

Supplementary Information

Cross-Platform Noise Thresholds in Quantum Teleportation:
Trapped-Ion and Neutral-Atom Architectures
v5.0

Celal Arda
Independent Researcher
`celal.arda@outlook.de`

February 2026

Contents

1 Data Availability and Code Repository

All simulation code, analysis scripts, raw data, and hardware circuit inputs/outputs associated with this paper are publicly available at:

GitHub: [unearthlyimprint/wormhole_stability](https://github.com/unearthlyimprint/wormhole_stability)

The repository contains the following components:

- **code/** — Core experiment scripts for the dephasing sweep and teleportation protocols.
- **data/** — All experimental data in CSV format, including dephasing sweeps, hardware results, and qubit scaling datasets.
- **hardware_gpu_input/** — Raw circuit submission (input JSON) and measurement results (output JSON) from the IonQ Forte-1 QPU.
- **scripts/** — Extended analysis, sweep, and plotting scripts, including Trotter-step scaling and control experiments.
- **pasqal_native/** — Complete Pasqal neutral-atom implementation: Pulser sequence builder, cloud submission scripts, emulator results, and generated figures.
- **manuscript/** — Full L^AT_EX source of the main manuscript and this Supplementary Information document.

Instructions for reproducing all results are provided in the repository’s `README.md`.

2 Quantum Circuit Architecture

2.1 3-Qubit Teleportation Protocol

The core protocol employs 3 qubits: Alice (A), Bob (B), and one message qubit (M).

Stage 1: Entanglement. A Bell pair is created between Alice and Bob:

$$|A, B\rangle \xrightarrow{H \otimes I} \xrightarrow{\text{CNOT}} \frac{1}{\sqrt{2}} (|00\rangle + |11\rangle) \quad (1)$$

followed by $R_z(\pi)$ and $R_z(-\pi)$ phase kicks on A and B respectively.

Stage 2: Message Injection. A test message state $|+\rangle = H|0\rangle$ is prepared and swapped onto Alice’s register via a SWAP gate (3 CNOT decomposition).

Stage 3: Parametric Dephasing. Controlled phase rotations of tunable strength γ are applied:

$$\mathcal{D}(\gamma) = R_z(\gamma\pi \cdot \xi_A) \otimes R_z(\gamma\pi \cdot \xi_B) \quad (2)$$

where $\xi_A = 1.0$ and $\xi_B = -1.5$ distribute the dephasing asymmetrically. At $\gamma = 0$, no perturbation is applied. This is a *unitary* operation modeling coherent phase errors.

Stage 4: Bridge Evolution. The Heisenberg-type coupling is implemented via first-order Trotter decomposition:

$$e^{-i\theta(XX+YY+ZZ)} \approx R_{XX}(\theta) \cdot R_{YY}(\theta) \cdot R_{ZZ}(\theta) \quad (3)$$

with $\theta = \pi/4$. Each R_{PP} decomposes into 2 CNOT gates, giving 6 CNOTs total.

Stage 5: Measurement. Bob’s qubit is measured in the Hadamard basis. Fidelity: $F = 2P(|0\rangle) - 1$.

Total entangling gate count: 1 (Bell) + 3 (SWAP) + 6 (bridge) = 10 CNOTs.

2.2 Extended 9-Qubit Architecture

The original 9-qubit architecture used two 4-qubit boundary registers (Alice: A_0 – A_3 , Bob: B_0 – B_3) plus one message qubit (M). As documented in Section ??, this architecture suffers from spectator qubits: only A_0 , B_0 , and M participate in information transfer.

3 Extended Experimental Data

3.1 IonQ Simulator — Full Dephasing Sweep

Table ?? presents the complete dephasing sweep results from the Azure Quantum IonQ simulator (100 shots per point).

Table 1: Full dephasing sweep on IonQ simulator.

	γ	$P(0\rangle)$	F	σ	Status
0.000	0.960	0.920	0.039		High fidelity
0.050	0.960	0.920	0.039		High fidelity
0.100	0.880	0.760	0.065		Degraded
0.200	0.600	0.200	0.098		Near-random
0.300	0.500	0.000	0.100		Random
0.400	0.450	−0.100	0.099		Below random
0.535	0.460	−0.080	0.100		Collapsed
0.600	0.460	−0.080	0.100		Collapsed
0.800	0.490	−0.020	0.100		Collapsed
1.000	0.500	0.000	0.100		Random

3.2 IonQ Forte-1 Hardware — Shot-Level Statistics

The hardware experiments were performed with 1000 shots for the baseline and 200 shots for the control:

Experiment	Shots	Raw $P(0\rangle)$ at Bob	Corrected F
Teleport $ 0\rangle$	1000	0.509 (pre-correction)	0.987 ± 0.004
Teleport $ 1\rangle$	1000	0.494 (pre-correction)	0.988 ± 0.003
Control (no ent.) $ 0\rangle$	200	0.995 (Bob stays $ 0\rangle$)	n/a
Control (no ent.) $ 1\rangle$	200	0.995 (Bob stays $ 0\rangle$)	n/a

Table 2: Shot-level summary of all IonQ Forte-1 hardware experiments.

γ	Mean $\langle n \rangle$	Ground state P_0	Shannon entropy S (bits)
0.00	0.48	12.0%	3.2
0.05	0.45	12.0%	3.1
0.10	0.38	22.5%	2.8
0.15	0.28	45.0%	2.3
0.20	0.14	71.5%	1.2
0.25	0.04	93.0%	0.4
0.30	0.01	99.0%	0.1
0.40	0.01	99.5%	0.0

Table 3: Pasqal neutral-atom EMU_FREE sweep: fine-grained γ resolution.

3.3 Pasqal Neutral-Atom — Full Sweep Data

3.4 Emulator Cross-Validation

3.5 FRESNEL_CAN1 QPU Hardware Results

The four-tier validation chain was completed with physical execution on Pasqal’s FRESNEL_CAN1 neutral-atom QPU (22 atoms: 9 core qubits + 13 spectators, 500 shots per checkpoint).

Three distinct noise regimes are identified:

- $\gamma = 0.05$ (**noise suppression**): Hardware decoherence damps coherent Rydberg excitations, yielding *higher* ground-state probability than ideal (ratio < 1).
- $\gamma = 0.20$ (**close agreement**): QPU matches ideal simulation within 1.4%, indicating the dephasing-induced transition is robust against hardware noise.
- $\gamma = 0.40$ (**noise floor**): Residual hardware noise creates a $\sim 2.6\%$ excitation floor, preventing the QPU from reaching the ideal ground state.

4 Hardware Platform Specifications

5 Spectator Qubit Analysis

5.1 Scaling Test Results

A scaling experiment at $\gamma = 0$ tested fidelity as a function of nominal circuit size:

γ	EMU_FREE P_0	EMU_SV P_0	Δ
0.05	12.0%	11.0%	+1.0%
0.20	71.5%	72.5%	-1.0%
0.40	93.0%	94.0%	-1.0%

Table 4: Cross-validation of EMU_FREE against exact state-vector simulation (EMU_SV). Agreement is within 1.5% at all checkpoints.

γ	QPU P_0	EMU_FREE P_0	Noise Ratio	Unique States
0.05	19.6%	8.0%	$0.67\times$	68
0.20	70.6%	72.0%	$1.13\times$	28
0.40	79.0%	93.0%	$2.84\times$	19

Table 5: FRESNEL_CAN1 QPU hardware results (core 9-qubit extraction). The noise ratio quantifies $(1 - P_0^{\text{QPU}})/(1 - P_0^{\text{ideal}})$.

5.2 Analysis

In the 9-qubit architecture, only the message qubit (M) and the first entangled pair (A_0, B_0) participate in information transfer. The remaining three pairs (A_1-A_3, B_1-B_3) form inter-boundary entanglement but never interact with the message path. IonQ’s transpiler detects these as spectator qubits and removes them.

This has two consequences:

1. The $F \approx 0$ result at $\gamma = 0.535$ is driven by the *injected dephasing parameter* γ , not by circuit depth or qubit count.
2. Testing depth-dependent fidelity degradation requires circuits where *all gates lie on the message’s critical path* (e.g., Trotter-step scaling).

6 Pasqal Neutral-Atom Implementation Details

6.1 Pulser Sequence Construction

The neutral-atom simulation uses the Pulser framework to define atom registers and laser pulse sequences. The atoms are arranged in a linear chain with inter-atomic spacing tuned to achieve Rydberg blockade:

- **Register:** 3 atoms at spacing $d = 6\mu\text{m}$ (within blockade radius $R_b \approx 9\mu\text{m}$).
- **Channel:** Global Rydberg channel with $\Omega_{\text{max}} = 2\pi \times 4\text{MHz}$.
- **Dephasing injection:** Detuning ramp $\Delta(\gamma)$ applied after entangling pulse.
- **Measurement:** Final state sampled from the ground/Rydberg basis.

Parameter	Value
<i>IonQ Forte-1 Trapped-Ion QPU</i>	
Ion species	$^{171}\text{Yb}^+$
Connectivity	All-to-all
Single-qubit gate fidelity	$> 99.7\%$
Two-qubit gate fidelity (MS gate)	$\sim 99.5\% - 99.7\%$
Two-qubit gate error	$\sim 0.3\% - 0.5\%$ per gate
Readout fidelity	$> 99.5\%$
Coherence time (T_2)	> 1 s
<i>Access Details</i>	
Cloud provider	Microsoft Azure Quantum
Backend (simulation)	<code>ionq.simulator</code>
Backend (hardware)	<code>ionq.qpu.forte-1</code>
<i>Pasqal Neutral-Atom Platform</i>	
Atom species	^{87}Rb
Interaction type	Rydberg blockade (van der Waals)
Emulator backends	EMU_FREE, EMU_SV, EMU_FRESNEL
QPU	FRESNEL_CAN1 (61 traps, 22 atoms used)
SDK	Pulser 1.7.0 + <code>pasqal-cloud</code> 0.20.8

Table 6: Hardware and emulator platform specifications.

6.2 Rydberg Blockade Hamiltonian

The Rydberg blockade Hamiltonian is:

$$H = \frac{\Omega}{2} \sum_i \sigma_x^i - \Delta \sum_i n_i + \sum_{i < j} \frac{C_6}{|r_i - r_j|^6} n_i n_j \quad (4)$$

where Ω is the Rabi frequency, Δ is the detuning, $n_i = |r\rangle\langle r|_i$ is the Rydberg number operator, and C_6 is the van der Waals coefficient.

The dephasing parameter γ maps to the detuning: increasing γ shifts the system from the resonant regime (where Rydberg excitations are favorable) to the off-resonant regime (ground state dominated).

7 Software Environment and Reproducibility

7.1 Python Dependencies

All code was developed and tested with Python ≥ 3.10 . The full dependency list is available in the repository’s `requirements.txt`:

7.2 Script Descriptions

Table ?? provides a summary of all scripts in the repository.

Nominal qubits	Entangling gates	Hardware F	Note
3	2	1.000	Baseline
5	6	1.000	Spectators removed
7	10	1.000	Spectators removed
9	14	1.000	Spectators removed

Table 7: Scaling test on IonQ Forte-1 at $\gamma = 0$. The hardware compiler identifies spectator qubits and optimizes them away. Note: at $\gamma = 0$, the parametric dephasing gates are identity operations, allowing the IonQ compiler to further optimize the nominal 10-CNOT baseline circuit to just 2 effective entangling gates.

Package	Version	Purpose
<code>numpy</code>	≥ 2.0	Numerical computation
<code>scipy</code>	≥ 1.12	Scientific computing
<code>matplotlib</code>	≥ 3.8	Visualization
<code>azure-quantum</code>	≥ 2.0	Azure Quantum SDK (IonQ)
<code>qsharp</code>	≥ 1.0	Q# integration
<code>pulser</code>	≥ 1.0	Pasqal sequence builder
<code>pasqal-cloud</code>	≥ 0.20	Pasqal Cloud SDK
<code>qutip</code>	≥ 5.0	Local quantum simulation
<code>python-dotenv</code>	≥ 1.0	Environment variable management

Table 8: Python package dependencies.

Table 9: Repository script index with descriptions.

Script	Description
<code>code/experiment_1_phase_transition.py</code>	Dephasing sweep: varies $\gamma \in [0, 1]$ on IonQ simulator.
<code>code/wormhole_pulser_continuous.py</code>	Continuous-mode Pulser simulation of the Rydberg Hamiltonian.
<code>scripts/tier1_analysis.py</code>	Result aggregation and statistical analysis.
<code>scripts/tier1_depth_sweep.py</code>	Depth sweep: varies circuit size at fixed γ .
<code>scripts/tier1v3_trotter_sweep.py</code>	Trotter-step scaling: varies Trotter steps 1–8 on the 3-qubit architecture.
<code>scripts/wormhole_control_experiment.py</code>	Control: teleportation circuit without Bell pair.
<code>scripts/wormhole_hardware_correct.py</code>	Hardware-corrected protocol with classical post-processing.
<code>scripts/wormhole_teleport_local_test.py</code>	Local teleportation experiment with shot-level analysis.
<code>scripts/wormhole_teleport_sweep.py</code>	Parameter sweep of teleportation fidelity.
<code>scripts/plot_azure_data.py</code>	Plotting utilities for Azure Quantum results.
<code>scripts/trotter_noisy_corrected.py</code>	Local noisy Trotter sweep with IonQ Forte noise model.
<code>pasqal_native/scripts/run_wormhole_pasqal.py</code>	Pasqal Cloud submission: builds and submits sequences for γ sweep.

Script	Description
<code>pasqal_native/scripts/run_fine_sweep.py</code>	Fine-grained γ sweep near the threshold region.
<code>pasqal_native/scripts/run_emulator_comparison.py</code>	Cross-validation between EMU_FREE and EMU_SV.
<code>pasqal_native/scripts/analyze_results.py</code>	Analysis and figure generation from emulator results.
<code>pasqal_native/scripts/merge_results.py</code>	Merges multiple emulator result files.
<code>pasqal_native/scripts/run_fresnel_validation.py</code>	Fresnel validation of the neutral-atom layout.
<code>pasqal_native/scripts/fetch_fresnel_results.py</code>	Retrieves completed FRESNEL_CAN1 QPU batch results.
<code>pasqal_native/scripts/analyze_fresnel_can1.py</code>	Extracts core-qubit statistics from QPU data.

7.3 Reproducing Results

To reproduce all results presented in the main manuscript:

1. Clone the repository:

```
git clone https://github.com/unearthlyimprint/
wormhole_stability.git
cd wormhole_stability
```

2. Install dependencies:

```
python -m venv venv && source venv/bin/
activate
pip install -r requirements.txt
```

3. Configure Azure Quantum credentials (see `.env.example`).

4. Run the dephasing sweep:

```
python code/experiment_1_phase_transition.py
```

5. For Pasqal neutral-atom results, follow the instructions in `pasqal_native/README.md`.