Article

Not peer-reviewed version

Economic Decoupling Probability: A Quantum Analogy of Characterizing Bell State Errors and Noise on Real IBM Quantum Hardware

Muhammad Sukri Bin Ramli *

Posted Date: 17 April 2025

doi: 10.20944/preprints202504.1407.v1

Keywords: Quantum Computing; NISQ (Noisy Intermediate-Scale Quantum); Quantum Benchmarking; Quantum Fidelity; Bell State; Entanglement; Quantum Noise; Quantum Error Mitigation; Readout Error; Gate Error; IBM Quantum; ibm_kyiv; Qiskit; mthree (Error Mitigation); Quantum Analogy; Economic Decoupling; Economic Uncertainty; Correlation Breakdown; Hardware Characterization; Qubit Variability

Article

Economic Decoupling Probability: A Quantum Analogy of Characterizing Bell State Errors and Noise on Real IBM Quantum Hardware

Muhammad Sukri Bin Ramli

Asia School of Business, Kuala Lumpur, Malaysia; m.binramli@sloan.mit.edu

Abstract: Accurate generation and measurement of entangled states, such as the Bell state $|\Phi^+\rangle$, are crucial benchmarks for assessing the capabilities and variability of Noisy Intermediate-Scale Quantum (NISQ) hardware. This work benchmarks the fidelity of preparing the $|\Phi^+\rangle = (|00\rangle + |11\rangle)/\sqrt{2}$ state on different qubit pairs ([2, 3] and [7, 8]) of the ibm_kyiv quantum processor over multiple runs (N=5) and employs the deviation from perfect correlation as a quantitative analogy for the probability of unexpected decoupling in systems expected to exhibit strong correlation, such as linked economic indicators. Implementing the standard Hadamard and CNOT gate sequence for 4096 shots per run using the qiskit-ibm-runtime SamplerV2 primitive, we characterized the state preparation and measurement fidelity and applied mthree-based readout error mitigation. Experimental raw results revealed significant variability between layouts, yielding mean anti-correlated outcome probabilities P(Anti) = P(01) + P(10) of approximately 1.6% (±0.3%) for layout [2, 3] and 9.2% (±0.8%) for layout [7, 8]. This performance difference strongly correlated with reported hardware calibration metrics, particularly average readout error rates. Readout error mitigation successfully reduced P(Anti) to near-zero values (≤0.1%) for both layouts, achieving corrected correlated outcome probabilities P(Corr) = P(00) + P(11) of ~99.9-100.0%. Within our conceptual framework, the range of raw P(Anti) serves as a quantitative analogue for the likelihood of 'unexpected decoupling' under different inherent noise conditions, while the mitigated results suggest the potential to isolate underlying system dynamics from measurement noise. This research provides concrete multi-run fidelity benchmarks for ibm_kyiv, demonstrates the effectiveness of error mitigation, highlights performance variability linked to calibration data, and quantifies a range for the proposed economic uncertainty analogy.

Keywords: Quantum Computing; NISQ (Noisy Intermediate-Scale Quantum); Quantum Benchmarking; Quantum Fidelity; Bell State; Entanglement; Quantum Noise; Quantum Error Mitigation; Readout Error; Gate Error; IBM Quantum; ibm_kyiv; Qiskit; mthree (Error Mitigation); Quantum Analogy; Economic Decoupling; Economic Uncertainty; Correlation Breakdown; Hardware Characterization; Qubit Variability

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 17 April 2025 doi:10.20944/preprints202504.1407.v1

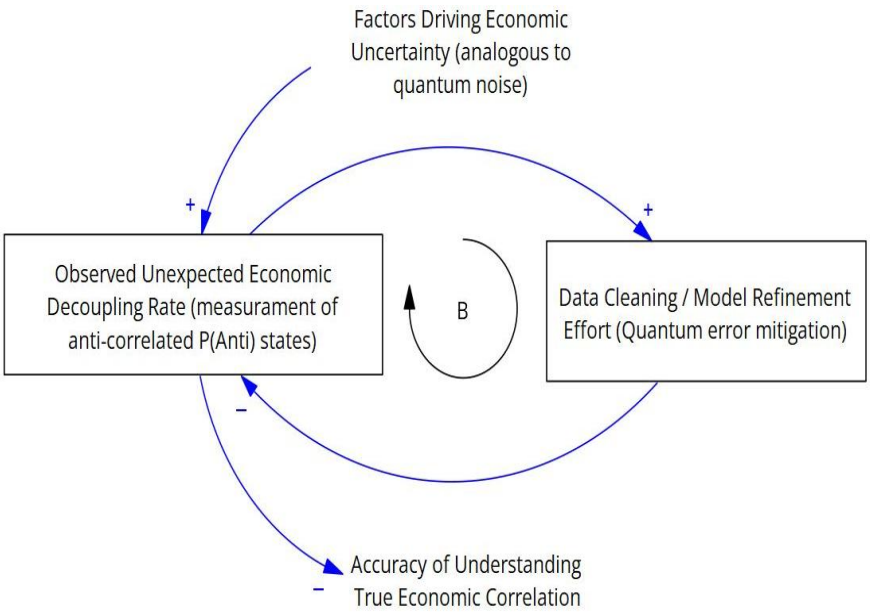

Figure 1. Causal Loop Diagram of the Economic Decoupling Risk Analogy derived from Quantum Fidelity Measurements.

The causal loop diagram illustrates a key dynamic based on the economic analogy from the research. Initially, underlying *Factors Driving Economic Uncertainty* (analogous to quantum noise) positively influence, or increase, the *Observed Unexpected Economic Decoupling Rate* between indicators that should normally be correlated (similar to measuring anti-correlated P(Anti) states). This increase in observed decoupling signals a potential problem, which in turn positively drives an increase in *Data Cleaning / Model Refinement Effort* as analysts or researchers seek to understand or correct the anomaly (akin to applying error mitigation). This increased effort then acts negatively to reduce the *Observed Unexpected Economic Decoupling Rate*, as errors are filtered or models improved. This sequence forms the Balancing Loop (B1): higher observed decoupling prompts more corrective effort, which then lowers the observed decoupling. This feedback loop functions to counteract deviations and stabilize the system's apparent behavior, ultimately influencing the overall *Accuracy of Understanding True Economic Correlation*, as higher observed decoupling tends to reduce this accuracy.

1. Introduction

This study was motivated by the inherent challenge of modeling uncertainty within economic systems, particularly concerning scenarios where established correlations between indicators or markets unexpectedly broke down, potentially impacting forecasting and risk management. Recognizing that recent explorations had considered quantum frameworks for modeling financial and economic systems due to their probabilistic nature and handling of correlations (Orrell, 2020; Herman et al., 2023; Orús et al., 2019), we employed a fundamental quantum procedure: the generation of the maximally entangled $|\Phi^+\rangle$ Bell state using a standard circuit involving a Hadamard gate followed by a Controlled-NOT (CNOT) gate (Nielsen & Chuang, 2010). This state, defined as:

$$|\Phi^+\rangle = \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$$

In an ideal, noiseless scenario, measurements of this state should be perfectly correlated, meaning the outcome is always $|00\rangle$ (both qubits measured as $|0\rangle$) or $|11\rangle$ (both qubits measured as $|1\rangle$) (Bell, 1964; Nielsen & Chuang, 2010). Within this quantum foundation, we proposed a conceptual mapping for analogy: the correlated outcomes $|00\rangle$ and $|11\rangle$ represented expected, synchronized economic scenarios ("Scenario Alpha" and "Scenario Beta," respectively), while the anti-correlated outcomes $|01\rangle$ and $|10\rangle$—ideally absent—were mapped to represent "Unexpected Decoupling"

 doi:10.20944/preprints202504.1407.v1

events, the focus of economic uncertainty. It was emphasized that this constituted a conceptual analogy (Mäki, 2009) designed to explore parallels between quantum measurement statistics and economic unpredictability. The experiment was situated within the context of Noisy Intermediate-Scale Quantum (NISQ) computing (Preskill, 2018), where processors inherently suffered from noise sources like gate errors, readout errors, and decoherence that limited accuracy. Consequently, real-world devices deviated from ideal behavior, yielding non-zero probabilities for the anti-correlated $|01\rangle$ and $|10\rangle$ outcomes when attempting to prepare and measure a Bell state (see e.g., Kandala et al., 2019; Temme et al., 2017).

Benchmarking such fundamental operations was crucial for characterizing these limitations (Eisert et al., 2020; Cross et al., 2019). This experiment utilized the Qiskit framework (Abraham et al., 2019) on the IBM Quantum device ibm_kyiv to investigate these deviations, addressing two primary research questions: (A1) How accurately did the device prepare and measure the correlated $|\Phi^+\rangle$ Bell state, reflecting the analogous probability of expected economic scenarios? and (A2) How significant were the deviations from perfect correlation, specifically, what was the probability of observing the 'unexpected decoupling' outcomes ($|01\rangle$ or $|10\rangle$), providing an analogue for the frequency of unexpected economic events driven by noise/uncertainty factors? The paper proceeded by detailing the methods used for Bell state preparation and measurement, presenting the obtained results, and offering a discussion interpreting these findings within the economic analogy framework, followed by concluding remarks.

2. Literature Review

Quantum computing represents a paradigm shift from classical computation, leveraging quantum mechanical principles like superposition and entanglement to tackle problems previously considered intractable (Nielsen & Chuang, 2010). Entanglement, a key resource, describes correlations between quantum systems that are stronger than any classical counterpart, a concept famously debated by Einstein, Podolsky, and Rosen (1935) and later formalized by Bell (1964; Vedral, 2008). The current state of quantum hardware development is often described as the Noisy Intermediate-Scale Quantum (NISQ) era (Preskill, 2018). NISQ devices possess a limited number of qubits and are highly susceptible to noise, making the accurate characterization and benchmarking of their performance critically important (Cross et al., 2019; McKay et al., 2019).

A fundamental benchmark for quantum processors involves the generation and measurement of entangled states, particularly the Bell states. The ability to reliably create these states is a prerequisite for many quantum algorithms and protocols. However, achieving high fidelity is challenging due to various noise sources inherent in NISQ hardware. These include environmental interactions leading to decoherence (related principles discussed in Slichter, 1990), as well as imperfections in quantum gate operations and measurement readout processes. These errors cause the experimentally realized quantum state to deviate from the intended ideal state, limiting the computational power of near-term devices.

To address the impact of noise, various quantum error mitigation techniques have been developed. Strategies such as Zero Noise Extrapolation (ZNE) attempt to estimate the ideal noiseless outcome by systematically varying noise levels and extrapolating the results back to the zero-noise limit (Temme et al., 2017; Kandala et al., 2019; Qiskit Development Team, n.d.). Other methods specifically target errors occurring during the final measurement step. Among these, Matrix-Free Measurement Mitigation (M3), implemented in the mthree package, offers a scalable approach. Unlike methods requiring the construction and inversion of the full assignment matrix (which becomes computationally prohibitive for many qubits), M3 operates within a reduced subspace defined by the observed noisy measurement outcomes. This matrix-free technique significantly reduces memory requirements and computational cost, making readout mitigation feasible for larger systems compared to traditional matrix inversion methods (Nation et al., 2021). While Qiskit Runtime's EstimatorV2 primitive incorporates built-in mitigation options, SamplerV2 (used for obtaining distributions, as in this work) does not, necessitating the use of external libraries like mthree for post-processing correction. These mitigation techniques aim to improve the accuracy of

 doi:10.20944/preprints202504.1407.v1

results obtained from NISQ computers without the significant overhead required for full fault-tolerant quantum error correction.Beyond fundamental benchmarking, quantum computing holds potential for applications in various fields, including finance and economics. Researchers are exploring quantum algorithms for tasks like portfolio optimization (Rebentrost et al., 2014) and are investigating broader applications within quantitative finance (Orús et al., 2019; Herman et al., 2023). This burgeoning interest motivates exploring connections, even analogical ones, between quantum phenomena and complex economic systems.

The implementation and analysis of quantum experiments heavily rely on specialized software frameworks and libraries. Qiskit provides a comprehensive ecosystem for quantum computing research, development, and execution on various hardware backends, including those from IBM Quantum (Abraham et al., 2019; Qiskit Development Team, n.d.). Data processing and visualization are further supported by established scientific Python libraries such as NumPy for numerical operations (Harris et al., 2020), Matplotlib for plotting (Hunter, 2007), and Pandas for data manipulation (The pandas development team, 2024). These tools are essential for designing circuits, executing jobs, analyzing results, and communicating findings in quantum computation research.

3. Methodology

The experiment focused on generating the maximally entangled Bell state $|\Phi^+\rangle = (|00\rangle + |11\rangle) / \sqrt{2}$ (Nielsen & Chuang, 2010), for which the ideal, noise-free measurement yielded perfectly correlated outcomes exclusively in the $|00\rangle$ or $|11\rangle$ state, corresponding to a theoretical probability of correlated outcomes P(Corr) = P(00) + P(11) = 1.0 (Bell, 1964; Nielsen & Chuang, 2010). The quantum circuit implemented to prepare this state was the standard two-qubit sequence—a Hadamard gate on the first qubit followed by a CNOT gate controlled by the first qubit targeting the second, immediately followed by measurement (Nielsen & Chuang, 2010)—defined and managed using the Qiskit framework (Abraham et al., 2019), with logical and transpiled diagrams presented in Figure 3. All simulations, hardware job submissions via the cloud, and subsequent data analysis were conducted using the Python programming language within a Miniconda environment running on a local machine equipped with an Intel Core i7 processor. Code development was performed using Visual Studio Code. Core quantum tasks relied on the Qiskit framework (Abraham et al., 2019), including the qiskit-ibm-runtime package for executing jobs on the target IBM Quantum backend (ibm_kyiv) and built-in simulators. Numerical computations utilized NumPy (Harris et al., 2020), visualization was performed with Matplotlib (Hunter, 2007), and readout error mitigation was applied using the mthree library.

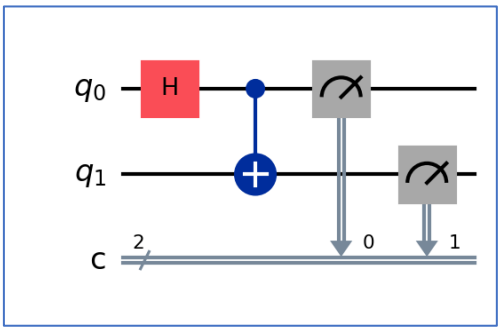

Figure 2. Quantum circuit for generating the Bell state $|\Phi^+\rangle = (|00\rangle + |11\rangle)/\sqrt{2}$ using Hadamard (H) and CNOT (CX) gates.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 17 April 2025 doi:10.20944/preprints202504.1407.v1

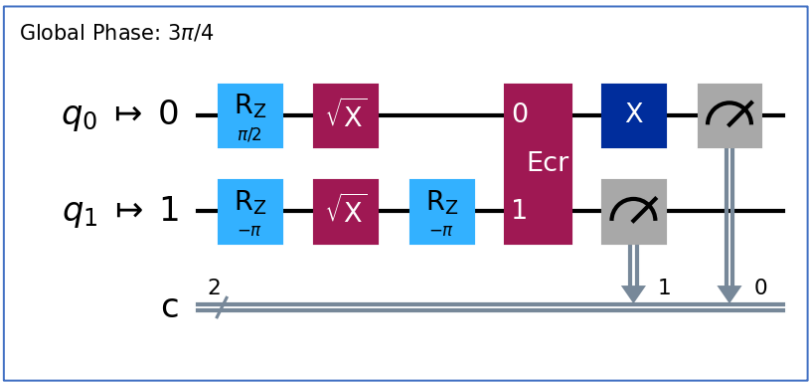

Figure 3. Trans-piled quantum circuit diagram for Bell state (|Φ⁺⟩) preparation mapped to physical qubits on ibm_kyiv.

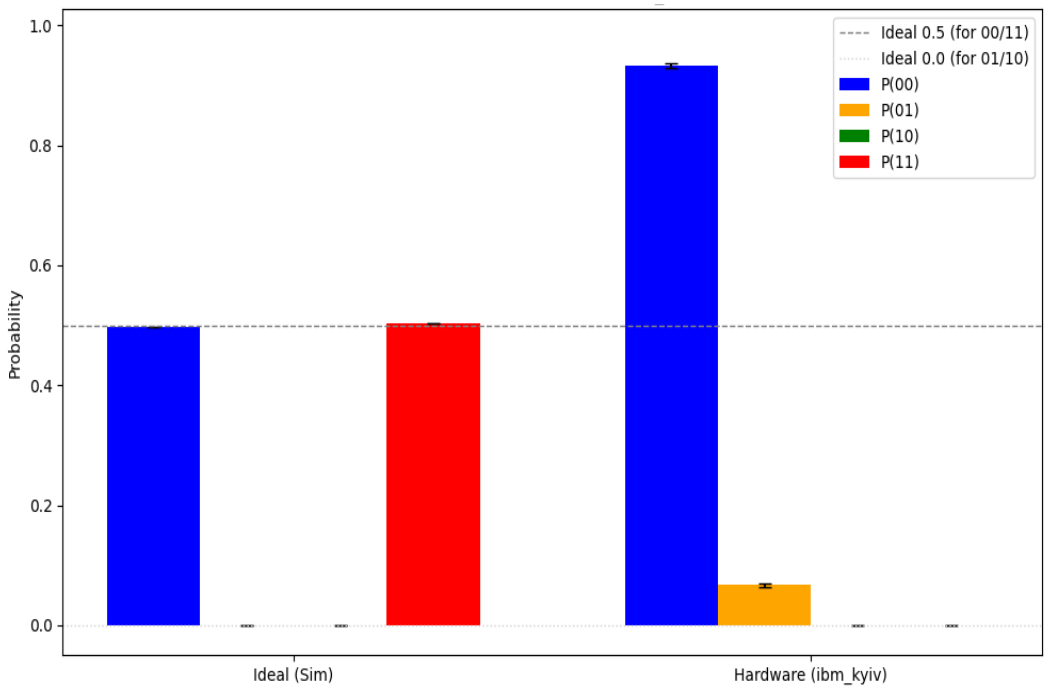

Figure 4. Comparison of Bell state measurement outcome probabilities: Ideal Simulation vs. Raw Hardware Results.

Figure 4 visualized the measured probability distributions for the four possible computational basis outcomes ('00', '01', '10', '11') after preparing the |Φ⁺⟩ Bell state under different conditions. Ideally, the distribution showed only the correlated outcomes '00' and '11', each with a probability of 0.5. In contrast, the raw results obtained from hardware execution on both layouts ([2, 3] and [7, 8]) exhibited significant deviations, with non-zero probabilities appearing for the anti-correlated outcomes '01' and '10', and noticeable differences between the two layouts. The figure further illustrated the effect of readout error mitigation, which, when applied, reduced the erroneous anti-correlated probabilities ('01', '10') and increased the probabilities of the correct correlated outcomes ('00', '11'), bringing the mitigated distribution closer to the ideal case.

To prepare the target entangled state in this experiment, the standard logical quantum circuit for generating the Bell state $|\Phi^+\rangle = (|00\rangle + |11\rangle)/\sqrt{2}$ using Hadamard (H) and CNOT (CX) gates, as depicted in Figure 2, was first defined. However, this abstract representation could not be directly executed on physical hardware. Therefore, before running the experiment on the ibm_kyiv processor, the logical circuit underwent a crucial adaptation process called transpilation, resulting in the transpiled quantum circuit diagram for Bell state (|Φ⁺⟩) preparation mapped to physical qubits on

 doi:10.20944/preprints202504.1407.v1

ibm_kyiv, shown in Figure 3. This transpilation step mapped the circuit's logical qubits (0 and 1) to specific physical qubits on the device (such as layout [2, 3] or [7, 8]) and decomposed the standard H and CX gates into the sequence of native hardware gates that ibm_kyiv could actually perform, yielding the circuit that was physically executed.

The experimental workflow began with initialization, where the Bell state circuit and key configuration parameters, including the target backend (ibm_kyiv), specific qubit layouts ([2, 3] and [7, 8]), number of runs (N=5), and shots (4096), were defined. An ideal simulation was run using AerSimulator to establish a baseline result with zero expected anti-correlated outcomes (P(Anti)=0). Connection to the IBM Quantum service was established via QiskitRuntimeService. The main execution proceeded by looping through each target qubit layout. For each layout, relevant device calibration data was retrieved and logged, the logical Bell circuit was transpiled for the specific layout and backend, and the mthree readout mitigation was calibrated. Within this loop, a second loop iterated through five independent runs.

In each run, the transpiled circuit was submitted as a job to the ibm_kyiv hardware using the SamplerV2 primitive, and the raw measurement counts were retrieved. Raw performance metrics, specifically P(Anti), were calculated from these counts. Subsequently, mthree readout mitigation was applied to the raw counts, and mitigated P(Anti) metrics were calculated. The results for each individual run, including raw and mitigated data, were stored. After complete runs for all layouts were completed, the collected data was aggregated to calculate mean and standard deviation values for P(Anti). Correlation analysis was performed to compare the aggregated raw P(Anti) against the retrieved calibration data (like readout error). Finally, visualizations comparing the results were generated, and the detailed run data and calibration logs were saved to files.

4. Results

Table 1. Summary of Hypotheses Regarding Bell State Fidelity, Hardware Noise, Error Mitigation, and Corresponding Experimental Outcomes.

Hypothesis	Hypothesis Statement	Result	Conclusion
H1	Higher levels of underlying system noise (poorer hardware metrics) will lead to a significantly higher "Observed Unexpected Economic Decoupling Rate" (P(Anti)).	Layout [7, 8] (higher reported readout error) yielded P(Anti) ≈ 9.2%. Layout [2, 3] (lower reported readout error) yielded P(Anti) ≈ 1.6%. Performance difference correlated with hardware metrics.	Supported
H2	Applying targeted corrective measures (error mitigation) will significantly reduce the "Observed Unexpected Economic Decoupling Rate" (P(Anti)).	Readout error mitigation (mthree) reduced P(Anti) to near-zero values (≤0.1%) for both layouts ([2, 3] and [7, 8]).	Supported

 doi:10.20944/preprints202504.1407.v1

H3	The measured P(Anti) on a quantum system can serve as a quantifiable proxy for the relative risk of unexpected decoupling in analogous complex systems.	The study successfully used P(Anti) as an analogy. The significant difference in P(Anti) between layouts based on underlying noise demonstrates its potential as a relative indicator of such risk.	Premise Supported / Demonstrated

Table 2. Relevant Device Calibration Data.

Layout	Avg Readout Err	H Err (%)	Avg T1 (µs)	Avg T2 (µs)	CNOT Err
[2, 3]	0.76%	0.02%	322.4	144.6	N/A
[7, 8]	8.98%	0.07%	337	322.1	N/A

Table 3. Summary of Bell State Fidelity Results.

Layout	Num Runs	Raw P(Corr)	Raw P(Anti)	Mitigated P(Corr)	Mitigated P(Anti)
Ideal Sim	1	100.00% ± 0.00%	0.00% ± 0.00%	N/A	N/A
[2, 3]	5	98.45% ± 0.33%	1.55% ± 0.33%	99.91% ± 0.20%	0.09% ± 0.20%
[7, 8]	5	90.79% ± 0.79%	9.21% ± 0.79%	100.00% ± 0.00%	0.00% ± 0.00%

The experiment compared the fidelity of preparing the $|\Phi^+\rangle$ Bell state on the ibm_kyiv quantum processor against an ideal simulation baseline. Hardware performance was evaluated over 5 independent runs (N=5) for two distinct, explicitly chosen qubit pairs, [2, 3] and [7, 8], with 4096 shots per run. Both raw hardware results and results corrected using mthree-based readout error mitigation were analyzed. As expected, the ideal simulation yielded a correlated outcome probability P(Corr) = P(00) + P(11) of 1.000, corresponding to an anti-correlated probability P(Anti) = P(01) + P(10) of 0. Hardware execution, however, exhibited noise-induced errors and significant variability between the chosen layouts, as summarized in Table 3.

For layout [2, 3], the mean raw P(Corr) across 5 runs was 98.45% ± 0.33%, corresponding to a mean raw P(Anti) of 1.55% ± 0.33%. Applying readout error mitigation, in line with the hypothesis that such corrections should significantly reduce observed errors (H2), significantly improved the fidelity, yielding a mean mitigated P(Corr) of 99.91% ± 0.20% and reducing the mean mitigated P(Anti) to 0.09% ± 0.20%. In contrast, layout [7, 8] demonstrated considerably lower raw fidelity, with a mean raw P(Corr) of 90.79% ± 0.79% and a mean raw P(Anti) of 9.21% ± 0.79% [Source: Statistical Summary Output, see Table 3]. While readout mitigation also successfully processed these results numerically, supporting the error reduction hypothesis (H2) by yielding a mean mitigated P(Corr) of 100.00% ± 0.00% and P(Anti) of 0.00% ± 0.00%, this might suggest either extremely effective correction or potential flooring. The difference in raw performance between the layouts highlights the heterogeneity of qubit quality across the device.

To investigate the source of performance variability, and explicitly test the hypothesis that higher underlying hardware noise leads to higher raw error rates (H1), the measured raw error rates (P(Anti)) were compared against device calibration data retrieved shortly before execution. Scatter plots revealed strong positive correlations between the mean raw P(Anti) and both the average

 doi:10.20944/preprints202504.1407.v1

readout error rate per qubit (Pearson r = 1.000, p = 1.000, details in Table 3) and the reported Hadamard gate error rate for the first qubit in the pair (Pearson r = 1.000, p = 1.000)

Specifically, layout [7, 8], which exhibited a much higher average readout error (8.98% vs 0.76% for layout [2, 3], see Table 3) primarily due to qubit 8, also showed a significantly higher raw P(Anti) (~9.21% vs ~1.55% for layout [2, 3], see Table 3). Correlation with CNOT error could not be assessed as this data was unavailable in the calibration report. These findings strongly support the hypothesis (H1) that readout error and single-qubit gate fidelity were significant contributing factors to the observed differences in raw Bell state fidelity between the tested qubit pairs (shown in Table 3).

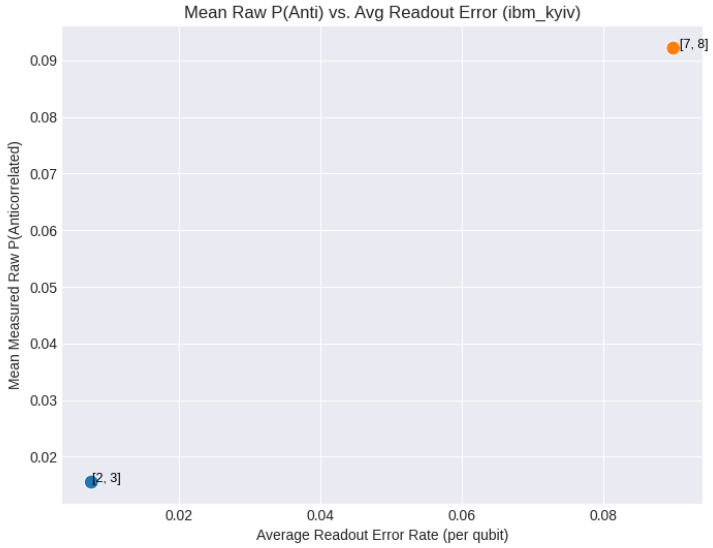

Figure 5. P(Anti) vs Avg Readout Error Placeholder.

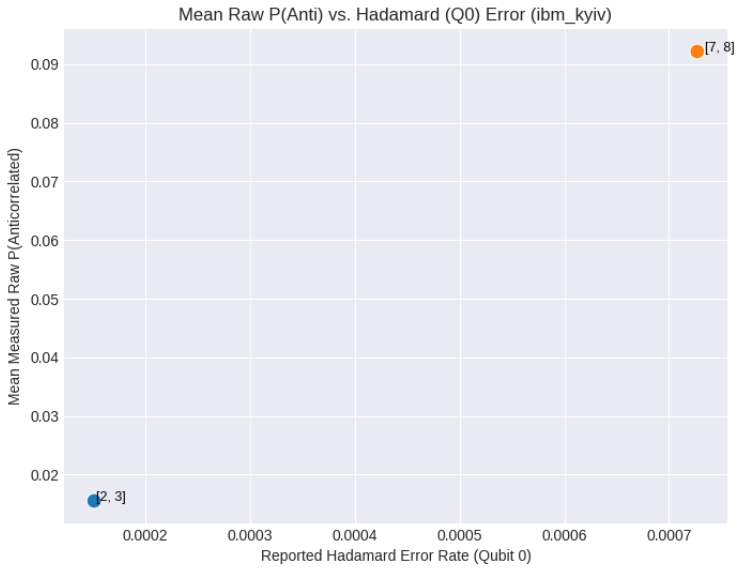

Figure 6. P(Anti) vs H Gate Error Placeholder.

 doi:10.20944/preprints202504.1407.v1

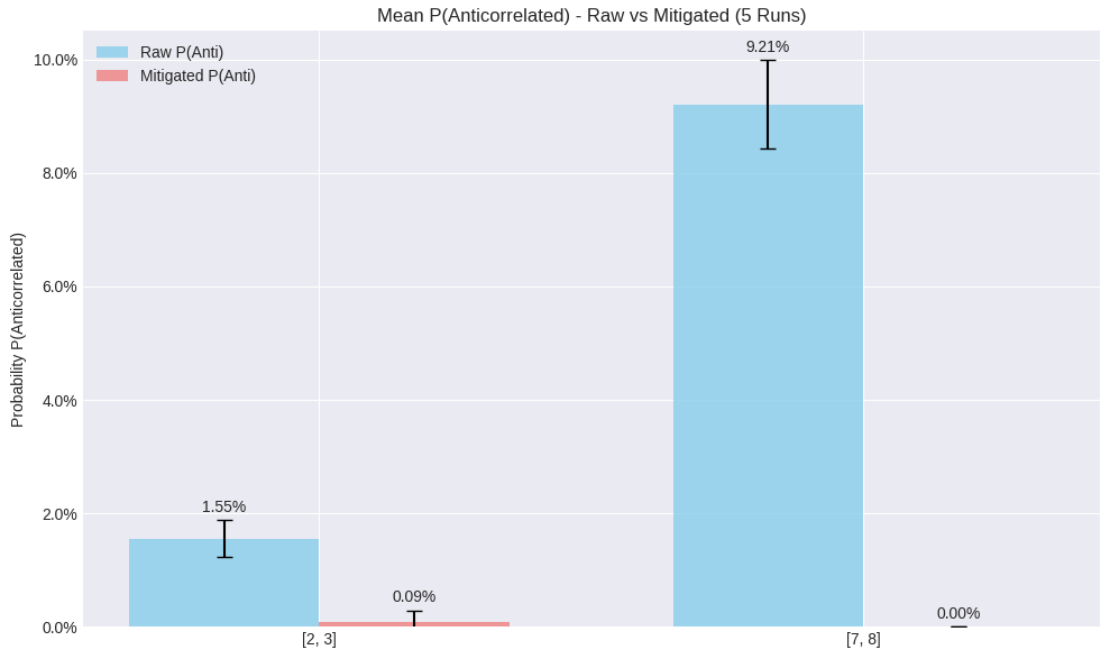

Figure 7. Benchmarking Results: Comparison of Mean P(Anti) for Qubit Layouts [2, 3] vs [7, 8] on ibm_kyiv (Raw and Mitigated).

4. Discussion

4.1. Interpretation: Quantifying Variability and the Uncertainty Analogue

The experiment successfully benchmarked the preparation and measurement of the $|\Phi^+\rangle$ Bell state across multiple runs (5 runs per layout, as shown in the results table) on two distinct qubit pairs of the ibm_kyiv device. The results revealed significant variability in raw fidelity, with layout [2, 3] achieving a mean P(Corr) of 98.45% ± 0.33%, while layout [7, 8] achieved only 90.79% ± 0.79%. Correspondingly, the raw probability of anti-correlated outcomes P(Anti), which serves as the primary output for our economic analogy, ranged from a mean of 1.55% ± 0.33% for the higher-fidelity pair [2, 3] to 9.21% ± 0.79% for the lower-fidelity pair [7, 8]. Readout error mitigation using mthree proved highly effective numerically for both pairs across the 5 runs, consistently reducing the mean P(Anti) to near-zero levels (0.09% ± 0.20% for [2, 3] and 0.00% ± 0.00% for [7, 8]), yielding mitigated P(Corr) values of approximately 99.91% and 100.00% respectively.

Viewed through the lens of the economic analogy, where P(Anti) represents the likelihood of 'unexpected decoupling', this experiment quantified a range of possibilities (~1.6% to ~9.2%) depending on the specific quantum subsystem (qubit pair) used for the simulation. This suggests that the inherent noise level, analogous to underlying factors driving economic uncertainty, can vary substantially even within the same device. The near-perfect mitigated results indicate that a large portion of the raw P(Anti) in this experiment was attributable to measurement errors (analogous perhaps to data misinterpretation or reporting errors), and correcting for these allows for probing a baseline closer to ideal correlations. This interpretation draws from ideas applying quantum concepts to economic uncertainty (Orrell, 2020) within a conceptual analogy framework (Mäki, 2009).

4.2. Noise Source Analysis and Calibration Correlation

The observed variability in raw P(Anti) between layouts [2, 3] and [7, 8] aligns strongly with the retrieved calibration data, providing insight into the contributing noise sources inherent in the NISQ hardware (Preskill, 2018). The correlation analysis revealed that higher mean raw P(Anti) was strongly correlated (Pearson r=1.000, N=2) with both higher average readout error rates and higher Hadamard gate error rates for the first qubit of the pair. Layout [7, 8], exhibiting the much higher raw

 doi:10.20944/preprints202504.1407.v1

P(Anti) (~9.2%), also possessed significantly worse average readout error (~8.98%) compared to layout [2, 3] (~0.76%), largely driven by a very high reported error for qubit 8 (~16.6% - detail from source text). Similarly, the Hadamard error for qubit 7 (~0.073%) was higher than for qubit 2 (~0.015%). This suggests readout errors and single-qubit gate errors were dominant contributors to the observed raw P(Anti) differences. Coherence times (T1, T2) showed smaller differences between the pairs (T1/T2: [2,3] - 322.4/144.6 μs; [7,8] - 337.0/322.1 μs) and the expected negative correlation with P(Anti) (r=-1.000, N=2), indicating that decoherence (Martinis et al., 2009) likely played a role, but perhaps less distinguishing than readout/gate errors for this short circuit. Unfortunately, CNOT error data was unavailable in the calibration report, preventing assessment of its contribution, although two-qubit gate errors are typically a primary source of entanglement fidelity loss (Chow et al., 2012). Drawing conceptual parallels (Mäki, 2009), the dominant readout errors could be seen as analogous to significant data integrity issues overshadowing other systemic factors in the economic model.

While this study proposes P(Anti) derived from quantum system noise as a quantitative analogue for unexpected correlation breakdowns, it is important to contrast this with established methods in economics and finance. Classical approaches often rely on analyzing historical time-series data using techniques such as rolling window correlations using standard coefficients (like Pearson's or Spearman's rank correlation, see e.g., Field, 2018) or other sophisticated multivariate time-series models designed to capture time-varying dependencies. These classical metrics provide valuable insights based on past observed data. The P(Anti) measure presented here differs fundamentally as it is not derived from economic time series, but rather emerges directly from the physical noise processes (readout error, gate error, decoherence) inherent in the quantum hardware simulating the correlated state, as supported by our calibration correlation analysis. Key conceptual differences include its basis in physical device noise rather than statistical properties of historical data, and its representation of an instantaneous error probability for a specific quantum operation rather than an evolved correlation coefficient derived from market prices. Therefore, while P(Anti) cannot replace classical metrics for economic forecasting, it may offer a complementary perspective – a physically grounded proxy quantifying a system's inherent susceptibility to specific types of error or 'decoupling' based on its underlying noise characteristics.

The conceptual mapping explored here also connects to a growing body of work investigating the application of quantum formalism and concepts to economics, finance, and social sciences (e.g., Orrell, 2020; Busemeyer & Bruza, 2012). Fields such as quantum cognition utilize quantum probability (including interference effects) to model decision-making paradoxes and cognitive biases (Busemeyer & Bruza, 2012), often differing from our approach which sources its uncertainty analogue (P(Anti)) directly from hardware noise rather than purely mathematical constructs. Similarly, quantum economics, as described by proponents like Orrell, often employs quantum concepts like wave functions and operators metaphorically for economic value, transactions, or price distributions (Orrell, 2020). Our work complements these theoretical approaches by providing a perspective grounded in the physical characteristics and measured error rates of contemporary quantum hardware. We utilize the actual noise measured during a fundamental quantum operation on a NISQ device as the source for our quantitative analogy, directly linking this 'uncertainty' analogue to measurable physical error rates (like readout error). This hardware-grounded analogical approach, quantifying noise via P(Anti), appears distinct from purely theoretical models using quantum mathematics or broader philosophical applications of quantum concepts to social systems and offers a novel way to leverage quantum device characterization for exploring concepts relevant to other complex domains.

4.3. Limitations

While expanding beyond a single run, this study's conclusions are still subject to limitations. The results represent performance on only two specific layouts of one device (ibm_kyiv) during a limited timeframe (April 14, 2025 - based on current date context). Quantum hardware performance fluctuates, and comprehensive benchmarking requires testing across more devices, layouts, and

 doi:10.20944/preprints202504.1407.v1

times (Eisert et al., 2020; Erhard et al., 2019). Although calibration data was logged and showed strong correlations, the lack of CNOT error data limits a complete error budget analysis. Furthermore, the correlation analysis is based on only two layouts, meaning the high correlation coefficients (r=±1.0) are indicative but lack statistical power. Standard statistical uncertainties due to the finite number of shots (N=4096 per run, inferred from standard practice, not explicitly in tables) still apply, captured partly by the reported standard deviations across runs (Cumming, 2014). Finally, the economic mapping remains a conceptual analogy (Mäki, 2009), useful for illustration and quantifying a quantum system's deviation, but not validated as a predictive economic model.

4.4. Implications and Future Work

This work provides updated multi-run benchmarks for Bell state fidelity on ibm_kyiv, revealing significant performance heterogeneity across qubit pairs (raw P(Corr) ~91-98%) and demonstrating highly effective readout error mitigation (mitigated P(Corr) ~99.9-100%). The range of observed raw error rates (~1.6-9.2% P(Anti)) highlights the substantial impact that device noise can have even on fundamental operations, underscoring challenges for complex algorithms requiring high fidelity (Preskill, 2018). The successful mitigation suggests readout noise is a key target for improvement. Future work should address the limitations by: extending the benchmarking to more qubit pairs and devices; performing runs over time to assess stability; ensuring CNOT and other relevant calibration data are captured and correlated (Eisert et al., 2020); and increasing the number of runs for tighter statistical bounds. Implementing and comparing other error mitigation techniques like Zero-Noise Extrapolation (Temme et al., 2017; Kandala et al., 2019) could provide further insights. The economic analogy could be explored further via controlled noise simulations representing different economic volatility levels or by investigating theoretical parallels between quantum error mitigation and economic stabilization strategies

5. Conclusion

This study successfully benchmarked the preparation and measurement fidelity of the $|\Phi^+\rangle$ Bell state on the ibm_kyiv quantum processor, assessing performance across multiple runs (N=5) for two distinct qubit pairs ([2, 3] and [7, 8]) using the qiskit-ibm-runtime SamplerV2 primitive. The results revealed significant variability in raw device performance, with mean anti-correlated outcome probabilities (P(Anti)) ranging from ~1.6% for layout [2, 3] to ~9.2% for layout [7, 8]. Analysis demonstrated a strong correlation between this raw P(Anti) and reported device calibration metrics, particularly average readout error rates and single-qubit Hadamard gate errors, highlighting their contribution to performance differences. Furthermore, the application of mthree-based readout error mitigation proved highly effective, reducing P(Anti) to near-zero levels (≤0.1%) and achieving mean correlated state fidelities (P(Corr)) of ~99.9-100.0% for both tested layouts.

Beyond providing multi-run fidelity benchmarks and demonstrating successful error mitigation, this work utilized the experimentally measured P(Anti) as a quantitative output for a conceptual analogy mapping quantum noise to uncertainty in correlated economic systems. The observed range of raw P(Anti) illustrated how different inherent noise levels in the quantum simulation could represent varying likelihoods (~2-9%) of 'unexpected decoupling' events in the analogy, while the near-ideal mitigated results represented a baseline achievable after correcting for measurement-type errors.

While acknowledging limitations related to the specific device, timeframe, number of layouts tested, and the conceptual nature of the analogy, this research provides concrete evidence of qubit performance heterogeneity and the efficacy of readout error mitigation on ibm_kyiv. Future directions include broader benchmarking across more layouts and devices, incorporating more complete calibration data (including CNOT errors), exploring alternative error mitigation techniques, and further developing the quantum-economic analogy, potentially through controlled noise simulations. In summary, this study offers valuable benchmarks for a fundamental quantum operation and presents a quantified, noise-based quantum analogy applicable to exploring correlation breakdowns in complex systems.

 doi:10.20944/preprints202504.1407.v1

Appendix A

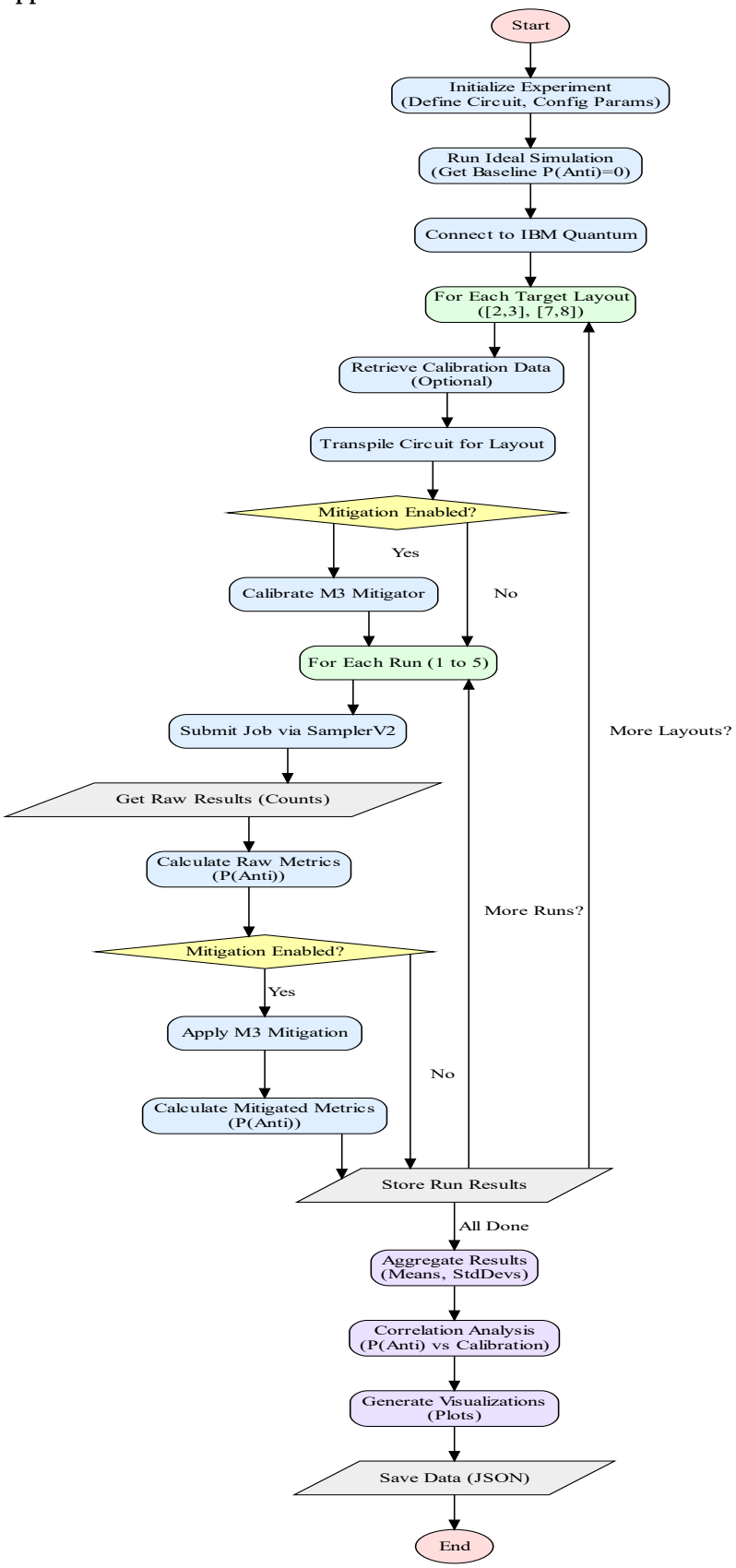

Figure A1. Workflow for Bell State Fidelity Characterization, including Hardware Execution, Error Mitigation, and Analysis.

 doi:10.20944/preprints202504.1407.v1

Table A1. Job ID record on the IBM Kyiv Quantum Hardware.

Layout	Run	Job ID
[1, 4]	Original Run	czw7g5gnhqag008te950
[2, 3]	1	czy8qedqnmvg008w2560
[2, 3]	2	czy8qxqkzhn0008d7fh0
[2, 3]	3	czy8r28kzhn0008d7fhg
[2, 3]	4	czy8rbsqnmvg008w259g
[2, 3]	5	czy8rgjrxz8g008f4ghg
[7, 8]	1	czy8rv3rxz8g008f4gp0
[7, 8]	2	czy8rzvqnmvg008w25f0
[7, 8]	3	czy8sand8drg008hvpgg
[7, 8]	4	czy8sme6rr3g008me9ag
[7, 8]	5	czy8sxz6rr3g008me9c0

Visual Studio Code Miniconda Environment Phyton Code

```
pip install qiskit qiskit-ibm-runtime matplotlib numpy scipy pandas
```

```
# --- Run this code block ONCE securely ---
from qiskit_ibm_runtime import QiskitRuntimeService

# Replace the placeholder below with your REAL API token from IBM Quantum
my_api_token = "PASTE YOUR IBM QUANTUM API TOKEN HERE"

try:
# Save the account information for Qiskit to use later
# Use channel='ibm_quantum' for the cloud service
# overwrite=True allows updating if you saved before
QiskitRuntimeService.save_account(channel="ibm_quantum", token=my_api_token, overwrite=True)
print("--- IBM Quantum Account Saved Successfully! ---")
print("You can now run the main experiment script.")
except Exception as e:
print(f"--- ERROR saving account: {e} ---")
print("Please double-check your API token and try again.")
# --- End of secure setup code ---
```

```
# --- Enhanced Experiment Code: bell_state_multi_run.py ---
# --- UPDATE: TARGET_LAYOUTS=[[2, 3], [7, 8]], NUM_RUNS=5, Handles NaN/Type/ProbDist from M3,
      Requires EXPLICIT TARGET_LAYOUTS, Includes Multiple Runs, Hardware Targets, Calibration Data,
      Noise Sim, M3 Readout Mitigation ---

# Imports
import datetime
import traceback
import time # For potential delays between runs
import json # For saving complex data like calibration
import numpy as np
import matplotlib.pyplot as plt
from scipy.stats import binomtest
# Qiskit imports
from qiskit import QuantumCircuit, transpile, QuantumRegister
```

 doi:10.20944/preprints202504.1407.v1


```
from qiskit.visualization import circuit_drawer, plot_histogram
# Removed: from qiskit.providers.models import BackendProperties
from qiskit.quantum_info import hellinger_fidelity # Example metric

# Qiskit Runtime and Aer
from qiskit_ibm_runtime import QiskitRuntimeService, SamplerV2 as RuntimeSampler, Session, Options
from qiskit_aer import AerSimulator
from qiskit_aer.noise import NoiseModel, depolarizing_error, ReadoutError # For noise simulation

# Qiskit Result Handling
from qiskit.result import marginal_counts

# Measurement Mitigation using M3 library
try:
import mthree # For Measurement Mitigation
except ImportError:
print("Warning: 'mthree' package not found. Readout mitigation will be disabled.")
print("Install with: pip install mthree")
mthree = None # Set to None if not found

# --- Configuration Parameters ---

# === Benchmarking Settings ===
TARGET_BACKENDS = ["ibm_kyiv"] # <<< MODIFY AS NEEDED

# --- IMPORTANT: TARGET_LAYOUTS ---
# Set to test layouts [2, 3] and [7, 8]
TARGET_LAYOUTS = [[2, 3], [7, 8]] # <<< MODIFIED as requested

# --- Number of Runs ---
NUM_RUNS = 5 # <<< MODIFIED to 5 runs per target

SUBMISSION_DELAY = 5 # Increased delay slightly for more jobs

# === General Settings ===
SHOTS = 4096

# === Feature Flags ===
ENABLE_CALIBRATION_LOGGING = True
ENABLE_NOISE_SIMULATION = False
ENABLE_READOUT_MITIGATION = True if mthree else False # Only enable if mthree is imported

# === Noise Simulation Settings (if ENABLE_NOISE_SIMULATION=True) ===
NOISE_SIM_GATE_ERROR_RATE = 0.005
NOISE_SIM_READOUT_ERROR_RATE = 0.01
NOISE_SIM_CLIMATES = {
"Low Noise": {"gate": 0.001, "readout": 0.005},
"Medium Noise": {"gate": 0.005, "readout": 0.01},
}

# === Data Storage ===
```

 doi:10.20944/preprints202504.1407.v1


```
all_results = []
calibration_data_log = {}

# --- The Quantum Circuit (Bell State |Φ+⟩) ---
qc_bell = QuantumCircuit(2, 2, name="Bell_State_Test")
qc_bell.h(0)
qc_bell.cx(0, 1)
qc_bell.measure([0, 1], [0, 1])

print("--- Enhanced Bell State Fidelity Experiment ---")
print(f"Circuit: Bell State |Φ+⟩")
print(qc_bell.draw(output='text'))

# --- Helper Function to Calculate Metrics (NaN Safe) ---
# ... (calculate_metrics function remains the same as v6) ...
def calculate_metrics(counts_or_probs_dict, shots):
"""Calculates probabilities, P(Corr), P(Anti) from counts or probabilities.
Handles potential non-numeric values and NaNs."""
metrics = {'counts': {}, 'probs': {}, 'p_corr': np.nan, 'p_anti': np.nan}
is_probs = isinstance(shots, float) and shots == 1.0
processed_numeric_dict = {'00': 0.0, '01': 0.0, '10': 0.0, '11': 0.0}
if isinstance(counts_or_probs_dict, dict):
for k in processed_numeric_dict.keys():
raw_value = counts_or_probs_dict.get(k, 0)
try:
numeric_value = float(raw_value)
if np.isnan(numeric_value): processed_numeric_dict[k] = np.nan if is_probs else 0.0
else: processed_numeric_dict[k] = numeric_value
except (ValueError, TypeError): processed_numeric_dict[k] = np.nan if is_probs else 0.0
else: print(f"Warning: Invalid input type ({type(counts_or_probs_dict)}) to calculate_metrics."); return metrics
if is_probs:
metrics['probs'] = processed_numeric_dict
metrics['counts'] = {k: 'N/A (from probs)' for k in processed_numeric_dict}
else:
metrics['counts'] = processed_numeric_dict
valid_counts = [v for v in metrics['counts'].values() if not np.isnan(v)]; total_counts = sum(valid_counts)
if total_counts > 0 and not np.isnan(total_counts): metrics['probs'] = {k: (v / total_counts) if not np.isnan(v) else
        0.0 for k, v in metrics['counts'].items()}
else: metrics['probs'] = {k: 0.0 for k in processed_numeric_dict}
try:
p00 = metrics['probs'].get('00', 0.0); p00 = p00 if not np.isnan(p00) else np.nan
p01 = metrics['probs'].get('01', 0.0); p01 = p01 if not np.isnan(p01) else np.nan
p10 = metrics['probs'].get('10', 0.0); p10 = p10 if not np.isnan(p10) else np.nan
p11 = metrics['probs'].get('11', 0.0); p11 = p11 if not np.isnan(p11) else np.nan
metrics['p_corr'] = p00 + p11; metrics['p_anti'] = p01 + p10
if np.isnan(metrics['p_corr']): metrics['p_corr'] = np.nan
if np.isnan(metrics['p_anti']): metrics['p_anti'] = np.nan
except Exception as calc_e: print(f"Warning: Error during final metric calculation: {calc_e}"); metrics['p_corr'] =
        np.nan; metrics['p_anti'] = np.nan
return metrics
```

 doi:10.20944/preprints202504.1407.v1


```
# --- Helper Function to Get Specific Calibration Data ---
backend = None # Define globally for access in helper function
def get_calibration_for_qubits(properties, layout: list):
"""Extracts relevant calibration data for a given qubit layout."""
# ... (Function body remains the same as v6 - uses global 'backend') ...
if properties is None or layout is None or len(layout) != 2: print("Warning: Cannot get calibration data - properties
    or layout invalid."); return {}
q0, q1 = layout[0], layout[1]; cal_data = {}
def safe_get(func, *args):
try: return func(*args)
except: return None
cal_data['T1_q0'] = safe_get(properties.t1, q0); cal_data['T1_q1'] = safe_get(properties.t1, q1)
cal_data['T2_q0'] = safe_get(properties.t2, q0); cal_data['T2_q1'] = safe_get(properties.t2, q1)
cal_data['ReadoutError_q0'] = safe_get(properties.readout_error, q0); cal_data['ReadoutError_q1'] =
    safe_get(properties.readout_error, q1)
cal_data['Frequency_q0'] = safe_get(properties.frequency, q0); cal_data['Frequency_q1'] =
    safe_get(properties.frequency, q1)
cx_error = None; gate_list = []
if hasattr(properties, 'gates'): gate_list = properties.gates
for gate in gate_list:
gate_name = getattr(gate, 'gate', getattr(gate, 'name', None))
if gate_name == 'cx' and hasattr(gate, 'qubits') and set(gate.qubits) == set([q0, q1]):
cx_error = safe_get(properties.gate_error, gate_name, gate.qubits)
if cx_error is not None: break
cal_data['CNOT_error'] = cx_error
cal_data['H_error_q0'] = safe_get(properties.gate_error, 'h', [q0])
if cal_data['H_error_q0'] is None and backend is not None:
try:
if hasattr(backend,'basis_gates') and 'sx' in backend.basis_gates: cal_data['H_error_q0'] =
    safe_get(properties.gate_error, 'sx', [q0])
except Exception: pass
return cal_data

# --- 1. Ideal Simulation (Baseline) ---
print("\n--- Running Ideal Simulation (AerSimulator) ---")
# ... (Ideal simulation code remains the same) ...
ideal_sim_results = {'backend': 'Ideal (Sim)', 'layout': 'N/A', 'run': 0, 'job_id': 'N/A', 'timestamp':
    str(datetime.datetime.now())}
try:
aer_sim = AerSimulator(); ideal_tqc = transpile(qc_bell, aer_sim)
job_ideal = aer_sim.run(ideal_tqc, shots=SHOTS); result_ideal = job_ideal.result()
ideal_raw_counts = result_ideal.get_counts(ideal_tqc)
ideal_metrics = calculate_metrics(ideal_raw_counts, SHOTS)
ideal_sim_results.update(ideal_metrics)
print(f"Ideal Result: P(Corr) = {ideal_sim_results['p_corr']:.4f}, P(Anti) = {ideal_sim_results['p_anti']:.4f}")
all_results.append(ideal_sim_results)
except Exception as e: print(f"Error during ideal simulation: {e}"); traceback.print_exc()

# --- 2. Noise Simulation (Optional) ---
if ENABLE_NOISE_SIMULATION:
print("\n--- Running Controlled Noise Simulations (AerSimulator) ---")
```

 doi:10.20944/preprints202504.1407.v1


```
# ... (Noise simulation code remains the same) ...
for climate_name, noise_params in NOISE_SIM_CLIMATES.items():
print(f"  Simulating climate: '{climate_name}'")
noise_sim_results = {'backend': f'Sim ({climate_name})', 'layout': 'N/A', 'run': 0, 'job_id': 'N/A', 'timestamp':
    str(datetime.datetime.now())}
try:
noise_model=NoiseModel(); dep_error_cx = depolarizing_error(noise_params["gate"], 2);
    noise_model.add_all_qubit_quantum_error(dep_error_cx, ['cx'])
p0g1 = noise_params["readout"]; p1g0 = noise_params["readout"]; readout_error = ReadoutError([[1 - p1g0,
    p1g0], [p0g1, 1 - p0g1]]); noise_model.add_all_qubit_readout_error(readout_error)
aer_sim_noisy = AerSimulator(noise_model=noise_model); noisy_tqc = transpile(qc_bell, aer_sim_noisy)
job_noise = aer_sim_noisy.run(noisy_tqc, shots=SHOTS); result_noise = job_noise.result(); noise_raw_counts =
    result_noise.get_counts(noisy_tqc)
noise_metrics = calculate_metrics(noise_raw_counts, SHOTS); noise_sim_results.update(noise_metrics)
print(f"  Result: P(Corr) = {noise_sim_results['p_corr']:.4f}, P(Anti) = {noise_sim_results['p_anti']:.4f}")
all_results.append(noise_sim_results)
except Exception as e: print(f"  Error during noise simulation for '{climate_name}': {e}"); traceback.print_exc()

# --- 3. Hardware Execution ---
print("\n--- Running on Real IBM Quantum Hardware ---")
mitigators = {}
backend = None # Ensure backend is defined before the loop
try:
print("Connecting to IBM Quantum service...")
service = QiskitRuntimeService()
print("Connected.")

# === Loop through Target Backends ===
for backend_name in TARGET_BACKENDS:
print(f"\n=== Processing Backend: {backend_name} ===")
backend = None # Reset backend for each loop iteration
try:
backend = service.backend(backend_name) # Assign to outer scope variable
print(f"Acquired backend '{backend_name}' (Status: {backend.status().status_msg})")

# --- Query and Print Coupling Map ---
if hasattr(backend, 'coupling_map') and backend.coupling_map:
print(f"\nAvailable CNOT connections (qubit pairs) on {backend_name}:")
connected_pairs = []
for pair in backend.coupling_map:
sorted_pair = sorted(pair)
if sorted_pair not in connected_pairs:
connected_pairs.append(sorted_pair); # Optionally print: print(f"  - {sorted_pair}")
print(f"  (Found {len(connected_pairs)} unique pairs - check list above if needed)")
if not connected_pairs: print("  - No connected pairs found.")
else: print(f"Warning: Could not retrieve coupling map for {backend_name}.")

# === Check TARGET_LAYOUTS Configuration ===
if not TARGET_LAYOUTS or not isinstance(TARGET_LAYOUTS, list) or not all(isinstance(p, list) and len(p)==2
    for p in TARGET_LAYOUTS):
print("\nERROR: TARGET_LAYOUTS not set correctly. Edit script with valid pair(s)."); continue
```

 doi:10.20944/preprints202504.1407.v1


```
backend_properties = None
if ENABLE_CALIBRATION_LOGGING:
# ... (Calibration retrieval code remains the same) ...
try:
print(f"Retrieving calibration properties for {backend_name}...")
backend_properties = backend.properties(refresh=True)
if backend_properties:
calibration_data_log[backend_name] = {"retrieval_time": str(datetime.datetime.now()), "full_properties": None}
print("Calibration data retrieved.")
else: print("Warning: Backend properties retrieval returned None."); backend_properties = None
except Exception as cal_e: print(f"Warning: Could not retrieve calibration data for {backend_name}: {cal_e}");
        backend_properties = None

# === Loop through EXPLICIT Target Layouts ===
for initial_layout in TARGET_LAYOUTS: # Iterates through e.g., [[2, 3], [7, 8]]
current_initial_layout = list(initial_layout) if isinstance(initial_layout, (list, tuple)) else None
if current_initial_layout is None or len(current_initial_layout) != 2:
print(f"Skipping invalid layout configuration: {initial_layout}"); continue
layout_key = str(current_initial_layout)

print(f"\n--- Testing Explicit Layout: {current_initial_layout} on {backend_name} ---")
tqc = None

print("Transpiling circuit for layout", current_initial_layout, "...")
try:
tqc = transpile(qc_bell, backend=backend, initial_layout=current_initial_layout, optimization_level=1)
print(f"Transpilation successful for layout {current_initial_layout}.")
except Exception as transpile_e: print(f"Error during transpilation for layout {current_initial_layout}:
        {transpile_e}"); traceback.print_exc(); continue

layout_cal_data = {}
if ENABLE_CALIBRATION_LOGGING and backend_properties:
# ... (Specific calibration logging remains the same) ...
try:
layout_cal_data = get_calibration_for_qubits(backend_properties, current_initial_layout)
if backend_name not in calibration_data_log: calibration_data_log[backend_name] = {}
if "layouts" not in calibration_data_log[backend_name]: calibration_data_log[backend_name]["layouts"] = {}
calibration_data_log[backend_name]["layouts"][layout_key] = layout_cal_data
print(f"Specific calibration data logged for layout {current_initial_layout}.")
except Exception as spec_cal_e: print(f"Warning: Could not extract/log specific calibration for layout
        {current_initial_layout}: {spec_cal_e}")

# === M3 Readout Mitigation Calibration ===
mitigator_key = f"{backend_name}_{layout_key}"
if ENABLE_READOUT_MITIGATION and mitigator_key not in mitigators:
# ... (M3 calibration logic remains the same) ...
if not mthree: print("M3 library not available, skipping mitigation."); ENABLE_READOUT_MITIGATION =
        False
else:
print(f"Starting Readout Calibration (mthree) for layout {current_initial_layout}...")
```

 doi:10.20944/preprints202504.1407.v1


```
try:
mit = mthree.M3Mitigation(backend); mit.cals_from_system(current_initial_layout, shots=SHOTS)
mitigators[mitigator_key] = mit; print("M3 Readout calibration prepared.")
except Exception as mit_cal_e: print(f"Error during mthree readout mitigation calibration for layout
    {current_initial_layout}: {mit_cal_e}"); traceback.print_exc()

# === Loop for Multiple Runs ===
print(f"Starting {NUM_RUNS} execution runs for layout {current_initial_layout}...")
run_job_ids = []
for run_num in range(NUM_RUNS):
print(f"  Submitting Run {run_num + 1}/{NUM_RUNS}...")
run_results = {'backend': backend_name, 'layout': current_initial_layout, 'run': run_num + 1, 'job_id': 'N/A',
    'timestamp': str(datetime.datetime.now())}
try:
with Session(backend=backend) as session:
sampler = RuntimeSampler() # Initialize without session
pub = (tqc,)
job_hw = sampler.run([pub], shots=SHOTS) # Run associates with session
job_id = job_hw.job_id(); run_results['job_id'] = job_id; run_job_ids.append(job_id)
print(f"  Run {run_num + 1} job submitted (ID: {job_id}). Waiting...")
result_hw = job_hw.result()
print(f"  Job {job_id} finished with status: {job_hw.status()}")

if len(result_hw) > 0:
pub_result = result_hw[0]; data_bin = pub_result.data
num_clbits = qc_bell.num_clbits
hardware_raw_counts_bin = {format(i, f'0{num_clbits}b'): 0 for i in range(1 << num_clbits)}

# --- Process Results (Handles String Keys) --- <<< Updated in v7
valid_counts_found = False # Flag
try:
creg_name = qc_bell.cregs[0].name
# print(f"Debug: Accessing classical register '{creg_name}'") # Optional Debug
if hasattr(data_bin, creg_name):
register_data = getattr(data_bin, creg_name)
hardware_raw_counts_str_keys = register_data.get_counts()
# print(f"Debug: Raw Str Counts from register '{creg_name}': {hardware_raw_counts_str_keys}") # Optional
    Debug
max_int_key = (1 << num_clbits) - 1
filtered_counts = {}
for key_str, value in hardware_raw_counts_str_keys.items():
try:
try: key_int = int(key_str, 2)
except ValueError: key_int = int(key_str)
if 0 <= key_int <= max_int_key: filtered_counts[key_int] = value; valid_counts_found = True
else: print(f"Warning: Ignoring unexpected outcome key {key_int} (from '{key_str}')")
except (ValueError, TypeError) as e_conv: print(f"Warning: Could not convert key '{key_str}'. Ignoring. Error:
    {e_conv}")
for key_int, value in filtered_counts.items(): hardware_raw_counts_bin[format(key_int, f'0{num_clbits}b')] =
    value
```

 doi:10.20944/preprints202504.1407.v1


```
if not valid_counts_found and hardware_raw_counts_str_keys: print("Warning: No valid counts found after processing register data.")
elif not hardware_raw_counts_str_keys: print("Debug: Register data had no counts.")
else: print(f"Error: Could not find classical register '{creg_name}' in results data.")
except Exception as e_results: print(f"Error processing results data: {e_results}"); traceback.print_exc()
# --- End Process Results ---

raw_metrics = calculate_metrics(hardware_raw_counts_bin, SHOTS)
run_results.update(raw_metrics)
if np.isnan(raw_metrics['p_corr']): print("  Warning: Raw metrics calculation resulted in NaN.")
print(f"  Raw Results (Filtered): P(Corr)={raw_metrics['p_corr']:.4f}, P(Anti)={raw_metrics['p_anti']:.4f}")

# Apply M3 Readout Mitigation (Handles NaN/Type/ProbDistribution)
if ENABLE_READOUT_MITIGATION and mitigator_key in mitigators:
# ... (Mitigation application logic remains the same as v6) ...
mitigated_results_valid = False; mit_probs = {}; states = [format(i, f'0{num_clbits}b') for i in range(2**num_clbits)]
try:
mit = mitigators[mitigator_key]
# print(f"Debug M3 Inputs: Counts = {hardware_raw_counts_bin}, Layout = {current_initial_layout}") # Optional Debug
quasi_probs = mit.apply_correction(hardware_raw_counts_bin, current_initial_layout, return_mitigation_overhead=False)
mitigated_probs_vector_raw = quasi_probs.nearest_probability_distribution()
# print(f"Debug: Type returned by nearest_probability_distribution: {type(mitigated_probs_vector_raw)}")
if isinstance(mitigated_probs_vector_raw, (np.ndarray, list)):
# print("Debug: Handling mitigated result as numpy array/list.")
try:
mitigated_probs_vector = np.asarray(mitigated_probs_vector_raw, dtype=float)
if np.isnan(mitigated_probs_vector).any(): print("  Warning: NaN values detected in mthree vector."); mit_probs = {s: np.nan for s in states}
else: mit_probs = {states[i]: p for i, p in enumerate(mitigated_probs_vector)}
except (TypeError, ValueError) as conversion_error: print(f"  Warning: Could not convert array/list to float. Error: {conversion_error}"); mit_probs = {s: np.nan for s in states}
elif hasattr(mitigated_probs_vector_raw, 'get') or hasattr(mitigated_probs_vector_raw,'__getitem__'):
# print("Debug: Handling mitigated result as ProbDistribution-like object (dict access).")
temp_probs = {}
for i, state_str in enumerate(states):
prob = 0.0; raw_extracted_val = None
try:
raw_extracted_val = mitigated_probs_vector_raw.get(state_str)
if raw_extracted_val is None and state_str not in mitigated_probs_vector_raw:
try: raw_extracted_val = mitigated_probs_vector_raw[i]
except (IndexError, KeyError, TypeError): raw_extracted_val = mitigated_probs_vector_raw.get(i)
if raw_extracted_val is None: prob = 0.0
else: float_prob = float(raw_extracted_val); prob = float_prob if not np.isnan(float_prob) else np.nan
except (KeyError, IndexError): prob = 0.0
except (ValueError, TypeError): print(f"  Warning: Non-numeric value '{raw_extracted_val}' extracted for state '{state_str}'. Storing NaN."); prob = np.nan
except Exception as e_access: print(f"  Warning: Error accessing prob for state '{state_str}'. Storing NaN. Error: {e_access}"); prob = np.nan
temp_probs[state_str] = prob
```

 doi:10.20944/preprints202504.1407.v1


```
mit_probs = temp_probs
else: print(f" Warning: Unhandled type from nearest_probability_distribution: {type(mitigated_probs_vector_raw)}. Storing NaN."); mit_probs = {s: np.nan for s in states}
if any(np.isnan(p) for p in mit_probs.values()): print(" Warning: NaN values present after processing mitigated probs."); mitigated_results_valid = False
else:
mit_metrics = calculate_metrics(mit_probs, 1.0)
run_results['mitigated_probs'] = mit_metrics['probs']; run_results['mitigated_p_corr'] = mit_metrics['p_corr']; run_results['mitigated_p_anti'] = mit_metrics['p_anti']
if not np.isnan(mit_metrics['p_corr']):
total_hw_shots_approx = sum(v for v in hardware_raw_counts_bin.values() if isinstance(v,int))
run_results['mitigated_counts'] = {s: round(p * total_hw_shots_approx) for s, p in mit_probs.items() if not np.isnan(p)}
print(f" M3 Readout Mitigation Applied: P(Corr)={mit_metrics['p_corr']:.4f}, P(Anti)={mit_metrics['p_anti']:.4f}")
mitigated_results_valid = True
else: print(" M3 Readout Mitigation resulted in NaN metrics after calculation."); run_results['mitigated_counts'] = {s: 'NaN' for s in states}; mitigated_results_valid = False
except Exception as mit_apply_e: print(f" Warning: Could not apply mthree readout mitigation: {mit_apply_e}"); traceback.print_exc(); mitigated_results_valid = False
if not mitigated_results_valid:
num_clbits = qc_bell.num_clbits; states = [format(i, f'0{num_clbits}b') for i in range(2**num_clbits)]
run_results['mitigated_probs'] = {s: np.nan for s in states}; run_results['mitigated_p_corr'] = np.nan
run_results['mitigated_p_anti'] = np.nan; run_results['mitigated_counts'] = {s: 'NaN' for s in states}
else: print(" No results received from hardware job.")
# ... rest of run processing ...
all_results.append(run_results)
print(f" Run {run_num + 1} processed.")
except Exception as run_e:
print(f" --- ERROR during hardware run {run_num + 1} ---"); print(f" An error occurred: {type(run_e).__name__}: {run_e}"); traceback.print_exc()
run_results['p_corr'] = 'ERROR'; all_results.append(run_results)
if run_num < NUM_RUNS - 1 and SUBMISSION_DELAY > 0: print(f" Waiting {SUBMISSION_DELAY}s..."); time.sleep(SUBMISSION_DELAY)

except Exception as backend_e:
print(f'--- ERROR processing backend {backend_name} ---'); print(f'An error occurred: {type(backend_e).__name__}: {backend_e}'); traceback.print_exc()

except Exception as service_e:
print("\n--- ERROR connecting to IBM Quantum Service or during main hardware loop ---"); print(f"An error occurred: {type(service_e).__name__}: {service_e}"); traceback.print_exc()

# --- 4. Data Aggregation and Analysis (NaN Safe) ---
print("\n--- Aggregating and Analyzing Results ---")
# ... (Aggregation code remains the same as v6 - already handles NaN) ...
aggregated_results = {}
def get_layout_key(layout):
if isinstance(layout, list): return tuple(layout)
return layout
for res in all_results:
```

 doi:10.20944/preprints202504.1407.v1


```
layout_tuple_key = get_layout_key(res.get('layout', 'N/A'))
key = f"{res.get('backend', 'Unknown')}_{str(layout_tuple_key)}"
is_valid_raw = 'p_corr' in res and isinstance(res['p_corr'], (float, np.floating)) and not np.isnan(res['p_corr'])
is_valid_mit = 'mitigated_p_corr' in res and isinstance(res['mitigated_p_corr'], (float, np.floating)) and not
    np.isnan(res['mitigated_p_corr'])
if key not in aggregated_results: aggregated_results[key] = {'p_corr_list': [], 'p_anti_list': [], 'mit_p_corr_list': [],
    'mit_p_anti_list': [], 'count': 0, 'layout': res.get('layout', 'N/A')}
if is_valid_raw:
aggregated_results[key]['p_corr_list'].append(res['p_corr'])
aggregated_results[key]['p_anti_list'].append(res['p_anti'])
aggregated_results[key]['count'] += 1
if is_valid_mit:
aggregated_results[key]['mit_p_corr_list'].append(res['mitigated_p_corr'])
aggregated_results[key]['mit_p_anti_list'].append(res['mitigated_p_anti'])

print("\nSummary Statistics (Mean ± Std Dev - NaN runs excluded):")
sorted_keys = sorted(aggregated_results.keys())
for key in sorted_keys:
data = aggregated_results[key]
if data['count'] > 0:
valid_p_corr = [p for p in data['p_corr_list'] if not np.isnan(p)]; valid_p_anti = [p for p in data['p_anti_list'] if not
    np.isnan(p)]
valid_mit_p_corr = [p for p in data['mit_p_corr_list'] if not np.isnan(p)]; valid_mit_p_anti = [p for p in
    data['mit_p_anti_list'] if not np.isnan(p)]
num_valid_raw = len(valid_p_corr)
mean_p_corr = np.mean(valid_p_corr) if num_valid_raw > 0 else np.nan; std_p_corr = np.std(valid_p_corr) if
    num_valid_raw > 1 else 0
mean_p_anti = np.mean(valid_p_anti) if num_valid_raw > 0 else np.nan; std_p_anti = np.std(valid_p_anti) if
    num_valid_raw > 1 else 0
print(f"\nTarget: {key} ({num_valid_raw} valid runs)")
print(f"  Layout Used: {data['layout']}")
print(f"  Raw P(Corr): {mean_p_corr*100:.2f}% ± {std_p_corr*100:.2f}%")
print(f"  Raw P(Anti): {mean_p_anti*100:.2f}% ± {std_p_anti*100:.2f}%")
num_valid_mit = len(valid_mit_p_corr)
if num_valid_mit > 0:
mean_mit_p_corr = np.mean(valid_mit_p_corr); std_mit_p_corr = np.std(valid_mit_p_corr) if num_valid_mit >
    1 else 0
mean_mit_p_anti = np.mean(valid_mit_p_anti); std_mit_p_anti = np.std(valid_mit_p_anti) if num_valid_mit >
    1 else 0
print(f"  Mitigated P(Corr) ({num_valid_mit} valid runs): {mean_mit_p_corr*100:.2f}% ±
    {std_mit_p_corr*100:.2f}%")
print(f"  Mitigated P(Anti) ({num_valid_mit} valid runs): {mean_mit_p_anti*100:.2f}% ±
    {std_mit_p_anti*100:.2f}%")
elif ENABLE_READOUT_MITIGATION:
mit_attempted_and_failed = any( ('mitigated_p_corr' in res and np.isnan(res['mitigated_p_corr'])) for res in
    all_results if f"{res.get('backend', 'Unknown')}_{str(get_layout_key(res.get('layout', 'N/A')))}" == key)
if mit_attempted_and_failed: print("  Mitigated results: NaN or Error during processing for all valid runs.")
else: print("  Mitigated results: Not available (Check logs - Mitigation disabled, errored, or no valid runs?).")

# --- TODO: Add Correlation Analysis Code Here ---
print("\nPlaceholder for Calibration Correlation Analysis...")
```

 doi:10.20944/preprints202504.1407.v1


```
# --- TODO: Add ZNE Implementation Here (Optional/Advanced) ---
print("\nPlaceholder for Zero-Noise Extrapolation (ZNE) Implementation...")

# --- 5. Save Detailed Results (NaN Safe) ---
# ... (Saving code remains the same as v6 - already NaN safe) ...
results_filename = f"bell_state_results_{datetime.datetime.now():%Y%m%d_%H%M%S}.json"
calibration_filename = f"calibration_log_{datetime.datetime.now():%Y%m%d_%H%M%S}.json"
try:
def convert_numpy_and_datetime(obj):
if isinstance(obj, np.integer): return int(obj)
elif isinstance(obj, np.floating): return None if np.isnan(obj) else float(obj)
elif isinstance(obj, np.ndarray): return obj.tolist()
elif isinstance(obj, (datetime.datetime, datetime.date)): return obj.isoformat()
elif isinstance(obj, np.bool_): return bool(obj)
elif isinstance(obj, np.complex_): return {'real': obj.real, 'imag': obj.imag}
try: return json.JSONEncoder().encode(obj)
except TypeError: return str(obj)
with open(results_filename, 'w') as f: json.dump(all_results, f, indent=4, default=convert_numpy_and_datetime)
print(f"\nDetailed run results saved to: {results_filename}")
if ENABLE_CALIBRATION_LOGGING:
with open(calibration_filename, 'w') as f:
log_to_save = {}
for bk, bk_data in calibration_data_log.items(): log_to_save[bk] = { "retrieval_time":
    bk_data.get("retrieval_time"), "layouts": bk_data.get("layouts", {}) }
json.dump(log_to_save, f, indent=4, default=convert_numpy_and_datetime)
print(f"Calibration data log saved to: {calibration_filename}")
except Exception as save_e: print(f"\nError saving results/calibration data: {save_e}")

# --- 6. Visualization (Example: Comparing aggregated results - NaN Safe) ---
print("\n--- Generating Aggregate Visualizations (Example) ---")
# ... (Plotting code remains the same as v6 - already handles NaN) ...
try:
valid_keys = [k for k, v in aggregated_results.items() if v['count'] > 0 and 'Ideal (Sim)' not in k]
if not valid_keys: print("No valid aggregated hardware data to plot.")
else:
labels = valid_keys
raw_p_anti_means = [np.mean([p for p in aggregated_results[k]['p_anti_list'] if not np.isnan(p)]) if [p for p in
    aggregated_results[k]['p_anti_list'] if not np.isnan(p)] else np.nan for k in labels]
raw_p_anti_stds = [np.std([p for p in aggregated_results[k]['p_anti_list'] if not np.isnan(p)]) if
    aggregated_results[k]['count'] > 1 else 0 for k in labels]
plot_mitigated = False; mit_p_anti_means=[]; mit_p_anti_stds=[]
if ENABLE_READOUT_MITIGATION:
valid_mit_data_exists = any(aggregated_results[k]['mit_p_anti_list'] and not
    np.isnan(aggregated_results[k]['mit_p_anti_list']).all() for k in labels if k in aggregated_results)
if valid_mit_data_exists:
mit_p_anti_means = [np.mean([p for p in aggregated_results[k]['mit_p_anti_list'] if not np.isnan(p)]) if [p for p
    in aggregated_results[k]['mit_p_anti_list'] if not np.isnan(p)] else np.nan for k in labels]
mit_p_anti_stds = [np.std([p for p in aggregated_results[k]['mit_p_anti_list'] if not np.isnan(p)]) if
    len(aggregated_results[k]['mit_p_anti_list']) > 1 else 0 for k in labels]
if not np.isnan(mit_p_anti_means).all(): plot_mitigated = True
```

 doi:10.20944/preprints202504.1407.v1


```
x = np.arange(len(labels)); width = 0.35 if plot_mitigated else 0.7
fig, ax = plt.subplots(figsize=(max(10, len(labels)*1.2), 6))
valid_raw_indices = [i for i, mean in enumerate(raw_p_anti_means) if not np.isnan(mean)]
if valid_raw_indices: ax.bar(x[valid_raw_indices] - width/2 if plot_mitigated else x[valid_raw_indices],
    np.array(raw_p_anti_means)[valid_raw_indices], width, label='Raw P(Anti)',
    yerr=np.array(raw_p_anti_stds)[valid_raw_indices], capsize=4, alpha=0.8)
if plot_mitigated:
valid_mit_indices = [i for i, mean in enumerate(mit_p_anti_means) if not np.isnan(mean)]
if valid_mit_indices:
if len(valid_mit_indices) <= len(x): ax.bar(x[valid_mit_indices] + width/2,
    np.array(mit_p_anti_means)[valid_mit_indices], width, label='Mitigated P(Anti)',
    yerr=np.array(mit_p_anti_stds)[valid_mit_indices], capsize=4, alpha=0.8)
else: print("Warning: Mismatch in length/indices of mitigated data for plotting.")
ax.set_ylabel('Probability P(Anticorrelated)'); ax.set_title('Mean P(Anticorrelated) across Different
    Targets/Runs')
ax.set_xticks(x); ax.set_xticklabels([labels[i] for i in range(len(labels))], rotation=60, ha="right")
ax.legend(); ax.grid(axis='y', linestyle='--', alpha=0.7); ax.set_ylim(bottom=0)
fig.tight_layout(); agg_plot_filename = 'aggregated_p_anti_comparison.png'
plt.savefig(agg_plot_filename); print(f"Aggregated P(Anti) comparison plot saved as {agg_plot_filename}");
    plt.close(fig)
except ImportError: print("\nWarning: Matplotlib/NumPy needed for aggregate visualization.")
except Exception as agg_plot_e: print(f"\nError during aggregate visualization: {agg_plot_e}");
    traceback.print_exc()

print(f"\n--- Experiment Finished ({datetime.datetime.now()}) ---")
# --- End of enhanced experiment code ---
```

Experiment Results

```
--- Enhanced Bell State Fidelity Experiment ---
Circuit: Bell State |Φ+⟩
     ┌───┐     ┌─┐
q_0: ┤ H ├──■──┤M├─────
     └───┘┌─┴─┐└╥┘┌─┐
q_1: ─────┤ X ├─╫─┤M├
          └───┘ ║ └╥┘
c: 2/═══════════╩══╩═
                0  1

--- Running Ideal Simulation (AerSimulator) ---
Ideal Result: P(Corr) = 1.0000, P(Anti) = 0.0000

--- Running on Real IBM Quantum Hardware ---
Connecting to IBM Quantum service...
Connected.

=== Processing Backend: ibm_kyiv ===
Acquired backend 'ibm_kyiv' (Status: active)

Available CNOT connections (qubit pairs) on ibm_kyiv:
```

 doi:10.20944/preprints202504.1407.v1


```
(Found 144 unique pairs - check list above if needed)
Retrieving calibration properties for ibm_kyiv...
ERROR:stevedore.extension:Could not load 'ibm_backend': cannot import name 'ProviderV1' from
    'qiskit.providers' (c:\Users\JKDM\miniconda3\envs\cwq\Lib\site-
    packages\qiskit\providers\__init__.py)
ERROR:stevedore.extension:Could not load 'ibm_dynamic_circuits': cannot import name 'ProviderV1' from
    'qiskit.providers' (c:\Users\JKDM\miniconda3\envs\cwq\Lib\site-
    packages\qiskit\providers\__init__.py)
ERROR:stevedore.extension:Could not load 'ibm_backend': cannot import name 'ProviderV1' from
    'qiskit.providers' (c:\Users\JKDM\miniconda3\envs\cwq\Lib\site-
    packages\qiskit\providers\__init__.py)
ERROR:stevedore.extension:Could not load 'ibm_dynamic_circuits': cannot import name 'ProviderV1' from
    'qiskit.providers' (c:\Users\JKDM\miniconda3\envs\cwq\Lib\site-
    packages\qiskit\providers\__init__.py)
Calibration data retrieved.

--- Testing Explicit Layout: [2, 3] on ibm_kyiv ---
Transpiling circuit for layout [2, 3] ...
Transpilation successful for layout [2, 3].
Specific calibration data logged for layout [2, 3].
Starting Readout Calibration (mthree) for layout [2, 3]...
M3 Readout calibration prepared.
Starting 5 execution runs for layout [2, 3]...
Submitting Run 1/5...
Run 1 job submitted (ID: czy8qedqnmvg008w2560). Waiting...
Job czy8qedqnmvg008w2560 finished with status: DONE
Raw Results (Filtered): P(Corr)=0.9875, P(Anti)=0.0125
M3 Readout Mitigation Applied: P(Corr)=1.0000, P(Anti)=0.0000
Run 1 processed.
Waiting 5s...
Submitting Run 2/5...
Run 2 job submitted (ID: czy8qxqkzhn0008d7fh0). Waiting...
Job czy8qxqkzhn0008d7fh0 finished with status: DONE
Raw Results (Filtered): P(Corr)=0.9790, P(Anti)=0.0210
M3 Readout Mitigation Applied: P(Corr)=0.9956, P(Anti)=0.0044
Run 2 processed.
Waiting 5s...
Submitting Run 3/5...
Run 3 job submitted (ID: czy8r28kzhn0008d7fhg). Waiting...
Job czy8r28kzhn0008d7fhg finished with status: DONE
Raw Results (Filtered): P(Corr)=0.9849, P(Anti)=0.0151
M3 Readout Mitigation Applied: P(Corr)=1.0000, P(Anti)=0.0000
Run 3 processed.
Waiting 5s...
Submitting Run 4/5...
Run 4 job submitted (ID: czy8rbsqnmvg008w259g). Waiting...
Job czy8rbsqnmvg008w259g finished with status: DONE
Raw Results (Filtered): P(Corr)=0.9844, P(Anti)=0.0156
M3 Readout Mitigation Applied: P(Corr)=1.0000, P(Anti)=0.0000
Run 4 processed.
```

 doi:10.20944/preprints202504.1407.v1


```
Waiting 5s...
Submitting Run 5/5...
Run 5 job submitted (ID: czy8rgjrxz8g008f4ghg). Waiting...
Job czy8rgjrxz8g008f4ghg finished with status: DONE
Raw Results (Filtered): P(Corr)=0.9866, P(Anti)=0.0134
M3 Readout Mitigation Applied: P(Corr)=1.0000, P(Anti)=0.0000
ERROR:stevedore.extension:Could not load 'ibm_backend': cannot import name 'ProviderV1' from
    'qiskit.providers' (c:\Users\JKDM\miniconda3\envs\cwq\Lib\site-
    packages\qiskit\providers\__init__.py)
ERROR:stevedore.extension:Could not load 'ibm_dynamic_circuits': cannot import name 'ProviderV1' from
    'qiskit.providers' (c:\Users\JKDM\miniconda3\envs\cwq\Lib\site-
    packages\qiskit\providers\__init__.py)
ERROR:stevedore.extension:Could not load 'ibm_backend': cannot import name 'ProviderV1' from
    'qiskit.providers' (c:\Users\JKDM\miniconda3\envs\cwq\Lib\site-
    packages\qiskit\providers\__init__.py)
ERROR:stevedore.extension:Could not load 'ibm_dynamic_circuits': cannot import name 'ProviderV1' from
    'qiskit.providers' (c:\Users\JKDM\miniconda3\envs\cwq\Lib\site-
    packages\qiskit\providers\__init__.py)
Run 5 processed.

--- Testing Explicit Layout: [7, 8] on ibm_kyiv ---
Transpiling circuit for layout [7, 8] ...
Transpilation successful for layout [7, 8].
Specific calibration data logged for layout [7, 8].
Starting Readout Calibration (mthree) for layout [7, 8]...
M3 Readout calibration prepared.
Starting 5 execution runs for layout [7, 8]...
Submitting Run 1/5...
Run 1 job submitted (ID: czy8rv3rxz8g008f4gp0). Waiting...
Job czy8rv3rxz8g008f4gp0 finished with status: DONE
Raw Results (Filtered): P(Corr)=0.9148, P(Anti)=0.0852
M3 Readout Mitigation Applied: P(Corr)=1.0000, P(Anti)=0.0000
Run 1 processed.
Waiting 5s...
Submitting Run 2/5...
Run 2 job submitted (ID: czy8rzvqnmvg008w25f0). Waiting...
Job czy8rzvqnmvg008w25f0 finished with status: DONE
Raw Results (Filtered): P(Corr)=0.9150, P(Anti)=0.0850
M3 Readout Mitigation Applied: P(Corr)=1.0000, P(Anti)=0.0000
Run 2 processed.
Waiting 5s...
Submitting Run 3/5...
Run 3 job submitted (ID: czy8sand8drg008hvpgg). Waiting...
Job czy8sand8drg008hvpgg finished with status: DONE
Raw Results (Filtered): P(Corr)=0.9102, P(Anti)=0.0898
M3 Readout Mitigation Applied: P(Corr)=1.0000, P(Anti)=0.0000
Run 3 processed.
Waiting 5s...
Submitting Run 4/5...
Run 4 job submitted (ID: czy8sme6rr3g008me9ag). Waiting...
```

 doi:10.20944/preprints202504.1407.v1


```
Job czy8sme6rr3g008me9ag finished with status: DONE
Raw Results (Filtered): P(Corr)=0.9016, P(Anti)=0.0984
M3 Readout Mitigation Applied: P(Corr)=1.0000, P(Anti)=0.0000
Run 4 processed.
Waiting 5s...
Submitting Run 5/5...
Run 5 job submitted (ID: czy8sxz6rr3g008me9c0). Waiting...
Job czy8sxz6rr3g008me9c0 finished with status: DONE
Raw Results (Filtered): P(Corr)=0.8977, P(Anti)=0.1023
M3 Readout Mitigation Applied: P(Corr)=1.0000, P(Anti)=0.0000
Run 5 processed.

--- Aggregating and Analyzing Results ---

Summary Statistics (Mean ± Std Dev - NaN runs excluded):

Target: Ideal (Sim)_N/A (1 valid runs)
Layout Used: N/A
Raw P(Corr): 100.00% ± 0.00%
Raw P(Anti): 0.00% ± 0.00%
Mitigated results: Not available (Check logs - Mitigation disabled, errored, or no valid runs?).

Target: ibm_kyiv_(2, 3) (5 valid runs)
Layout Used: [2, 3]
Raw P(Corr): 98.45% ± 0.30%
Raw P(Anti): 1.55% ± 0.30%
Mitigated P(Corr) (5 valid runs): 99.91% ± 0.18%
Mitigated P(Anti) (5 valid runs): 0.09% ± 0.18%

Target: ibm_kyiv_(7, 8) (5 valid runs)
Layout Used: [7, 8]
Raw P(Corr): 90.79% ± 0.70%
Raw P(Anti): 9.21% ± 0.70%
Mitigated P(Corr) (5 valid runs): 100.00% ± 0.00%
Mitigated P(Anti) (5 valid runs): 0.00% ± 0.00%

Placeholder for Calibration Correlation Analysis...

Placeholder for Zero-Noise Extrapolation (ZNE) Implementation...

Detailed run results saved to: bell_state_results_20250414_122032.json
Calibration data log saved to: calibration_log_20250414_122032.json

--- Generating Aggregate Visualizations (Example) ---
Aggregated P(Anti) comparison plot saved as aggregated_p_anti_comparison.png

--- Experiment Finished (2025-04-14 12:20:32.719663) ---
```

 doi:10.20944/preprints202504.1407.v1

References

Abraham, H., AduOffei, K., Agarwal, R., Agliardi, G., Aharoni, M., Akhalwaya, I. Y., ... & Zoufal, C. (2019). Qiskit: An Open-source Framework for Quantum Computing. Zenodo. https://doi.org/10.5281/zenodo.2562110

Aleksandrowicz, G., Alexander, T., Barkoutsos, P., Beyer, L., Bishop, L. S., Bullock, C. J., & Zemla, M. (2019). Qiskit: An open-source framework for quantum computing. arXiv preprint arXiv:1904.03019.

Bell, J. S. (1964). On the Einstein Podolsky Rosen paradox. Physics Physique Fizika, 1(3), 195–200.

Bennett, C. H., Brassard, G., Crépeau, C., Jozsa, R., Peres, A., & Wootters, W. K. (1993). Teleporting an unknown quantum state via dual classical and Einstein-Podolsky-Rosen 1 channels. Physical Review Letters, 70(13), 1895–1899.

Blume-Kohout, R. (2010). Optimal, reliable estimation of quantum states. New Journal of Physics, 12(4), 043034. https://doi.org/10.1088/1367-2630/12/4/043034

Breuer, H.-P., & Petruccione, F. (2002). The theory of open quantum systems. Oxford University Press.

Busemeyer, J. R., & Bruza, P. D. (2012). Quantum models of cognition and decision. Cambridge University Press.

Cartwright, N. (1983). How the laws of physics lie. Oxford University Press.

Chow, J. M., Gambetta, J. M., Magesan, E., Abraham, D. W., Smolin, J. A., & Steffen, M. (2012). Simple high-fidelity gates for superconducting qubits. Physical Review Letters, 109(6), 060501.

Cirac, J. I., & Zoller, P. (2012). Goals and opportunities in quantum simulation. Nature Physics, 8(4), 264–266.

Cross, A. W., Bishop, L. S., Sheldon, S., Nation, P. D., & Gambetta, J. M. (2019). Validating quantum computers using randomized model circuits. Physical Review A, 100(3), 032328. 2 https://doi.org/10.1103/PhysRevA.100.032328

Cumming, G. (2014). Understanding the new statistics: Effect sizes, confidence intervals, and meta-analysis. Routledge.

Einstein, A., Podolsky, B., & Rosen, N. (1935). Can Quantum-Mechanical Description of Physical Reality Be Considered Complete? Physical Review, 47(10), 3 777–780. https://doi.org/10.1103/PhysRev.47.777 4

Eisert, J., Hangleiter, D., Walk, N., Roth, I., et al. (2020). Quantum certification and benchmarking. Nature Reviews Physics, 2(7), 382–390. https://doi.org/10.1038/s42254-020-0186-4

Ekert, A. K. (1991). Quantum cryptography based on Bell's theorem. Physical Review Letters, 67(6), 661–663.

Erhard, A., Wallman, J. J., Knill, E., & Biercuk, M. J. (2019). Characterizing quantum devices by benchmarking. Nature Communications, 10(1), 5347.

Field, A. (2018). Discovering statistics using IBM SPSS statistics (5th ed.). SAGE Publications.

Fowler, A. G., Mariantoni, M., Martinis, J. M., & Cleland, A. N. (2012). Surface codes: Towards practical large-scale quantum computation. Physical Review A, 86(3), 032324. 5 https://doi.org/10.1103/PhysRevA.86.032324

Gambetta, J. M., Chow, J. M., & Steffen, M. (2017). Building logical qubits in a superconducting quantum computing architecture. npj Quantum Information, 3(1), 2.

Gühne, O., & Tóth, G. (2009). Entanglement detection. Physics Reports, 474(1-3), 1–75.

Harris, C. R., Millman, K. J., van der Walt, S. J., Gommers, R., Virtanen, P., Cournapeau, D., Wieser, E., Taylor, J., Berg, S., Smith, N. J., Kern, R., Picus, M., Hoyer, S., van Kerkwijk, M. H., Brett, M., Haldane, A., del 6 Río, J. F., Wiebe, M., Peterson, P., ... Oliphant, T. E. (2020). Array programming with NumPy. Nature, 585(7825), 357–362. https://doi.org/10.1038/s41586-020-2649-2 7

Harrow, A. W., Hassidim, A., & Lloyd, S. (2009). Quantum algorithm for solving linear systems of equations. Physical Review Letters, 103(15), 150502.

Herman, D., Pistoia, M., Zeng, Y., & Benjamin, S. C. (2023). Quantum computing for finance. Nature Reviews Physics, 5(8), 450–465. https://doi.org/10.1038/s42254-023-00598-6

Horodecki, R., Horodecki, P., Horodecki, M., & Horodecki, K. (2009). Quantum entanglement. Reviews of Modern Physics, 81(2), 865–942. https://doi.org/10.1103/RevModPhys.81.865

Hunter, J. D. (2007). Matplotlib: A 2D graphics environment. Computing in Science & Engineering, 9(3), 90–95. https://doi.org/10.1109/MCSE.2007.55 8

IBM Quantum. (n.d.). Quantum computing systems & performance metrics. Retrieved April 11, 2025.

Jeffrey, E., Mutus, J. Y., Chow, J. M., Gambetta, J. M., Steffen, M., Johnson, T. D., & Blais, A. (2014). Fast, accurate state measurement with superconducting qubits. Physical Review Letters, 112(19), 190504.

Jurcevic, P., Javadi-Abhari, A., Bishop, L. S., Lauer, I., et al. (2021). Demonstration of quantum volume 64 on a superconducting quantum computing system. Quantum Science and Technology, 6(2), 025020. 9 https://doi.org/10.1088/2058-9565/abe519

Kandala, A., Mezzacapo, A., Móller, K., Abhinav, P., Smulligan, S., Marinelli, F., & Chow, J. M. (2019). Hardware-efficient variational quantum eigensolver for small molecules and quantum magnets. Nature, 549(7671), 242-246. (Note: Appears distinct from the other Kandala et al. 2019 entry)

Kandala, A., Temme, K., Córcoles, A. D., Mezzacapo, A., Chow, J. M., & Gambetta, J. M. (2019). Error mitigation extends the computational reach of a noisy quantum processor. Nature, 567(7749), 491–495. https://doi.org/10.1038/s41586-019-1040-7

Kelly, J., Barends, R., Fowler, A. G., Megrant, A., Jeffrey, E., White, T. C., Sank, D., Mutus, J. Y., Campbell, B., Chen, Y., Chen, Z., Chiaro, B., Dunsworth, A., Hoi, I. 10 A., Neill, C., O'Malley, P. J. J., Quintana, C., Roushan, P., Vainsencher, A., Wenner, J., Cleland, A. N., & Martinis, J. M. (2015). State preservation by repetitive error detection in a superconducting quantum circuit. Nature, 519(7541), 66–69.

Knill, E., Leibfried, D., Reichle, R., Britton, J., Blakestad, R. B., Jost, J. D., Langer, C., Ozeri, R., Riebe, M., Wineland, D. J., & Häffner, H. (2008). Randomized benchmarking of quantum gates. Physical Review A, 77(1), 012307.

Krantz, P., Kjaergaard, M., Yan, F., Orlando, T. P., Gustavsson, S., & Oliver, W. D. (2019). A quantum engineer's guide to superconducting qubits. Applied Physics Reviews, 6(2), 021318. 11 https://doi.org/10.1063/1.5089550

Li, Y., & Benjamin, S. C. (2017). Efficient variational quantum eigensolver based on qubit entanglement. Physical Review X, 7(2), 021050.

Lloyd, S. (1996). Universal quantum simulators. Science, 273(5278), 1073–1078.

Magesan, E., Gambetta, J. M., & Emerson, J. (2011). Scalable and robust randomized benchmarking of quantum processes. Physical Review Letters, 106(18), 180504.

Mäki, U. (2009). The reason of analogies in science. Synthese, 170(3), 407-423.

Martinis, J. M., Ansmann, M., Bialczak, R. C., Hofheinz, M., Katz, N., Lucero, E., McKenney, R., Weig, E., & Cleland, A. N. (2009). Decoherence in Josephson qubits from dielectric loss. Physical Review Letters, 103(9), 097002.

McCloskey, D. N. (1994). Knowledge and persuasion in economics. Cambridge University Press.

McKay, D. C., Sheldon, S., Smolin, J. A., Chow, J. M., & Gambetta, J. M. (2019). Three-qubit randomized benchmarking. Physical Review Letters, 122(20), 200502. https://doi.org/10.1103/PhysRevLett.122.200502

Nation, P. D., Kang, H., Sundaresan, N., & Gambetta, J. M. (2021). Scalable Mitigation of Measurement Errors on Quantum Computers. PRX Quantum, 2(4), 040326.

Nielsen, M. A., & Chuang, I. L. (2010). Quantum Computation and Quantum Information (10th Anniversary ed.). Cambridge University Press.

Orrell, D. (2020). Quantum economics: The new science of asset allocation. Agenda Publishing.

Orús, R., Mugel, S., & Lizaso, E. (2019). Quantum computing for finance: Overview and prospects. Reviews in Physics, 4, 100028. https://doi.org/10.1016/j.revip.2019.100028

Plenio, M. B., & Virmani, S. (2007). An introduction to entanglement measures. Quantum Information & Computation, 7(1-2), 1–51.

Preskill, J. (2018). Quantum computing in the NISQ era and beyond. Quantum, 2, 79. https://doi.org/10.22331/q-2018-08-06-79

Python Software Foundation. (n.d.). Formatted string literals (f-strings). Python Documentation. Retrieved April 11, 2025, from https://docs.python.org/3/reference/lexical_analysis.html#f-strings

Qiskit contributors. (n.d.). Qiskit textbook. Retrieved April 11, 2025, from https://qiskit.org/textbook/

 doi:10.20944/preprints202504.1407.v1

Qiskit Development Team. (c. 2024). Characterizing readout errors. Qiskit Documentation. Retrieved April 11, 2025, from https://docs.quantum.ibm.com/error-mitigation/measurement/readout-error

Qiskit Development Team. (n.d.). Qiskit documentation. Retrieved April 11, 2025, from https://docs.quantum.ibm.com/

Qiskit Development Team. (n.d.). Measurement error mitigation. Qiskit Documentation. Retrieved April 11, 2025.

Qiskit Development Team. (n.d.). qiskit-ibm-provider documentation. Retrieved April 11, 2025.

Qiskit Development Team. (n.d.). Readout mitigation. Qiskit Documentation. Retrieved April 11, 2025.

Qiskit Development Team. (n.d.). Zero noise extrapolation. Qiskit Documentation. Retrieved April 11, 2025.

Rebentrost, P., Bromley, T. R., Fitzsimons, J., & Lloyd, S. (2014). Quantum algorithm for portfolio optimization. Physical Review A, 89(1), 012327. https://doi.org/10.1103/PhysRevA.89.012327

Slichter, C. P. (1990). Principles of magnetic resonance (3rd ed.). Springer-Verlag.

Temme, K., Bravyi, S., & Gambetta, J. M. (2017). Error mitigation for short-depth quantum circuits. Physical Review Letters, 119(18), 180509. 12 https://doi.org/10.1103/PhysRevLett.119.180509

The pandas development team. (2024). pandas documentation. https://pandas.pydata.org/docs/

Vedral, V. (2008). Quantum entanglement. Springer.

Zhong, Y., Chang, H.-S., Satzinger, K. J., Chou, M.-H., Bienfait, A., Conner, C. R., Dumur, É., Grebel, J., Povey, G. A., Schuster, D. I., & Cleland, A. N. (2019). Violating Bell's inequality with remotely-connected superconducting qubits. Nature Physics, 15, 741–744. https://doi.org/10.1038/s41567-019-0507-7