MAQCAL: A BOUNDED SEQUENTIAL DECISION WORKFLOW FOR QUANTUM READOUT CALIBRATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Quantum readout calibration repeatedly requires experimenters to locate a resonator frequency and select a readout power, with sweep ranges, resolutions, and stopping decisions often configured from device-specific experience. We present MAQCal, a bounded sequential decision workflow for these two coupled stages. Stage-specific planners propose typed sweep actions from measurement feedback. A deterministic runtime enforces stage order, numerical constraints, and iteration limits, and selects the returned estimates. We provide the planner interfaces and execution rules to support reproducibility. We also study a weighted k-nearest-neighbor (kNN) prior for frequency-search initialization using simulated chip records. On a simulation-driven dataset of 2006 chip designs and three synthetic noise families, the mean of MAQCal's relative SNR gains over the strongest fixed preset in each of 15 reported settings is 13.7%, with setting-specific gains from 4.4% to 23.4%. Stage-wise and scan-budget experiments show consistent improvements in the evaluated frequency and power decisions. MAQCal automates repeated sweep choices and improves readout quality over the tested fixed configurations in simulation under matched downstream processing.

1 INTRODUCTION

Superconducting and trapped-ion quantum processors have been scaled to systems comprising tens to hundreds of qubits (Kjaergaard et al., 2020; Murali et al., 2020), yet these devices remain in the Noisy Intermediate-Scale Quantum (NISQ) regime (Chen et al., 2023), where noise, decoherence, and temporal drift introduce systematic shifts in qubit frequencies, coupling parameters, and readout characteristics (Burnett et al., 2019; Wise et al., 2021). Maintaining reliable operation therefore requires precise and repeated calibration across multiple layers of the hardware stack, including qubit frequency tuning, control-pulse optimization, and readout-chain characterization (Preskill, 2018; Krantz et al., 2019). As Berritta et al. (2025) demonstrate, even parameters whose behavior is well understood in isolation require continuous re-estimation throughout the course of an experiment, making calibration a recurring operational cost.

In practice, much of this calibration is still carried out manually or run as predefined scan routines in standard calibration toolchains (Zwolak & Taylor, 2023; Kelly et al., 2018; Kanazawa et al., 2023; Pasquale et al., 2024). Two steps in quantum readout calibration are especially common and consequential: selecting an appropriate readout tone frequency from resonator spectroscopy and verifying a readout power that provides sufficient signal contrast without driving the resonator into an unstable regime (Jeffrey et al., 2014; Heinsoo et al., 2018; Gao et al., 2021; Sank et al., 2025). In these workflows, a frequency sweep or transmission spectrum is fitted first, after which a selected probe frequency is used for subsequent readout. Power is then chosen from a separately characterized operating range. The settings and stopping conditions are often specified before the next measurement, while experimentalists may still adjust them between runs (Kelly et al., 2015). Repeated adjustment of these choices requires expert attention across devices and experimental sessions (Kjaergaard et al., 2020; Krantz et al., 2019) (see Figure 1).

Our research question is whether a bounded workflow that updates frequency and power sweeps from measurement feedback can improve readout quality over fixed scan presets under matched downstream processing. We formulate the two stages as a sequential decision process. They are coupled because the localized frequency defines the search region for power selection. MAQCal implements this

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

process with stage-specific planners and a finite-state runtime. The planners receive structured measurement summaries and propose typed sweep actions. The runtime executes measurements, applies deterministic numerical guards, checks termination, and selects the returned estimates. This separation makes the physical stage order, action constraints, and stopping rules explicit. We evaluate the complete workflow against six fixed scan presets under three synthetic noise families in simulation.

Our contributions are as follows.

- We present MAQCal, a bounded workflow for coupled frequency localization and power selection. It separates planning from execution through typed sweep actions and a deterministic runtime that enforces stage order, numerical guards, and stopping rules.
- We specify the planner interfaces and execution rules through measurement summaries, action schemas, exact prompts, and a deterministic fallback execution path. Recorded proposals, runtime adjustments, and measurement outcomes support inspection and reproduction of the workflow.
- We evaluate the complete workflow against six fixed scan presets in simulation under matched downstream processing, with stage-wise and matched frequency-budget analyses. We separately evaluate a weighted kNN prior from 200 simulated chip records for frequency-window initialization.

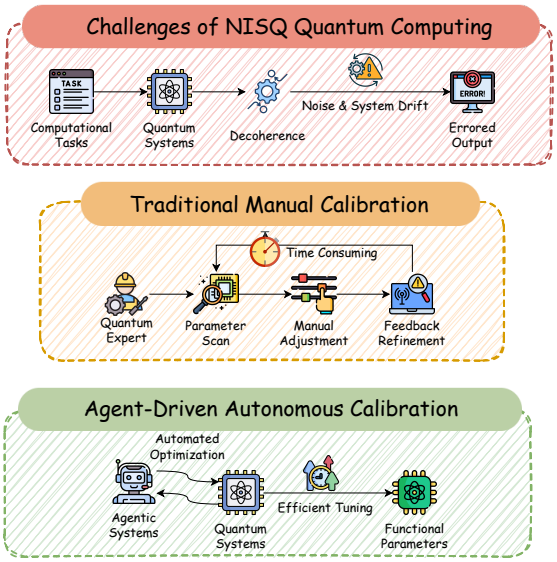

Figure 1: Quantum readout calibration in context. **Top:** noise, decoherence, and drift in NISQ processors. **Middle:** manual scans and expert judgment. **Bottom:** sweep planning from measurement feedback. The "agent-driven" and "agentic" labels refer to the implementation of the evaluated bounded workflow.

2 Related Work

Quantum Readout Calibration. Calibration is essential for maintaining high fidelity operations on quantum processors. Prior work has studied parameter optimization at the gate level, including numerical optimal control methods such as GRAPE (Khaneja et al., 2005) and CRAB (Müller et al., 2022). Beyond gate control, resonator spectroscopy is a standard tool for identifying readout frequencies and characterizing qubit-resonator interactions (Wallraff et al., 2004; Blais et al., 2004). Studies of dispersive readout further show that readout fidelity depends on probe frequency and power through trade-offs among signal contrast, measurement backaction, and nonlinear effects (Gambetta et al., 2007; Jeffrey et al., 2014). In representative experimental workflows, a spectroscopy or transmission sweep is first used to characterize the resonator response. A fitted resonance or a frequency selected from the measured IQ separation is then used as the probe setting for subsequent readout (Jeffrey et al., 2014; Heinsoo et al., 2018). This scan-and-select procedure leaves the sweep window, resolution, and power schedule as settings chosen before or outside the next measurement. Such programmatic scan routines are embodied in standard calibration toolchains and device bring-up procedures, and they remain labor-intensive and dependent on operator experience as device counts grow (Kelly et al., 2018; Wittler et al., 2021; Gao et al., 2021). Adaptive methods based on experimental feedback have also been explored for parameter estimation and drift compensation (Ma et al., 2025; Berritta et al., 2025). MAQCal applies measurement feedback to the coupled frequency

and power decisions within a constrained workflow. We evaluate stored chip records as a separate frequency initialization component.

Agent Systems for Sequential Decision-Making. Agent methods have been studied for tasks that involve sequential decisions under observational feedback. ReAct (Yao et al., 2023b) interleaves reasoning and action, while Tree of Thoughts (Yao et al., 2023a) explores structured search over intermediate decisions. AutoGPT (Yang et al., 2023) and Toolformer (Schick et al., 2023) investigate tool use, task decomposition, and iterative refinement. Reflexion (Shinn et al., 2023), ChemCrow (M. Bran et al., 2024), and Voyager (Wang et al., 2023) study feedback-driven operation in other domains. MAQCal uses a structured observation and action interface for the repeated sweep decisions in readout calibration.

These two lines of work motivate a narrower systems question. Fixed calibration scripts require sweep choices before measurements, whereas adaptive methods update those choices from observed responses. MAQCal combines this feedback loop with a finite-state implementation of the frequency-to-power dependency. We also evaluate a retrieval prior for frequency-search initialization. The resulting contribution is an application-specific, auditable automation workflow evaluated against six fixed scan presets.

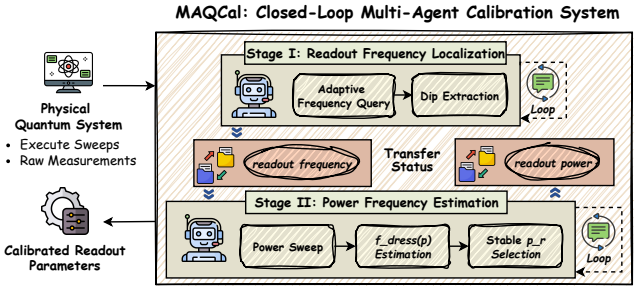

Figure 2: Overview of MAQCal. **Stage I** localizes the readout resonator frequency. **Stage II** uses this frequency to select readout power. A finite-state runtime coordinates the stages, constrains actions, and checks termination. The "multi-agent" label names the two stage-specific planner modules in the implementation. The experiments evaluate the complete bounded workflow.

3 Workflow Design and Components

We first formalize readout calibration as a sequential decision problem, then describe the executable workflow and its components. The overall architecture is shown schematically in Figure 2.

3.1 Problem Formulation

We consider calibration of the superconducting qubit readout chain, focusing on selecting the readout tone frequency f_r and readout power p_r, which directly affect readout SNR and state discrimination. We model the system as $\mathcal{S} = (\mathcal{Q}, \mathcal{P})$, where $\mathcal{Q}$ denotes the quantum hardware (chip and readout structure) and $\mathcal{P}$ denotes the calibration parameters. In this work, $\mathcal{P}$ is restricted to $\{f_r, p_r\}$.

Because suitable values cannot be reliably determined from analytical device models alone, they must be obtained through measurement. We formulate readout calibration as a finite budget sequential decision process. At each iteration, the workflow observes a summary of previous measurement responses and selects an executable sweep action. The action space is constrained. The frequency module sets sweep center, span, and point count. The power module sets the power range and step size while the local frequency sweep remains fixed. The workflow terminates when a convergence criterion is met or when its iteration limit is reached, and returns the best available estimates of f_r and p_r as operating points for downstream calibration stages.

3.2 Workflow Graph

Graph Abstraction. We organize calibration as an executable finite-state directed graph driven by iterative measurement feedback (Algorithm 1). Its modular nodes use dependency-aware transitions and follow the loop: *sweep* $\rightarrow$ *summarize observations* $\rightarrow$ *update* $\rightarrow$ *stop or continue*. In implementation, a software simulator returns responses for the requested sweep configurations. Beyond implementation, the finite-state structure encodes the physical dependency between readout frequency

and readout power: meaningful power search requires a localized frequency reference. This prevents invalid parameter combinations and makes transitions auditable. At the iteration limit, the workflow returns its best available estimate.

Nodes and Control Flow. The graph contains two executable agents. FreqAgent executes Stage I and updates $\hat{f}_{\mathrm{res}}$ together with associated measurement summaries. PowerAgent executes Stage II conditioned on the output of Stage I, and updates $(\hat{f}_{\mathrm{dress}}, \hat{p}_r)$ accordingly during execution.

Loops and Termination. Edges encode conditional loops and stage transitions:

$$\text{Start} \to \text{FreqAgent}$$

$$\text{FreqAgent} \to \begin{cases} \text{FreqAgent}, & \text{if } \texttt{done} = \texttt{false}, \\ \text{PowerAgent}, & \text{if } \texttt{done} = \texttt{true}, \end{cases}$$

$$\text{PowerAgent} \to \begin{cases} \text{PowerAgent}, & \text{if } \texttt{done} = \texttt{false}, \\ \text{End}, & \text{if } \texttt{done} = \texttt{true}. \end{cases}$$

The workflow terminates when $\texttt{phase}=$ End or when its iteration budget is exhausted. In both cases, it returns the best available estimates of $\hat{f}_{\mathrm{res}}$, $\hat{f}_{\mathrm{dress}}$ (if identified), and $\hat{p}_r$ as operating points for downstream calibration stages.

3.3 Agent Design

Terminology. We retain the names FreqAgent and PowerAgent to match the reference implementation and figure labels. An agent denotes a stage-specific planner wrapper with structured observations and typed actions. The runtime controls execution, validation, and stage transitions. The experiments evaluate the complete bounded workflow. Isolating the effect of the stage decomposition requires a separate comparison.

Algorithm 1: Executable two-agent workflow.

Require: Budget B
Ensure: $\hat{f}_{\mathrm{res}}, \hat{f}_{\mathrm{dress}}, \hat{p}_r$
Initialize $\mathcal{S}$ with phase *Freq*
while $\mathcal{S}.\mathrm{phase} \neq$ *End* and $\mathcal{S}.\mathrm{budget} > 0$ **do**
  if $\mathcal{S}.\mathrm{phase} =$ *Freq* **then**
    $\mathcal{S} \leftarrow \Phi_{\textit{FreqExpert}}(\mathcal{S})$
    if $\mathcal{S}.\mathrm{done}$ **then**
      $\mathcal{S}.\mathrm{phase} \leftarrow$ *Power*
    end if
  else
    $\mathcal{S} \leftarrow \Phi_{\textit{PowerExpert}}(\mathcal{S})$
    if $\mathcal{S}.\mathrm{done}$ **then**
      $\mathcal{S}.\mathrm{phase} \leftarrow$ *End*
    end if
  end if
  $\mathcal{S}.\mathrm{budget} \leftarrow \mathcal{S}.\mathrm{budget} - 1$
end while
return $(\hat{f}_{\mathrm{res}}, \hat{f}_{\mathrm{dress}}, \hat{p}_r)$

Role Decomposition. The calibration procedure follows a strictly ordered two-stage sequence. In Stage I, the system identifies the readout resonator frequency by localizing the resonator dip and obtaining $\hat{f}_{\mathrm{res}}$ through iterative frequency sweeps. Stage II is then performed conditioned on $\hat{f}_{\mathrm{res}}$: the system searches within a narrow neighborhood around $\hat{f}_{\mathrm{res}}$ to estimate the dispersively shifted (dressed) resonator frequency $\hat{f}_{\mathrm{dress}}$ under different readout powers, and selects the corresponding readout power $\hat{p}_r$ at a stable operating point.

Frequency Expert. The frequency expert aims to obtain a reliable estimate of the readout resonance $\hat{f}_{\mathrm{res}}$. At each iteration, the planner receives fit summaries such as dip location, response contrast, and residual magnitude. It proposes the next sweep center, span, point count, and stop flag. The runtime applies numerical guards, executes the sweep, and updates $\hat{f}_{\mathrm{res}}$ from the fitted response. Low confidence can trigger a wider sweep through the deterministic guards. The runtime checks the convergence criterion, the stop flag, and the iteration limit before continuing the measurement loop.

Power Expert. Conditioned on $\hat{f}_{\mathrm{res}}$, the power expert searches for a stable readout power $\hat{p}_r$ and estimates the power-dependent dressing effect, denoted as $f_{\mathrm{dress}}(p)$ or an equivalent frequency shift summary. At each iteration, the planner receives power-sweep summaries and proposes a power window, step size, and stop flag. The runtime clamps the numerical sweep settings, executes the sweep, and ranks fitted candidates using a fixed scoring rule. At termination, it returns $\hat{p}_r$ and the corresponding $\hat{f}_{\mathrm{dress}}$ from the retained candidates.

3.4 RETRIEVAL-BASED EMPIRICAL PRIOR

Different chip designs can have different offsets between the nominal readout frequency and the resonator frequency. To study an empirical initialization prior, we construct a retrieval database from simulated chip records. From a pool of approximately 3000 simulated chip designs (see § 4.1 for details), we filter out physically unreasonable samples and uniformly sample 200 valid chips to form the database. Each entry stores a scalar readout geometry descriptor $L_{\mathrm{RO},i}$, the design readout frequency $f_{r,i}^{\mathrm{design}}$, the simulated frequency offset Δf_i in MHz, and the resulting shifted frequency $f_{r,i}^{\mathrm{shifted}}$, related by

$$f_{r,i}^{\mathrm{shifted}} = f_{r,i}^{\mathrm{design}} + \Delta f_i/1000,$$

where the division converts MHz to GHz. This database supplies an empirical prior from previously simulated chips.

Retrieval and Offset Prediction. Given an unseen chip specified by $(L_{\mathrm{RO}}, f_r^{\mathrm{design}})$, we retrieve the top-$K$ nearest entries and apply a weighted kNN regression to predict the frequency offset $\widehat{\Delta f}$:

$$\begin{aligned} d_i^2 &= \left(\frac{L_{\mathrm{RO}} - L_{\mathrm{RO},i}}{s_L}\right)^2 + \left(\frac{f_r^{\mathrm{design}} - f_{r,i}^{\mathrm{design}}}{s_f}\right)^2, \\ w_i &= \frac{1}{\varepsilon + d_i^2}, \qquad \widehat{\Delta f} = \frac{\sum_{i\in\mathcal{N}_K} w_i\, \Delta f_i}{\sum_{i\in\mathcal{N}_K} w_i}. \end{aligned} \tag{1}$$

where $\mathcal{N}_K$ denotes the retrieved neighbor set, s_L and s_f are scaling factors, and ε is a small constant for numerical stability. The predicted offset centers the initial spectroscopy window via

$$\hat{f}_r^{\mathrm{init}} = f_r^{\mathrm{design}} + \widehat{\Delta f}/1000,$$

again converting the predicted offset from MHz to GHz and reducing the search burden on Stage I.

In the retrieval variant, the prior centers the initial Stage I sweep on a predicted frequency region. In Appendix A.4.3, we vary its size while keeping the evaluation set fixed. The 200-record configuration is a fixed operating point chosen to provide a small memory bank while retaining most of the filtered design pool for evaluation. The size study reports how this choice affects initialization and downstream calibration quality.

The retrieval database and evaluation set contain disjoint chip instances from the same simulated design-generation pipeline. The experiments evaluate held-out performance within this pipeline.

3.5 EXECUTION CONSTRAINTS AND REPRODUCIBILITY

Coordination is centralized by the finite-state runtime, which enforces allowed transitions and serializes node execution. GPT-4o mini proposes actions through the `gpt-4o-mini` interface. Python validates their structure and applies numerical sweep constraints before execution. The runtime controls execution through three mechanisms: (i) a constrained action space limited to sweep parameters and stopping decisions, (ii) schema validation, numerical guards, and a deterministic fallback execution path after request or parsing failures, and (iii) bounded iteration that returns the best available estimate at termination. The matched-budget frequency variant also caps the total number of measurement points.

At each planning step, the frequency module receives a chip descriptor and the latest fit diagnostics. It proposes a scan center, span, point count, and stop flag. The power module receives the frequency setting, its numerical constraints, the configured simulator noise level, and the latest power-fit summary. It proposes a power interval, step size, and stop flag. Each call uses a fresh system message and JSON payload with temperature set to 0.0. The free-text reason is logged, while Python executes only the typed action fields. If a request or parsing step fails, the stage uses its deterministic fallback action for that iteration. This interface identifies the decisions delegated to GPT-4o mini within the numerical calibration loop.

Agent traces contain proposed actions, guard adjustments, measurement summaries, and stopping decisions. Appendix C specifies the model interface, decoding settings, exact prompts, payloads, schemas, and fallback execution path.

Table 1: End-to-End readout quality under the high noise stress setting $\mathrm{nl} = 0.09$. All methods share the same downstream evaluation pipeline after the initial frequency and power decisions. SNR (↑ higher is better); $p_{0\to1}$, $p_{1\to0}$ (↓ lower is better); σ_0, σ_1 (↓ lower is better).

Method	IQ-Chain					Chain					Realistic-Chain				
	SNR	$p_{0\to1}$	$p_{1\to0}$	σ_0	σ_1	**SNR**	$p_{0\to1}$	$p_{1\to0}$	σ_0	σ_1	**SNR**	$p_{0\to1}$	$p_{1\to0}$	σ_0	σ_1
NE24_v1	9.79	0.0483	0.0423	0.2386	0.2435	9.10	0.0530	0.0457	0.2701	0.2756	8.99	0.0582	0.0493	0.2845	0.2903
NE24_v2	8.98	0.0632	0.0553	0.3233	0.3300	8.61	0.0650	0.0563	0.3244	0.3310	9.49	0.0549	0.0464	0.2612	0.2665
QA-Google_v1	10.55	0.0286	0.0254	0.1270	0.1294	9.81	0.0313	0.0259	0.1324	0.1349	10.42	0.0271	0.0235	0.1201	0.1224
QA-Google_v2	10.07	0.0331	0.0280	0.1405	0.1433	9.62	0.0370	0.0310	0.1487	0.1515	10.31	0.0284	0.0246	0.1278	0.1301
QE-IBM_v1	7.90	0.0396	0.0344	0.1745	0.1777	8.25	0.0392	0.0333	0.1702	0.1734	7.47	0.0384	0.0331	0.1665	0.1695
QE-IBM_v2	7.43	0.0466	0.0392	0.2004	0.2042	7.86	0.0411	0.0361	0.1827	0.1860	8.03	0.0424	0.0367	0.1862	0.1897
MAQCal	**11.47**	**0.0237**	**0.0192**	**0.1127**	**0.1152**	**11.09**	**0.0232**	**0.0173**	**0.0967**	**0.0986**	**12.01**	**0.0247**	**0.0203**	**0.1125**	**0.1150**

4 Experiments and Results

We evaluate whether MAQCal improves readout quality over predefined sweep-and-fit configurations in simulation. All methods use the same downstream evaluation pipeline under three synthetic noise families. The comparison tests whether choosing sweep parameters from measurement feedback improves on fixed presets within the simulated chip distribution.

4.1 Experimental Setup

Chip Dataset and Split. We generate 3000 candidate chips using Qiskit Metal and electromagnetic simulation, then retain 2006 designs after physical and engineering filters. We uniformly sample 200 chips for the retrieval database in Section 3.4 and evaluate on the remaining 1806. The two sets contain disjoint chip instances from the same generated design distribution. Appendix A.1 describes the generation and filtering procedure, with an example layout in Appendix B.1.

Noise Settings. We use *IQ_Chain* (white noise with occasional spikes), *Chain* (random walk drift with spikes), and *Realistic_Chain* (white noise, drift, and spikes). For each family, we sweep $n \in \{0.00, 0.01, \ldots, 0.10\}$ to test increasing perturbation strengths.

4.2 Overall Evaluation

End-to-End Protocol. Following Section 3.2, Agent1 selects the readout frequency in Step 1 and Agent2 selects power with dressed frequency tracking in Step 2. F-CircleSlope extracts frequency from complex transmission $S_{21}(f)$. P-ParetoFit uses local spectra $S_{21}(f; P)$ at successive power settings around the selected frequency. All methods share the same fixed reference settings for Steps 3 through 8, so the comparison isolates their initial frequency and power decisions. The common fitting routines and evaluation protocol are detailed in Appendix A.1, with an execution trace in Appendix B.2. Retrieval initialization is evaluated separately in Appendix A.4.3.

Baselines. We compare against six fixed presets using the same F-CircleSlope → P-ParetoFit solver sequence and downstream pipeline. NE24, QA-Google, and QE-IBM each have two configurations that vary the frequency span and resolution, power range, and power step. These labels identify scan presets implemented for this evaluation. The complete settings are listed in Appendix A.2. This comparison measures sensitivity to scan parameters selected before measurement.

Evaluation Metrics. After completing Step 1 through Step 8, we compute five readout quality metrics: (i) **SNR** (higher is better), (ii) $p_{0\to1}$ (false positive probability for $|0\rangle$; lower is better), (iii) $p_{1\to0}$ (false negative probability for $|1\rangle$; lower is better), (iv) σ_0 (IQ cloud spread for $|0\rangle$; lower is better), and (v) σ_1 (IQ cloud spread for $|1\rangle$; lower is better). Concretely, let $s \in \{0, 1\}$ denote the prepared state, and let $\{\mathbf{x}_i^{(s)}\}_{i=1}^{n_s} \subset \mathbb{R}^2$ be the corresponding single shot IQ samples with empirical mean μ_s and covariance Σ_s. We define

$$\mathrm{SNR} = \frac{\|\mu_1 - \mu_0\|_2}{\sqrt{\mathrm{tr}(\Sigma_0)} + \sqrt{\mathrm{tr}(\Sigma_1)}}, \qquad \sigma_s = \sqrt{\frac{1}{2}\,\mathrm{tr}(\Sigma_s)}. \tag{2}$$

We obtain $p_{0\to1}$ and $p_{1\to0}$ using a standard linear discriminant decision boundary on the IQ plane, with a fixed calibration rule and identical fitting code across all methods.

Overall Results ($nl = 0.09$). Table 1 reports end-to-end readout quality under a high noise stress setting ($nl = 0.09$) across three noise families. Because all methods share the same downstream evaluation pipeline after the initial readout decisions, observed differences reflect the quality of the frequency and power choices made in Step 1 and Step 2. Among the compared methods, MAQCal achieves the highest SNR and the lowest assignment errors and IQ cloud spreads across all three noise families. Relative to the strongest fixed sweep baseline (QA-Google_v1), MAQCal improves SNR by roughly 9% on IQ-Chain (11.47 vs. 10.55), 13% on Chain (11.09 vs. 9.81), and 15% on Realistic-Chain (12.01 vs. 10.42). Across the three families, the aggregate gap grows with noise complexity, with the largest gains under Realistic-Chain, which combines drift and spikes. The baselines themselves show substantial variation across configurations. For example, under Realistic-Chain, QA-Google_v1 achieves an SNR of 10.42 while QE-IBM_v2 reaches only 8.03 despite both using the same fitting routines. This illustrates the sensitivity of predefined sweep scripts to their scan hyperparameters and motivates the core argument: within the evaluated simulator, MAQCal reduces sensitivity to the preset selected before measurement by choosing sweep settings from intermediate responses. Full results at other noise levels are in Appendix A.3.

The improvements also appear in assignment metrics. Under Realistic-Chain noise, MAQCal has $p_{0\to1} = 0.0247$ and $p_{1\to0} = 0.0203$, compared with 0.0271 and 0.0235 for QA-Google_v1. Its IQ cloud spreads are 0.1125 and 0.1150, compared with 0.1201 and 0.1224 for that preset. Under Chain noise, the corresponding error probabilities fall from 0.0313 and 0.0259 to 0.0232 and 0.0173. The SNR gains therefore coincide with lower assignment errors and narrower IQ clouds in the same high-noise setting. All five metrics use the common downstream pipeline after the selected frequency and power are passed forward.

4.3 Ablation Studies

The overall results combine both stages. For consistency with the implementation and plotted legends, Agent1 and Agent2 denote the Stage I frequency planner and Stage II power planner, respectively. The following experiments evaluate each stage with its runtime guards and fitting routines.

4.3.1 Frequency Localization (Step 1)

Setup. We isolate **Step 1** and evaluate **Agent1 (MAQCal)** against the same fixed sweep baselines. All methods are run end to end under multiple noise types and noise levels using the same evaluation protocol. This ablation compares frequency localization at the first decision stage, before downstream steps are executed.

Metric. We report the mean absolute error (MAE) between the estimated resonator frequency $\hat{f}_{\mathrm{res}}$ and the ground-truth frequency f_{res}, defined as $\mathrm{MAE} = \mathrm{E}\big[\,|\hat{f}_{\mathrm{res}} - f_{\mathrm{res}}|\,\big]$.

Results. Figure 3 shows higher frequency MAE for the fixed sweep baselines as noise increases in the reported settings. The values also differ across baseline configurations: under Chain noise at $nl = 0.09$, the best fixed sweep baseline (NE24_v1) achieves a MAE of 9.59 MHz while the worst (QE-IBM_v2) reaches 20.64 MHz. In contrast, Agent1 maintains a MAE below 2.50 MHz across the entire noise range, with only mild high-noise degradation.

This pattern is consistent with the frequency module using previous responses to allocate measurements, while fixed windows can be poorly aligned with the resonance under some noise settings.

4.3.2 Power Selection (Step 2)

Setup. We further isolate **Step 2** (power selection) and evaluate **Agent2 (MAQCal)** against the same baselines. To remove confounding factors from frequency exploration, all methods receive the same high-quality frequency operating point and the same fixed 30 MHz local spectroscopy span in Step 2. Methods differ in their power-selection policy: the fixed baselines use preset power windows and step sizes, whereas Agent2 adaptively chooses the power range and step size based on feedback from previous sweeps.

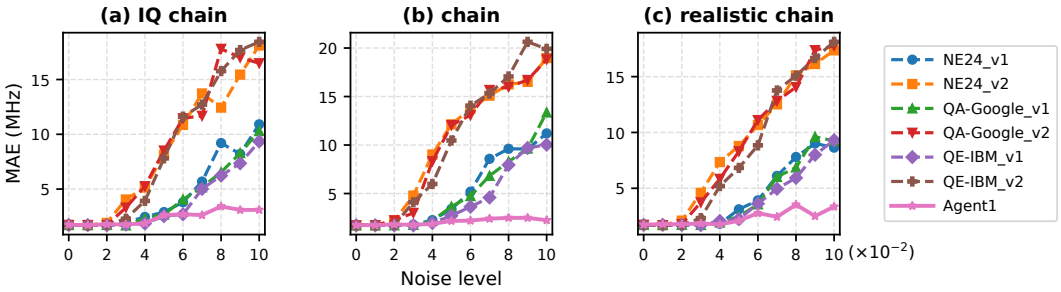

Figure 3: Frequency localization under increasing noise. MAE of the estimated resonance frequency $\hat{f}_{\mathrm{res}}$ versus noise level across three noise families. The plotted aggregate values show lower error for Agent1 than the fixed presets in the reported settings.

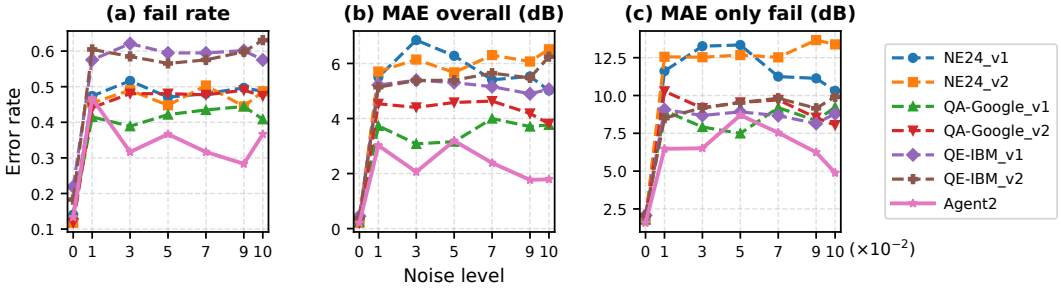

Figure 4: Power selection under Chain noise. (a) Failure rate of punchout power selection. (b) MAE of the returned readout power. (c) MAE conditioned on failed runs. The plotted aggregate values show lower failure frequency and severity for Agent2 in the reported settings.

Metrics. Under *Chain* noise at levels $n \in \{0, 1, 3, 5, 7, 9, 10\} \times 10^{-2}$, Figure 4 reports the power-selection failure rate, overall power-decision MAE, and MAE restricted to failed runs. The power errors are in dB.

Results. As shown in Figure 4, the fixed baselines degrade noticeably as noise increases: both fail rate and power decision error rise, with uneven trends across noise levels. Agent2 has a lower failure rate and error in these settings. At $nl = 0.09$, Agent2 achieves a fail rate of 28% compared to 44% for the best fixed baseline (QA-Google_v1), and an overall MAE of 1.77 dB compared to 3.71 dB. The **MAE (only fail)** is also lower for Agent2 in the reported settings.

4.4 Extended Experimental Analyses

We evaluate three auxiliary settings, with detailed protocols and results in Appendix A.4. The scan-budget experiment holds noise at $n = 0.05$ and gives each method an identical number of frequency samples. Increasing uniform scan density does not reliably improve the fixed baselines in the tested settings. MAQCal maintains lower aggregate localization error across the reported budgets and noise families. The drift experiment runs $T = 20$ calibration rounds with Gaussian increments at $\sigma_{\mathrm{drift}} \in \{0.2, 0.5, 1.0\}$ MHz. All methods start from the same initial condition. MAQCal then uses its previous calibration output as a prior and updates each scan from new feedback. The fixed baselines repeat their predefined windows. The aggregate results show lower regret and higher SNR for MAQCal in these settings.

The retrieval experiment varies the number of stored records over $K_{\mathrm{mem}} \in \{0, 50, 100, 200\}$ and keeps the evaluation set fixed. Each nonzero variant samples its records uniformly from the original database. Without retrieval, Stage I starts from the nominal design frequency. Larger databases reduce the initial frequency-offset error and improve downstream calibration quality in this simulation split. This comparison measures the contribution of stored simulated chip records to initialization under noise. These auxiliary evaluations cover the same simulated design distribution and fixed baseline families as the main experiments.

5 DISCUSSION

The results support feedback-driven sweep selection in the two evaluated readout stages. Under $nl = 0.09$, the SNR improvement over the strongest fixed preset is approximately 9% on IQ-Chain, 13% on Chain, and 15% on Realistic-Chain. The stage-wise experiments locate gains in both decisions. Under Chain noise at this level, Agent1 has a frequency MAE below 2.50 MHz, compared with 9.59 MHz for the best fixed preset. With a common frequency input, Agent2 reduces power-selection failures from 44% to 28% and power MAE from 3.71 to 1.77 dB.

The matched frequency-budget experiment shows lower localization error when sample counts are held equal. The retrieval study examines initialization separately. Under Chain noise at $nl = 0.10$, frequency-offset MAE is approximately 5.7 MHz without retrieval and 2.3 MHz with 200 stored records. The largest change occurs between zero and 50 records. At low noise, all four memory sizes have MAE near 1.7 to 2.0 MHz. The prior helps place the first sweep before feedback from the current chip is available.

The drift test extends the workflow over 20 calibration rounds. At a drift scale of 0.5 MHz, MAQCal obtains a final SNR of 11.79, compared with 6.73 for the best fixed preset, and median cumulative regret of 30.47, compared with 75.27. The workflow carries its previous result into the next round and updates sweeps from new measurements. Appendix C documents the planner proposals, host constraints, and fallback execution path underlying these decisions.

Scope of Empirical Claim. The empirical claim is bounded to the stated simulation protocol. The evaluation uses 2006 filtered chip designs generated by one simulation pipeline, with 200 records for retrieval and 1806 held-out chips from the same design distribution, three synthetic noise families, the reported noise levels, and six fixed presets. All methods share the downstream processing settings, and the matched-budget experiment controls the number of frequency samples. These results support improvement of the complete workflow over the tested fixed presets under this protocol. A2 receives the configured simulator noise level as an input, as detailed in Appendix C.2. The evidence therefore assumes access to this metadata.

The separate contributions of the LLM planner, deterministic guards, and final candidate scoring remain unresolved by these comparisons. A matched comparison with a rule-based adaptive controller is a priority for future evaluation, using the same observations, priors, action constraints, and budgets. The fallback path documented in Appendix C.3 provides an executable starting point for that comparison. Behavior on real hardware, other device-generation pipelines, and further calibration stages also requires separate evaluation. Calibration time, model-call overhead, and reductions in operator effort remain to be measured.

6 CONCLUSION

We presented MAQCal, a constrained two-stage workflow for simulation-based quantum readout calibration. The workflow separates planning from deterministic execution through typed sweep actions. Its finite-state runtime coordinates frequency localization and power selection, transfers the localized frequency between stages, and enforces numerical guards and iteration limits. We separately evaluate a weighted kNN initialization prior using 200 simulated chip records. Across three noise families, the mean of the 15 setting-specific relative SNR gains over the strongest fixed preset in each setting is 13.7%. The stage-wise and matched-budget results show lower errors in the evaluated frequency and power decisions.

MAQCal uses new measurement summaries to select successive frequency and power sweeps. The design makes the ordered decisions, validation, and termination explicit. At equal frequency-sample counts, its localization error remains below the fixed presets across the tested budgets. The simulated drift test examines this feedback sequence over repeated calibration rounds. The separate retrieval experiment shows how stored chip records improve initialization. Together, these results support a bounded workflow that automates sweep selection and improves readout quality over the tested fixed presets under matched downstream processing in simulation.

AI USE STATEMENT

GPT-4o mini is used as the stage-specific planner inside the MAQCal agents, as described in Section 3.3 and Appendix C. AI-assisted tools were also used to inspect implementation details and assist with manuscript editing. The authors retain responsibility for the experimental results and the accuracy of the manuscript.

REFERENCES

Fabrizio Berritta, Jacob Benestad, Lukas Pahl, Melvin Mathews, Jan A Krzywda, Réouven Assouly, Youngkyu Sung, David K Kim, Bethany M Niedzielski, Kyle Serniak, et al. Efficient qubit calibration by binary-search hamiltonian tracking. *PRX Quantum*, 6(3):030335, 2025. doi: 10.1103/77qg-p68k.

Alexandre Blais, Ren-Shou Huang, Andreas Wallraff, Steven M Girvin, and R. J. Schoelkopf. Cavity quantum electrodynamics for superconducting electrical circuits: An architecture for quantum computation. *Physical Review A—Atomic, Molecular, and Optical Physics*, 69(6):062320, 2004. doi: 10.1103/physreva.69.062320.

Jonathan J Burnett, Andreas Bengtsson, Marco Scigliuzzo, David Niepce, Marina Kudra, Per Delsing, and Jonas Bylander. Decoherence benchmarking of superconducting qubits. *npj Quantum Information*, 5(1):54, 2019. doi: 10.1038/s41534-019-0168-5.

Sitan Chen, Jordan Cotler, Hsin-Yuan Huang, and Jerry Li. The complexity of nisq. *Nature Communications*, 14(1):6001, 2023. doi: 10.1038/s41467-023-41217-6.

Jay Gambetta, WA Braff, A Wallraff, SM Girvin, and RJ Schoelkopf. Protocols for optimal readout of qubits using a continuous quantum nondemolition measurement. *Physical Review A—Atomic, Molecular, and Optical Physics*, 76(1):012325, 2007. doi: 10.1103/physreva.76.012325.

Yvonne Y Gao, M Adriaan Rol, Steven Touzard, and Chen Wang. Practical guide for building superconducting quantum devices. *PRX Quantum*, 2(4):040202, 2021. doi: 10.1103/prxquantum.2.040202.

Johannes Heinsoo, Christian Kraglund Andersen, Ants Remm, Sebastian Krinner, Theodore Walter, Yves Salathé, Simone Gasparinetti, Jean-Claude Besse, Anton Potočnik, Andreas Wallraff, et al. Rapid high-fidelity multiplexed readout of superconducting qubits. *Physical Review Applied*, 10 (3):034040, 2018. doi: 10.1103/physrevapplied.10.034040.

Evan Jeffrey, Daniel Sank, JY Mutus, TC White, J Kelly, R Barends, Y Chen, Z Chen, B Chiaro, A Dunsworth, et al. Fast accurate state measurement with superconducting qubits. *Physical review letters*, 112(19):190504, 2014. doi: 10.1103/physrevlett.112.190504.

Naoki Kanazawa, Daniel J Egger, Yael Ben-Haim, Helena Zhang, William E Shanks, Gadi Aleksandrowicz, and Christopher J Wood. Qiskit experiments: A Python package to characterize and calibrate quantum computers. *Journal of Open Source Software*, 8(84):5329, 2023. doi: 10.21105/joss.05329.

Julian Kelly, Rami Barends, Austin G Fowler, Anthony Megrant, Evan Jeffrey, Theodore C White, Daniel Sank, Josh Y Mutus, Brooks Campbell, Yu Chen, et al. State preservation by repetitive error detection in a superconducting quantum circuit. *Nature*, 519(7541):66–69, 2015. doi: 10.1038/nature14270.

Julian Kelly, Peter O'Malley, Matthew Neeley, Hartmut Neven, and John M Martinis. Physical qubit calibration on a directed acyclic graph. *arXiv preprint arXiv:1803.03226*, 2018. doi: 10.48550/arXiv.1803.03226.

Navin Khaneja, Timo Reiss, Cindie Kehlet, Thomas Schulte-Herbrueggen, and Steffen J. Glaser. Optimal control of coupled spin dynamics: Design of nmr pulse sequences by gradient ascent algorithms. *Journal of Magnetic Resonance*, 172(2):296–305, 2005. doi: 10.1016/j.jmr.2004.11.004.

Morten Kjaergaard, Mollie E Schwartz, Jochen Braumüller, Philip Krantz, Joel I-J Wang, Simon Gustavsson, and William D Oliver. Superconducting qubits: Current state of play. *Annual Review of Condensed Matter Physics*, 11(1):369–395, 2020. doi: 10.1146/annurev-conmatphys-031119-050605.

Philip Krantz, Morten Kjaergaard, Fei Yan, Terry P Orlando, Simon Gustavsson, and William D Oliver. A quantum engineer's guide to superconducting qubits. *Applied physics reviews*, 6(2): 021318, 2019. doi: 10.1063/1.5089550.

Andres M. Bran, Sam Cox, Oliver Schilter, Carlo Baldassari, Andrew D White, and Philippe Schwaller. Augmenting large language models with chemistry tools. *Nature Machine Intelligence*, 6(5):525–535, 2024. doi: 10.1038/s42256-024-00832-8.

Hailan Ma, Bo Qi, Ian R Petersen, Re-Bing Wu, Herschel Rabitz, and Daoyi Dong. Machine learning for estimation and control of quantum systems. *National Science Review*, 12(8):nwaf269, 2025. doi: 10.1093/nsr/nwaf269.

Matthias M Müller, Ressa S Said, Fedor Jelezko, Tommaso Calarco, and Simone Montangero. One decade of quantum optimal control in the chopped random basis. *Reports on progress in physics*, 85(7):076001, 2022. doi: 10.1088/1361-6633/ac723c.

Prakash Murali, Dripto M Debroy, Kenneth R Brown, and Margaret Martonosi. Architecting noisy intermediate-scale trapped ion quantum computers. In *2020 ACM/IEEE 47th Annual International Symposium on Computer Architecture (ISCA)*, pp. 529–542. IEEE, 2020. doi: 10.1109/isca45697.2020.00051.

Andrea Pasquale, Edoardo Pedicillo, Juan Cereijo, Sergi Ramos-Calderer, Alessandro Candido, Gabriele Palazzo, Rodolfo Carobene, Marco Gobbo, Stavros Efthymiou, Yuanzheng Paul Tan, Ingo Roth, Matteo Robbiati, Jadwiga Wilkens, Alvaro Orgaz-Fuertes, David Fuentes-Ruiz, Andrea Giachero, Frederico Brito, José Ignacio Latorre, and Stefano Carrazza. Qibocal: an open-source framework for calibration of self-hosted quantum devices. *arXiv preprint arXiv:2410.00101*, 2024. doi: 10.48550/arXiv.2410.00101.

John Preskill. Quantum computing in the nisq era and beyond. *Quantum*, 2:79, 2018. doi: 10.22331/q-2018-08-06-79.

Daniel Sank, Alex Opremcak, Andreas Bengtsson, Mostafa Khezri, Jimmy Chen, Ofer Naaman, and Alexander Korotkov. System characterization of dispersive readout in superconducting qubits. *Physical Review Applied*, 23(2):024055, 2025. doi: 10.1103/physrevapplied.23.024055.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. Toolformer: Language models can teach themselves to use tools. *Advances in neural information processing systems*, 36:68539–68551, 2023. doi: 10.52202/075280-2997.

Noah Shinn, Federico Cassano, Beck Labash, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. *arXiv preprint arXiv:2303.11366v3*, 2023. URL `https://arxiv.org/abs/2303.11366v3`.

Andreas Wallraff, David I Schuster, Alexandre Blais, Luigi Frunzio, R-S Huang, Johannes Majer, Sameer Kumar, Steven M Girvin, and Robert J Schoelkopf. Strong coupling of a single photon to a superconducting qubit using circuit quantum electrodynamics. *Nature*, 431(7005):162–167, 2004. doi: 10.1038/nature02851.

Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar. Voyager: An open-ended embodied agent with large language models. *arXiv preprint arXiv:2305.16291*, 2023. doi: 10.48550/arXiv.2305.16291.

David F Wise, John JL Morton, and Siddharth Dhomkar. Using deep learning to understand and mitigate the qubit noise environment. *PRX Quantum*, 2(1):010316, 2021. doi: 10.1103/prxquantum.2.010316.

Nicolas Wittler, Federico Roy, Kevin Pack, Max Werninghaus, Anurag Saha Roy, Daniel J Egger, Stefan Filipp, Frank K Wilhelm, and Shai Machnes. Integrated tool set for control, calibration, and characterization of quantum devices applied to superconducting qubits. *Physical Review Applied*, 15(3):034080, 2021. doi: 10.1103/physrevapplied.15.034080.

Hui Yang, Sifu Yue, and Yunzhong He. Auto-gpt for online decision making: Benchmarks and additional opinions. *arXiv preprint arXiv:2306.02224*, 2023. doi: 10.48550/arXiv.2306.02224.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Thomas L Griffiths, Yuan Cao, and Karthik Narasimhan. Tree of Thoughts: Deliberate problem solving with large language models. In *Advances in Neural Information Processing Systems*, volume 36, pp. 11809–11822, 2023a. doi: 10.52202/075280-0517.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. ReAct: Synergizing reasoning and acting in language models. In *The Eleventh International Conference on Learning Representations*, 2023b. URL https://openreview.net/forum?id=WE_vluYUL-X.

Justyna P Zwolak and Jacob M Taylor. Colloquium: Advances in automation of quantum dot devices control. *Reviews of modern physics*, 95(1):011006, 2023. doi: 10.1103/RevModPhys.95.011006.

A Additional Experimental Details and Results

A.1 Simulation and Evaluation Protocol

We generate parametric chip layouts with Qiskit Metal and use Ansys HFSS (EPR) and Ansys Q3D (LOM) for electromagnetic analysis and parameter extraction. Each chip contains a hanger-type resonator coupled to a feedline through an interdigital tee coupler. We vary the layout under a fixed circuit topology and simulate 3000 candidate chips. Physical and engineering filters remove designs with excessively small external linewidth κ_{out}, insufficient coupling strength g, or nonphysical outcomes. This procedure retains 2006 valid designs. We uniformly sample 200 for the frequency-offset retrieval database and use the remaining 1806 for evaluation. The split is disjoint at the chip level and tests held-out instances from the same simulation pipeline. Appendix B.1 shows an example chip.

The common fitting sequence consists of ***F-CircleSlope*** and ***P-ParetoFit***. F-CircleSlope extracts the readout frequency from complex transmission $S_{21}(f)$ measured in a sweep around the readout band. P-ParetoFit then sweeps drive power P after fixing f_{r}. At each power level, it acquires a local frequency sweep $S_{21}(f; P)$ around f_{r} for power selection. All six fixed presets use this sequence, with scan settings specified in Appendix A.2. Retrieval initialization is evaluated in Appendix A.4.3.

In the end-to-end protocol, Agent1 (FreqAgent) controls resonator spectroscopy and frequency localization in Step 1. Agent2 (PowerAgent) controls the power scan and dressed frequency tracking in Step 2. Steps 3 through 8 cover the downstream readout-control and evaluation stages. Every method uses identical, fixed high-quality reference settings for those stages. This controls their contribution to measured readout quality and isolates the initial frequency and power decisions. Appendix B.2 provides a representative execution trace of the full pipeline.

A.2 Baseline Configurations

The six presets differ in manually selected scan hyperparameters. Their frequency windows, point counts, power ranges, and steps are listed in Table 2. These labels are shorthand for the fixed configurations implemented in the simulator. Every preset uses the same fitting routines and downstream evaluation settings described in Appendix A.1.

Table 2: Full configurations of predefined sweep-and-fit baselines.

Method	**Frequency sweep**	**Power range (dBm)**	**Power step**
NE24_v1	50 MHz / 201 points	$[-150, -100]$	$1dB$
NE24_v2	50 MHz / 401 points	$[-150, -100]$	$2dB$
QA-Google_v1	60 MHz / 201 points	$[-150, -110]$	$1dB$
QA-Google_v2	60 MHz / 401 points	$[-150, -110]$	$2dB$
QE-IBM_v1	70 MHz / 201 points	$[-140, -110]$	$1dB$
QE-IBM_v2	70 MHz / 401 points	$[-140, -110]$	$2dB$

A.3 Additional End-to-End Results

Table 3: End-to-end readout quality under noise setting $\mathrm{nl} = 0.01$.

Method	**IQ-Chain**					**Chain**					**Realistic-Chain**				
	SNR	$p_{0\to1}$	$p_{1\to0}$	σ_0	σ_1	**SNR**	$p_{0\to1}$	$p_{1\to0}$	σ_0	σ_1	**SNR**	$p_{0\to1}$	$p_{1\to0}$	σ_0	σ_1
NE24_v1	10.14	0.0416	0.0349	0.1953	0.1992	9.41	0.0549	0.0457	0.2723	0.2781	9.27	0.0530	0.0451	0.2541	0.2594
NE24_v2	9.96	0.0459	0.0388	0.2232	0.2278	9.11	0.0587	0.0496	0.2845	0.2905	9.46	0.0564	0.0465	0.2642	0.2697
QA-Google_v1	10.08	0.0362	0.0296	0.1447	0.1475	10.82	0.0325	0.0260	0.1280	0.1305	10.17	0.0361	0.0283	**0.1423**	**0.1451**
QA-Google_v2	10.50	0.0301	0.0239	0.1248	0.1273	10.45	0.0414	0.0336	0.1641	0.1673	10.08	0.0363	0.0291	0.1480	0.1509
QE-IBM_v1	8.44	0.0400	0.0330	0.1685	0.1718	8.60	0.0428	0.0340	0.1687	0.1719	8.15	0.0419	0.0345	0.1742	0.1776
QE-IBM_v2	8.52	0.0404	0.0326	0.1687	0.1720	8.33	0.0406	0.0338	0.1683	0.1715	8.31	0.0403	0.0340	0.1729	0.1762
MAQCal	**12.96**	**0.0248**	**0.0200**	**0.0906**	**0.0925**	**11.67**	**0.0227**	**0.0188**	**0.1013**	**0.1034**	**11.74**	**0.0317**	**0.0226**	0.1577	0.1613

Table 4: End-to-end readout quality under noise setting $\mathrm{nl} = 0.03$.

Method	IQ-Chain					Chain					Realistic-Chain				
	SNR	$p_{0\to1}$	$p_{1\to0}$	σ_0	σ_1	**SNR**	$p_{0\to1}$	$p_{1\to0}$	σ_0	σ_1	**SNR**	$p_{0\to1}$	$p_{1\to0}$	σ_0	σ_1
NE24_v1	9.85	0.0450	0.0382	0.2306	0.2355	10.28	0.0458	0.0405	0.2222	0.2270	10.19	0.0386	0.0331	0.1846	0.1884
NE24_v2	9.58	0.0467	0.0396	0.2229	0.2276	9.07	0.0555	0.0494	0.2791	0.2850	9.81	0.0444	0.0389	0.2239	0.2285
QA-Google_v1	10.76	0.0251	0.0212	0.1062	0.1082	10.58	0.0265	0.0220	0.1125	0.1147	10.90	0.0282	0.0230	0.1168	0.1191
QA-Google_v2	10.65	0.0256	0.0207	0.1109	0.1130	10.79	0.0279	0.0231	0.1224	0.1248	10.52	0.0340	0.0278	0.1307	0.1331
QE-IBM_v1	8.12	0.0338	0.0265	0.1447	0.1475	7.85	0.0375	0.0315	0.1610	0.1641	8.01	0.0435	0.0333	0.1716	0.1749
QE-IBM_v2	8.73	0.0350	0.0276	0.1460	0.1488	8.20	0.0408	0.0335	0.1750	0.1784	8.17	0.0465	0.0376	0.1853	0.1890
MAQCal	**11.23**	**0.0206**	**0.0176**	**0.0985**	**0.1006**	**11.32**	**0.0213**	**0.0181**	**0.1043**	**0.1065**	**12.81**	**0.0202**	**0.0170**	**0.0984**	**0.1005**

Table 5: End-to-end readout quality under noise setting $\mathrm{nl} = 0.08$.

Method	IQ-Chain					Chain					Realistic-Chain				
	SNR	$p_{0\to1}$	$p_{1\to0}$	σ_0	σ_1	**SNR**	$p_{0\to1}$	$p_{1\to0}$	σ_0	σ_1	**SNR**	$p_{0\to1}$	$p_{1\to0}$	σ_0	σ_1
NE24_v1	9.37	0.0473	0.0407	0.2256	0.2300	9.37	0.0548	0.0463	0.2612	0.2664	8.50	0.0541	0.0454	0.2558	0.2610
NE24_v2	8.90	0.0621	0.0526	0.3123	0.3187	9.14	0.0566	0.0494	0.3055	0.3120	8.58	0.0580	0.0490	0.2863	0.2922
QA-Google_v1	9.75	0.0316	0.0277	0.1390	0.1416	9.85	0.0286	0.0241	0.1249	0.1272	10.08	**0.0273**	0.0228	0.1210	0.1232
QA-Google_v2	10.59	0.0240	0.0211	0.1097	0.1116	10.25	0.0322	0.0269	0.1354	0.1380	10.29	0.0300	0.0262	0.1345	0.1370
QE-IBM_v1	7.87	0.0361	0.0307	0.1591	0.1621	7.90	0.0424	0.0358	0.1822	0.1856	8.30	0.0345	0.0297	0.1605	0.1635
QE-IBM_v2	7.83	0.0421	0.0361	0.1855	0.1889	7.92	0.0389	0.0345	0.1733	0.1765	7.93	0.0424	0.0380	0.1842	0.1877
MAQCal	**12.02**	**0.0222**	**0.0167**	**0.1013**	**0.1034**	**11.18**	**0.0266**	**0.0193**	**0.1115**	**0.1137**	**12.27**	0.0286	**0.0209**	**0.1040**	**0.1061**

Table 6: End-to-end readout quality under noise setting $\mathrm{nl} = 0.10$.

Method	IQ-Chain					Chain					Realistic-Chain				
	SNR	$p_{0\to1}$	$p_{1\to0}$	σ_0	σ_1	**SNR**	$p_{0\to1}$	$p_{1\to0}$	σ_0	σ_1	**SNR**	$p_{0\to1}$	$p_{1\to0}$	σ_0	σ_1
NE24_v1	8.74	0.0551	0.0470	0.2727	0.2780	8.96	0.0625	0.0512	0.2950	0.3010	8.68	0.0604	0.0529	0.2943	0.3002
NE24_v2	8.40	0.0635	0.0547	0.3272	0.3340	8.81	0.0627	0.0525	0.3114	0.3179	8.57	0.0553	0.0476	0.2800	0.2856
QA-Google_v1	10.56	0.0317	0.0271	0.1337	0.1361	9.81	0.0353	0.0307	0.1510	0.1539	10.76	0.0271	0.0232	0.1170	0.1193
QA-Google_v2	10.56	0.0294	0.0258	0.1307	0.1332	10.48	0.0302	0.0260	0.1302	0.1326	10.18	0.0328	0.0273	0.1365	0.1391
QE-IBM_v1	7.00	0.0435	0.0380	0.1863	0.1898	7.02	0.0429	0.0372	0.1901	0.1936	7.67	0.0391	0.0318	0.1689	0.1720
QE-IBM_v2	7.93	0.0400	0.0352	0.1779	0.1812	7.50	0.0413	0.0369	0.1871	0.1904	8.22	0.0361	0.0315	0.1637	0.1667
MAQCal	**12.64**	**0.0176**	**0.0144**	**0.0778**	**0.0794**	**11.69**	**0.0153**	**0.0147**	**0.0773**	**0.0787**	**13.11**	**0.0190**	**0.0161**	**0.0983**	**0.1006**

A.4 Extended Experiments

A.4.1 Sensitivity to Scan Point Budget

Setup. We test whether denser fixed sweeps reproduce the frequency-localization improvements under equal measurement budgets. We constrain the total number of frequency evaluations within the spectroscopy interval and run Agent1 against budget-matched versions of the same baselines, where each method is restricted to an identical number of frequency samples. All experiments in this section use a fixed noise level of $n = 0.05$.

Results. Figure 5 shows that increasing the scan budget does not reliably improve baseline performance. Across all three noise families, baseline MAE remains flat or even increases at higher budgets. A fixed window can remain poorly centered when the resonance shifts or local noise obscures its response. By contrast, Agent1 stays at low aggregate error across all tested budget levels.

This experiment tests one alternative explanation for the gains in Table 1 and Figure 3: a denser uniform scan. Under the tested budgets, adding uniformly distributed points does not reliably improve the fixed presets. The results are consistent with feedback-based allocation contributing to localization accuracy under a matched number of samples.

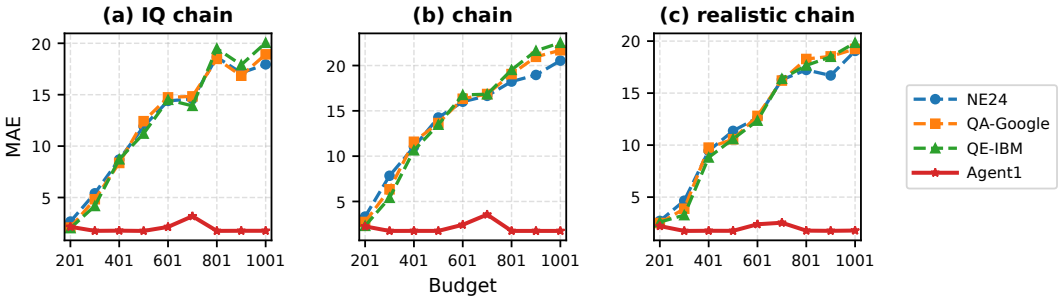

Figure 5: Sensitivity to scan point budget. MAE of $\hat{f}_{\text{res}}$ versus total scan points at noise level $n = 0.05$. Increasing the budget does not reliably improve fixed sweep baselines, while Agent1 has lower aggregate error in the reported budget levels and noise families.

A.4.2 Drift Tracking Calibration

Setup. The experiments above evaluate MAQCal in a single-episode setting where the device parameters are stationary within each run. To test whether the workflow can extend to time-varying conditions, we simulate a scenario in which the target resonator frequency drifts over $T = 20$ successive calibration rounds according to a Gaussian drift process:

$$f_c(t+1) = f_c(t) + \epsilon_t, \qquad \epsilon_t \sim \mathcal{N}(0, \sigma^2_{\text{drift}}). \tag{3}$$

We consider three drift intensities,

$$\sigma_{\text{drift}} \in \{0.2, 0.5, 1.0\}\ \text{MHz}, \tag{4}$$

corresponding to mild, moderate, and strong temporal variation. At the first time step, all methods perform calibration from the same initial condition. For subsequent steps, MAQCal uses the previous calibration output as a prior and updates its scan plan based on newly observed feedback, while the fixed sweep baselines repeat with their predefined spectroscopy windows. This experiment is a controlled simulation stress test. Its purpose is to evaluate whether the feedback-driven workflow can reuse previous outputs as priors under gradual frequency drift.

Metrics. We report five drift tracking metrics. (i) **Final cumulative regret** (↓): the total gap between oracle calibration quality and achieved quality accumulated over all T rounds, defined as $\mathcal{R}(T) = \sum_{t=1}^{T}(J_t^\star - J_t)$. (ii) **Final SNR** (↑): the readout SNR achieved at the last calibration round $t = T$. (iii) **Mean SNR over time** (↑): the average readout SNR across all T rounds, $\bar{S} = \frac{1}{T}\sum_{t=1}^{T}\text{SNR}(t)$. (iv) **Mean regret over time** (↓): the average per round regret, $\bar{r} = \frac{1}{T}\sum_{t=1}^{T}(J_t^\star - J_t)$. (v) **Median cumulative regret** (↓): the median of cumulative regret across chips, providing a measure that is robust to outlier runs.

Results. Figure 6 summarizes drift tracking performance across the reported metrics and three drift intensities. The plotted aggregate results favor MAQCal across these settings. The pattern appears in both regret-based and SNR-based measures: for example, at moderate drift ($\sigma = 0.5$), MAQCal reaches a final SNR of 11.79, compared to 6.73 for the best fixed baseline, and reduces median cumulative regret to 30.47, compared to 75.27 for the best fixed baseline. As drift intensity increases to $\sigma = 1.0$, all methods degrade, and MAQCal remains ahead of the fixed baselines in these plotted settings while reusing previous calibration outputs as priors for subsequent rounds. The SNR and regret comparisons describe sequential recalibration under this simulated drift process.

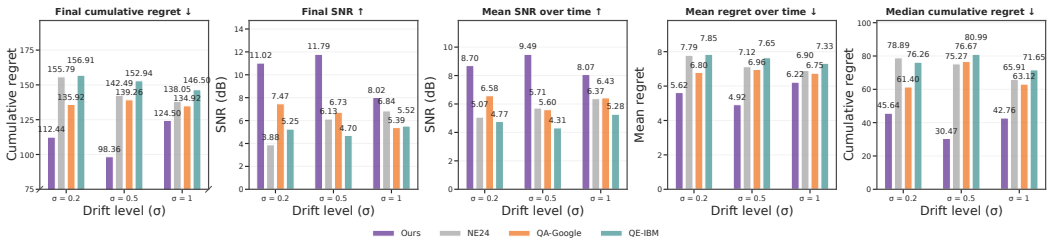

Figure 6: Drift tracking over $T = 20$ rounds under three drift intensities ($\sigma \in \{0.2, 0.5, 1.0\}$ MHz). From left to right: final cumulative regret (↓), final SNR (↑), mean SNR over time (↑), mean regret over time (↓), and median cumulative regret (↓). The plotted aggregate values favor MAQCal across the reported settings.

A.4.3 Retrieval Database Scaling

Setup. We investigate how the size of the retrieval database affects calibration performance. As described in § 3.4, MAQCal constructs an empirical prior from stored simulated chip records to center the initial spectroscopy window. To quantify the contribution of this component, we vary the number of stored chip records as $K_{\text{mem}} \in \{0, 50, 100, 200\}$. Here, $K_{\text{mem}} = 0$ denotes a variant in which MAQCal performs calibration without any retrieval-based prior; the frequency expert starts from the nominal design frequency and relies entirely on feedback to localize the resonance. For $K_{\text{mem}} \in \{50, 100, 200\}$, we uniformly sample the corresponding number of chip records from the original database. The evaluation set is kept fixed across all sizes, ensuring that the variants differ in the number of available prior records.

Results. Figure 7 shows that increasing the database size reduces the frequency offset MAE across all three noise families. Since this metric measures the error of a frequency offset, all MAE values in this experiment are reported in MHz. At low noise, the variants have similar error. As noise increases, the curves diverge: at $\text{nl} = 0.10$ under Chain noise, $K_{\text{mem}} = 0$ reaches approximately 5.7 MHz while $K_{\text{mem}} = 200$ remains near 2.3 MHz. The largest improvement occurs in the transition from $K_{\text{mem}} = 0$ to $K_{\text{mem}} = 50$, indicating that even a modest collection of simulated chip records provides a useful starting point. Further increases to 100 and 200 yield smaller additional gains in the tested split. The same pattern holds across IQ-Chain and Realistic-Chain noise.

At low to moderate noise, the $K_{\text{mem}} = 0$ variant also has low error. The gap between variants widens at higher noise levels, showing the value of a more accurate starting frequency in these simulated settings.

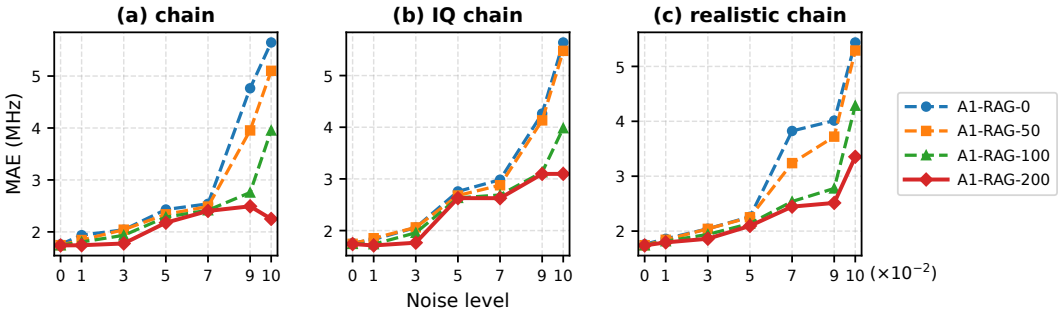

Figure 7: Retrieval database scaling. Frequency offset MAE, reported in MHz, versus noise level under different database sizes $K_{\text{mem}} \in \{0, 50, 100, 200\}$, evaluated across three noise families. Larger databases provide more accurate empirical priors, with the benefit most pronounced under high noise.

B Workflow and Device Illustrations

B.1 Quantum Chip Layout

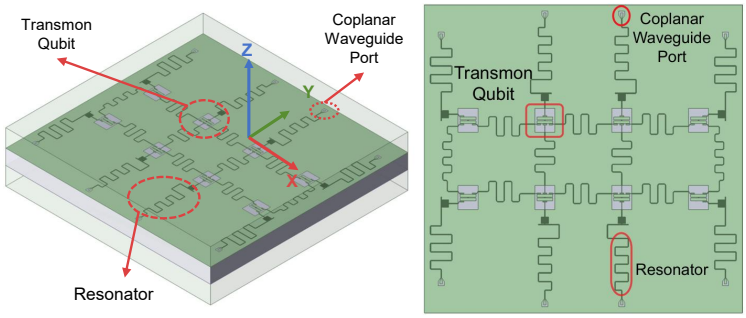

Figure 8: Quantum chip layout. Each chip contains a hanger type resonator coupled to a feedline, a transmon qubit, and a coplanar waveguide port.

B.2 Example Execution Trace

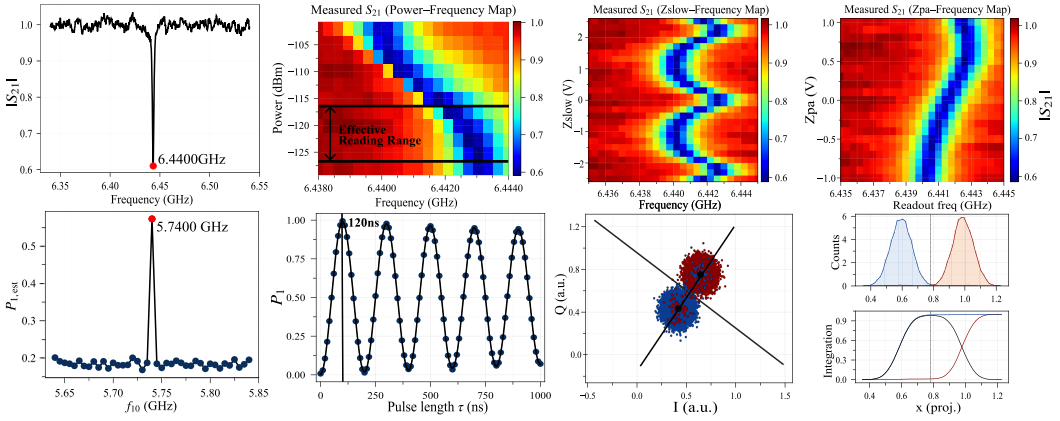

Figure 9: Example execution trace of MAQCal. The workflow produces intermediate spectroscopy, power sweep, and readout quality summaries at each stage, allowing the calibration process to be inspected through its intermediate outputs.

C AGENT IMPLEMENTATION AND REPRODUCIBILITY

This appendix specifies the language-model interface and the Python rules that turn proposed actions into frequency and power sweeps. It includes the system prompts, user-message construction, output schemas, action constraints, and deterministic fallback execution path. The accompanying `reproducibility/source_manifest.json` identifies the reference implementation files by SHA-256 hashes. The reported experiments used GPT-4o mini, requested as `gpt-4o-mini`.

C.1 MODEL CONFIGURATION AND EXECUTION PATH

The reference implementation uses the OpenAI Python client through `chat.completions.create`. The experiment entry points resolve the model from `--llm-model`, with a default taken from `OPENAI_MODEL` or `gpt-4o-mini` when that variable is absent. They set temperature to 0.0. Each request contains one system message and one JSON user message.

Table 7: Planner request and response settings.

Setting	Implementation
Model selection	Explicit CLI argument, then `OPENAI_MODEL`, then `gpt-4o-mini`.
Decoding	Temperature 0.0. Other decoding parameters use API defaults.
Initial response mode	Strict JSON schema generated from the stage's Pydantic model. Schema names are `A1Step`, `A2Step`, and `BudgetedA1Step`.
Request exception	One additional application-level request in JSON-object mode after an exception in the initial request.
Response validation	Nonempty content and Pydantic parsing. A propagated exception invokes the stage's deterministic fallback execution path.
Conversation state	Fresh two-message request at each planning step. The latest measurement summary is supplied in the user payload.

Frequency and power agents run as direct Python loops. Python executes the measurements, fits their outputs, applies numerical constraints, checks termination, and selects the returned estimate. The LLM proposes the next sweep parameters and a stop flag. The stage loops enforce iteration limits. The matched-budget frequency variant also checks its remaining measurement-point budget before each sweep. A planning step can stop before taking a measurement.

C.2 OBSERVABLE INPUTS AND STRUCTURED ACTIONS

The user message is the JSON serialization of the payload, using `ensure_ascii=False` and `allow_nan=False`. A recursive preprocessing function converts nonfinite numerical observations to `null`. No additional natural-language user instruction is appended. At the first iteration, the previous-result field is `null`. Later calls receive the latest stage result in a fresh conversation. Table 8 specifies the payload fields. The exact constructor expressions are also included in `reproducibility/user_payloads.py.txt`.

Table 8: User-message content passed to each planner.

Planner	Payload fields and contents
A1	`iter_idx`, `max_iters`, `converge_eps_hz`, `chip_json`, `last_fit`, and `output_keys_exactly`.
A1 chip prior	Sample identifier, resonator length in mm, designed readout frequency in Hz and GHz, linewidth in Hz, dispersive shift in MHz, shift-to-linewidth ratio, and qubit frequency in Hz. Retrieval-enabled frequency entry points can add a predicted offset, shifted frequency, and neighbor records.
A1 feedback	Fit success, fitted frequency, previous scan center, span and point count, grid spacing, relative position in the scan, peak SNR, relative circle-fit residual, minimum-transmission frequency, and normalized frequency mismatch. The payload retains the latest fit event, including its diagnostic fields.
Budgeted A1	A1 fields plus `point_budget_total`, `points_used_total`, and `points_remaining`.
A2	`iter_idx`, `max_iters`, `noise_level`, `expert_prior`, `fixed_freq_scan`, `last_p2`, `constraints`, and `output_keys_exactly`.
A2 fixed inputs	Frequency center, span and point count, sweep method `analytic`, selection label `physical_lower_half`, power prior $[-150, -110]$ dBm with 2 dB spacing, and numerical action constraints. The noise level comes from the simulator configuration.
A2 feedback	Selected power, index and edge flag, power grid, depth, fit residual and absolute frequency shift, local means and medians over up to five adjacent power levels, global 50th and 90th percentiles, percentile spreads, and the fraction of nonfinite fit metrics.

The `noise_level` field is a dimensionless perturbation parameter configured by the simulator for the current episode. In the experiments it takes values in $\{0.00, 0.01, \ldots, 0.10\}$ within each synthetic noise family. It scales perturbations relative to the simulated signal amplitude, with different components in each family. A2 reads this value directly from `dev.noise_params` and supplies it to both the planner and the fallback rule. It is known simulation metadata available before the sweep. Estimating noise from measurement traces is outside the evaluated protocol. Hardware use would require an experimentally available estimate or a planner evaluated without this input.

The structured actions contain the five required fields listed in Table 9. Extra fields are forbidden by the Pydantic models. Numerical ranges are imposed by host code after parsing, so schema validity and physical admissibility are distinct checks. The free-text `reason` is stored in the trace and does not select a measurement or change a stopping condition. Python operates on the numerical fields and `should_stop`. The complete generated schemas are included in `reproducibility/action_schemas.json`.

Table 9: Required output fields for the frequency and power planners.

Planner	Field	Type	Meaning
A1	`scan_center_freq_hz`	number	Next center in Hz
	`scan_span_hz`	number	Next span in Hz
	`num_freq_points`	integer	Sweep point count
	`should_stop`	boolean	Stop before the next sweep
	`reason`	string	Short explanation
A2	`power_min_dbm`	number	Lower power in dBm
	`power_max_dbm`	number	Upper power in dBm
	`power_step_db`	number	Power spacing in dB
	`should_stop`	boolean	Stop before the next sweep
	`reason`	string	Short explanation

C.3 Validation and Fallback Execution Path

An exception from the first completion request triggers a second request with the same messages in JSON-object mode. An empty response or a parsing failure occurs after this request block and propagates directly to the agent. The agent then uses its deterministic fallback action for that iteration. There is no additional response-repair loop. Successfully parsed actions pass through the numerical guards in Table 10. The guards can alter a valid model proposal before execution. This separates request recovery, parsing fallback, and numerical action correction in the control flow.

Table 10: Host-side action constraints and stopping rules.

Stage	Rule
Standard A1	Clamp the center to 1 to 15 GHz and span to 1 Hz to 2 GHz. Require at least 21 points. This variant has no explicit total frequency-point cap.
A1 confidence	Low confidence is triggered by peak SNR below 3 or missing, relative circle residual above 0.15 or missing, or a finite relative position outside $[0.15, 0.85]$. Replace the proposed center by the previous center plus 0.5 times the fitted shift clipped to ± 0.25 of the previous span.
A1 span update	After a valid fit, retain a span of at least $\max(80\,\mathrm{MHz}, 20\kappa_f)$, within the span clamps. At low confidence, do not narrow the previous span. An edge or residual warning widens it by at least a factor of 1.5, up to the cap. A failed fit also requests a wider scan.
A1 termination	Stop after a valid fitted frequency lies within 10 kHz of the preceding scan center, an accepted stop proposal, or the iteration limit. The standard loop checks the fit-based gate after planning.
Budgeted A1	Clamp the point count to the remaining total budget and stop below 21 remaining points. Check convergence before planning. Default confidence settings use a 0.20 circle-residual threshold, 0.6 damping, and a 15 MHz absolute span floor within the 200 MHz cap.
A2	Clamp endpoints to $[-200, -20]$ dBm, reorder reversed endpoints, and clamp spacing to $[0.5, 5]$ dB. A further spacing adjustment targets 7 to 301 power levels. Stop on a stop proposal or the iteration limit.

Here κ_f is the input linewidth in Hz. The standard A1 fallback starts from the retrieved frequency when available, then the design frequency, with a 6 GHz default if neither field is present. It uses $s_0 = \max(30\kappa_f, 80\,\mathrm{MHz})$ and $N_0 = \max(21, \lceil s_0 / \max(\kappa_f/5, 1\,\mathrm{Hz})\rceil + 1)$. If the linewidth is unavailable or nonpositive, it estimates $\kappa_f = f_0/5000$. A successful fit recenters the next scan without changing its span or point count. A failed fit doubles the span up to 2 GHz and uses at least 201 points. The same confidence guards and stopping gate are then applied. The budgeted fallback instead initializes with $\max(15\kappa_f, 40\,\mathrm{MHz})$ within its span bounds and caps the requested points by the remaining budget. After a successful fit, it uses a damped center update and scales the span by 1.5 at low confidence or 0.6 otherwise, before the host guards are applied.

The A2 fallback starts from the power prior in Table 8. A selected point within two indices of an edge expands that boundary by 10 dB. An interior point defines a new window centered on the selected power, with half-width 12 dB for noise level at least 0.05 and 10 dB otherwise. At lower noise the spacing is clipped to 0.5 to 1 dB. Python retains fit summaries across iterations and selects the final estimate using fixed scoring functions. A1 scores valid fits using peak SNR, circle residual, edge position, and frequency mismatch. A2 scores fitted power candidates using depth, residual, frequency shift, invalid-fit fraction, edge position, and a power preference. It returns the lowest power among candidates scoring at least 98% of the best score. These selection rules are part of the controller and do not depend on the generated explanation.

Setting `planner=None` in `A1FrequencyAgent` and `A2PowerAgent` selects this fallback execution path at every iteration. The fitting routines, guards, and final candidate selection remain active. The reported experiments compare MAQCal with the six fixed presets. A dedicated evaluation with the fallback path active throughout each episode remains future work. This configuration makes that adaptive comparison executable with the same measurement interface, guards, and candidate scoring. The fallback decision rule is deterministic for identical inputs and measurement summaries. Simulator randomness is a separate source of variation.

C.4 System Prompts

The following A1 and A2 prompts are copied from `agent_core/prompts.py`. Line wrapping is adjusted for display. Their statement about deterministic execution assigns measurement execution to Python. The budget experiment overrides the generic A1 prompt with a Chinese strategy prompt from its active entry point. An English translation is provided below. The original Chinese text is the model input. Its constants resolve to 21 minimum points and a 10000.0 Hz convergence threshold. Exact prompt strings, including the original language and whitespace, are supplied in `reproducibility/system_prompts.json`.

A1 System Prompt.

```
You are A1, a bounded planner for superconducting readout
frequency scans.

You do not run experiments, call tools, fit data, or decide
hidden state.  Python runs every experiment
deterministically.
Your only job is to propose the next bounded action from the
observable payload.

Output exactly one JSON object with only these fields:
scan_center_freq_hz, scan_span_hz, num_freq_points,
should_stop, reason.

Do not output markdown, comments, code fences, or extra
fields.
```

A2 System Prompt.

```
You are A2, a bounded planner for superconducting readout
power sweeps.

You do not run experiments, call tools, fit data, or decide
hidden state.  Python runs every experiment
deterministically.
Your only job is to propose the next bounded action from the
observable payload.

Output exactly one JSON object with only these fields:
power_min_dbm, power_max_dbm, power_step_db, should_stop,
reason.

Do not output markdown, comments, code fences, or extra
fields.
```

Budgeted A1 System Prompt (English Translation).

```
You are defined as A1:  an agent for planning readout
frequency-sweep parameters and iterative convergence on
superconducting quantum chips.

At each call you receive:
- chip_json:  chip design information, most important at the
first iteration, possibly including
rag_pred_shifted_readout_f_hz.
- last_fit:  the preceding F5 fit and quality metrics,
available when iter_idx > 0.
- iter_idx / max_iters:  iteration progress.
- points_remaining:  the remaining point budget, the only
budget information.

Your task:  output the next frequency-sweep parameters and
decide whether to stop using should_stop.

Hard rules:  strictly respect the budget.
```

```
1) num_freq_points must satisfy 21 <= num_freq_points <=
points_remaining.
2) If points_remaining < 21, set should_stop=true because
another sweep is impossible.
3) Never propose num_freq_points > points_remaining.
4) Multiple iterations are allowed.  Each deducts points
from the budget.  Improve the final readout-frequency
estimate within that budget.

Recommended strategy:  follow these recommendations whenever
possible.
A) First sweep, iter_idx == 0:
- center:  prefer rag_pred_shifted_readout_f_hz, otherwise
readout_f_design_hz.
- span:  about 40 MHz is recommended, adjustable from 20 MHz
to 200 MHz to cover offsets and noise.
- points:  use enough points within points_remaining for
sufficient resolution, with at least 21 points.

B) Later sweeps, iter_idx >= 1:
- If last_fit.ok == True, move the center toward
last_fit.f_ro_hz without a large jump that could follow
noise.
- If last_fit.rel_pos is near an edge (< 0.15 or > 0.85),
last_fit.peak_snr is very low, or last_fit.circle_rel_std is
large, keep or widen the span.
- If last_fit.mismatch_frac is large, f_ro differs clearly
from the minimum-transmission frequency.  If peak_snr is
also low, treat the fit as unreliable.  Widen the span and
return to a conservative center such as the design value or
preceding center.

Available stopping conditions:
- last_fit.ok == True and abs(last_fit.f_ro_hz -
last_fit.scan_center_hz) <= 10000.0:  set should_stop=true.
- Or points_remaining < 21.

Strict output format:
Output only one JSON object containing only the following
keys:
scan_center_freq_hz, scan_span_hz, num_freq_points,
should_stop, reason.
Do not output markdown, explanations, or code fences.
```