pay for the simplification with the extra Pauli gates at each noise event.

Pauli twirling is the foundation of **randomized benchmarking** (the standard gate fidelity measurement protocol) and is used implicitly in many VQA mitigation schemes.

Depolarizing Noise In Practice

A few practical observations:

1. Real hardware is not purely depolarizing. Superconducting qubits have amplitude damping (energy loss) as the dominant noise; trapped ions have Rabi frequency errors; neutral atoms have loss and Rydberg decay. Depolarizing is an approximation.

2. The approximation is often good enough. For theoretical estimates and order-of-magnitude calculations, depolarizing is sufficient. Deviations matter at the factor-of-2 level, not the factor-of-100 level.

3. Simulation vs hardware. Depolarizing noise is the standard model in simulators (Qiskit Aer, Cirq, PennyLane). When you run a "noisy simulation," you're typically getting depolarizing unless you specify otherwise.

4. Randomized benchmarking returns depolarizing rates. The standard gate-fidelity measurement protocol assumes the noise is depolarizing (or has been twirled). Reported gate fidelities are depolarizing fidelities, which may differ from the "true" fidelity of the underlying physical noise.

5. Mitigation techniques built for depolarizing often extend. Zero-noise extrapolation (node 9.5) was originally developed for depolarizing noise and extends to broader noise classes. Similarly, probabilistic error cancellation is depolarizing-first.

So: depolarizing noise is both the theoretical default *and* the practical default, and knowing it deeply pays off.

Structural Role

This node is the **canonical noise example** of Module 9. It defines the single- and multi-qubit depolarizing channels, derives their action on observables, computes the depth-fidelity tradeoff explicitly, and establishes the connection to noise-induced plateau theory.

It is the benchmark against which other noise models (amplitude damping in 9.3, coherent errors in later sections) are compared.

Relations to Other Concepts

- **Nielsen-Chuang** — standard reference for the depolarizing channel.
- **Wang et al. (2021)** — noise-induced barren plateau theorem.
- **Randomized benchmarking (Magesan et al. 2011)** — fidelity measurement assuming depolarizing noise.
- **Pauli twirling** — converting arbitrary noise to depolarizing.
- **Module 4.5 (Noise-induced plateaus)** — the qualitative version.

Worked Example — Depth Limit For A VQA On 2026 Hardware

Consider a 20-qubit VQE on a chemistry Hamiltonian, running on superconducting hardware with: - Single-qubit gate fidelity: 99.9% $\to p_1 = 10^{-3}$. - Two-qubit gate fidelity: 99.5% $\to p_2 = 5 \times 10^{-3}$. - Shot budget: 10^6 per cost evaluation.

Ansatz: Hardware-Efficient Ansatz with 10 layers, each layer having 20 single-qubit gates + 19 two-qubit gates = 39 gates/layer. Total: 390 gates per circuit, of which 200 are two-qubit.

Total error: approximating each gate as an independent depolarizing channel, the circuit fidelity is

$$F \approx (0.999)^{200} \cdot (0.995)^{200} = 0.819 \cdot 0.367 = 0.30.$$

Only 30% of the signal survives. Combined with shot noise at $1/\sqrt{10^6} = 10^{-3}$, the effective SNR is $0.30/10^{-3} = 300$ — good enough for training.

Doubled depth (20 layers): $F \approx 0.30^2 = 0.09$. SNR ≈ 90. Still workable but degraded.

Quadrupled depth (40 layers): $F \approx 0.30^4 = 0.008$. SNR ≈ 8. Near the noise floor.

Conclusion: for this hardware, 10-20 layer circuits are usable, 40+ layers are beyond the noise-limited depth. This is consistent with the state-of-the-art 2026 experimental VQE demonstrations, which typically use 5-20 layer ansätze.

Visualization

Pending. A log-scale plot of gate fidelity vs circuit depth for several values of p (from 10^{-4} to 10^{-2}) showing the exponential decay. The shot noise floor is a horizontal line; the intersections give the maximum usable depth. Caption: "Depolarizing noise sets the NISQ depth ceiling."

Exercises

1. Compute the Pauli transfer matrix of a single-qubit depolarizing channel at rate p.
2. Show that the composition of two depolarizing channels at rates p_1, p_2 is a depolarizing channel. What is its rate?
3. (VQA) For a 30-qubit VQA with 500 two-qubit gates at fidelity 99.8%, estimate the circuit fidelity and the shot budget needed for a useful cost estimate.

Research Extensions

- **Non-uniform depolarizing rates** — different rates for different gates or qubits.
- **Coherent depolarizing deviations** — what if the noise is almost-depolarizing but not exactly?
- **Active depolarization** — intentionally adding depolarizing noise as a form of regularization for training stability.
- **Depolarizing-equivalent circuit compilation** — transforming circuits so their effective noise is depolarizing.

Cross-links to Other Courses

- **Open Quantum Systems** — Lindblad master equations for depolarizing noise.
- **Randomized Benchmarking** — the standard gate fidelity protocol.
- **Module 4.5 (Noise-induced plateaus)** — the qualitative version.
- **Module 9.5 (Zero-noise extrapolation)** — the main mitigation technique for depolarizing noise.

Gate Errors and Coherence Times

Position in Knowledge Graph

System: Variational Quantum Algorithms **Structural Domain:** Hardware Noise Models **Node:** 9.3

Node 9.2 treated depolarizing noise as the canonical model. This node gets **physical**: it explains where noise actually comes from in real hardware, what "gate error" and "coherence time" mean operationally, and how the abstract quantum channels of 9.1 connect to the physical parameters T_1, T_2, gate duration, and fidelity that appear on hardware spec sheets.

The goal: after this node, you can read a hardware specification (e.g., "IBM Heron r1 processor, 133 qubits, median T_1 150 µs, median T_2 80 µs, median 2Q fidelity 99.5%") and know exactly what it implies for VQA depth, shot budget, and expected performance.

This is the **bridge between theory and practice** — the point where abstract channels become concrete numbers.

The Sources Of Noise

Real quantum hardware has several distinct noise sources, each with its own physics and its own channel representation.

1. Energy relaxation (amplitude damping)

Physics: the excited state $|1\rangle$ can decay to the ground state $|0\rangle$ via spontaneous emission (superconducting), photon scattering (ions, atoms), or other dissipative mechanisms. The rate is characterized by the **energy relaxation time** T_1.

Channel. The amplitude damping channel at rate $\gamma = 1 - e^{-t/T_1}$ (where t is the elapsed time):

$$\mathcal{A}_\gamma(\rho) = E_0 \rho E_0^\dagger + E_1 \rho E_1^\dagger,$$

with

$$E_0 = \begin{pmatrix} 1 & 0 \\ 0 & \sqrt{1-\gamma} \end{pmatrix}, \quad E_1 = \begin{pmatrix} 0 & \sqrt{\gamma} \\ 0 & 0 \end{pmatrix}.$$

Key properties: - **Non-unital:** amplitude damping moves states *toward* $|0\rangle$, not toward $I/2$. There is a preferred direction. - **Breaks symmetry:** a state initialized to $|1\rangle$ decays, but $|0\rangle$ does not. Noise depends on the state. - **Cumulative:** at time t, the channel parameter is $\gamma(t) = 1 - e^{-t/T_1}$.

2. Dephasing (phase damping)

Physics: the coherence between $|0\rangle$ and $|1\rangle$ is destroyed by fluctuations in the environment (charge noise, flux noise, magnetic field variations). The rate is T_ϕ, and combines with T_1 to give the **total coherence time** T_2:

$$\frac{1}{T_2} = \frac{1}{2T_1} + \frac{1}{T_\phi}.$$

Channel. The phase damping channel at rate $\lambda = 1 - e^{-t/T_\phi}$:

$$\mathcal{P}_\lambda(\rho) = \begin{pmatrix} \rho_{00} & \sqrt{1-\lambda}\rho_{01} \\ \sqrt{1-\lambda}\rho_{10} & \rho_{11} \end{pmatrix}.$$

Key properties: - **Unital:** it preserves $I/2$. - **Diagonal-preserving:** computational basis populations are unchanged. - **Eigenvalue structure:** exponential decay of off-diagonal elements (coherences).

3. Coherent errors

Physics: miscalibrations in the control pulses cause systematic over- or under-rotations. Not random — deterministic but unknown.

Channel (example: over-rotation). $U_{\text{actual}} = R_Y(\theta + \delta)$ instead of $R_Y(\theta)$. No probability mixing — just a wrong unitary.

Key properties: - **Unitary:** still a valid quantum operation. - **Coherent accumulation:** many small errors can add up to a large error if they are correlated. - **Not captured by Pauli channels:** the standard depolarizing model misses coherent errors, which is a frequent source of theory-experiment discrepancy.

4. Crosstalk

Physics: a gate on qubit A affects qubit B due to unintended couplings. Purely a multi-qubit effect.

Channel. The ideal single-qubit gate U_A is replaced by a correlated two-qubit channel that depends on the state of qubit B. Hard to characterize, hard to model.

Key properties: - **Non-local:** creates correlations between distant errors. - **Context-dependent:** the effect depends on what other gates are happening in parallel. - **One of the dominant error sources** in 2026 superconducting hardware.

5. Measurement errors

Physics: the measurement operation itself is imperfect. A qubit in state $|0\rangle$ may be reported as "1" with some probability, and vice versa.

Channel. Modeled as a classical confusion matrix applied to the measurement outcome:

$$\begin{pmatrix} P(0_{\text{reported}}|0_{\text{true}}) & P(0_{\text{reported}}|1_{\text{true}}) \\ P(1_{\text{reported}}|0_{\text{true}}) & P(1_{\text{reported}}|1_{\text{true}}) \end{pmatrix}.$$

Typical values: 1-5% error rate for current superconducting hardware, $\sim 0.5\%$ for trapped ions.

Key properties: - **Classical:** can be corrected by post-processing if the confusion matrix is known. - **Asymmetric:** typically different rates for $0 \to 1$ and $1 \to 0$ errors. - **Dominates at shallow depth:** for very shallow circuits, measurement error can be the largest contribution.

The Relaxation-Time Parameters

The headline numbers for any piece of quantum hardware:

T_1 **(energy relaxation time):** the characteristic timescale for the excited state to decay. Typical values: - Superconducting (transmon): 50-300 µs. - Trapped ion: seconds. - Neutral atom (Rydberg): hundreds of µs. - Photonic: effectively infinite (no relaxation). - Silicon spin: milliseconds.

T_2 **(total dephasing time):** the characteristic timescale for coherences to decay, always $\leq 2T_1$. Typical values: - Superconducting: 20-200 µs (often $T_2 \ll 2T_1$ because of dephasing). - Trapped ion: hundreds of ms to seconds. - Neutral atom: 0.5-5 ms. - Photonic: effectively infinite. - Silicon spin: hundreds of µs (isotope-enriched).

T_2^* **(pure dephasing time without echo):** the dephasing time measured by a Ramsey experiment (no echo pulse). Usually shorter than T_2.

Gate time: the duration of a gate operation. Typical values: - Superconducting single-qubit: 20-50 ns. - Superconducting two-qubit: 100-500 ns. - Trapped ion: 10-500 µs. - Neutral atom: 100 ns-1 µs.

The gate error budget: a gate of duration τ on a qubit with coherence time T_2 has coherence-limited error $\sim \tau/T_2$. For superconducting hardware with $\tau = 300$ ns and $T_2 = 100$ µs, this is 3×10^{-3} — a ∼0.3% error floor from coherence alone. Actual two-qubit errors are usually 2-3x larger due to calibration and other non-idealities.

The Circuit Time Budget

For a circuit with L gates of duration τ, the total circuit time is $L\tau$. For the circuit to be useful, this must be much less than T_2:

$$L\tau \ll T_2 \quad \Longleftrightarrow \quad L \ll T_2/\tau.$$

Examples:

- Superconducting ($T_2 = 100$ µs, $\tau = 300$ ns): $L \ll 333$ gates. Usable depth ~ 100 gates.
- Trapped ions ($T_2 = 1$ s, $\tau = 100$ µs): $L \ll 10^4$ gates. Usable depth $\sim 1000 - 3000$ gates.
- Neutral atoms ($T_2 = 2$ ms, $\tau = 500$ ns): $L \ll 4000$ gates.

Observation: the gate-count-per-coherence-time is surprisingly similar across technologies in the low hundreds. The different technologies have different absolute speeds but similar "relative" circuit budgets.

For VQAs: the usable depth $\sim 100 - 1000$ gates is consistent with the depolarizing analysis of 9.2. Real depth is more like $\sim 10 - 50$ layers of an HEA for 20-30 qubits — limited by both coherence and noise-induced plateaus.

Fidelity As An Operational Metric

Gate fidelity is the headline metric, but it can mean several things:

1. Average gate fidelity $\bar{F}$: the average over Haar-random input states of the fidelity between the ideal and actual output. The number reported on spec sheets.

2. Process fidelity F_{proc}: the overlap between the ideal and actual process, normalized. Related to average fidelity by $\bar{F} = (dF_{\text{proc}} + 1)/(d + 1)$.

3. Entanglement fidelity: for two-qubit gates, the fidelity when the input is a maximally entangled state. Sometimes more informative.

4. Benchmarking fidelity: what **randomized benchmarking** reports. This is a specific operational protocol that effectively measures the twirled depolarizing rate, which is close to but not identical with the average gate fidelity.

For most practical purposes, **the randomized benchmarking fidelity is what hardware vendors report**, and that's what you should use in your depth-fidelity calculations. It's the depolarizing-equivalent fidelity after Pauli twirling.

The Amplitude Damping Vs Depolarizing Comparison

A practically important comparison: how different is amplitude damping from depolarizing?

At low rates ($\gamma \ll 1$): very similar. Both are approximately "error with probability $\sim \gamma$," and the detailed structure doesn't matter much for shallow circuits.

At higher rates: amplitude damping is *preferentially* toward $|0\rangle$, so it biases expectation values of Z toward $+1$ (the $|0\rangle$ eigenvalue). Depolarizing biases all Pauli expectations toward zero symmetrically.

For VQAs: if your cost function has significant Z character, amplitude damping will shift the cost systematically (upward for Z with positive coefficient). The optimizer will find a *different* minimum than the ideal one. This is a real practical issue with superconducting hardware.

Quantitative comparison: at average gate fidelity 99.5%, depolarizing noise shifts $\langle Z\rangle$ to $0.995\langle Z\rangle$; amplitude damping shifts $\langle Z\rangle$ to $\langle Z\rangle + (\text{const}) \cdot (1 - \text{rate})$ — a state-dependent additive shift.

Upshot: for theoretical analysis, assume depolarizing for clean bounds. For real hardware, use amplitude damping + dephasing. For quick practical estimates, depolarizing is usually accurate enough.

The Coherence-Limited Regime

A useful conceptual division: a VQA circuit is either **coherence-limited** or **gate-error-limited**.

Coherence-limited: the total circuit time is close to T_2, so the whole state decays before the circuit finishes. Further reductions in per-gate error don't help — the state is already decohering. This is the regime for ions (where individual gate fidelities are excellent but circuits are slow).

Gate-error-limited: the circuit is fast compared to T_2, but individual gates have non-trivial error rates. The total error is dominated by summing per-gate errors. This is the regime for superconducting qubits (fast gates, shorter coherence).

For VQA design:

- In the coherence-limited regime, **reducing gate count** is the right strategy. Every gate fewer is better because it saves circuit time.
- In the gate-error-limited regime, **reducing per-gate error** is the right strategy. Every calibration improvement compounds.

Most 2026 superconducting hardware is gate-error-limited for shallow circuits and coherence-limited for deep ones. The crossover depth depends on the specific machine.

Calibration And Drift

Real hardware parameters **drift** on timescales of minutes to hours:

- **Qubit frequencies** drift by tens of kHz over hours.
- **Gate fidelities** drift by 10-20% (relative) over a day.
- T_1, T_2 can vary 2x over a week.

Hardware providers run periodic **recalibration** (typically every 1-4 hours) to keep parameters within spec. Between calibrations, the noise model is slowly changing.

For VQAs: if a VQA training run takes 12 hours, the noise model at the end of the run is different from the model at the beginning. A static noise-aware compilation won't be valid throughout. The right response is **adaptive noise characterization** — periodically re-measuring the noise parameters and updating the compilation or the optimizer accordingly.

This is one of the practical deployment challenges that Module 10 (Adaptive Control Systems) addresses.

Structural Role

This node is the **physical-hardware** node of Module 9. It connects the abstract noise channels of 9.1-9.2 to the concrete parameters (T_1, T_2, gate time, fidelity) that appear on hardware spec sheets, and gives the practitioner the information needed to translate between hardware specs and VQA performance estimates.

It is the prerequisite for nodes 9.4 (mitigation) and 9.6 (noise-adaptive ansatz design), which build on the physical understanding.

Relations to Other Concepts

- **Bloch (1946), Redfield (1957)** — foundational work on relaxation theory.
- **Randomized benchmarking (Magesan et al. 2011)** — the standard fidelity measurement.
- **Quantum process tomography** — full characterization of a gate.
- **Module 9.1 (Quantum noise channels)** — the mathematical framework.
- **Module 10 (Adaptive control)** — handling drift via adaptation.

Worked Example — Noise Budget For A 15-Qubit QAOA

Target hardware: superconducting, $T_1 = 150$ µs, $T_2 = 80$ µs, single-qubit gate time 30 ns, two-qubit gate time 300 ns, single-qubit fidelity 99.9%, two-qubit fidelity 99.5%.

Circuit: 15-qubit QAOA at depth $p = 3$. Each layer has 15 single-qubit gates + 14 two-qubit gates = 29 gates. Three layers = 87 gates, of which 42 are two-qubit.

Time budget: 87 gates times average time ~ 200 ns ≈ 17 µs. **Well within $T_2 = 80$ µs.** Coherence is not the bottleneck.

Fidelity budget: - Single-qubit error contribution: $45 \cdot 10^{-3} = 0.045$. - Two-qubit error contribution: $42 \cdot 5 \times 10^{-3} = 0.21$. - Total depolarizing error: ≈ 0.26. Circuit fidelity ≈ 0.74.

Measurement error: typically 2% per qubit, so 15 qubits of measurement error contribute another ~ 0.3 to the total error.

Net: about 40-50% of the signal survives. Shots needed to resolve signal above shot noise: for a cost that is ~ 0.5 in magnitude on a good run and 0.25 after noise, shot noise of 10^{-3} (i.e., 10^6 shots) is comfortably below the signal.

Conclusion: this QAOA is workable. For depth 6, we'd double the gate count and the signal fidelity would drop to ~ 0.3. At depth 10, ~ 0.1 — workable but getting marginal. At depth 20, signal is lost.

This is the kind of calculation every experimentalist does before running a VQA.

Visualization

Pending. A chart showing, for several hardware platforms (superconducting, trapped ion, neutral atom, silicon), the typical T_1, T_2, gate time, and gate error. Overlaid: the usable circuit depth for each platform. Caption: "Hardware comes in flavors. Each flavor has a different depth budget."

Exercises

1. A qubit has $T_1 = 100$ µs, $T_2 = 60$ µs. What is the pure dephasing time T_ϕ?

2. For a superconducting two-qubit gate of duration 400 ns and fidelity 99.2%, what fraction of the error is coherence-limited vs gate-error-limited?
3. (VQA) For a 30-qubit VQE on trapped-ion hardware ($T_2 = 1$ s, two-qubit gate time 300 µs, fidelity 99.9%), estimate the maximum circuit depth.

Research Extensions

- **Time-dependent noise models** — capturing drift between calibrations.
- **Crosstalk characterization** — systematic methods for identifying and mitigating crosstalk.
- **Coherent-error-aware compilation** — reducing coherent errors via gate-sequence design.
- **Noise tomography** — full process characterization of hardware gates.

Cross-links to Other Courses

- **Open Quantum Systems** — Lindblad master equations.
- **Quantum Measurement Theory** — measurement errors and corrections.
- **Hardware Physics (superconducting/ion/atom)** — the underlying physics of specific platforms.
- **Module 9.4 (Noise mitigation)** — the response to the errors characterized here.

Noise Mitigation Techniques

Position in Knowledge Graph

System: Variational Quantum Algorithms **Structural Domain:** Hardware Noise Models **Node:** 9.4

Noise mitigation is the collection of **post-processing techniques** that reduce the effect of noise on the *expectation value* you extract from a noisy quantum circuit — without doing full fault-tolerant quantum error correction. The distinction is critical:

- **Error correction** (QEC) fixes the underlying *quantum state* using redundant encoding and active syndrome measurement. It is the long-term answer. It requires thousands of physical qubits per logical qubit and is not yet practical.
- **Error mitigation** does not fix the state. It uses classical post-processing to produce a better *estimate* of the noiseless expectation value from noisy runs. It is the NISQ-era answer.

Mitigation techniques pay for their accuracy with **sample complexity** — more circuit runs, not more qubits. This matches the VQA resource profile well, where qubits are scarce but classical compute is cheap.

This node surveys the major mitigation techniques: zero-noise extrapolation (ZNE), probabilistic error cancellation (PEC), readout error correction, Clifford data regression, symmetry verification, dynamical decoupling, and virtual distillation. Node 9.5 goes deep on ZNE specifically. Node 9.6 covers noise-adaptive ansatz design, which is a structurally different mitigation strategy.

What Mitigation Can And Cannot Do

Can:

1. **Estimate the noiseless expectation value** $\langle O \rangle_{\text{ideal}}$ with reduced bias, at the cost of increased variance.
2. **Remove systematic error** from specific known noise sources (e.g., readout errors, coherent calibration errors).
3. **Exploit symmetries** to reject errors that break a conserved quantity.
4. **Extend the usable depth** of a circuit by a factor of 2-5x under typical NISQ conditions.

Cannot:

1. **Recover the full quantum state** — you only get expectation values, not wavefunctions.
2. **Work past a fundamental sampling floor** — the sample complexity grows exponentially in the effective noise, so eventually you can't afford the shots.
3. **Fix errors that are indistinguishable from the signal** — e.g., if noise has the same symmetry structure as the observable.
4. **Replace error correction** for deep circuits — the overhead makes deep circuits unreachable.

Mitigation is the **right tool for the NISQ regime**: shallow-to-moderate circuits, moderate noise, tolerance for increased shot count. Outside that regime (either very shallow or very deep), different tools apply.

The Sample-Complexity Cost

A universal fact about mitigation: the reduction in bias comes with an increase in variance. Formally, if a mitigation technique multiplies the estimator by a factor $\gamma > 1$ (to undo noise shrinkage), the estimator variance grows by γ^2, and the required shot count to reach fixed precision grows the same way.

For modest noise ($p \sim 10^{-3}$ per gate, depth ~ 100), typical γ values are 2-10, meaning 4-100x more shots. Manageable.

For large noise ($p \sim 10^{-2}$, depth ~ 500), γ can blow up to 10^3-10^4, meaning 10^6-10^8x more shots. Unreachable.

The practical sweet spot: NISQ circuits with gate fidelity 99.5%-99.9% and depth 50-200 gates. Mitigation works here; it fails outside.

This sample-complexity barrier is why mitigation is not a long-term answer. It's a bridge to fault-tolerance.

Readout Error Correction

The simplest and most impactful mitigation. Measurement errors on current hardware are typically 1%-5% per qubit — much larger than gate errors. Luckily, they are also the easiest to characterize and correct.

Procedure:

1. **Characterize** the confusion matrix M, where M_{ij} is the probability of observing outcome i given the true outcome was j. This is done by preparing each computational basis state and measuring, typically with a few thousand shots.
2. **Invert** the confusion matrix: M^{-1} maps observed probabilities back to ideal ones.
3. **Apply** M^{-1} to the observed probability vector to recover the corrected distribution.

For n qubits, the full confusion matrix is $2^n \times 2^n$ — prohibitive beyond ~ 10 qubits. Practical variants:

- **Tensor product assumption:** treat each qubit's readout error independently. The confusion matrix factors as $M = M_1 \otimes M_2 \otimes \cdots \otimes M_n$. Characterization is $O(n)$ and inversion is $O(n)$.
- **Iterative Bayesian unfolding:** avoid explicit matrix inversion; iteratively refine the distribution using Bayes' rule. Handles correlated readout errors.
- **M3 (Matrix-free Measurement Mitigation):** solves a linear system to mitigate without forming the matrix. Scales to hundreds of qubits (Nation et al. 2021).

Effectiveness: removes most readout error bias at modest cost. Should be applied to *all* VQA experiments — it's essentially free relative to the underlying circuits.

Zero-Noise Extrapolation (ZNE) — Brief

Core idea: run the circuit at several artificially amplified noise levels, fit the expectation value as a function of noise strength, and extrapolate to zero noise.

Procedure:

1. Define a noise-amplification method (unitary folding, pulse stretching, or identity insertion) that scales the effective noise by factor $\lambda \in \{1, \lambda_2, \lambda_3, \ldots\}$.
2. Run the circuit at each amplification level and estimate $\langle O \rangle(\lambda)$.
3. Fit a curve (linear, exponential, Richardson) to $\langle O \rangle$ vs λ.
4. Extrapolate to $\lambda = 0$ to get the estimated noiseless value.

Pros: requires no detailed noise model, easy to implement, works with any observable.

Cons: extrapolation is extrapolation — the fit can be wrong, especially with large noise or non-monotonic curves. Typical accuracy improvement is 3-10x for NISQ regimes.

Node 9.5 goes into full detail: folding methods, fit choices, error analysis, practical recipes.

Probabilistic Error Cancellation (PEC)

A more ambitious technique: **invert the noise channel directly** via a quasi-probability decomposition.

Idea. Suppose the noise channel $\mathcal{N}$ is known and invertible. Then $\mathcal{N}^{-1}$ is a (quasi-)map: it can be written as a linear combination of physically implementable channels, possibly with negative coefficients:

$$\mathcal{N}^{-1} = \sum_k c_k \mathcal{E}_k, \quad \sum_k c_k = 1, \quad c_k \in \mathbb{R}.$$

Procedure. At each circuit run, sample a k with probability $|c_k|/\gamma$ (where $\gamma = \sum_k |c_k| \geq 1$ is the one-norm), apply the $\mathcal{E}_k$ operation, and multiply the measurement outcome by $\text{sign}(c_k) \cdot \gamma$.

Averaging many such runs produces an unbiased estimator of the noiseless expectation value.

The cost. The variance blowup factor is γ^2, and γ grows exponentially in the circuit depth for most realistic noise ($\gamma \sim e^{c \cdot L \cdot p}$ for noise rate p over depth L). So PEC works for shallow circuits but becomes prohibitively expensive for deep ones.

Pros: unbiased (not an extrapolation), gives mathematically rigorous guarantees.

Cons: requires precise knowledge of the noise channel (often unavailable), has exponential sample overhead, and the "quasi-probability" implementation is operationally awkward.

Status in 2026: PEC works on gate counts of 50-200 for superconducting hardware. Above that, sample complexity kills it. It is typically combined with ZNE (PEC for the short-range errors, ZNE for what remains).

Reference: Temme, Bravyi, Gambetta (2017) introduced PEC. Subsequent work by Takagi et al. (2022) gave sample complexity lower bounds showing the exponential is fundamental.

Clifford Data Regression (CDR)

A clever mitigation strategy using the fact that Clifford circuits are **classically simulable**.

Procedure.

1. Take your VQA circuit $U(\boldsymbol{\theta})$.
2. Construct a set of "nearby" Clifford circuits by replacing non-Clifford gates with their closest Clifford approximants.
3. Classically simulate each Clifford circuit to get the ideal expectation values $\{C_i^{\text{ideal}}\}$.
4. Run each Clifford circuit on the noisy hardware to get $\{C_i^{\text{noisy}}\}$.
5. Fit a regression model $f : C_i^{\text{noisy}} \mapsto C_i^{\text{ideal}}$ — typically linear.
6. Apply f to the noisy expectation value from the original (non-Clifford) circuit.

Intuition. The noise process is (approximately) consistent across Clifford and non-Clifford circuits — if you know how noise distorts the Cliffords, you can invert it for the non-Cliffords.

Pros: doesn't require a noise model, handles *systematic* bias in a principled way, often works when simple ZNE fails.

Cons: needs classical simulation of Clifford circuits (fine for moderate qubit counts; up to ~ 100 qubits for $\mathcal{O}(100)$-depth Cliffords), the "nearby Clifford" construction is heuristic, the regression can overfit.

Reference: Czarnik, Arrasmith, Coles, Cincio (2020) introduced CDR. Later extended to include polynomial regression and neural network models.

Practical note: CDR is the technique that most often outperforms ZNE on realistic VQA problems in the 50-qubit range. It should be considered a first-line mitigation alongside ZNE.

Symmetry Verification

Many physical Hamiltonians have **conserved quantities** — particle number, spin parity, total charge. If the ideal quantum circuit preserves these symmetries, then measurement outcomes that violate them are *known to be wrong* and can be discarded.

Procedure.

1. Identify a conserved quantity Q (typically a Pauli sum that commutes with the cost Hamiltonian).
2. Measure Q (either as part of the cost estimation or via a separate ancilla-based check).
3. Discard any shot where Q does not match the expected value.
4. Use only the remaining shots to estimate the cost.

Example (chemistry VQE). For a molecule with N electrons, the particle number $\hat{N} = \sum_i \hat{n}_i$ is conserved. Discard any Fock-space outcome with $\hat{N} \neq N$. This removes errors that change particle number (which amplitude damping does).

Pros: cheap (no extra circuits), principled (error rejection is unbiased), removes a physically meaningful class of errors.

Cons: only works if the symmetry is actually preserved by the circuit (not all ansätze preserve particle number), reduces the *effective* shot count (shots violating the symmetry are discarded), doesn't help with errors that preserve the symmetry.

Reference: McArdle, Yuan, Benjamin (2019) developed symmetry verification for VQE. It's now standard in chemistry VQA pipelines.

Dynamical Decoupling (DD)

A mitigation technique at the **pulse level** rather than the circuit level. It reduces coherent and low-frequency noise by rapid unitary "refocusing" pulses.

Intuition. If you're sitting idle on a qubit for time τ and there's a slowly varying environment field, the qubit acquires a phase error. But if you apply an X pulse halfway through, the phase error accumulated in the first half gets reversed in the second half — the net error is small.

Standard DD sequences.

- **Hahn echo:** a single π pulse in the middle. Cancels static inhomogeneous dephasing.
- **CPMG:** a train of π pulses at regular intervals. Cancels frequency noise up to a higher frequency.
- **XY-4, XY-8:** pulse sequences that protect against more general noise (not just dephasing).
- **Uhrig DD:** non-uniform pulse spacing optimized for specific noise spectra.

Application to VQAs. In a VQA, during the "dead time" when qubits are idle (waiting for other gates to finish, or during measurement of other qubits), DD pulses can be inserted to refocus errors. This is especially useful for superconducting qubits with long dead times.

Pros: reduces coherent and low-frequency noise at pulse level, no computational overhead, transparent to the circuit (the DD pulses are inserted into the idle periods).

Cons: doesn't help with high-frequency (Markovian) noise — and most gate errors *are* Markovian. Primarily helps with idle errors.

Reference: Viola, Lloyd (1998) introduced DD theoretically. Modern usage in VQAs: Pokharel et al. (2018) showed 3-5x fidelity improvement on IBM hardware via DD insertion.

Virtual Distillation

A more recent (2020+) technique that uses **multiple copies** of the noisy state to purify it.

Idea. If the noisy state is $\rho = (1-\epsilon)|\psi\rangle\langle\psi| + \epsilon\sigma$ with $\epsilon \ll 1$, then $\rho^2/\mathrm{Tr}(\rho^2)$ is closer to $|\psi\rangle\langle\psi|$ than ρ is. Higher powers ρ^M are closer still.

Procedure.

1. Prepare M copies of the noisy state.
2. Use a circuit that implements the "derangement" operation on the copies.
3. Measure an observable O along with the derangement.
4. The combined measurement gives an unbiased estimator of $\mathrm{Tr}(\rho^M O)/\mathrm{Tr}(\rho^M)$ — which is close to $\langle\psi|O|\psi\rangle$.

Pros: no noise model required, exponential suppression in M for the coherent error part.

Cons: requires M copies of the state (so $M\times$ qubit count), the derangement circuit introduces its own noise, the method is biased in a specific way that needs correction.

Reference: Huggins et al. (2021), Koczor (2021) independently introduced virtual distillation. Still somewhat research-stage; not yet standard in VQA pipelines but is the main 2024-2026 research direction for error mitigation.

Mitigation In Practice: A Decision Tree

For VQA practitioners, the practical question is: *which* mitigation technique should I apply?

1. **Always apply readout error correction.** Cost is negligible, benefit is significant. Non-negotiable.
2. **If your Hamiltonian has a conserved symmetry, apply symmetry verification.** Also cheap and principled.
3. **If your hardware has long idle times, apply dynamical decoupling.** Costs nothing in shots; reduces idle errors.
4. **For the "main" mitigation**, choose between ZNE and CDR based on depth:
   - Shallow ($L < 100$): ZNE with linear or exponential fit. Simple and effective.
   - Moderate depth ($100 < L < 300$): CDR often outperforms ZNE. Use if you can classically simulate the Cliffords.
   - Deep ($L > 300$): Neither works well. Consider reducing depth or switching to a shallower ansatz.
5. **For research-grade VQAs**, consider PEC (if noise model is known) or virtual distillation (if qubit count allows). Higher effort, can give best-in-class results.

Combining techniques. Readout correction, symmetry verification, and DD can be stacked with ZNE or CDR. Be careful about combining PEC with anything else — the quasi-probability math gets messy.

Mitigation And The Thesis

The thesis view of mitigation:

1. **Mitigation is a form of classical side-computation applied to the noisy optimizer.** It is not part of the quantum dynamics — it is outer-loop post-processing.
2. **HOPSO's structure makes mitigation cleaner.** Because HOPSO separates the dynamical update from the objective generator, mitigation can be applied to the objective generator (which produces C_{noisy}) without touching the optimizer dynamics. This is architecturally clean.
3. **Mitigation reduces bias but not the fundamental signal-to-noise tradeoff.** The usable depth is set by the fidelity budget plus the shot budget, and mitigation just shuffles the budget between the two. No free lunch — the question is whether your VQA can afford the tradeoff.
4. **The future of mitigation is integration with optimization.** Mitigation techniques applied in *series* (observe, correct, optimize) are one architecture; applied in *parallel* (the optimizer is aware of the mitigation cost and adjusts accordingly) is another. Module 10 takes up the latter.

Structural Role

This node is the **survey of the mitigation toolkit** for Module 9. It introduces each major mitigation technique, describes what it does and its cost, and gives a practical decision tree for when to use each.

It is the parent node for 9.5 (ZNE in depth) and 9.6 (noise-adaptive ansatz design). Together, these three nodes cover the main lines of attack against NISQ noise for VQAs.

Relations to Other Concepts

- **Temme, Bravyi, Gambetta (2017)** — PEC original.
- **Li, Benjamin (2017)** — ZNE original.
- **Czarnik et al. (2020)** — CDR original.
- **McArdle et al. (2019)** — symmetry verification for VQE.
- **Huggins et al. (2021), Koczor (2021)** — virtual distillation.
- **Viola, Lloyd (1998)** — DD theory.
- **Nation et al. (2021)** — M3 readout mitigation.
- **Cai et al. (2023)** — review of quantum error mitigation.

Worked Example — Mitigation Stack For A 10-Qubit VQE

Consider a 10-qubit VQE for a chemistry problem, running on superconducting hardware with: - Two-qubit gate fidelity 99.5%. - Readout fidelity 97%. - Circuit depth $L = 80$ (including 40 two-qubit gates). - Particle number conservation (electrons fixed at 5). - 10^5 shots budget per cost evaluation.

Without mitigation. - Circuit fidelity: $(0.995)^{40} \approx 0.818$. Then readout on 10 qubits: $(0.97)^{10} \approx 0.737$. Combined: 0.60. - Cost estimate is biased — off by roughly 40% relative to true value.

With readout mitigation. - Readout bias corrected by tensor-product M matrix inversion. Residual readout error: $\sim 1\%$. Combined fidelity: $0.818 \cdot 0.99 \approx 0.81$. - Cost estimate bias: $\sim 19\%$.

Plus symmetry verification (particle number). - Discards shots that violate $\hat{N} = 5$. Roughly 15% of shots are discarded. - Effective shots: $0.85 \cdot 10^5 = 8.5 \times 10^4$. - Bias from remaining errors (those preserving particle number): $\sim 10\%$.

Plus ZNE (linear fit, $\lambda = 1, 1.5, 2$). - Uses 3x the shot budget ($3 \times 8.5 \times 10^4 = 2.55 \times 10^5$) to get the extrapolation points. - After linear fit, residual bias: $\sim 2\%$. - Variance roughly doubled compared to single $\lambda = 1$ run.

Final result: cost estimate with $\sim 2\%$ bias, requires $\sim 3\times$ shot count relative to raw. Acceptable for VQA optimization.

Observation. Each mitigation layer halves (or better) the residual bias. The stack is additive. Getting below 2% bias requires either CDR or PEC, both of which are significant investments.

Visualization

Pending. A bar chart showing bias and variance for each mitigation layer applied sequentially: (no mitigation) → +readout → +symmetry → +ZNE. Bias decreases; variance increases. Caption: "The bias-variance tradeoff of mitigation stacks."

Exercises

1. For a depolarizing channel at rate $p = 0.01$ and a single-qubit observable X, compute the PEC quasi-probability decomposition and the resulting γ.
2. Show that symmetry verification is *unbiased* — the estimator using only symmetric shots has the same expectation as the ideal estimator, provided the symmetry is exactly preserved by the ideal circuit.
3. (VQA) For a 20-qubit VQA with depth 100, gate fidelity 99.5%, readout fidelity 97%, and shot budget 10^6, design a mitigation stack and estimate the final cost bias and effective shot count.

Research Extensions

- **Adaptive mitigation** — selecting the mitigation technique based on the current circuit properties.
- **Learning-based mitigation** — neural networks trained to correct noisy expectation values given side information about the circuit.
- **Hybrid PEC/ZNE** — combining the two techniques to trade off bias and variance optimally.
- **Mitigation-aware optimizer design** — optimizers that account for the mitigation cost in their loss function.
- **Scalable virtual distillation** — reducing the qubit overhead of copy-based techniques.

Cross-links to Other Courses

- **Quantum Error Correction** — the long-term answer that mitigation approximates.
- **Statistical Estimation Theory** — bias-variance tradeoff as a fundamental constraint.
- **Module 9.5 (Zero-Noise Extrapolation)** — the detailed treatment of the most widely used technique.
- **Module 9.6 (Noise-Adaptive Ansatz Design)** — a structurally different mitigation strategy.
- **Module 10 (Adaptive Control Systems)** — mitigation-aware optimizer design.

Zero-Noise Extrapolation

Position in Knowledge Graph

System: Variational Quantum Algorithms **Structural Domain:** Hardware Noise Models **Node:** 9.5

Zero-noise extrapolation (ZNE) is the single most widely deployed error mitigation technique for VQAs. It is conceptually simple, needs no detailed noise model, works with any observable, and can be applied to almost any circuit with modest overhead. The basic idea:

1. Run the circuit at several **artificially amplified** noise levels.
2. Fit the expectation value as a function of noise strength.
3. Extrapolate back to zero noise.

The key insight is that you can deliberately make your circuits noisier in controlled amounts, and from the resulting data points you can infer where the line (or curve) would cross at zero noise. Under the right conditions, this gives a substantially better estimate than just running the circuit at nominal noise.

ZNE was introduced by Li-Benjamin (2017) and Temme-Bravyi-Gambetta (2017) more or less simultaneously. It has since become the default mitigation technique in Qiskit, Mitiq (a dedicated mitigation library), PennyLane, and most VQA benchmarking studies.

This node covers the details: noise amplification methods, fit choices, error analysis, practical implementation, and its integration with VQA optimization.

The Basic Idea, Formally

Let $\langle O \rangle(\lambda)$ be the expectation value of observable O when the circuit's noise is amplified by factor $\lambda \geq 1$. The nominal circuit corresponds to $\lambda = 1$. By construction, $\lambda = 0$ corresponds to *no noise*.

Assumption. The function $\lambda \mapsto \langle O \rangle(\lambda)$ is smooth and can be extrapolated from points $\lambda_1 = 1, \lambda_2, \lambda_3, \ldots$ back to $\lambda = 0$.

Procedure.

1. Choose a set of amplification levels $\{\lambda_i\}$ (e.g., $\{1, 1.5, 2, 2.5\}$).
2. For each λ_i, amplify the circuit's noise by that factor and run N_{shots} shots to get $\hat{O}(\lambda_i)$.
3. Fit a model $f(\lambda)$ to the data points $\{(\lambda_i, \hat{O}(\lambda_i))\}$.
4. Report the extrapolated value $\hat{O}_{\text{ZNE}} = f(0)$ as the mitigated estimate.

The critical choices are: **how to amplify the noise** (without changing the logic of the circuit) and **what functional form to fit**.

Noise Amplification Methods

Three main strategies exist for increasing the effective noise without changing the ideal operation of the circuit.

1. Unitary folding

Replace each gate U with $U \cdot U^\dagger \cdot U$, which is logically equivalent to U but has three times the gate count (and hence three times the noise).

More generally, folding by factor $\lambda = 2k + 1$ replaces U with $U \cdot (U^\dagger U)^k$. **Global folding** applies this to the entire circuit U_{circ}; **local folding** applies it to individual gates.

Pros: noise scaling is discrete ($\lambda = 1, 3, 5, \ldots$) but can be made continuous by folding a random subset of gates. Works in circuit-level simulators. Widely supported.

Cons: the assumption that noise scales linearly with gate count is only approximate (different gates have different noise, and noise doesn't always compose linearly).

Reference. Giurgica-Tiron et al. (2020) introduced the modern unitary folding framework.

2. Pulse stretching

At the pulse level, lengthen each gate's control pulses by a factor $\lambda \geq 1$. The gate does the same ideal operation but takes longer, during which more decoherence accumulates.

Pros: the noise amplification is *physical* rather than synthetic — you're actually running the hardware in a noisier regime. Fewer assumptions.

Cons: requires pulse-level hardware access (not available via simple circuit APIs), the stretching may not preserve gate fidelity exactly (coherent errors can get worse rather than just scaling), and λ is bounded by the coherence time (you can't stretch indefinitely).

Reference. Kandala et al. (2019) used pulse stretching for ZNE in the IBM hardware stack.

3. Identity insertion

Insert identity operations (physically realized as $UU^\dagger$ for some U) at various points in the circuit, adding noise without changing the ideal logic.

Similar to unitary folding in effect but more flexible about *where* the extra noise goes.

Pros: can target specific gates or qubits for amplification.

Cons: the specific choice of U matters (different U's have different noise profiles), so the amplification is not cleanly parameterized.

In practice. Unitary folding is the default for circuit-level ZNE; pulse stretching is the default when pulse-level access is available; identity insertion is used for specialized studies.

Extrapolation Fits

With data $\{(\lambda_i, \hat{O}_i)\}$, several fit choices exist.

Linear fit

$$f(\lambda) = a\lambda + b.$$

Extrapolated value: $f(0) = b$.

Simplest, robust, appropriate when the noise is small and the dependence is approximately linear.

Pros: stable, easy to implement, small variance.

Cons: can have significant bias when the true dependence is nonlinear.

Polynomial fit

$$f(\lambda) = \sum_{k=0}^{d} c_k \lambda^k.$$

Degree d fit; extrapolation is $f(0) = c_0$. Richardson extrapolation is a special case.

Pros: more flexible.

Cons: higher-degree polynomials overfit and amplify statistical noise in the extrapolation. Numerical instability for large d.

Typical choice: $d = 2$ or $d = 3$, no higher.

Exponential fit

$$f(\lambda) = Ae^{-\Gamma\lambda} + B.$$

Extrapolated value: $f(0) = A + B$.

The exponential form is motivated by the actual dependence of Pauli expectation values on depolarizing noise: $\langle P\rangle(\lambda) \approx \langle P\rangle_{\text{ideal}} \cdot (1 - 4p/3)^{L\lambda}$, which is exponential in λ.

Pros: matches the true depolarizing dependence.

Cons: the fit is nonlinear and can fail to converge with sparse data. Requires at least 3 data points and often 4+.

Typical choice: when $\geq$4 data points are available and noise is known to be approximately depolarizing.

Richardson extrapolation

A specific polynomial extrapolation that eliminates leading-order error terms one at a time. For example, with $\lambda = 1, 2$:

$$f(0) \approx 2\hat{O}(1) - \hat{O}(2).$$

With $\lambda = 1, 2, 3$: a weighted sum of the three values that cancels both linear and quadratic terms.

Pros: closed-form, known error bounds, well-studied in numerical analysis.

Cons: assumes specific polynomial structure, amplifies variance at higher orders.

Error Analysis

Two sources of error affect ZNE:

1. Statistical error. Each $\hat{O}(\lambda_i)$ has shot noise of order $1/\sqrt{N_{\text{shots}}}$. The extrapolation amplifies this noise. For linear extrapolation with two points $\lambda = 1, 2$, the extrapolated variance is:

$$\text{Var}[\hat{O}_{\text{ZNE}}] = \text{Var}[2\hat{O}(1) - \hat{O}(2)] = 4\sigma^2 + \sigma^2 = 5\sigma^2,$$

so the variance is 5x the single-run variance. For higher-order fits, the variance amplification is larger (up to $\sim 30\sigma^2$ for Richardson with $\lambda = 1, 2, 3$).

2. Extrapolation bias. If the true dependence is not the model you fit, extrapolation introduces a bias. For a linear fit applied to data that is actually exponential, the bias is $\sim$ (curvature) $\cdot\, \lambda_{\text{min}}^2$.

Tradeoff. Higher-order fits reduce bias but increase variance. The optimal choice depends on the circuit, noise level, and shot budget.

Rule of thumb. For moderate noise and shallow circuits, use linear fits. For moderate-to-deep circuits, use exponential. For high-precision work, use several fit types and compare — if they agree, trust them; if they disagree, the extrapolation is unreliable.

Pitfalls Of ZNE

Several situations where ZNE fails or misleads:

1. Non-monotonic dependence. If the noisy expectation value is not monotonic in λ (e.g., because of coherent errors that build up non-linearly), the extrapolation has no fixed point to converge to. Fit functions can give nonsense values.

2. Noise that's not a "factor". ZNE assumes noise can be amplified *uniformly*. For heterogeneous noise (different gates have very different error rates), this assumption breaks down. The effective λ is ambiguous.

3. Saturation at small λ. If the nominal noise is already close to saturation (noise level so high that expectation values are near zero), you cannot extrapolate linearly — the curve is in its flat region and the fit is ill-conditioned.

4. Cost landscape distortion. Applying ZNE to every cost evaluation in a VQA optimization slows convergence (because shots are 3-10x more expensive) *and* introduces correlated errors across iterations. The optimizer sees a slightly noisier and slightly warped landscape.

5. The expensive folded circuits. Folded circuits have more gates, so they take longer to run *and* they're more error-prone. At high λ, the amplified circuit may be so noisy that the data point is useless.

Mitigating the pitfalls. Use multiple λ values (at least 3), try multiple fit functions, check consistency, and validate on benchmark problems where the ideal answer is known (Clifford circuits work well for this).

ZNE In VQA Optimization

A subtle question: when should ZNE be applied *during* VQA optimization?

Option A — Every iteration. Apply ZNE at every cost evaluation. Guarantees the optimizer sees the mitigated cost, but the shot overhead is 3-10x per iteration.

Option B — Only at the final evaluation. Optimize with raw (unmitigated) cost, then apply ZNE once at the end to estimate the true cost at the converged point. Saves shots during training but relies on the raw cost having the same minimum as the mitigated cost.

Option C — Periodic mitigation. Apply ZNE every k iterations to check convergence, use raw cost otherwise. Compromise between A and B.

The tradeoff. Option B is correct *if* noise is depolarizing (since depolarizing preserves the minimum location — see node 9.2). For more structured noise, the raw minimum may differ from the mitigated minimum, and Option A is needed.

Current practice. Most 2026 VQA experiments use Option B or C for compute efficiency, with Option A reserved for small-scale precision studies.

ZNE As A Basic Building Block

ZNE is the "default" mitigation in most VQA libraries and is the first technique to reach for. Its main virtues:

1. **Model-free.** Doesn't require knowledge of the noise channel.
2. **Universal.** Works for any observable, any circuit, any noise type.
3. **Cheap.** Overhead is 3-10x in shots, which is manageable.
4. **Stackable.** Can be combined with readout mitigation, symmetry verification, DD without conflict.
5. **Transparent.** The extrapolation curve gives a visual diagnostic — if the curve looks wrong, you know ZNE is unreliable.

The main virtues of *alternatives* (CDR, PEC, virtual distillation) are corrections to specific ZNE failure modes. ZNE is the baseline; other techniques are improvements for specific conditions.

Structural Role

This node is the **detailed treatment** of the most widely used mitigation technique, expanding on the overview in node 9.4. It covers noise amplification methods, fit choices, error analysis, pitfalls, and VQA integration.

It is the reference node for ZNE in the course, and the practical companion to node 9.4's survey view.

Relations to Other Concepts

- **Li, Benjamin (2017)** — ZNE origin.
- **Temme, Bravyi, Gambetta (2017)** — ZNE + PEC origin.
- **Kandala et al. (2019)** — hardware ZNE demonstration with pulse stretching.
- **Giurgica-Tiron et al. (2020)** — unitary folding framework and Mitiq library.
- **LaRose et al. (2022)** — Mitiq: software package for error mitigation.
- **Richardson extrapolation (numerical analysis classic)** — the parent technique.
- **Module 9.2 (Depolarizing noise)** — why the minimum is preserved under depolarizing.
- **Module 9.4 (Mitigation techniques)** — the broader context.

Worked Example — ZNE On A 10-Qubit QAOA

Setup. A 10-qubit QAOA at depth $p = 3$, two-qubit gate fidelity 99.5%, shot budget 10^5 per cost evaluation. The cost is $\langle H_C \rangle = -0.53$ (noiseless, known from small simulation) and -0.32 (noisy, directly measured).

Step 1 — Choose amplification levels. $\lambda \in \{1, 2, 3\}$ via unitary folding.

Step 2 — Run circuits. - $\lambda = 1$: fold factor 1 (nominal), $\hat{O}_1 = -0.32$. - $\lambda = 2$: fold each gate to $UU^\dagger U$, 3 gate count, $\hat{O}_2 = -0.19$. - $\lambda = 3$: fold each gate twice, 5 gate count, $\hat{O}_3 = -0.11$.

Each shot budget 10^5, so total 3×10^5 shots.

Step 3 — Fit.

Linear fit: $f(\lambda) = -0.41 + 0.105\lambda$. $f(0) = -0.41$.

Exponential fit: $f(\lambda) = -0.55 \cdot e^{-0.42\lambda} + 0.01$. $f(0) = -0.54$.

Richardson (with $\lambda = 1, 2$): $f(0) = 2 \cdot (-0.32) - (-0.19) = -0.45$.

Step 4 — Compare to the truth. The noiseless value is -0.53. - Linear: -0.41, error 0.12. - Exponential: -0.54, error 0.01. **Best.** - Richardson: -0.45, error 0.08. - Unmitigated: -0.32, error 0.21.

Observation. Exponential is the best fit for this depolarizing-dominated regime. Linear is significantly biased because the underlying decay is not linear. Richardson is between the two.

Shot cost. 3x (three amplification levels $\times$ 10^5 shots each).

Variance amplification. Roughly 5x for linear, more for Richardson, less for exponential (because the fit is "softer").

Conclusion. Exponential ZNE gives a 20x bias reduction at 3x shot cost. Good tradeoff.

Visualization

Pending. A plot of $\langle H_C \rangle$ vs λ showing three data points (at $\lambda = 1, 2, 3$), three fit curves (linear, exponential, Richardson), and the true value at $\lambda = 0$. The exponential curve should hit the true value closely; the linear should miss. Caption: "ZNE in action: fit choice matters."

Exercises

1. For a single-qubit depolarizing channel at rate p, show that the expectation value of X under λ-folded noise is $(1 - 4p/3)^{2\lambda-1} \cdot \langle X \rangle_{\text{ideal}}$. Use this to derive the exact exponential fit formula.
2. Prove that linear ZNE is unbiased if the true $\langle O \rangle(\lambda)$ is linear in λ, and biased by $\mathcal{O}(\lambda^2 \cdot |O''|)$ if it is quadratic.
3. (VQA) For a 12-qubit VQE with depth 60 and two-qubit gate fidelity 99.3%, compare linear, exponential, and Richardson ZNE using simulated data. Which gives the smallest total error (bias + statistical)?

Research Extensions

- **Adaptive λ-selection** — automatically choosing the amplification levels based on the observed curvature.
- **Learning-based extrapolation** — using neural networks to extrapolate rather than parametric fits.
- **Joint cost/gradient ZNE** — mitigating both the cost and its gradient simultaneously with a shared shot budget.
- **Exponential ZNE with variance reduction** — variance reduction techniques specific to the exponential fit.
- **ZNE in the deep-circuit regime** — adapting ZNE for circuits that are past the saturation depth.

Cross-links to Other Courses

- **Numerical Analysis** — Richardson extrapolation, polynomial interpolation.
- **Statistical Estimation** — bias-variance tradeoff, extrapolation risk.
- **Module 9.4 (Mitigation techniques)** — overview context.
- **Module 9.8 (Taxonomy of Noise)** — where ZNE fits in the mitigation landscape.
- **Module 10.5 (Closing the Loop)** — integrating mitigation with the optimizer.

Noise-Adaptive Ansatz Design

Position in Knowledge Graph

System: Variational Quantum Algorithms **Structural Domain:** Hardware Noise Models **Node:** 9.6

Noise mitigation (node 9.4-9.5) treats noise as a fact to correct *after* the circuit runs. **Noise-adaptive ansatz design** takes the opposite view: modify the ansatz *before* the circuit runs, so that the noise profile of the specific hardware has the least harmful effect.

The motivating observation: noise on real hardware is *heterogeneous*. Some qubits are better than others. Some two-qubit gates (on specific qubit pairs) have much higher fidelity than others. The hardware connectivity graph has topology (not every pair of qubits can interact directly). And the noise spectrum changes slowly over time with drift and recalibration.

A noise-blind ansatz ignores all this and treats the circuit as a uniform object. A noise-adaptive ansatz uses the hardware information to:

1. **Place the best-fidelity qubits where the circuit needs them most.**
2. **Route two-qubit gates through the lowest-noise pairs.**
3. **Reduce depth in regions where coherence is weakest.**
4. **Choose the ansatz shape itself** to match the hardware's native gate set.

The result: better circuit fidelity for the same logical operation, at zero quantum runtime cost (all the optimization is classical). This is essentially a form of *hardware-aware compilation* specialized for VQAs.

This node covers the main techniques: qubit selection, routing, Hamiltonian-to-hardware mapping, hardware-efficient ansätze, and the ADAPT-VQE family of noise-aware ansatz growth.

Qubit Selection

The simplest form of noise-adaptive design. Modern superconducting hardware has 50-1000 qubits; most VQA problems use far fewer. The question: which subset of qubits do you pick?

Naive approach: use qubits 0 through $n-1$ on the device. Simple but ignores fidelity differences.

Better approach: rank qubits by fidelity metrics and pick the best n.

Useful metrics for ranking:

- **T1, T2 coherence times** — higher is better.
- **Readout fidelity** — higher is better.
- **Single-qubit gate fidelity** — higher is better.
- **Two-qubit gate fidelity on connecting edges** — higher is better.
- **Calibration freshness** — more recently calibrated qubits are usually better.

The ranking is done per-VQA-run using the daily calibration data published by the hardware provider (IBM, Rigetti, IonQ, etc.). The best qubits change over time as the hardware drifts.

Gotcha: the "best" qubits individually may not have the best two-qubit couplings. You need to optimize over *subgraphs*, not individual qubits. This is a graph optimization problem and can be hard in general (subgraph isomorphism is NP-hard), but practical heuristics work well for small problem sizes.

Reference. Tannu-Qureshi (2019) showed 3-5x fidelity improvement on IBM hardware via hardware-aware qubit selection alone.

Gate Routing

A VQA circuit usually implies a *logical* connectivity pattern (which qubits interact in the cost Hamiltonian). The hardware has a *physical* connectivity graph (which qubits are directly coupled). These rarely match exactly.

The **routing problem** is: map logical qubits to physical qubits, inserting SWAP gates where needed, to realize the logical circuit on the physical hardware with minimum noise cost.

Naive routing minimizes the total SWAP count — fewer SWAPs means fewer noisy operations.

Noise-aware routing weights each potential SWAP by the error rate of the specific two-qubit gate used. A high-fidelity SWAP pathway is preferred even if it's longer.

Formalization. This is a weighted graph optimization problem. Given the logical connectivity graph G_L and the physical graph G_P with edge weights w_{ij} (error rates), find a mapping and SWAP sequence that minimizes the total weighted error.

Practical tools. Qiskit's `SabreSwap`, `SabreLayout`, and `NoiseAdaptiveLayout` passes do noise-aware routing. Similar tools exist in Cirq (Google) and tket (Quantinuum).

Impact. On hardware with significant fidelity heterogeneity, noise-aware routing gives 2-5x fidelity improvement compared to naive routing.

Hardware-Efficient Ansätze (HEA)

A deeper form of adaptation: choose the ansatz *shape* to match the hardware's native gate set and connectivity.

Standard HEA:

$$U(\boldsymbol{\theta}) = \prod_{l=1}^{L} \left[\left(\prod_i R_Y(\theta_i^l) R_Z(\phi_i^l) \right) \cdot \left(\prod_{(i,j)\in E} \mathrm{CNOT}_{ij} \right) \right],$$

where E is the hardware's native two-qubit connectivity. Key properties:

- **Uses only native gates** — no expensive decomposition of arbitrary U into single-qubit rotations and CNOTs.
- **Respects connectivity** — no SWAP insertion required, since two-qubit gates are only between connected qubits.
- **Linearly parameterized in layers** — depth is the main knob.

Tradeoff. HEA has the best hardware fidelity *but* the worst expressibility per parameter. Hardware-native gates often don't span the full unitary group cleanly, so you need more layers to reach the same expressibility as a problem-tailored ansatz.

Practical observation. For NISQ VQAs, the fidelity savings often outweigh the expressibility loss. HEA with $L = 5 - 20$ layers is competitive with much deeper "theoretically better" ansätze on noisy hardware.

Reference. Kandala et al. (2017) — the original hardware-efficient VQE on IBM hardware for small molecules.

Hamiltonian-to-Hardware Matching

A more sophisticated technique: match the *structure* of the Hamiltonian to the hardware topology.

Observation. A chemistry Hamiltonian has locality structure — some terms involve adjacent orbitals, others involve distant ones. Mapping this to the hardware:

1. **Place orbitals that interact strongly on well-connected physical qubits.**
2. **Use SWAP networks that match the locality pattern.**
3. **Schedule the simulation to minimize communication between distant qubits.**

For Hamiltonians derived from lattice models (Hubbard, Heisenberg, Ising), this is even cleaner: the lattice structure directly maps to a hardware connectivity pattern, and the best ansatz is one that mirrors that lattice.

For chemistry VQEs specifically, the **unitary coupled cluster (UCC)** ansatz has a natural locality pattern inherited from Slater-Condon rules. Recent work (Lee et al. 2021) shows that matching UCC excitation operators to hardware topology improves fidelity 2-10x for 10-20 qubit molecules.

For QAOA on graph problems, a good rule is to *assign graph vertices to physical qubits such that adjacent vertices are adjacent on the hardware*. This reduces the SWAP count dramatically on native problems.

ADAPT-VQE And Growing Ansätze

The **Adaptive Derivative-Assembled Pseudo-Trotter (ADAPT-VQE)** method grows the ansatz one gate at a time, selecting at each step the gate that gives the largest gradient component.

Procedure.

1. Start with an empty ansatz (just the reference state).
2. Compute the gradient of the cost with respect to adding each candidate gate from a pre-defined pool.
3. Add the gate with the largest gradient magnitude.
4. Optimize the parameters of the current ansatz.
5. Repeat until convergence.

Why this is noise-adaptive. The gradient estimate is based on the noisy circuit, so the *noise-favored* gates tend to be selected — gates that are high-fidelity on the specific hardware give larger and more reliable gradients, so they enter the ansatz first. The result is an ansatz that is implicitly hardware-aware.

Pros: extremely compact ansätze (much fewer parameters than HEA), naturally hardware-aware, grows only as much as needed for the specific problem.

Cons: the gradient computation at each growth step is expensive (you're computing the gradient over the full operator pool). For large pools, this becomes a significant overhead.

Reference. Grimsley, Economou, Barnes, Mayhall (2019) introduced ADAPT-VQE. Subsequent work extended it to fermionic-qubit mappings and noise-aware variants.

Gate Fidelity As A Cost Term

A direct approach to noise-adaptive design: add the gate-noise cost directly to the training objective.

Idea. Define a modified cost:

$$C_{\text{adapted}} = C_{\text{quantum}} + \alpha \cdot C_{\text{noise}},$$

where $C_{\text{noise}} = \sum_g p_g$ is the sum of noise rates over all gates in the circuit, and α is a regularization parameter.

Optimization now trades off the quantum cost against the noise cost: the optimizer is encouraged to find low-gate-count, low-error solutions.

Caveat. The gate count is discrete (number of gates in the circuit), so gradient descent doesn't directly apply. You need either: - A soft penalty (α times a smooth approximation of gate count). - Discrete search

(genetic algorithms, reinforcement learning, Bayesian optimization). - A structured ansatz family where gate count is controlled by a continuous parameter.

Practical use. Primarily a research direction — not yet standard. The thesis's HOPSO framework can naturally incorporate a noise-aware cost term in its objective generator, which is a potential future extension.

Dynamic Ansatz Reconfiguration

A futuristic version: the ansatz structure is **reconfigured during training** based on observed hardware fidelity changes.

Procedure.

1. Monitor the hardware fidelity in real time (via periodic benchmarking).
2. Detect when specific gates have degraded (e.g., due to drift or a recalibration).
3. Re-route or re-layer the ansatz to avoid the degraded gates.
4. Continue optimization with the new ansatz structure.

This is a form of **adaptive circuit compilation during training** and fits naturally into the control-theoretic view of Module 7. It's the logical extension of node 7.6's feedback control framework to the ansatz level.

Research status. Early experiments (Ziman et al. 2023, Hetzinger et al. 2024) show this can improve final fidelity by 2-5x on drifting hardware. Not yet standard; active research.

Noise-Adaptive Vs Mitigation: Structural Comparison

Dimension	Mitigation (9.4, 9.5)	Adaptation (9.6)
When applied	After runs	Before runs
Cost domain	Shot count	Classical compile time
What it fixes	Residual bias	Systematic fidelity
Depth limit	Extends by 3-10x	Extends by 2-5x
Combines with	Other mitigation	Mitigation techniques
Hardware info needed	Minimal	Full calibration data
Best for	Moderate noise	Heterogeneous noise

The two approaches are complementary. You should apply *both*: a noise-adaptive ansatz to minimize the underlying error, plus mitigation to further reduce residual bias. The combined improvement is roughly multiplicative.

The Thesis View

Noise-adaptive design fits the thesis framework in a clean way:

1. **It is a structural modification of the objective generator.** The objective function C is unchanged conceptually, but its hardware representation changes.
2. **HOPSO can incorporate hardware data as a parameter.** The optimizer can be given the current fidelity map as a static input, and it can be designed to prefer parameter settings that activate high-fidelity gates more than low-fidelity ones.

3. **Separation of dynamics and objective.** The *dynamics* of HOPSO (how particles move) is unchanged; only the *objective landscape* (what they're searching) is adapted. This is the right way to think about it architecturally.

4. **Adaptive regime.** The static noise-adaptive design of section 9.6 extends naturally to the dynamic one of Module 10 — the objective generator can be updated during training as hardware drifts, closing the loop between hardware monitoring and circuit structure.

Structural Role

This node is the **ansatz-level mitigation** complement to nodes 9.4-9.5 (post-processing mitigation). It covers qubit selection, gate routing, hardware-efficient ansätze, Hamiltonian-to-hardware matching, ADAPT-VQE, and dynamic reconfiguration.

It is the structural precondition for Module 10's adaptive VQA architectures, which take the noise-adaptive ideas and embed them in a feedback-control framework.

Relations to Other Concepts

- **Tannu, Qureshi (2019)** — hardware-aware qubit selection.
- **Kandala et al. (2017)** — original hardware-efficient ansatz.
- **Grimsley et al. (2019)** — ADAPT-VQE.
- **Li, Benjamin, Proctor (2020)** — noise-aware compilation with SabreSwap-style algorithms.
- **Module 9.4 (Mitigation)** — complementary post-processing approach.
- **Module 7.6 (Feedback control)** — dynamic reconfiguration as closed-loop control.
- **Module 10 (Adaptive VQA)** — the deployment endpoint.

Worked Example — 8-Qubit Chemistry VQE On A Heterogeneous Device

Setup. An 8-qubit H_2O chemistry VQE on a 27-qubit IBM device (call it `ibm_kyoto` for concreteness). Hardware calibration data: - Single-qubit gate fidelities: 99.95%-99.99%. - Two-qubit gate fidelities: 98.5%-99.6% (varies by edge). - Best 8-qubit connected subgraph: qubits $\{3, 5, 8, 11, 14, 16, 19, 22\}$ with average two-qubit fidelity 99.4%. - Worst 8-qubit connected subgraph: $\{0, 1, 2, 4, 6, 7, 9, 10\}$ with average two-qubit fidelity 98.8%.

Circuit. A UCCSD ansatz with 50 two-qubit gates. Run with 10^5 shots per cost evaluation.

Comparison:

Strategy	Qubit subset	Avg 2q fid	Circuit fid	Cost bias
Naive (qubits 0-7)	worst	0.988	0.55	45%
Noise-aware selection	best	0.994	0.74	26%
Best + noise-aware routing	best	0.994 + routing	0.78	22%
Best + routing + HEA switch	best	—	0.83	17%
Best + routing + HEA + ZNE	best	—	—	6%
Best + routing + HEA + ZNE + CDR	best	—	—	2%

Observation. Noise-adaptive design alone reduces bias by ~30% (45% → 17%). Adding mitigation on top reduces it further (17% → 2%). The stack multiplies: the total improvement is ~20x.

Compute cost. Qubit selection and routing: seconds. HEA conversion: minutes. Training: same as baseline. ZNE: 3x shots. CDR: 1.5x more compute for Clifford regression. Net: ~5x compute cost total for 20x bias reduction. Worth it.

Visualization

Pending. A diagram of a 27-qubit device with two 8-qubit subsets highlighted: the naive one (bottom-left, mostly-worst) and the noise-aware one (best-fidelity subset). Edges are colored by two-qubit gate fidelity. Caption: "Qubit selection: picking the right 8 out of 27 matters."

Exercises

1. Given a hardware graph $G_P = (V, E, w)$ with edge weights w_{ij} (error rates) and a desired subset size n, formulate the noise-aware qubit selection problem as a graph optimization problem. Is it tractable?
2. For a QAOA on a 10-vertex ring-graph MaxCut, design the noise-aware qubit-to-vertex mapping on a linear-connectivity hardware device. Compare SWAP count to naive mapping.
3. (VQA) Implement a noise-aware variant of the parameter-shift gradient that weights each gate's gradient by its fidelity. Does this improve convergence on a noisy 6-qubit VQA?

Research Extensions

- **Adaptive noise-aware compilation** — recompile during training as hardware drifts.
- **Noise-aware parameter initialization** — start near parameter settings that favor high-fidelity gates.
- **Multi-objective ansatz search** — jointly optimize expressivity and hardware fidelity.
- **Hardware-co-designed ansätze** — design the ansatz *and* the hardware gate set together for specific target problems.
- **Transfer learning for ansatz structure** — reuse ansatz designs across similar problems on similar hardware.

Cross-links to Other Courses

- **Compilers and code generation** — routing and layout as compilation problems.
- **Graph theory** — subgraph selection and minimum-weight routing.
- **Machine learning architecture search** — ADAPT-VQE as a form of NAS (neural architecture search) for quantum circuits.
- **Module 9.4 (Mitigation)** — the post-processing complement.
- **Module 10 (Adaptive Control Systems)** — the closed-loop ansatz adaptation.

Simulation Modes: Statevector, Density Matrix, Aer

Position in Knowledge Graph

System: Variational Quantum Algorithms **Structural Domain:** Hardware Noise Models **Node:** 9.7

Before you run a VQA on real hardware, you almost always simulate it classically. Simulation serves several purposes: developing and debugging the algorithm, evaluating expressivity and trainability, testing noise mitigation, and producing baselines against which to compare the hardware results. Getting simulation right is foundational to the whole VQA research enterprise.

But there isn't one kind of quantum simulation — there are several, each with different capabilities, costs, and use cases. The main modes:

1. **Statevector simulation** — exact, noiseless, stores the full 2^n-dimensional amplitude vector. Fastest for small n, exponentially expensive.
2. **Density matrix simulation** — exact, supports mixed states and general noise channels, stores a $2^n \times 2^n$ matrix. Doubles the memory cost compared to statevector.
3. **Stochastic / trajectory simulation (Qiskit Aer, quantum trajectory methods)** — Monte Carlo sampling of pure-state trajectories under noise. Memory like statevector but with a shot overhead for noise averaging.
4. **Tensor network simulation (MPS, PEPS)** — exact or approximate, uses low-entanglement structure. Scales better for certain classes of circuits.
5. **Clifford simulation (stabilizer formalism)** — exact and efficient for Clifford-only circuits. Limited to Clifford gates + Pauli observables.
6. **Approximate classical simulation** — methods that produce an approximation of the state with a budget.

This node covers the main modes relevant to VQAs: statevector (the fastest for small n), density matrix (the most expressive noise model), Aer-style stochastic trajectories (the practical middle ground), and tensor networks (the scaling solution for structured states). Understanding the tradeoffs is essential for designing VQA experiments that can actually be simulated within a reasonable compute budget.

Statevector Simulation

The simplest mode: store the amplitude vector $|\psi\rangle \in \mathbb{C}^{2^n}$ directly, and apply each gate as a matrix-vector multiplication.

Memory. $2^n \times 16$ bytes (complex doubles) $= 2^{n+4}$ bytes. For $n = 30$, that's 16 GB (feasible on a workstation). For $n = 35$, 512 GB (big machine). For $n = 40$, 16 TB (supercomputer). The effective limit is around $n = 30 - 35$ on commodity hardware.

Compute. Each gate is an $O(2^n)$ operation (for single-qubit gates) or $O(2^n)$ for two-qubit gates (with careful implementation). A depth-L circuit costs $O(L \cdot 2^n)$.

Noise handling. Statevector simulation is *noiseless*. If you want to simulate noise, you need to either switch to a different mode (density matrix, trajectory) or inject the noise as classical randomness (e.g., stochastically applying Pauli errors between gates).

Pros: fastest for small n, simplest code, no approximation.

Cons: exponential in n, no native noise support.

Use cases for VQAs: - Developing and debugging the algorithm structure. - Computing exact expectation values to serve as ground truth for noisier simulations. - Evaluating expressibility (via frame potential sampling) and trainability (via exact gradient variance). - Small-scale proof-of-concept experiments.

Qiskit's `AerSimulator(method='statevector')`, Cirq's `Simulator`, PennyLane's `default.qubit`, and Intel QS are typical implementations.

Density Matrix Simulation

Stores the full density matrix $\rho \in \mathbb{C}^{2^n \times 2^n}$. This is the most general state representation — it can handle mixed states, which are required for anything involving noise, partial traces, or decoherence.

Memory. $2^{2n} \times 16$ bytes. For $n = 15$, 16 GB. For $n = 18$, 1 TB. Effective limit is $n \approx 15$ on a workstation.

Compute. Each gate application is a superoperator action on ρ, costing $O(4^n)$ naively but can be made $O(2^n \cdot k)$ for k-local gates with smart implementation. Noise channels are applied the same way as gates, using Kraus operators.

Pros: full generality, exact noise simulation with any Kraus channel, clean handling of partial traces and reduced states.

Cons: memory cost is squared compared to statevector — half the qubit limit ($n \approx 15$ vs 30).

Use cases for VQAs: - Testing noise mitigation techniques on small systems. - Comparing different noise models (depolarizing vs amplitude damping). - Computing exact noisy gradients and their variances. - Validating noise-aware ansatz designs.

When to use. When you need to simulate noise and your system is small enough to afford the memory cost.

Qiskit's `AerSimulator(method='density_matrix')` and `qiskit_aer.noise.NoiseModel` is the standard implementation.

Stochastic Trajectory Simulation

The practical middle ground: use pure-state (statevector) simulation, but randomly inject noise at each gate according to the noise channel's Kraus decomposition.

How it works. For each gate followed by a noise channel with Kraus operators $\{K_k\}$, sample a k with probability $\|K_k|\psi\rangle\|^2$ and apply $K_k|\psi\rangle / \|K_k|\psi\rangle\|$. Repeat many times and average.

Memory. Same as statevector: $O(2^n)$. Can simulate up to $n \approx 30$.

Compute. Same as statevector per trajectory, but you need many trajectories to average out the noise. Typically 100-10000 trajectories per data point.

Pros: scales to large n, can handle any Kraus channel, no exponential blowup from density matrix.

Cons: each trajectory is noisy — the simulated observable estimate has variance from *both* the original shot noise *and* the Monte Carlo noise averaging. Can require many more trajectories than naive shot count.

Use cases for VQAs: - Simulating noisy VQAs on 20-30 qubits with realistic noise. - Testing mitigation techniques at scale. - Producing the baseline "noisy simulation" against which hardware runs are compared.

When to use. When you need noisy simulation *and* density matrix is too expensive (typically $n > 15$).

Qiskit Aer's `AerSimulator(method='matrix_product_state')` and `method='stabilizer'` are not trajectory methods, but the default `method='automatic'` often uses trajectory methods when noise is present. PennyLane's `default.mixed` and Cirq's `density_matrix_simulator` are alternatives.

Tensor Network Simulation (MPS, PEPS)

For quantum states with *low entanglement*, tensor network methods scale much better than the general-state methods.

Matrix Product States (MPS). Write the state as

$$|\psi\rangle = \sum_{i_1,\ldots,i_n} \mathrm{Tr}(A_1^{i_1} A_2^{i_2} \cdots A_n^{i_n}) |i_1 \cdots i_n\rangle,$$

where $A_k^{i_k}$ are matrices of dimension $\chi \times \chi$ (called the "bond dimension").

Memory. $O(n \cdot \chi^2)$. Grows polynomially in n and in χ.

Compute. Each two-qubit gate costs $O(n \cdot \chi^3)$ in the worst case.

Key observation. The bond dimension χ is related to the *entanglement* in the state: it must be at least $2^{S/\ln 2}$ where S is the entanglement entropy between a subsystem. For *low-entanglement states*, χ stays small and simulation is efficient.

Pros: can simulate very large systems (hundreds of qubits) for low-entanglement states. Handles noise via purification (adding ancilla legs).

Cons: fails for highly entangled states — the bond dimension explodes. Also awkward for non-local interactions (two-qubit gates between distant qubits).

Use cases for VQAs: - Benchmarking VQAs on large systems (50-200 qubits) when the state remains low-entanglement. - Testing circuits on lattice Hamiltonians where the ansatz preserves locality. - Dequantization studies — seeing if the quantum state could be simulated classically.

When *not* to use. For highly entangled circuits (random HEA, deep QAOA), MPS fails. Statevector or trajectory is the better choice.

Reference. Vidal (2004) introduced time-evolving block decimation (TEBD); modern packages include ITensor, quimb, and TeNPy.

2D generalization (PEPS). Projected Entangled Pair States work on 2D lattices. Useful for 2D VQAs but have their own convergence issues.

Clifford Simulation

A very different mode: the **stabilizer formalism** tracks quantum states as collections of "stabilizers" (Pauli operators that the state is an eigenstate of) rather than amplitudes.

Gottesman-Knill theorem. Any quantum circuit consisting of: - Clifford gates (CNOT, H, S, their conjugates) - Pauli measurements - Preparation in computational basis states

can be simulated classically in polynomial time — specifically, $O(n^3)$ per gate for n qubits.

Pros: exponentially faster than statevector for Clifford-only circuits. Can simulate thousands of qubits.

Cons: *only* for Clifford circuits. Any non-Clifford gate (T, parameterized rotations) breaks the formalism. Can be extended to "T-doped" circuits with $O(2^t)$ cost for t T gates.

Use cases for VQAs: - Testing the Clifford variants of a VQA (e.g., a VQA reduced to Clifford for benchmarking). - Clifford Data Regression (node 9.4) uses this as a subroutine. - Noise characterization via randomized benchmarking (Clifford circuits with varied lengths).

When to use. Whenever you can reduce your VQA to a Clifford subcircuit, use Clifford simulation for that part.

Qiskit Aer's `method='stabilizer'`, Cirq's Clifford simulator, and Aaronson's CHP are standard implementations.

Approximate Classical Simulation

Several methods produce *approximate* classical simulations of quantum states at controlled accuracy:

1. Truncated amplitude methods. Only track the largest amplitudes, discard small ones. Works when the state is sparse in the computational basis.

2. Classical shadows (Huang-Kueng-Preskill 2020). Measure the state in random bases and use the measurement record to compute arbitrary k-local observables. Not a full simulation but sufficient for VQA expectation values.

3. Slicing methods. Break the circuit into smaller slices that are simulated independently and then combined. Scales with circuit depth but with cost reduction from parallelism.

4. Machine-learning surrogates. Train a neural network to approximate the expectation value function $\boldsymbol{\theta} \mapsto C(\boldsymbol{\theta})$ from a small number of exact evaluations. Useful for exploring the landscape without running the simulator.

Use cases. These are for extreme scaling or exploration. Not primary simulation modes but complementary tools.

Simulation Mode Decision Tree

For a VQA practitioner, the choice of simulation mode depends on:

1. **System size n.**
   - $n \leq 15$: density matrix is affordable, use it for noise simulation. Statevector for noiseless.
   - $15 < n \leq 30$: statevector for noiseless, stochastic trajectory for noisy.
   - $n > 30$: tensor network if low entanglement, classical shadows for expectation values, or restrict to Clifford variants.
2. **Noise model needed?**
   - No noise: statevector, tensor network (if large).
   - Depolarizing only: stochastic trajectory is cheap.
   - General noise model: density matrix (if small enough), trajectory (if large).
3. **What's being computed?**
   - Full state: density matrix or statevector.
   - Expectation values only: classical shadows can work.
   - Gradient: parameter shift rule with any underlying simulator.
4. **Expressibility/trainability study?**
   - Noiseless statevector — no noise needed for the structural analysis.
5. **Mitigation validation?**
   - Density matrix for small, trajectory for medium, unavailable for very large.

Rule of thumb. Start with the cheapest mode that answers your question, and only upgrade when the cheaper mode is insufficient. Noise simulation is expensive; avoid it when possible.

Thesis View: Simulation Modes And The Objective Generator

From the thesis framework:

1. **The simulation mode *is* the objective generator for classical pretraining.** Before deploying to hardware, the VQA's cost function is computed via simulation; the simulation mode determines the accuracy and speed of that computation.

2. **Different simulation modes give different signal-to-noise tradeoffs.** Exact (statevector) simulation gives perfect signal; trajectory simulation matches hardware noise. Choosing the mode is a design decision that affects the training dynamics.
3. **HOPSO can use different simulators for different purposes.** Exact for expressibility checks, trajectory for realistic cost evaluation, classical shadow for large-scale exploration. The separation of objective and dynamics lets you swap simulators without changing the optimizer.
4. **Simulation is the "oracle" for the optimizer.** The optimizer doesn't know whether its cost evaluations come from hardware or simulation — as long as the interface is the same, the optimizer works. This is the practical manifestation of the thesis's abstraction.

Structural Role

This node is the **practical simulation reference** for VQA research. It covers the main simulation modes, their costs, and when to use each.

It is the bridge between the theoretical noise models (9.1-9.3) and the practical experiments that use them. Module 9's material is only useful if you can actually simulate it, and this node is the guide to doing so.

Relations to Other Concepts

- **Gottesman, Knill (1998)** — stabilizer formalism and Clifford simulation.
- **Vidal (2004)** — TEBD and MPS for quantum simulation.
- **Aaronson, Gottesman (2004)** — improved Clifford simulation.
- **Huang, Kueng, Preskill (2020)** — classical shadows.
- **Qiskit Aer documentation** — the reference implementation.
- **Module 9.1 (Quantum noise channels)** — what the simulator needs to represent.
- **Module 9.4 (Mitigation)** — what the simulator is used to test.

Worked Example — Choosing A Simulator For A 20-Qubit VQE

Problem. A 20-qubit chemistry VQE on LiH with a hardware-efficient ansatz at depth 10. Goal: evaluate the exact noiseless cost and a noisy cost with 99.5% two-qubit gate fidelity, 99.9% single-qubit fidelity, 97% readout fidelity.

Option 1 — Statevector, noiseless. - Memory: $2^{20} \cdot 16$ bytes = 16 MB. Trivial. - Compute: $10 \cdot 20 \cdot 2^{20} \approx 2 \times 10^8$ operations per data point. Fast (seconds). - Use: for the noiseless baseline.

Option 2 — Density matrix, noisy. - Memory: $2^{40} \cdot 16$ bytes = 16 TB. Unfeasible on workstation. - Skip this option.

Option 3 — Stochastic trajectory, noisy. - Memory: $2^{20} \cdot 16$ bytes = 16 MB per trajectory. - Compute: same as statevector per trajectory. Need ~1000 trajectories per data point. - Total compute: 1000 $\times$ (statevector cost) $\approx 2 \times 10^{11}$ operations $\approx$ minutes per data point. - Use: for the noisy baseline.

Option 4 — Hardware. - Compute: 1-10 minutes per cost evaluation including queue time. - Noise level: realistic. - Cost: real compute time and shot budget.

Workflow. Use Option 1 for landscape exploration and hyperparameter tuning; Option 3 for validating noise mitigation; Option 4 for the final runs.

Observation. For 20-qubit VQEs, density matrix simulation is already out of reach. Trajectory methods become essential. This is typical for realistic VQA experiments.

Visualization

Pending. A plot comparing simulation cost (memory, time) vs qubit count for each mode: statevector, density matrix, trajectory, MPS (at fixed bond dimension), Clifford. Log-log scale. Caption: "Simulation modes scale differently. Pick the right one."

Exercises

1. Compute the memory cost of density matrix simulation for $n = 10, 15, 20, 25$ qubits. At what point does it exceed your workstation's RAM?
2. For stochastic trajectory simulation, show that the variance of the noisy expectation value estimate is proportional to $1/N_{\text{traj}}$ and derive the constant.
3. (VQA) For a 25-qubit MPS simulation of a depth-20 HEA, estimate the bond dimension needed for 1% accuracy. Is this tractable?

Research Extensions

- **Hybrid simulation modes** — switching between simulators during a VQA run based on resource availability.
- **GPU-accelerated statevector simulation** — scaling statevector beyond $n = 30$ using multi-GPU clusters.
- **Smart noise injection** — reducing the trajectory count needed by stratified sampling.
- **Approximate density matrix methods** — low-rank or tensorized density matrices for $n > 15$.
- **Machine-learning surrogates for cost landscapes** — neural networks replacing expensive simulation during optimization.

Cross-links to Other Courses

- **Classical Computing and Algorithms** — cost models, memory hierarchy, parallelization.
- **Tensor Network Methods in Physics** — MPS, PEPS, DMRG.
- **Module 9.1 (Quantum noise channels)** — what simulators represent.
- **Module 9.4 (Mitigation techniques)** — what simulators are used to validate.
- **Module 8.7 (Quantum advantage criteria)** — the line between tractable and intractable simulation.

Taxonomy of Noise in VQA

Position in Knowledge Graph

System: Variational Quantum Algorithms **Structural Domain:** Hardware Noise Models **Node:** 9.8

This is the **closing synthesis** of Module 9. The earlier nodes covered noise as individual objects: quantum channels (9.1), the depolarizing model (9.2), real hardware parameters (9.3), mitigation techniques (9.4-9.5), noise-adaptive ansatz design (9.6), and simulation modes (9.7).

This node steps back and organizes the whole landscape into a **taxonomy**: a structured map of the kinds of noise that affect VQAs, the kinds of damage they do, and the kinds of defenses that apply to each. The aim is to give a single chart that a practitioner can consult when diagnosing a noisy VQA — *"what kind of noise is killing me, and what do I do about it?"*

The taxonomy is organized along three axes:

1. **Where in the stack the noise lives** (physical, gate, circuit, classical post-processing).
2. **What kind of effect it has on the quantum state** (unital vs non-unital, Markovian vs non-Markovian, coherent vs incoherent).
3. **What it does to the VQA cost landscape** (shrinks, shifts, flattens, warps).

Each cell in the taxonomy has a characteristic signature, a set of mitigation techniques, and a connection to the thesis framing. At the end, I close the module with a summary of where the VQA literature stands on noise (2026) and where the open problems are.

Axis 1: Where The Noise Lives

Quantum computing has a layered architecture, and noise can enter at any layer. Knowing the layer tells you the characteristic timescale and the available countermeasures.

Layer 1 — Physical (sub-gate)

Examples. Thermal excitation, spontaneous emission, flux noise in superconducting qubits, laser phase noise for ion traps, atom loss for neutral atoms, nuclear spin coupling for silicon spins.

Timescale. Continuous during the circuit. Parameterized by T_1, T_2, T_ϕ (node 9.3).

Effect. Coherent and incoherent decay of the quantum state. The dominant source of decoherence in all current platforms.

Countermeasures. Hardware engineering (better qubits, better isolation, dynamical decoupling). Cannot be fixed at the circuit level; it's intrinsic to the physical layer.

Layer 2 — Gate (control-level)

Examples. Calibration drift, imperfect pulse shape, crosstalk between neighboring gates, leakage outside the computational subspace, finite control precision.

Timescale. Per-gate (nanoseconds to microseconds for superconducting, microseconds for ions).

Effect. Gate fidelity less than 1. Causes coherent errors (when control parameters are slightly wrong) and incoherent errors (when control parameters vary randomly).

Countermeasures. Recalibration (periodic benchmarking and correction), pulse shaping (DRAG pulses, Slepian pulses), randomized compiling (converting coherent to incoherent), gate-level Pauli twirling.

Layer 3 — Circuit (compilation-level)

Examples. SWAP overhead from routing, suboptimal layout, unnecessary gate count from poor decomposition, idle-time errors during synchronization.

Timescale. Circuit-level (microseconds to milliseconds).

Effect. Inflates the effective noise rate compared to the optimal circuit. Essentially a "compilation inefficiency" that translates to extra noise.

Countermeasures. Better compilation: noise-aware routing, native-gate ansätze, circuit optimization passes, ADAPT-style ansatz growth.

Layer 4 — Measurement (readout)

Examples. Readout confusion (measuring 0 when true state is 1), dispersive shift errors, thermal population.

Timescale. Measurement time (microseconds).

Effect. Biases the measurement outcome. Typically 1-5% on current hardware — larger than gate errors.

Countermeasures. Readout error correction (confusion matrix inversion, M3), repeated measurement, ancilla-based parity checks.

Layer 5 — Classical post-processing

Examples. Numerical precision in the classical optimizer, shot noise in expectation value estimation, optimizer instability, hyperparameter sensitivity.

Timescale. Classical loop iteration (milliseconds to seconds).

Effect. Adds noise to the effective cost function. Not a quantum noise per se, but functionally similar from the optimizer's perspective.

Countermeasures. More shots, better estimator (Bayesian posteriors, filters), optimizer improvements (Adam, adaptive learning rates), regularization.

Axis 2: What Kind Of Effect On The State

Within the quantum channel formalism, noise channels can be classified by several structural properties.

Unital vs Non-Unital

A channel is **unital** if $\mathcal{E}(I) = I$ — it preserves the maximally mixed state.

- **Unital examples:** depolarizing, Pauli noise, dephasing.
- **Non-unital examples:** amplitude damping (has a preferred state $|0\rangle$), thermal relaxation, measurement disturbance.

Practical significance. Unital noise has no "preferred direction" in state space — it symmetrically shrinks expectation values. Non-unital noise *biases* the state toward a specific fixed point, which means it can *move the minimum* of the VQA cost landscape, not just shrink its amplitude. This is a more insidious kind of error, because the optimizer converges to the wrong answer.

Markovian vs Non-Markovian

A noise process is **Markovian** if the error at each step depends only on the current state (no memory). It is **non-Markovian** if there's memory — the error depends on the history.

- **Markovian examples:** depolarizing channels (applied per gate), random Pauli noise.

- **Non-Markovian examples:** low-frequency flux noise, 1/f noise, crosstalk between distant gates via a shared bath.

Practical significance. Markovian noise accumulates predictably (multiplicatively per gate). Non-Markovian noise is harder — it can cause correlated errors that break standard analysis. Most VQA theory assumes Markovian; real hardware is often not.

Coherent vs Incoherent

Coherent errors are unitary but wrong — e.g., a Z rotation by 1.01π instead of π. They preserve the purity of the state; the state evolves to a slightly wrong pure state.

Incoherent errors are mixing operations — they increase the state's entropy. Decoherence, depolarizing noise, amplitude damping.

Practical significance. Coherent errors are *more damaging per error rate* because they can constructively interfere over many gates, compounding in a correlated way. Incoherent errors average out (variance scales as $\sqrt{L}$). So a 1% coherent error can do more damage than a 1% incoherent error over a deep circuit.

Countermeasure. *Randomized compiling* converts coherent errors to incoherent by randomly conjugating each gate with a random Pauli. It's essentially free and recommended for VQAs on coherent-error-dominated hardware.

Axis 3: What It Does To The VQA Cost Landscape

This is the VQA-specific axis — what the noise does to the optimizer's experience of the landscape.

Shrinks The Contrast

The most common effect: depolarizing-like noise shrinks the amplitude of all Pauli expectation values uniformly. The cost landscape is compressed toward a constant, but its *shape* (location of minima, directions of gradients) is unchanged.

Signature. Cost values all closer to the noiseless average $\mathrm{Tr}(H)/d$; minimum unchanged in parameter space; gradients scaled down.

Impact on VQA. Reduces signal-to-noise ratio; optimizer still finds the right answer but needs more shots. Mitigable via ZNE or CDR.

Shifts The Minimum

Non-unital noise (amplitude damping, thermal population) biases the state toward a specific fixed point, which shifts the cost function's minimum in parameter space. The optimizer converges, but to the wrong place.

Signature. Noisy minimum is at different parameters than noiseless minimum; even with infinite shots, the optimizer converges to a biased answer.

Impact on VQA. Systematic bias that cannot be removed by averaging. Requires either PEC (unbiased mitigation) or symmetry verification (if a symmetry is broken by the noise).

Flattens The Landscape

Depolarizing noise at high rate or over deep circuits causes the entire landscape to become flat — the exponential decay of contrast reaches the shot noise floor everywhere. This is the **noise-induced barren plateau** (Wang 2021).

Signature. Gradient variance is indistinguishable from shot noise; optimization stalls; no parameter update direction is reliable.

Impact on VQA. Training fails. No amount of shots helps (shot noise floor is the problem). Only defense is shallower circuits or noise reduction at the hardware level.

Warps The Landscape

Coherent errors and correlated noise can introduce *wrong* features into the landscape — spurious minima, new saddle points, distorted gradient directions.

Signature. Hard to detect; the optimizer converges but the convergence point is wrong in a non-obvious way. Validation against a known answer reveals the issue.

Impact on VQA. Insidious. Testing on a known-answer benchmark (small H_2 or a Clifford-reducible problem) is essential to catch this.

The Full Taxonomy Table

Putting it all together:

Noise type	Layer	Unital?	Markov?	Cost effect	Mitigation
Depolarizing	Gate	Yes	Yes	Shrinks	ZNE, CDR
Amplitude damping	Physical/Gate	**No**	~Yes	Shifts + shrinks	PEC, symmetry verification
Dephasing	Physical	Yes	Yes (approx)	Shrinks Z contrast	DD, ZNE
Coherent (cal drift)	Gate	Yes	Yes	Warps	Recalibration, randomized compiling
Crosstalk	Gate	Varies	**No**	Warps	Scheduling, DD, isolation
Leakage	Gate	**No**	Varies	Warps	Leakage-reducing pulses, filtering
Readout	Measurement	N/A	Yes	Bias observations	Confusion matrix inversion
1/f flux noise	Physical	Yes	**No**	Shrinks + warps	DD, hardware engineering
Shot noise	Classical	N/A	Yes	Flattens	More shots, Bayesian filter
Calibration drift	Control	Varies	**No**	Warps over time	Periodic recalibration, adaptive control

Reading the table. Each row is a distinct noise source. The columns tell you how it behaves and what to do about it. The "Mitigation" column points at specific techniques.

Key observations: - Non-unital and non-Markovian noise are the hardest cases. - Depolarizing is the best-case scenario — everything else is worse. - Classical-layer "noise" (shot noise, drift) is a major factor in practice.

The VQA Noise Hierarchy

On current (2026) hardware, the noise sources rank roughly as follows, in order of total impact on VQAs:

1. **Two-qubit gate errors.** By far the dominant source. 0.1%-1% per gate. All mitigation efforts focus here first.
2. **Readout errors.** Second-biggest. 1%-5% per qubit. Cheap to fix, always apply readout mitigation.
3. **Single-qubit gate errors.** 0.01%-0.1%. Smaller but accumulate over many gates.
4. **Decoherence during idle.** T2 effects. Mitigable via DD and smart scheduling.
5. **Crosstalk.** Usually smaller than direct gate errors but can be correlated and insidious.
6. **Coherent errors / calibration drift.** Slow-varying. Recalibration handles it.
7. **Leakage.** Usually $< 0.1\%$ on mature platforms. Reduced by careful pulse design.

This hierarchy is *specific to superconducting hardware in 2026*. Trapped ions have different rankings (less readout error, more laser noise). Neutral atoms add atom loss. The taxonomy is platform-dependent but the axes are universal.

Where The Field Stands (2026)

A summary of the state of noise + VQA research as of 2026:

1. Theory is mature for depolarizing noise. Wang 2021's plateau theorem, Ragone-style bounds, and ZNE theory give a comprehensive picture for the depolarizing case.

2. Non-unital noise is less well-understood. Analytical results for amplitude damping are scattered. Most theoretical work still uses depolarizing as a stand-in.

3. Mitigation is well-developed but has hit a shot-complexity wall. ZNE, PEC, CDR, symmetry verification are all established. The open question is how to extend them without exponential shot blow-up.

4. Hardware fidelities are improving but slowly. Two-qubit gate fidelities have gone from 99% (2018) to 99.9% (2026) on the best platforms — a factor of 10. This gives a factor-10 increase in usable depth. Steady but not transformational progress.

5. Circuit-level mitigation is well-established. Noise-aware compilation, ADAPT-VQE, and HEA are in every production VQA stack.

6. Open frontiers: - **Non-Markovian noise modeling** — how to treat it in VQA theory. - **Adaptive mitigation** — auto-selecting techniques based on circuit properties. - **Noise-aware optimizer design** — optimizers that know about mitigation cost. - **Error-corrected VQAs** — using minimal error correction (a few logical qubits) inside VQAs. - **Noise-tailored ansätze** — designing ansätze that are specifically resistant to the dominant noise types.

The field has matured from "we don't know how noise affects VQAs" (2017) to "we have a comprehensive taxonomy and standard tools" (2026). The remaining questions are about *how much* mitigation and adaptation can push the NISQ-era depth, and when fault-tolerance will take over.

Thesis Framing

From the thesis perspective:

1. **Noise is a perturbation of the objective generator, not of the optimizer dynamics.** The optimizer (HOPSO or otherwise) operates on whatever noisy cost function it's given; the noise lives in the evaluation of the objective. Separating these cleanly is architecturally important.
2. **Mitigation is a classical side-computation attached to the objective generator.** Not part of the quantum dynamics. This matches HOPSO's abstraction of the optimizer-objective boundary.

3. **Noise-adaptive ansatz design is a modification of the objective generator's functional form.** Changing the ansatz changes what function $C(\boldsymbol{\theta})$ the optimizer sees. This is a form of dynamical restriction that complements regularization.

4. **Adaptive noise handling (monitoring hardware drift and reconfiguring) is the natural extension.** This is what Module 10 takes up explicitly: the closed-loop adaptive VQA that uses feedback control at the outer level.

The thesis claim from this module: **noise is a solvable problem at the depths current hardware allows, provided you use the full toolkit (adaptation + mitigation + hardware tuning) and choose the right optimizer for the noisy landscape**.

Closing Module 9

Module 9 has traversed the noise landscape:

- **9.1:** Quantum channels and CPTP maps — the formal language.
- **9.2:** Depolarizing noise — the workhorse model and its implications.
- **9.3:** Gate errors and coherence times — the physical parameters.
- **9.4:** Mitigation techniques — the post-processing toolkit.
- **9.5:** Zero-noise extrapolation — the dominant mitigation in depth.
- **9.6:** Noise-adaptive ansatz design — the pre-processing complement.
- **9.7:** Simulation modes — how to model all this classically.
- **9.8 (this node):** Taxonomy — the full chart and the state of the field.

The arc: from abstract formalism → to physical reality → to defensive techniques → to a unified map. Each step adds structure that the previous lacked.

This module is the *practical* companion to Modules 7 (control theory) and 10 (adaptive deployment). Those modules tell you how to *design* a VQA for noise; this module tells you what the noise *is*.

Structural Role

This node is the **synthesis and closing** of Module 9. It organizes the module's content into a three-axis taxonomy, provides a summary table, ranks noise sources by impact, and sketches the state of the field in 2026.

It is the capstone reference node for VQA practitioners who need to quickly diagnose noise issues and select countermeasures.

Relations to Other Concepts

- **Nielsen-Chuang** — the formal channel language.
- **Wang et al. (2021)** — noise-induced plateau theorem.
- **Cai et al. (2023)** — comprehensive review of quantum error mitigation.
- **Module 9.1-9.7** — the components being synthesized.
- **Module 10** — the next step, where adaptive control handles drift and time-varying noise.

Worked Example — Diagnosing A Failing VQA

Scenario. You're running a 12-qubit VQE for a molecule. You know the noiseless answer (from exact diagonalization): ground state energy $E_0 = -7.88$ Hartree. Your noisy VQA converges to -7.72, off by 0.16 Hartree (way too much).

Diagnosis workflow:

Step 1. Apply readout error correction. New result: -7.79. Improvement 0.07; readout was a significant contributor.

Step 2. Apply symmetry verification on electron count. New result: -7.81. Improvement 0.02; non-unital (amplitude-damping-like) errors were shifting the minimum.

Step 3. Apply ZNE with linear fit. New result: -7.85. Improvement 0.04; depolarizing-like errors were shrinking contrast.

Step 4. Switch to noise-aware qubit selection (best 12 qubits on the device). New result: -7.87. Improvement 0.02; heterogeneous fidelity was inflating effective noise.

Step 5. Apply DD during idle periods. New result: -7.875. Improvement 0.005.

Remaining error: 0.005 Hartree. Possibly unmitigable or requires deeper techniques (PEC, CDR).

Diagnosis summary. - Main source: readout error (44% of total bias). - Second: depolarizing contrast loss (25%). - Third: non-unital shift (12%). - Fourth: heterogeneous fidelity (12%). - Fifth: idle errors (3%).

Each source is attacked by a different mitigation. The taxonomy directs which tool to use for which problem.

Visualization

Pending. A treemap or Sankey diagram showing the five-layer stack (physical, gate, circuit, measurement, classical), with noise sources in each layer sized by their typical contribution to VQA error. Arrows connecting each source to its mitigation technique. Caption: "The noise taxonomy at a glance."

Exercises

1. For each noise type in the taxonomy table, identify whether it can shift the VQA minimum (non-unital) or just shrink contrast (unital).
2. (Conceptual) Why is a 1% coherent error typically more damaging than a 1% incoherent error for a deep VQA? Quantify using a circuit of depth 100.
3. (VQA, project-scale) Design a noise diagnostic pipeline: given a noisy VQA result, determine the dominant error source by applying mitigation layers one at a time and measuring the improvement. Output the diagnosis as a table.

Research Extensions

- **Non-Markovian noise for VQAs** — extending the theory to time-correlated errors.
- **Adaptive mitigation selection** — auto-picking the right mitigation per circuit.
- **Error-correction-aware VQAs** — running VQAs on partially error-corrected circuits.
- **Noise-tailored ansatz families** — ansätze specifically designed to resist the dominant error type on the target hardware.
- **Unified VQA noise simulator** — simulating all noise axes simultaneously with controllable knobs.

Cross-links to Other Courses

- **Module 9.1-9.7** — all the components of the taxonomy.
- **Module 7 (Control-Theoretic View)** — the design side.
- **Module 10 (Adaptive Control Systems)** — the deployment endpoint.
- **Module 6 (Optimization Dynamics)** — the optimizer's interaction with noise.
- **Quantum Error Correction** — the long-term answer beyond NISQ.