Feedback Control in Quantum Systems

Variational Quantum Algorithms · Module 7 — Control Theoretic View

Node 7.6

Position in Knowledge Graph

System: Variational Quantum Algorithms **Structural Domain:** Control-Theoretic View **Node:** 7.6

This is the **closing node** of Module 7. It is the synthesis: having developed observability (7.2), controllability (7.3), optimal control (7.4), and Pontryagin (7.5) as building blocks, this node combines them into the central object of modern control theory — the **closed-loop feedback controller**.

A feedback controller is a system that **observes** the state, **decides** what control input to apply based on the observation, and **acts**. The defining feature is that the controller's decisions depend on the observed state — closing the loop. This is what makes control systems robust to noise, drift, and disturbances; it is the technological basis of every modern engineered system from cruise control to power grids.

For VQAs, feedback control is the right framework for the *adaptive* extension. A non-adaptive VQA has a fixed optimizer schedule and ignores the system's response. An adaptive VQA observes the system (gradient norms, cost trajectory, hardware drift) and modifies its strategy accordingly. The control-theoretic view says: this is a feedback controller, and you should design it as one — using the same principles that have worked for 70 years in classical engineering.

This node walks through feedback control concepts, the quantum measurement complications, and the implications for adaptive VQA design. It is the natural bridge to Module 10 (VQA as Adaptive Control Systems).

What Feedback Control Is

A **feedback control system** has:

- A **plant** (the system being controlled).
- A **sensor** (measures the plant's output).
- A **controller** (decides the input based on the measurement).
- An **actuator** (applies the input to the plant).

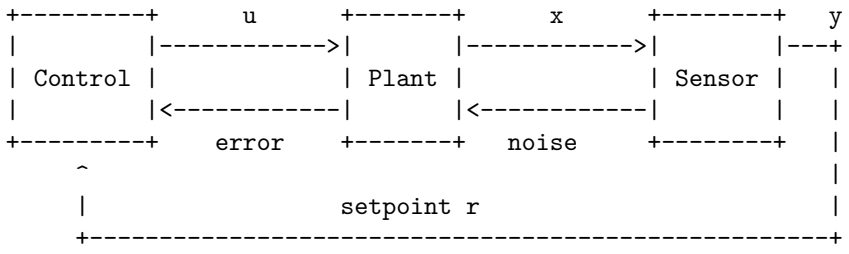
The signal flow is: plant → sensor → controller → actuator → plant. The loop is *closed* because the controller's output depends on the plant's measured output.

Compare to open-loop control: in open loop, the controller just runs a pre-computed input sequence and doesn't observe the plant. Open loop is simple but fragile — any disturbance or model error throws it off.

Closed loop is robust because errors get observed and corrected. This is why every real engineered system uses feedback.

The Standard Feedback Block Diagram

The canonical block diagram:



The controller compares the measured output y against a desired setpoint r, computes an error, and produces a control input u. The plant state x evolves under u, the sensor measures $y = h(x) + \text{noise}$, and the loop continues.

The **controller design problem** is: given the plant's dynamics and the sensor characteristics, design the controller logic so that the closed-loop system has desired properties (stability, fast convergence, disturbance rejection, etc.).

Classical Feedback Designs

A few standard patterns:

1. Proportional-Integral-Derivative (PID).

$$u(t) = K_p \cdot e(t) + K_i \int_0^t e(\tau)d\tau + K_d \cdot \dot{e}(t),$$

where $e(t) = r - y(t)$ is the error. Three tunable gains. Simple, robust, dominates classical industrial control. Most servos, thermostats, and chemical reactors use PID.

2. State feedback. $u = -K\mathbf{x}$ for a constant gain matrix K. Requires full state observation. Optimal under LQR assumptions.

3. Output feedback. $u = -K\mathbf{y}$ — directly from measurements without estimating the state. Simpler but less powerful than state feedback.

4. Observer-based feedback (separation principle). First estimate the state $\hat{\mathbf{x}}$ from measurements (Kalman filter), then apply state feedback to the estimate: $u = -K\hat{\mathbf{x}}$. Decouples state estimation from control design.

5. Model Predictive Control (MPC). At each time step, solve a finite-horizon optimal control problem online, apply only the first action, then repeat. Handles constraints naturally. Used in process control and (more recently) robotics.

6. Adaptive control. The controller's parameters update online as the plant is observed. Useful when the plant model is uncertain or slowly varying. Module 10 takes this up.

Why Quantum Feedback Is Different

Translating these to the quantum setting runs into the basic constraint of quantum measurement: **measurement collapses the state**.

This means:

1. You cannot continuously observe the state. Each measurement destroys the state (or at least disturbs it). To get a continuous trace of the state, you need many copies of the system or a *weak* measurement scheme.

2. The measurement is intrinsically noisy. Even an ideal projective measurement gives a sample from a probability distribution, not the underlying state directly.

3. The feedback delay matters. If the time between measurement and feedback action is non-negligible, the state has already evolved during that delay, and the action is based on stale information.

These constraints have spawned a whole sub-field: **quantum feedback control**. The two main paradigms:

(a) Discrete feedback. Measure projectively, then apply a unitary or reset based on the measurement outcome. Repeat. Used in error correction, state preparation, and discrete quantum computation.

(b) Continuous feedback. Use weak measurements that perturb the state minimally, integrate the measurement record to estimate the state in real time, and apply Hamiltonian feedback proportional to the estimate. The framework is **stochastic master equations** plus **conditional dynamics** — Wiseman-Milburn formalism.

For VQAs, the relevant paradigm is **discrete feedback at the optimizer level**: measure the cost (or gradient), update the parameters, then repeat. The "measurement" is the cost estimation, the "controller" is the optimizer, the "actuator" is the parameter update mechanism.

VQA As A Discrete Feedback Loop

The VQA outer loop is *exactly* a discrete feedback control loop:

Feedback element	VQA element
Plant	Quantum circuit + cost evaluation
Sensor	Cost estimator (with shot noise)
Controller	Optimizer (Adam, GD, HOPSO, etc.)
Actuator	Parameter update
Setpoint	Target cost (often $-\infty$ or known optimum)
Loop frequency	Iterations per second
Loop delay	Time per iteration (state prep + measurement + classical compute)

Implications of this framing:

1. **The optimizer is a controller.** It can be designed using control-theoretic principles, not just machine-learning intuitions.
2. **Loop delay matters.** If iterations are slow, the controller's actions are based on stale information. This affects optimizer choice — fast-converging optimizers (Adam) are better when iterations are slow.
3. **Sensor noise matters.** Shot noise in cost estimation is exactly sensor noise in the feedback loop. It can be filtered (Kalman-style), averaged (running mean), or compensated (Robbins-Monro).
4. **Stability is a concern.** In feedback loops, large gains can cause instability (oscillations, divergence). In VQAs, large learning rates have the same effect. The control framing tells you to think about gain scheduling and stability margins, not just convergence rates.
5. **Disturbance rejection is a concern.** Hardware drift acts as a disturbance on the plant. A good optimizer should reject this disturbance — adapt as the hardware changes. This is the adaptive control problem.

The Separation Principle For VQAs

Classical control theory has a beautiful result: under linearity and Gaussian noise, the optimal controller can be **separated** into:

1. A **state estimator** (Kalman filter) that produces a best-estimate state from the measurement record.
2. A **feedback law** (LQR-style) that acts on the estimate.

The two can be designed *independently* and the result is jointly optimal. This is the **separation principle**.

For VQAs, the natural analogue:

1. **State estimator:** an estimator that maintains a posterior over the parameter values (or the cost landscape) given the noisy cost evaluations.
2. **Feedback law:** an optimizer that acts on the posterior estimate.

A naive VQA optimizer skips the estimator and feeds raw measurements into the controller — equivalent to a "no-filter" controller. Adding even a simple moving-average filter (or a Kalman-like estimator) significantly improves performance under noise.

Bayesian VQA optimizers (Bayesian optimization, Gaussian process bandits) implement this separation explicitly: they maintain a posterior over the cost landscape and update parameters based on the posterior. They are the cleanest example of separation-principle thinking in VQAs.

Robust Control For VQAs

A more advanced control-theoretic concept: **robust control** handles uncertainty in the plant model. Classical methods include H_∞ control, μ-synthesis, and Lyapunov-based robust design.

For VQAs, the "uncertainty" comes from:

1. **Hardware drift** (gates calibrate slightly differently over time).
2. **Noise model uncertainty** (you don't know the exact noise rates).
3. **Cost evaluation noise** (shot noise, plus systematic errors).
4. **Optimizer hyperparameter uncertainty** (you don't know the right learning rate).

A robust VQA controller would be designed to **achieve good performance for all plants in some uncertainty set**. This is conservative but reliable — and it is the right framework when you can't trust your plant model perfectly.

Practical robust techniques for VQAs:

- **Conservative learning rates:** stay below the stability threshold for all plausible plants.
- **Trust regions:** limit the parameter step size so a single bad measurement doesn't move you far.
- **Worst-case cost evaluation:** average over multiple shots or multiple circuits to reduce noise sensitivity.
- **Adaptive robust controllers:** learn the plant uncertainty online and tighten the bounds as more data arrives.

These are not standard in the VQA literature but are routine in classical control. Importing them is one of the underexploited bridges between fields.

Adaptive Optimizers As Adaptive Controllers

The thesis's central claim: **adaptive VQA optimizers (HOPSO, Adam with gain scheduling, regularized SGD) are adaptive controllers**, and they should be designed as such.

The argument:

1. The standard fixed-hyperparameter optimizer is the equivalent of an open-loop controller — runs a pre-computed schedule, ignores system response.
2. An optimizer that observes its own performance (gradient norms, cost trajectory, gradient variance) and modifies its hyperparameters online is a closed-loop adaptive controller.
3. The mature literature on adaptive control gives explicit design principles, stability theorems (Lyapunov-based), and convergence guarantees that apply with appropriate translation.
4. Most VQA optimizers are designed by hand-tuning rather than by adaptive control principles. There is significant room for principled improvement.

This is the bridge to **Module 10**, which takes the adaptive control framing and develops it into specific architectures for VQA deployment.

Quantum Feedback Control: A Brief Survey

For completeness, the dominant approaches in *quantum* feedback control (the inner-loop, gate-level version):

1. Wiseman-Milburn continuous quantum measurement. Weak measurements with Wiener-process measurement records. Stochastic master equations describe the conditional dynamics. The optimal controller is a function of the measurement record.

2. Quantum error correction. Repeatedly measure stabilizers, decode the syndrome, apply correction. The discrete version of feedback. Surface codes, color codes, Bacon-Shor codes all use this paradigm.

3. Measurement-based quantum computation. The whole computation is measurement and feed-forward. Each measurement determines the next, and the final state encodes the result. Inverts the usual "prepare then measure" structure.

4. Quantum Kalman filtering. Uses Wiseman-Milburn formalism to maintain a real-time state estimate, then applies feedback to drive the estimate toward a target. Used in cooling experiments and quantum metrology.

These are not VQA techniques per se, but they show that quantum feedback control is a well-developed field with established methods. Importing them into VQA design (especially for hardware-aware optimizers) is an open research direction.

Closing Module 7

Module 7 has now traversed the control-theoretic view of VQAs:

- **7.1:** The framing — VQA as a control system, two-loop structure.
- **7.2:** Observability — what we can learn from measurements, gauge symmetries.
- **7.3:** Controllability — what states we can reach, the DLA equivalence with expressivity.
- **7.4:** Optimal control theory — the algorithmic core, GRAPE, Krotov, CRAB.
- **7.5:** Pontryagin minimum principle — the structural conditions for optimality.
- **7.6 (this node):** Feedback control — closing the loop, adaptive optimizers.

The arc: from passive structural analysis (observability, controllability) to active design (optimal control, PMP) to closed-loop adaptation (feedback). Each step adds a layer that the previous one lacked.

The thesis's central claim from this module: **VQAs are control systems, and 70 years of control theory has techniques the VQA community has not fully imported**. Modules 8 (theory) and 10 (deployment) build on this premise.

Structural Role

This node is the **closing synthesis** of Module 7. It combines observability, controllability, and optimal control into the closed-loop feedback framework, identifies VQA optimization as a discrete feedback loop, and previews the adaptive control deployment of Module 10.

It is also the strongest argument for the control-theoretic view: closed-loop adaptive controllers are the technology that makes modern engineering work, and importing them into VQA design is one of the best-leveraged research directions for making VQAs robust to real hardware.

Relations to Other Concepts

- **Wiener (1948)** — *Cybernetics*, founding text.
- **Kalman (1960)** — Kalman filter, separation principle.
- **Wiseman-Milburn (2010)** — *Quantum Measurement and Control*.
- **Module 6 (Optimization Dynamics)** — the optimizer-as-controller view.
- **Module 10 (Adaptive Control Systems)** — the deployment endpoint.

Worked Example — A Feedback VQA Optimizer

Consider a VQA optimizer that:

1. Estimates the cost $\hat{C}_k$ at iteration k.
2. Computes a moving-average filtered cost $\bar{C}_k = \alpha \bar{C}_{k-1} + (1-\alpha)\hat{C}_k$ (Kalman-like filter).
3. Uses $\bar{C}_k$ (not $\hat{C}_k$) to compute the gradient via parameter shift.
4. Adapts the learning rate based on the variance of recent gradients: low variance → larger step, high variance → smaller step.
5. Detects "stuck" behavior (cost not decreasing) and applies a random kick to the parameters.

As a feedback controller: - **Plant:** the quantum circuit + Hamiltonian. - **Sensor:** raw cost estimator (noisy). - **Filter:** the moving average — denoises the sensor. - **Controller:** the gradient-based parameter update with adaptive gain. - **Actuator:** parameter modification. - **Disturbance rejection:** the random kick when stuck.

This is *exactly* how a competent classical control engineer would design a controller for a noisy plant. None of the techniques are novel in control theory; all are well-established. But applied to VQAs, this kind of design is uncommon — most optimizers are hand-tuned with a single learning rate and no filtering.

The control framing motivates the use of these techniques and gives the principled justification.

Visualization

Pending. A block diagram of a closed-loop adaptive VQA optimizer, with all the control components labeled (plant, sensor, filter, controller, actuator, adaptation, disturbance rejection). Caption: "VQA optimization, redrawn as classical control."

Exercises

1. For a VQA optimizer with cost noise std σ, what is the equivalent "sensor noise" in a classical feedback loop? How would a Kalman filter improve it?
2. The separation principle says estimator and controller can be designed independently under LQG assumptions. For a VQA, what would the "estimator" be?

3. (VQA) Design a feedback controller for a VQA optimizer that handles slow hardware drift (gate fidelities changing on the hour timescale). Specify the sensor, filter, controller logic.

Research Extensions

- **Kalman-filtered VQA optimizers** — explicit Kalman filtering of the cost trajectory.
- **Robust control for hardware-noise VQAs** — applying H_∞ techniques to optimizer design.
- **MPC-style VQA optimizers** — finite-horizon lookahead with constraint handling.
- **Quantum feedback at the gate level for VQAs** — combining inner-loop quantum feedback with outer-loop optimization.

Cross-links to Other Courses

- **Classical Control (Ogata)** — PID, state feedback, observer-based feedback.
- **Quantum Measurement and Control (Wiseman-Milburn)** — quantum-specific feedback theory.
- **Adaptive Control (Åström-Wittenmark)** — the parent literature for Module 10.
- **Module 6 (Optimization Dynamics)** — the optimizer-only view.
- **Module 10 (Adaptive Control Systems)** — the deployment chapter.

Variational Quantum Algorithms · Module 7 — Control Theoretic View · Node 7.6
Concepts: feedback, control-theory, adaptive-systems, stability
Prerequisites: 7.2, 7.4, 7.5

hbar.university — own.your.cognition