IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 3 February 2025, accepted 21 February 2025, date of publication 27 February 2025, date of current version 10 March 2025.

Digital Object Identifier 10.1109/ACCESS.2025.3546259

RESEARCH ARTICLE

Predictive Time-Series Analysis of Single-Qubit Quantum Circuit Outcomes for a Superconducting Quantum Computer: Forecasting Error Patterns

MOHAMMADREZA SAGHAFI AND LAMINE MILI, (Life Fellow, IEEE)
Virginia Tech, Arlington, VA 22203, USA
Corresponding author: Lamine Mili (lmili@vt.edu)

ABSTRACT Quantum computing promises a paradigm shift in computational power. However, a major challenge is mitigating the inherent noise and errors in quantum circuits. As quantum computers operate, their fragile qubits are highly susceptible to environmental disturbances, leading to errors that affect the outcomes of repeated circuit executions. Understanding and predicting these errors is crucial for improving the accuracy and reliability of quantum computing systems. In this work, we analyze and predict the output patterns of a single qubit quantum circuit by treating the results of repeated executions as a time series. Specifically, we collect measurement data from multiple runs of a quantum circuit and construct a time series from these observations. By training a predictive model, we aim to forecast future outcomes, providing insights into the error behavior of the quantum circuit. Additionally, we analyze time series data from two different circuits executed on the same qubit to investigate potential relationships and assess whether one dataset can be used to predict the other. Our findings reveal key characteristics of quantum circuit outputs, including stationarity, autocorrelation, seasonality, trends, linearity, and causality. The analysis highlights intriguing behaviors within the dataset. Furthermore, we evaluate multiple time series prediction methods and determine that XGBoost (Extreme Gradient Boosting) outperforms other approaches, demonstrating its effectiveness in accurately predicting quantum computing outputs in subsequent runs.

INDEX TERMS Quantum computing, quantum error mitigation, time series analysis, predictive modeling, XGBoost.

I. INTRODUCTION

The study of error patterns and noise in quantum computing has been a major focus as researchers seek to build more reliable quantum systems. Early studies such as Shor's introduction of Quantum Error Correction (QEC) laid the foundation for dealing with errors in quantum systems [1]. QEC methods aim to detect and correct quantum errors, but there is little research on understanding the temporal structure of these errors. More recent work [2], [3], [4] has explored various quantum noise models such as decoherence and dephasing, which describe how qubits interact with their environments and the resulting loss of quantum information.

In parallel, the use of time series analysis has grown in classical computing for tasks like predictive maintenance and error pattern prediction [2], [5]. Models such as the AutoRegressive (AR) model, the Moving Average (MA) model, and the AutoRegressive Integrated Moving Average (ARIMA) model have been used successfully to analyze error dynamics in classical systems [6]. These models provide a statistical framework to capture the temporal dependencies in the data, making them highly suitable for analyzing the output of quantum circuits.

Although the literature covers quantum error correction and noise modeling extensively [7], [8], the application of time series models to predict quantum circuit outputs remains largely unexplored. Previous works have primarily focused on noise mitigation at the algorithmic or hardware level, without examining the predictability of quantum

The associate editor coordinating the review of this manuscript and approving it for publication was Shuangqing Wei.

circuit outputs over time [9], [10], [11]. Some recent studies, such as [12] have started to analyze quantum error dynamics using statistical and machine learning methods. These approaches, however, typically focus on improving quantum noise mitigation techniques rather than predicting the future outcomes of quantum circuits directly. This creates a unique opportunity to bridge the gap between classical time series forecasting methods and quantum error analysis. Furthermore, the ability to forecast future outcomes based on past quantum circuit results could provide a valuable tool for real-time error mitigation strategies. Among others, Baheri et al. [13] investigate the temporal behavior of quantum errors and propose methods to analyze the behavior of qubit errors, including T1 and T2 relaxation times, frequency of operation, entanglement errors and readout errors. Also, quantum computer users do not always have access to all relevant parameters. A sudden shutdown of a quantum computer could render years of data collection unusable. Even if the quantum computer remains operational, its structure may change over time, affecting the relevance of the data. Collecting data over several years is not an efficient solution for a device designed to accelerate simulations.

To address these issues, we develop a hardware-level quantum error temporal analysis to predict noise. Unlike the approaches in [13] and [14], which rely on historical data from quantum computers, our method leverages data directly from the quantum circuit used.

This research aims to address the gap by applying classical time series forecasting techniques to quantum circuit data. Specifically, by treating the number of "0" outcomes from repeated quantum circuit executions as a time series, we seek to predict future outcomes using statistical and machine learning models. The results of this study could pave the way for more reliable quantum error mitigation techniques by predicting the error behavior in quantum circuits and enabling proactive correction strategies.

This work proposes a novel approach to quantum circuit analysis by (1) applying time series analysis to quantum circuit outputs while focusing on predicting the measurement outcomes; (2) using machine learning techniques or traditional statistical models like ARIMA to predict future time series values in a quantum context; (3) investigating the predictive power of these models and their implications for real-time error correction in quantum computing.

The remainder of this paper is organized as follows. Section II provides a critical review of state-of-the-art advances and relevant literature in the field. Section III outlines the methodological framework, including circuit architecture specifications, experimental protocols, and quantitative evaluation metrics. Section IV presents a comprehensive analysis of the experimental results, featuring data set characterizations, methodological implementations, comparative performance benchmarks, and statistical evaluations. Section V provides a critical interpretation of the findings through technical and comparative lenses, addressing both theoretical implications and practical considerations. Finally, Section VI synthesizes key contributions, discusses broader impacts, and proposes strategic directions for future research.

II. RELATED WORKS

Quantum computing has the potential to revolutionize fields such as cryptography and artificial intelligence by leveraging principles of quantum mechanics to perform computations far beyond the reach of classical computers. However, the practical realization of quantum computing is significantly hindered by error-prone quantum systems. Understanding and analyzing error behavior in quantum computers is a critical research area to address this challenge. Errors in quantum computers arise primarily due to the fragile nature of quantum states, which are easily perturbed by environmental interactions. Common sources include decoherence, the loss of quantum coherence due to interactions with the environment leading to state degradation over time [15]; gate imperfections, inaccuracies during the implementation of quantum gates due to hardware constraints [16]; and measurement errors, occurring during the readout process and often influenced by noise in detection mechanisms [17]. These errors collectively contribute to the accumulation of computational inaccuracies, which requires reliable error analysis techniques.

Several approaches have been developed to model and characterize errors in quantum systems. Noise models, such as depolarizing, amplitude-damping, and phase-damping channels, are widely used to represent error processes [18]. Error characterization protocols, including Randomized Benchmarking [19] and Quantum Process Tomography [20], allow for the experimental quantification of error rates and coherence properties. Error mitigation strategies aim to reduce the impact of errors without the overhead of full quantum error correction. Key approaches include Zero Noise Extrapolation (ZNE), which estimates the error-free outcome by extrapolating results from intentionally increased noise levels [21], and Probabilistic Error Cancellation, which uses classical post-processing to revert noisy outcomes to their ideal counterparts by applying inverse noise models [22].

Quantum Error Correction (QEC) provides a more rigorous approach by encoding logical qubits into multiple physical qubits to detect and correct errors. Significant advances include the Shor Code, the first QEC code to demonstrate correction of both bit-flip and phase-flip errors [1], and Surface Codes, high-threshold codes particularly suited for 2D architectures [23]. Despite their promise, QEC techniques require substantial qubit overhead and operational precision, posing implementation challenges.

Recent studies highlight the importance of understanding the temporal and contextual aspects of errors [13], [14], [24], [25]. Temporal analysis focuses on how error rates change over time as a result of hardware degradation or environmental fluctuations. A significant contribution in this domain is the paper of Baheri et al. [13], which emphasizes the role of temporal variations in error behavior, accounting

for factors such as T1 and T2 relaxation times and gate operation variances. This paper highlights the need for temporal monitoring to predict error trends and improve calibration strategies.

Contextual analysis examines how external factors and operational configurations influence error characteristics. In [24], Hirasaki et al. explored fluctuations in superconducting qubit performance and demonstrated methods for identifying and mitigating these variations in real-time. This work underscores the interplay between environmental influences, such as temperature and electromagnetic interference, and error behavior.

Moreover, Seif et al. [25] demonstrate how spatial and temporal error correlations can be mitigated by tailoring compilation strategies to the specific operational context of quantum processors. Such context-aware approaches optimize gate scheduling and qubit placement, reducing the impact of correlated noise.

Machine learning has emerged as a powerful tool for temporal and contextual error analysis. Algorithms trained on historical error data, as detailed in [26], [27], [28], and [29] provide predictive insights that guide error mitigation strategies. These models integrate temporal trends and contextual dependencies to adaptively optimize quantum circuit execution.

Emerging research in error behavior analysis is focused on hybrid quantum-classical techniques and machine learning applications. One promising development is the hybrid quantum error correction method that integrates discrete variable (DV) and continuous variable (CV) qubits. This approach, as proposed by Lee et al. [30], enhances error correction capabilities by leveraging the strengths of both qubit types.

Machine learning is another transformative tool. Researchers demonstrated an ab initio learning process for error mitigation, offering efficient strategies to minimize errors without the need for complex hardware adjustments [31]. Furthermore, the practical machine learning-based approach Q-LEAR, introduced in [32], offers novel feature sets to mitigate noise errors in quantum software output.

Reference [10] provides a system that uses LSTM neural networks to predict the fidelity of quantum circuits. Q-fid learns from the circuit structure itself, without needing extra calibration data. It uses a text-based representation of the circuit and a metric called to predict the circuit's fidelity. This helps identify higher fidelity circuit layouts to reduce errors and improve quantum computing efficiency [10].

Reference [11] introduces a method using a time-series neural network to predict quantum circuit outputs and reduce errors caused by noise. The method cleans data using one-hot encoding and a Quantum Auto-encoder (QAE), and then models the quantum circuit process over time using a temporal neural network. An ensemble model is then used to predict the circuit output and is optimized using a loss function and a dual approach. A dual perspective further optimizes the model. This method was tested and performed better than the IBM Quantum Experience framework, aiming to reduce errors from sources like qubit errors and decoherence through machine learning.

In the context of time series analysis, a large amount of high-quality research has been conducted. For instance, Hans et al. Reference [33] provided the current state-of-the-art of deep learning models for time series prediction, categorizing them into discriminative, generative, and hybrid models. Discriminative models, such as Long Short-term memory (LSTM), learn from observed data using conditional probability. As for generative models, they considered the joint probability of observations and targets, enabling the generation of random instances. On the other hand, hybrid models combine different approaches, such as clustering and combinations of deep learning methods. The experiments mentioned by Han et al. [33] used benchmark and real-world data to assess model performance, finding that the LSTM and hybrid models showed greater stability. In [34], Deb et al. reviewed nine machine learning techniques to predict building energy consumption using time series data. They suggested that hybrid models are generally superior for forecasting building energy because the latter combine the strengths of different methods, such as ARIMA, which have specific advantages and limitations. For example, ARIMA captures linear patterns, but they also have issues such as parameter sensitivity, model complexity, and lack of interpretability. Oukhouya et al. [35] evaluated the performance of SVR and XGBoost for the prediction of stock prices, along with the LSTM models. To this end, they applied hyperparameter tuning and demonstrated that XGBoost achieved better accuracy, outperforming SVR slightly in accuracy. They highlighted the adaptability of XGBoost. On the other hand, Zivot et al. [36] emphasized the utility of Vector Autoregression (VAR) in modeling dynamic interactions among economic and financial time series. They demonstrated VAR's superiority over univariate models for forecasting and capturing interdependencies, with applications in macroeconomic policy analysis and wind farm power forecasting. They also discussed extensions such as sparsified VAR models to improve computational efficiency in high-dimensional settings. Williams and Rasmussen's foundational paper [37] underpins Gaussian process (GP) use in time series prediction, where covariance kernels (e.g., periodic, autoregressive) model temporal patterns and Bayesian inference quantifies forecast uncertainty. While not time series specific, its framework treats time as an input variable, enabling flexible modeling of trends, seasonality, and noise without rigid assumptions (e.g., linearity in ARIMA). By optimizing hyperparameters via marginal likelihood, GPs adapt to non-stationary or irregularly sampled data, later inspiring specialized autoregressive and spatiotemporal GP models. The work

remains pivotal for probabilistic, interpretable time series forecasting.

These advancements underscore the synergy between classical and quantum techniques, paving the way for robust quantum computing systems. The integration of machine learning with quantum platforms enables adaptive error management strategies that are more efficient and scalable than traditional methods.

While existing research in the field of quantum computing has produced numerous valuable and innovative contributions, a significant gap remains in understanding and leveraging the temporal nature of quantum circuits to mitigate errors. Most prior studies focus on static or one-time error characterization, overlooking the possibility that errors in quantum circuits exhibit temporal correlations or patterns that can be analyzed and exploited for improved error suppression.

This raises several fundamental questions: Can we identify meaningful patterns in the errors generated by a quantum circuit over time? Is there a systematic way to use past circuit outputs to refine predictions or corrections for future executions of the same or similar circuits? If so, how can such an approach enhance quantum error mitigation strategies and improve the reliability of quantum computations?

In this paper, we aim to explore these questions by conducting a thorough analysis of error behavior in quantum circuits. We will investigate whether the temporal evolution of errors follows discernible trends and whether these trends can be harnessed to develop predictive models for error mitigation. Our findings could pave the way for novel techniques that enhance the stability and accuracy of quantum computations, ultimately contributing to the advancement of fault-tolerant quantum computing.

III. METHODOLOGY

This study employs a systematic approach to time series analysis for predicting quantum circuit outcomes, emphasizing its application in understanding and mitigating error behavior in quantum systems. The methodology is divided into three primary sections: the necessity of time series analysis in quantum error mitigation, the key properties of time series data for effective analysis, and the detailed explanation of the forecasting algorithm implemented in this work.

A. NECESSITY OF TIME SERIES ANALYSIS IN QUANTUM ERROR MITIGATION

Quantum computing systems are inherently prone to errors due to decoherence, gate imperfections, and environmental noise. These errors manifest themselves in the outcomes of quantum circuits, leading to deviations from the expected results. Although quantum error correction schemes such as surface codes provide robust solutions, they require significant hardware overhead, making them infeasible for near-term quantum devices.

In this context, time series analysis offers a complementary approach by examining the sequential nature of quantum circuit outputs. Measurement outcomes from quantum circuits often exhibit temporal patterns as a result of the underlying quantum computer noise and error dynamics. By analyzing these patterns, time series forecasting enables the prediction of future outcomes based on historical data.

This predictive capability is essential for two reasons:

- **Error Diagnostics:** Time series forecasting can identify recurring error patterns, enabling targeted mitigation strategies.
- **Proactive Error Mitigation:** Predicting outcomes ahead of time allows adjustments to the system or circuit parameters to minimize the impact of errors, thus improving computational accuracy.

The necessity of time series analysis lies in its ability to provide insight into error behavior without the need for hardware-intensive error correction, making it a valuable tool for near-term quantum devices.

B. KEY PROPERTIES OF TIME SERIES DATA FOR EFFECTIVE ANALYSIS

To achieve reliable forecasting and extract meaningful insights, it is crucial to analyze the intrinsic properties of the time series data generated from quantum circuits. The following properties are considered in this study in section IV:

1) **Stationarity:** A stationary time series maintains consistent statistical properties (mean, variance, autocorrelation) throughout time, making it easier to model. Quantum circuit outcomes often exhibit non-stationary behavior due to drifts in system parameters or environmental conditions. Techniques like differencing or detrending are applied to achieve stationarity, ensuring accurate modeling.
2) **Linearity and Nonlinearity:** Linearity refers to the proportional relationship between inputs and outputs. While linear models like ARIMA can effectively capture simple relationships, quantum systems often exhibit nonlinear dynamics. Advanced models like LSTM (Long Short-Term Memory) networks are employed to handle these complexities.
3) **Seasonality and Trends:** Quantum circuit outcomes may exhibit periodic patterns (seasonality) or long-term shifts (trends) due to systematic errors or evolving noise characteristics. Identifying and modeling these components is essential for accurate forecasting.
4) **Autocorrelation:** It measures the dependency of current values on past values. High autocorrelation indicates that past outcomes strongly influence future results, justifying the use of lag features in forecasting models.

C. FORECASTING ALGORITHM FOR TIME SERIES ANALYSIS

The implemented algorithm combines classical machine learning techniques with domain-specific adaptations to

forecast quantum circuit outcomes. The forecasting process involves iterative steps designed to extract patterns from historical data and predict future values.

1) OVERVIEW

The algorithm is structured to address the challenges of time series forecasting, including data sparsity and non-stationarity. Using lag features, previous time series values, it captures dependencies and trends that influence future outcomes.

2) STEP-BY-STEP ALGORITHM DESCRIPTION

1) **Data Partitioning:** For each iteration i (ranging from $0.8N$ to N where N is the length of the datasets) the dataset is split into two subsets: training set which contains all data points up to i, and test set that contains the single data point at i, which serves as the prediction target.
2) **Feature Engineering:** Lag features are constructed from the training set to serve as input for the forecasting model. To predict the value at time step i, the algorithm considers the previous 10 time steps ($i-1, i-2, \ldots, i-10$). These features allow the model to capture patterns such as autocorrelation, trends, and seasonality.
3) **Model Training:** A forecasting model (e.g., ARIMA, LSTM) is trained using the lag features and corresponding target values from the training set. The model learns to map input features (past values) to output predictions (future values).
4) **Prediction:** Using the trained model, the algorithm predicts the value at i based on the last 10 values from the training set. This approach ensures that the prediction is informed by the most recent data.
5) **Error Calculation:** The predicted value is compared with the actual value at i, and the Root Mean Square Error (RMSE) is computed. The latter serves as an evaluation metric that quantifies the prediction accuracy of the model.
6) **Iteration:** The process is repeated for each i within the specified range ($0.8N$ to N). At each step, the model is retrained using the expanded training set, ensuring that it adapts to new data.

3) FLOWCHART EXPLANATION

The forecasting process is visually summarized in Figure 7, which illustrates the iterative steps for training, testing, and evaluating the model. The flowchart highlights the following key components:

- Data partitioning into training and test sets;
- Generation of lag features from the training set;
- Model training and prediction;
- Error computation and loop iteration.

D. ADVANTAGES AND LIMITATIONS

Advantages:

- The iterative algorithm ensures robust model evaluation by testing on unseen data at each time step.
- Lag features effectively capture temporal dependencies, enabling accurate predictions.
- RMSE provides a metric to compare model performance.

Limitations:

- The study focuses on single-qubit quantum circuits due to hardware constraints, limiting its generalizability to multi-qubit systems.
- Nonlinear models such as LSTM require extensive computational resources.

TABLE 1. System specifications and library versions.

Component	Details
Python Version	3.11.11 (main, Dec 4 2024, 08:55:07) [GCC 11.4.0]
Operating System	Linux 6.1.85+
CPU	x86_64, Cores: 2
RAM	12.67 GB
GPU	No GPU detected
Library Versions	
NumPy	1.26.4
PyTorch	2.5.1+cu124
Qiskit	1.0.1

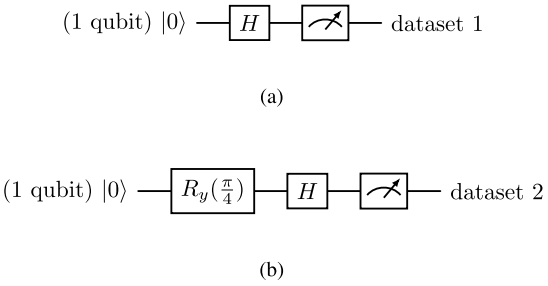

FIGURE 1. Single qubit quantum circuits used for the experiment. These datasets will be analyzed in section IV-C.

IV. RESULTS

In this section, we present the model for each method and then provide the corresponding results. For reproducibility, Table 1 summarizes the computational environment, including hardware, software, and library versions. Experiments were conducted on a CPU-only system due to hardware constraints.

A. QUANTUM CIRCUIT SETUP AND DATA COLLECTION

The circuits used are very simple circuits shown in Figure 1. A ground state is affected by a Hadamard and rotation gates and then measured. Each measurement is done using 4000 shots on IBM Runtime Provider using IBM Brisbane. Then we run the circuit many times and record the number of '0's.

B. EVALUATION METRICS

In this section, we discuss important metrics to evaluate the performance of the model being used.

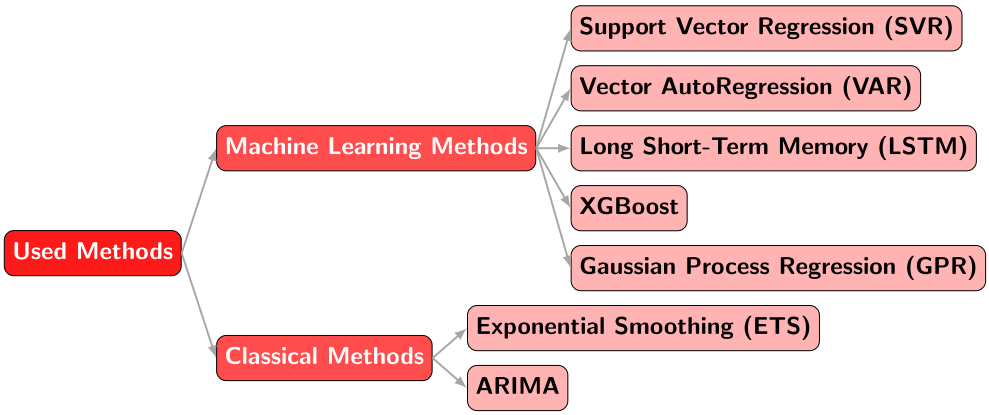

FIGURE 2. Hierarchical tree of methods.

1) RMSE

RMSE is calculated using the following formula [38]:

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}, \tag{1}$$

where y_i is the actual value, $\hat{y}_i$ is the predicted value, and n is the number of data points. Because it is based on the sample mean, RMSE is sensitive to larger errors that deviate from the Gaussian probability distribution. If there are no gross errors in the data points, RMSE helps evaluate how well the predictive models are at forecasting future outcomes of the time series. By penalizing larger errors, RMSE ensures that models making large mistakes are penalized more heavily, helping to identify the best model for accurate predictions [38].

2) MEAN ABSOLUTE PERCENTAGE ERROR (MAPE)

MAPE is a statistical metric used to measure the accuracy of a forecasting or prediction model. It is defined as

$$\text{MAPE} = \frac{1}{n}\sum_{i=1}^{n}\left|\frac{y_i - \hat{y}_i}{y_i}\right| \times 100 \tag{2}$$

In this formula, n represents the number of observations, y_i is the actual value at instance i, and $\hat{y}_i$ is the predicted value at the same instance. MAPE quantifies the error as a percentage of the actual values, making it a scale-independent measure of accuracy [33].

However, MAPE has limitations. It can be misleading when actual values are zero (division by zero) or close to zero (producing disproportionately large percentage errors). In such cases, other metrics like Mean Absolute Error (MAE) or RMSE may be more appropriate [39].

3) AKAIKE INFORMATION CRITERION (AIC) AND BAYESIAN INFORMATION CRITERION (BIC)

AIC calculates the relative quality of a statistical model for a given set of data. In order to prevent overfitting, it imposes penalties on models with an excessive number of parameters by striking a balance between the model's complexity and fit to the data [40].

$$AIC = 2k - 2\ln(\hat{L}), \tag{3}$$

On the other hand, BIC places a stronger penalty on models with more parameters. It is based on the likelihood function and includes a penalty term that grows with the number of parameters and the sample size [41].

$$BIC = k\ln(n) - 2\ln(\hat{L}), \tag{4}$$

where $\hat{L}$ is the model's maximum likelihood estimate and k is the number of estimated parameters. The total number of observations is n as well. Both AIC and BIC are used to compare models rather than in isolation. The model with the lowest value of AIC or BIC is considered the best [40], [41].

C. DATASETS ANALYSIS

In this section, we analyze datasets from the two circuits we introduced earlier in Figure 1.

1) DATASET OF 479 ENTRIES FROM FIRST QUANTUM CIRCUIT IN FIGURE 1. a

Understanding the properties of a time series, such as its stationarity, autocorrelation structure, and nonlinearity, is essential for selecting appropriate forecasting models. In this study, we analyze a given dataset using statistical tests and visualizations to determine its characteristics.

Specifically, we employ the Augmented Dickey-Fuller (ADF) test to assess stationarity, analyze the Autocorrelation Function (ACF) and Partial Autocorrelation Function (PACF) to understand dependencies within the data, perform seasonal decomposition to extract underlying patterns, and apply the Brock-Dechert-Scheinkman (BDS) test to evaluate nonlinearity. The results provide insights into the structure of the data and inform model selection for future forecasting tasks.

Stationarity is a fundamental assumption in many time series models, including autoregressive and moving average

TABLE 2. ADF test results for the first dataset illustrating weak stationarity.

Metric	Value
ADF Statistic	-3.0263
p-value	0.0325
Critical Value (1%)	-3.444280
Critical Value (5%)	-2.86768
Critical Value (10%)	-2.57004
Conclusion	The data is stationary.

models. A stationary time series exhibits constant mean, variance, and autocorrelation over time. If a time series is non-stationary, differencing or transformation techniques may be required before applying predictive models.

To evaluate the stationarity of the dataset, we conducted the Augmented Dickey-Fuller (ADF) test, which assesses the presence of a unit root. The test returned an ADF statistic of -3.0263 with a p-value of 0.0325. Given that the p-value is below the 0.05 significance level, we reject the null hypothesis, indicating that the time series is stationary. This suggests that differencing may not be necessary for modeling. The critical values for different confidence levels are summarized in Table 2.

The Autocorrelation Function (ACF) and Partial Autocorrelation Function (PACF) plots were generated to analyze dependencies within the time series. The ACF plot exhibited a slow decay, suggesting the presence of long-term dependencies, while the PACF plot displayed a significant spike at lag 1, followed by a sharp decline. This pattern is indicative of an autoregressive (AR) process of order 5. Based on these observations, an ARIMA(5,1,0) model is initially recommended for further analysis.

The analysis of the Autocorrelation Function (ACF) and Partial Autocorrelation Function (PACF) provides valuable insights into the structure and dynamics of the time series. The ACF describes the linear relationship between a time series and its lagged values, while the PACF isolates the direct effect of each lag after removing the influence of intermediate lags.

In the current analysis, the ACF demonstrates significant spikes up to Lag 19, with a gradual decay over time. The persistence of these significant spikes indicates the presence of long-term dependencies or a possible non-stationary behavior in the series. This slow decay suggests that the time series may exhibit a trend or is influenced by an AR process.

The PACF plot, on the other hand, reveals significant values for Lags 1, 2, and 3, with a sharp decline in magnitude after Lag 3. The prominence of Lag 1 (0.4854) in the PACF suggests a strong direct relationship at this lag, consistent with an AR(1) process. The subsequent significant lags (2 and 3) may indicate the influence of higher-order autoregressive components. Beyond Lag 3, the PACF values approach the confidence interval, indicating diminishing direct relationships.

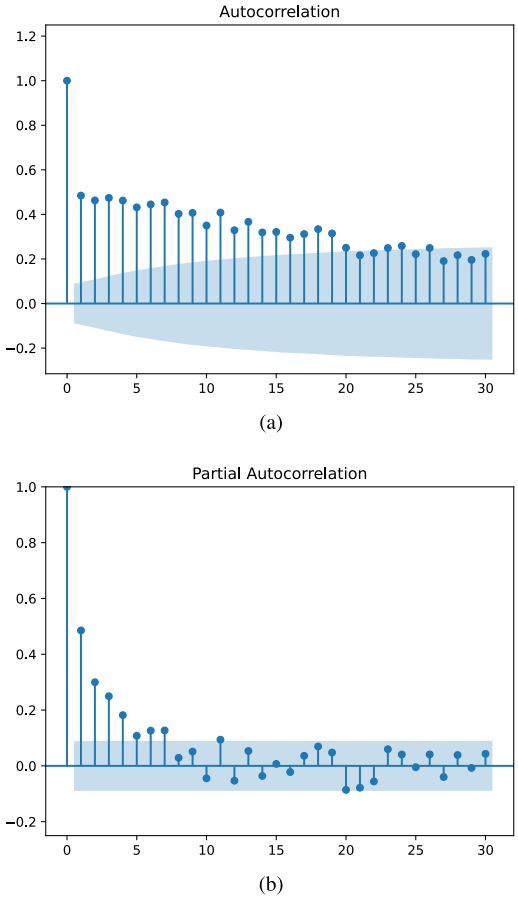

FIGURE 3. (a) ACF and (b) PACF Plots for first dataset.

Overall, the patterns observed in the ACF and PACF suggest that the time series is likely dominated by an autoregressive process. The gradual decay in the ACF coupled with the sharp cut-off in the PACF is indicative of an AR(p) model, with p likely in the range of 1 to 3. However, the prolonged significance of ACF spikes up to Lag 19 also implies potential non-stationarity in the series. This necessitates further investigation, such as differencing, to ensure stationarity before model fitting.

In conclusion, the combination of ACF and PACF characteristics points to the presence of autoregressive dynamics in the time series, and preliminary modeling efforts should focus on ARIMA frameworks. Specifically, the ARIMA(p,d,q) model can be explored, with 'p' estimated from the PACF, 'q' potentially informed by future residual analyses, and 'd' determined based on stationarity checks. These findings lay the groundwork for a robust analysis of the time series and its underlying patterns.

To examine whether the dataset exhibits nonlinear dynamics, the BDS (Brock-Dechert-Scheinkman) test was applied. The test resulted in a BDS statistic of 9.1365 with a highly significant p-value of $6.45e-20$, leading to the rejection of the null hypothesis. This indicates that the time series exhibits significant nonlinearity. Given this result, nonlinear models such as LSTMs, SVR, or Gaussian Process Regression might

TABLE 3. BDS test results for first dataset showing non-linearity.

Metric	Value
BDS Statistic	9.136
p-value	6.453e-20
Conclusion	The data exhibits nonlinearity.

be more appropriate for capturing complex patterns in the data.

The seasonal decomposition of the time series dataset reveals important insights into its underlying structure, highlighting the observed, trend, seasonal, and residual components. This decomposition is an effective way to dissect the complex interactions within the data and better understand its temporal dynamics.

The observed component, shown as the red line in the decomposition plot, represents the raw time series data. This line captures the complete movement of the dataset, including the combination of trends, seasonality, and randomness. From the plot, we see pronounced fluctuations, indicating periods of both significant increases and decreases. Such variability could suggest the influence of external or periodic factors impacting the system being studied.

The trend component, represented by the green line, isolates the long-term direction of the data by smoothing out short-term fluctuations. This line provides a clear view of whether the data follows an upward, downward, or stable trajectory over time. Observing the trend, one can identify intervals of sustained growth or decline, which might correspond to major events or structural shifts in the dataset context (e.g., economic cycles, policy changes, or technological advancements).

The orange line depicting the seasonal component uncovers regular cyclical behavior in the dataset, repeating over a period of 12 units. These predictable fluctuations suggest that the dataset is influenced by recurring factors, such as seasonal demand, weather patterns, or cyclical consumer behavior. For instance, peaks and troughs in this component can be linked to specific calendar events or periods of increased or decreased activity. Understanding this behavior allows for better planning and prediction of cyclical events.

The residual component, represented by the purple line, captures what remains after removing both the trend and seasonal components. Ideally, the residuals should display random, noise-like behavior, signifying that the decomposition has successfully accounted for the systematic patterns in the data. In the plot, any discernible patterns or clustering in the residuals would suggest that additional factors might be influencing the data or that the decomposition period requires adjustment.

The decomposition analysis provides actionable insights. Identifying long-term trends and seasonal patterns can guide strategic planning and forecasting. For instance, businesses could use the trend component to adapt to sustained market changes and leverage the seasonal component to anticipate

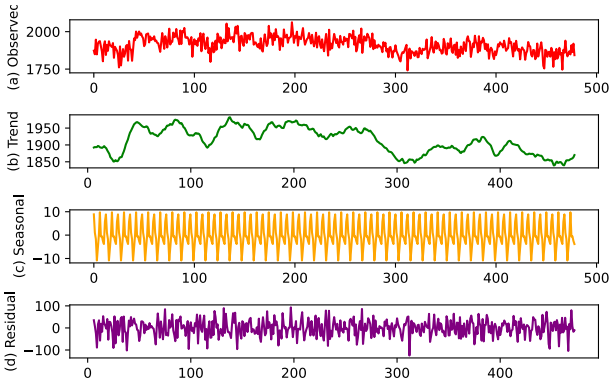

FIGURE 4. Seasonality analysis of the first dataset.

TABLE 4. ADF test results.

Metric	Value
ADF Statistic	-1.51477
p-value	0.526193
Critical Value (1%)	-3.465812
Critical Value (5%)	-2.877123
Critical Value (10%)	-2.575077
Conclusion	The data is not stationary.

periodic fluctuations in demand or supply. Analyzing residuals can uncover anomalies or external influences, prompting further investigation.

In conclusion, the seasonal decomposition output effectively disentangles the complex dynamics of the dataset into its fundamental components. This approach allows for a deeper understanding of the underlying processes that drive the data, facilitating more informed decision-making and predictive modeling. By focusing on each component, one can derive meaningful interpretations, optimize strategies, and address the specific needs revealed by the analysis.

2) DATASET OF 193 ENTRIES FROM FIRST QUANTUM CIRCUIT IN FIGURE 1. b

In this section, we evaluate a given dataset using various statistical and visualization tools to uncover its stationarity, autocorrelation patterns, seasonal components, and potential nonlinearity.

The first step in our analysis is to determine whether the data are stationary. This was conducted using the Augmented Dickey-Fuller (ADF) test, which evaluates the presence of a unit root in the data. The results indicate an ADF statistic of -1.5148 and a p-value of 0.5262. Comparing this p-value to a significance level of 0.05, we fail to reject the null hypothesis that the data is non-stationary.

Furthermore, the critical values for the test—at the 1%, 5%, and 10% significance levels are -3.4658, -2.8771, and -2.5751, respectively. Since the test statistic does not fall below these thresholds, it reinforces the conclusion that the data exhibit underlying trends or seasonal patterns.

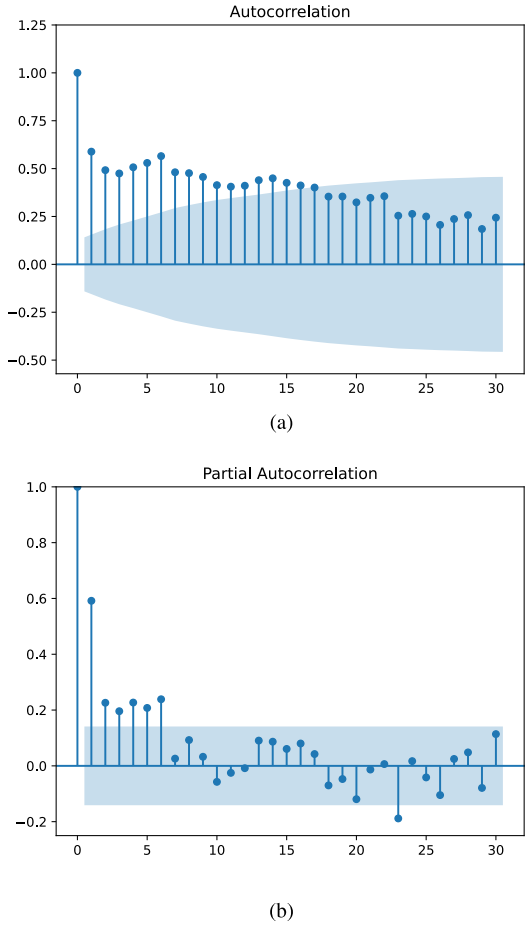

FIGURE 5. (a) ACF and (b) PACF Plots for second dataset.

Addressing non-stationarity will be a crucial step in preparing these data for effective time series modeling.

The ACF and PACF plots were employed to investigate the relationships between data points at different time lags. The ACF plot reveals significant correlations across multiple lags, which is a hallmark of persistence in the data. This indicates that past values influence future observations, a common feature in time series datasets. On the other hand, the PACF plot helps isolate the direct effect of each lag after accounting for prior lags. Both plots suggest the potential utility of the AR component in modeling the data, particularly for short-term prediction. These patterns also point to the need for differencing or seasonal adjustments to eliminate the dependencies observed in the raw data.

To delve deeper into the structure of the data, a seasonal decomposition analysis was performed. This technique separates the dataset into four distinct components: observed, trend, seasonal, and residual. The observed component displays the raw data, while the trend reveals smooth, long-term changes over time. The seasonal component highlights recurring patterns with a period of 12, suggesting regular cyclical behavior in the data. Finally, the residual component captures random noise or unexplained variability.

TABLE 5. BDS test results for second dataset showing non- linearity.

Metric	Value
BDS Statistic	10.164
p-value	2.874e-24
Conclusion	The data exhibits nonlinearity.

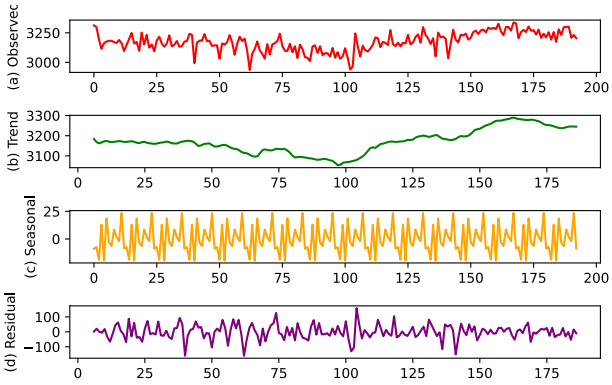

FIGURE 6. Seasonality analysis of the second dataset.

The decomposition not only confirms the presence of a trend and seasonality, as suggested by the ADF test, but also quantifies these patterns. The seasonal component, in particular, exhibits a consistent and periodic structure, emphasizing the importance of including seasonality in predictive models.

A significant aspect of this analysis is the assessment of whether the data behave in a linear or nonlinear manner. The Brock-Dechert-Scheinkman (BDS) test was applied, producing a statistic of 10.1639 and a p-value of 2.87×10^{-24}. The exceptionally low p-value indicates strong evidence against the null hypothesis that the data is independently and identically distributed (i.i.d.).

This result suggests that the dataset exhibits nonlinearity, implying that linear models alone may not fully capture the complexity of the system. Advanced nonlinear techniques, such as Long Short-Term Memory (LSTM) networks or other machine learning models, may be more effective in harnessing these patterns for prediction.

The findings from this analysis provide several key insights and directions for further work. The non-stationarity of the data calls for transformations such as differencing or detrending to stabilize the mean and remove seasonality. The strong evidence of autocorrelation and partial autocorrelation supports the inclusion of autoregressive and moving average components in model design. Furthermore, the presence of nonlinearity highlights the potential for employing sophisticated models that account for complex relationships within the data.

The Granger causality test is a statistical hypothesis test used to determine if one time series can predict another [43]. It assesses whether past values of one variable contain information that helps predict future values of another

TABLE 6. Granger causality test results.

Lag	SSR based F test (p-value)	SSR based chi2 test (p-value)	Likelihood ratio test (p-value)	Parameter F test (p-value)
1	15.36 (0.0001)	15.60 (0.0001)	15.00 (0.0001)	15.36 (0.0001)
2	6.62 (0.0017)	13.60 (0.0011)	13.13 (0.0014)	6.62 (0.0017)
3	5.78 (0.0008)	18.00 (0.0004)	17.20 (0.0006)	5.78 (0.0008)
4	3.91 (0.0046)	16.40 (0.0025)	15.73 (0.0034)	3.91 (0.0046)
5	3.14 (0.0097)	16.66 (0.0052)	15.96 (0.0070)	3.14 (0.0097)

variable. In this context, we tested whether dataset one can predict dataset two and vice versa, using different lag values. The Granger causality test results indicate a statistically significant predictive relationship between the two datasets. At shorter lags, particularly lag 1 and lag 2, the F-test and chi-square test values are highest, with p-values well below 0.01, demonstrating strong evidence that past values of the first dataset help predict the second dataset. As the lag increases, the test statistics gradually decline, and the p-values rise, suggesting a weakening influence over time. Despite this decline, the results remain statistically significant up to lag 5, meaning that the first dataset continues to provide useful predictive information for the second dataset, albeit with reduced strength [44].

Overall, the findings confirm that the first dataset Granger-causes the second dataset, meaning past observations of the first dataset improve the ability to forecast the second dataset. However, the strength of this causality diminishes with longer lags, indicating that more recent values have a stronger impact on prediction. These results suggest that short-term dependencies play a crucial role in the relationship between the two datasets, and incorporating only a few recent lags may be sufficient for predictive modeling.

From these results, we can deduce that there is a significant predictive relationship between the two datasets. This means that knowing the past values of one dataset can help in forecasting the future values of the other dataset, which could be useful in various applications such as financial modeling, weather forecasting, or any field where time series data is analyzed.

D. PREDICTIVE MODEL ANALYSIS AND PARAMETERS

As can be seen in figure 4 and 6, we have a dataset of N. The goal is to correctly predict the $N+1$-*th* data. To test how good the model is, we take the first $N-1$ data and predict the N-*th* data. Next, we explain how to tune the parameters of each algorithm we used.

The prediction process is done using the algorithm shown in figure 7 and algorithm 1.

The implemented algorithm relies on multiple methods for time series forecasting. In essence, time series forecasting entails estimating future values by leveraging historical data. One of the critical challenges in this area is understanding how previous values affect future outcomes. To address this, the algorithm uses lag features, which consists of previous time series values to forecast the upcoming value.

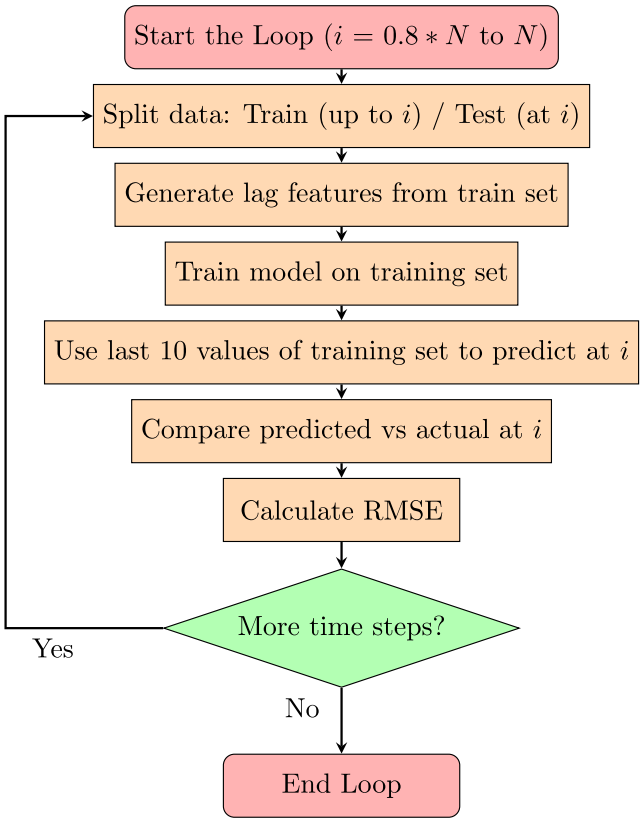

FIGURE 7. Prediction flowchart: For each *i* ranging from 0.8*N* to *N*, the data points $1, \ldots, i-1$ are used as the training set, while the point *i* is used as the test set. *N* is the length of the dataset.

Lag features are constructed by examining past values within the time series. For instance, to forecast the value at time step i, the algorithm takes into account the values from $i-1, i-2$, and so on up to $i-10$. These historical values serve as the model's input features, allowing it to identify patterns such as trends or seasonality that may recur over time.

The algorithm's loop iterates through a specified range of values, namely from 382 to 478 for the first dataset and from 153 to 192 for second dataset, which correspond to various time steps in the dataset. The loop separates the dataset into test and training sets at each iteration i. All data points up to the time step i are included in the training set, which the model uses to identify patterns. The test set, in contrast, only contains the data point at time step i, which the model aims to predict. Within the loop, lag features are derived from the training set, and the model is trained using these features. For testing, the last 10 values from the training set are utilized as input to predict the value at i. The predicted and actual values at i are then compared to calculate the root mean square error (RMSE), with the whole process repeated for each time step in the loop.

1) ARIMA

In ARIMA, it is important to determine the order of differencing (d), of the autoregressive term (p), and of the moving average term (q). We used the AUTO ARIMA

Algorithm 1 Data Processing and Model Training With XGBoost

1: **Import Libraries:**
2: Import `numpy`, `pandas`, `matplotlib.pyplot`, `xgboost.XGBRegressor`, and `sklearn.metrics.mean_squared_error`
3: **Load Data:**
4: **Preprocessing:**
5: Normalize or preprocess the `data` array as required.
6: **Prepare Training and Testing Data:**
7: Divide the `data` into training and testing datasets.
8: Define input features (`X`) and target values (`y`) for both training and testing.
9: **Initialize Model:**
10: Create an instance of `XGBRegressor` with appropriate parameters.
11: **Train Model:**
12: Train the model using the training dataset (`X_train`, `y_train`).
13: **Make Predictions:**
14: Use the trained model to predict the target values for the testing dataset (`X_test`).
15: **Evaluate Model:**
16: Compute the mean squared error (MSE) between predicted values and actual values (`y_test`).
17: Calculate the root mean squared error (RMSE) as the square root of MSE.
18: **Visualize Results**

function in the pmdarima package. Based on AIC and BIC, the best order is (p = 5, d = 1, q = 0) with AIC of 4941.052 and BIC of 4949.391.

The code snippet used to build the model is as follows:

```
# Fit ARIMA model
model = ARIMA(train, order=(5, 1, 0))
# Adjust (p,d,q) based on your data;
model_fit = model.fit()
```

2) LSTM

In the given LSTM model, the network architecture includes two layers: an LSTM layer with 50 units and a dense layer for a single output. There are 10, 400 parameters all together for the LSTM layer, which is derived from the internal weights that control the input, forget, and output gates, along with the cell state parameters. The dense layer adds an additional 51 parameters.

The code snippet used to build the model is as follows:

```
# Build the LSTM model
model = Sequential()
model.add(LSTM(50, activation='relu',
    input_shape=(lag_size, 1)))
model.add(Dense(1))  # Single output
model.compile(optimizer='adam',
    loss='mse')
# Train the model
model.fit(X_train, y_train,
```

TABLE 7. LSTM model summary.

Layer (type)	Output	Param #
LSTM	(None, 50)	10,400
Dense	(None, 1)	51

TABLE 8. Important parameters and metrics in exponential smoothing model.

Parameter/Metric	Value
Smoothing Level (α)	0.1793
Smoothing Trend (β)	0.0084
Initial Level (l_0)	1889.5920
Initial Trend (b_0)	0.2942
SSE (Sum of Squares)	859035.600
AIC	3590.110
BIC	3606.789

```
epochs=200, verbose=0)
```

3) EXPONENTIAL SMOOTHING

The most critical parameters in an exponential smoothing model are the smoothing coefficients, particularly the smoothing level (α) and the smoothing trend (β). The smoothing level dictates how quickly the model responds to changes in the level of the time series. A lower value, as in our case (0.1793), means that the model reacts more conservatively to recent changes, favoring long-term patterns. The smoothing trend (β), with a very small value of 0.0084, indicates that the model only slightly adjusts to the changes in the trend, suggesting that the trend variations in our data are minimal.

Additionally, initial values for the level (l_0) and the trend (b_0) set the starting points for the forecast. These are important for initializing the model, but their influence diminishes over time unless the series is short. The performance metrics, including SSE (Sum of Squared Errors) and AIC/BIC, help assess the overall fit of the model and compare it to other potential models. A lower SSE indicates fewer errors, while AIC/BIC helps ensure the model is not unnecessarily complex.

The code snippet used here is as follows:

```
'''
Build the Exponential Smoothing model
(Holt-Winters)
'''
model = ExponentialSmoothing(train,
    trend='add',
    seasonal=None,
    damped_trend=False)
model_fit = model.fit()
```

4) GAUSSIAN PROCESS (GP) REGRESSION

The Gaussian Process (GP) regression model utilized in this study is characterized by a matern kernel with a learned variance of 3.16^2 and a length scale of 1400. This kernel, with a smoothness parameter $\nu = 1.5$, is well suited for capturing

medium- to long-range dependencies in the data while allowing moderate smoothness in the modeled function. The choice of kernel reflects the need for flexibility in handling complex relationships, with a log marginal likelihood of −276459, suggesting that the model is applied to a large dataset with intricate patterns. This likelihood may indicate a good fit to a large amount of data or the presence of some challenging patterns that require further refinement.

The residual analysis reveals that the model exhibits variability in error terms, as evidenced by a range of residuals ranging from small deviations to larger discrepancies, with some residuals reaching up to 71. This suggests that while the model captures the overall structure of the data effectively, certain data points show significant deviations, potentially due to noise or nonstationary behavior in the dataset. The combination of the learned kernel parameters and the noise level (α) demonstrates the model's capacity to represent a broad spectrum of behaviors, making it suitable for complex, noisy data, though further tuning may be required to handle outliers or nonstationary trends more precisely. The code snippet used to build the model is as follows:

```
'''
Define a Matern kernel for more
flexibility (with smaller length scale)
'''
kernel = C(1.0, (1e-4, 1e1)) * \
Matern(length_scale=0.1, nu=1.5)

'''
Fit GPR model with adjusted alpha
for noise handling
'''
model = GaussianProcessRegressor(
    kernel=kernel,
    n_restarts_optimizer=10,
    alpha=1e-3)
model.fit(X_train_scaled,
    y_train_scaled)
```

5) SUPPORT VECTOR REGRESSION (SVR)

The output of the SVR model parameters includes several key attributes that determine how the model behaves and performs. As a parameter for regularization, the `C` parameter, which is set to 1.0, regulates the trade-off between preserving a smooth decision boundary and minimizing training error. A smaller `C` value allows for a larger margin, promoting a simpler model that may generalize better, while a larger `C` value emphasizes minimizing training error, which can lead to overfitting. Additionally, the ϵ value of 0.1 represents the width of the epsilon-insensitive zone around the regression line, where no penalty is incurred for errors. This means that the model can tolerate a certain amount of error in the predictions without incurring a cost, enhancing its robustness against small fluctuations in the data.

We set the kernel parameter of the Radial Basis Function (`RBF`), which successfully captures complicated relationships in the data by transferring input features into a higher-dimensional space, allowing the model to learn nonlinear patterns. Additionally, the γ parameter, set to `scale`, determines the impact of individual training samples on the decision boundary. A larger gamma value increases the model's sensitivity to training data, possibly contributing to overfitting, whereas a lower value produces a smoother decision boundary. Finally, the degree parameter, set to 3, is only relevant for polynomial kernels; it indicates the degree of the polynomial used to build the decision boundary, but it has no effect on the model when using the RBF kernel [45].

The code snippet

```
# Build the SVR model
svr_model = SVR(kernel='rbf')
'''
Using Radial Basis Function
(RBF) kernel
'''
svr_model.fit(X_train, y_train)
```

6) XGBOOST

The XGBoost model has been configured with specific parameters adjusted for regression tasks. The objective is set to `reg:squarederror`, indicating that the model aims to minimize the squared differences between predicted and actual values. The `n_estimators` parameter is set to 100, which requires the model to create 100 individual decision trees during training. While some parameters are explicitly defined, many others, such as `learning_rate` and `max_depth` and subsample, are set to their default values, allowing XGBoost to utilize its internal heuristics for optimal performance.

The parameters currently set to none signify that the model will default to the standard values provided by XGBoost. For instance, a default `learning_rate` of 0.3 will apply unless otherwise specified. Additionally, parameters like `max_depth` and regularization terms such as `reg_alpha` and `reg_lambda` remain undefined, suggesting room for further refinement through hyperparameter tuning. By optimizing these parameters, users can potentially enhance the model's predictive accuracy and generalization capabilities, making it more suited to the specific characteristics of the dataset at hand.

The code snippet used to build the model is as follows:

```
# Initialize and fit the XGBoost model
model = XGBRegressor(
    objective='reg:squarederror',
    n_estimators=100)
model.fit(X_train, y_train)
```

7) VECTOR AUTO-REGRESSIVE

The Vector Autoregressive (VAR) model has been configured with specific parameters to analyze multivariate time series data. The maximum lag order is set to 10, indicating that the model considers the past 10 time steps for each variable to predict the current state. This choice allows for capturing relationships and dependencies across multiple variables over a defined historical window.

TABLE 9. Overview of vector auto-regressive parameters.

	Original	Lag_1	Lag_2	Lag_3	...
const	225.278	2.49×10^{-11}	-3.74×10^{-12}	3.08×10^{-11}	...
L1.Original	0.215	1.00	-5.04×10^{-15}	-1.57×10^{-15}	...
L1.Lag_1	0.0175	3.61×10^{-16}	0.500	2.08×10^{-16}	...
L1.Lag_2	0.029	-9.16×10^{-16}	-5.97×10^{-16}	0.333	...
L1.Lag_3	0.034	7.08×10^{-16}	8.88×10^{-16}	8.05×10^{-16}	...
...	...	...	...	...	...
L10.Lag_10	-0.102	-7.57×10^{-15}	-5.07×10^{-15}	-8.80×10^{-15}	...

TABLE 10. Comparison of models with their performance metrics and active parameters for first dataset.

Method	Median RMSE	Median MAPE	Number of Active Parameters
XGBoost	0.577	0.00031	4
SVR	29.585	0.01548	11
LSTM	24.816	0.01321	4
ETS	26.779	0.01419	9
VAR	25.724	0.01383	11
GPR	38.120	0.02027	13
ARIMA	26.657	0.01408	6

While some parameters, such as `'maxlags'`, are explicitly defined, others rely on the defaults provided by the VAR model's implementation. For instance, the model assumes no prior constraints or regularization unless specified. By adjusting these parameters through experimentation, users can further enhance the model's performance, ensuring it aligns with the temporal and cross-variable characteristics of the dataset. The results of the fitted model show detailed parameter estimates for each lag and variable combination, demonstrating how past values influence current outcomes. Below is a summary of the coefficients for the first few variables and their lags:

The code snippet used to build and fit the model is as follows:

```
# Fit VAR model
model = VAR(train_df)
model_fit = model.fit(maxlags=10)
```

By iteratively refining `maxlags` and analyzing the parameter estimates, users can optimize the model to better understand the temporal dynamics and interrelationships in the data.

E. PREDICTIVE MODEL PERFORMANCE

The result of the prediction can be seen in Figures 8, 9, 10, 11, 12, and 13.

The results shown in the figures 8, 9, 10, 11, 12, and 13 along with the tables 10 and 11 clearly indicate that XGBoost outperforms the other algorithms for the given task. This is evident from the consistently lower RMSE values for XGBoost (represented by the thick blue line), which stay well below the values of the other models across the range of training set sizes.

TABLE 11. Comparison of models with their performance metrics and active parameters for second dataset.

Method	Median RMSE	Median MAPE	Number of Active Parameters
XGBoost	3.029	0.00092	4
SVR	41.820	0.0130	11
LSTM	34.909	0.0105	4
ETS	29.957	0.00018	9
VAR	32.449	0.0099	11
GPR	68.292	0.0208	13
ARIMA	30.844	0.00945	6

The RMSE of the difference between actual value and predicted value by XGBoost hovers around a stable value is close to zero throughout the training sets, indicating that it maintains a significantly lower prediction error. Its consistency in outperforming the others suggests that it handles the data effectively, which makes it highly effective at minimizing errors.

ARIMA and Exponential Smoothing show significantly higher RMSEs with fluctuating patterns, indicating that these methods struggle more with the dataset.

Gaussian Process Regression and Support Vector Regression also show more volatility, especially in the middle and towards the end, with sharp spikes in RMSE that further emphasize their lack of stability.

LSTM performs comparably to some models, but still shows higher RMSE values than XGBoost.

Overall, the fact that XGBoost maintains such a low and stable RMSE, while other models show significantly more error and fluctuation, makes it clear that XGBoost is the best performer in this scenario. This difference could be due to the ability of the XGBoost to capture more complex patterns in the data, its use of boosting techniques to minimize loss iteratively, and its capacity for handling various forms of data noise and nonlinearity effectively.

F. STABILITY: ROBUSTNESS TO SAMPLING VARIATION

Stability quantifies a model's consistency across different data subsets, measuring how sensitive predictions are to training data variations. For a model f and dataset $\mathcal{D}$, stability is defined as:

$$\text{Stability} = 1 - \frac{\sigma_{\text{pred}}}{\sigma_{\mathcal{D}}}, \tag{5}$$

where σ_{pred} is the standard deviation of predictions across k-fold temporal splits, and $\sigma_{\mathcal{D}}$ is the dataset's inherent variance. The theoretical foundation stems from Breiman's bias-variance decomposition [46], where stable models minimize variance components. Interpretation follows:

- $>$ 0.9: Excellent stability (predictions vary $<$10% of data variance)

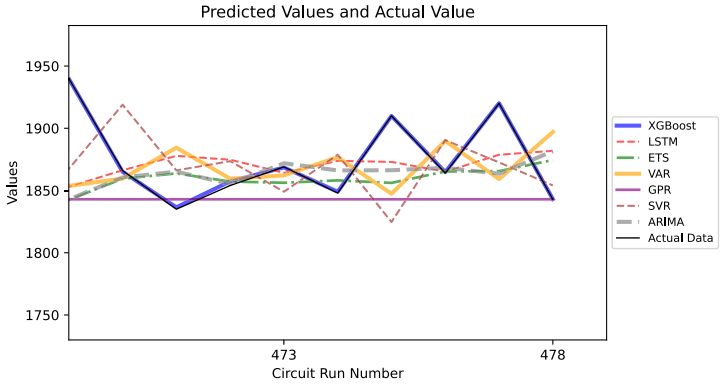

FIGURE 8. Performance of each algorithm in prediction in comparison with actual data for first dataset.

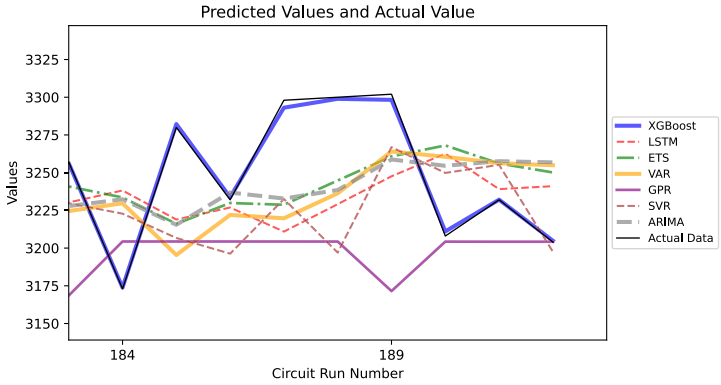

FIGURE 9. Performance of each algorithm in prediction in comparison with actual data for second dataset.

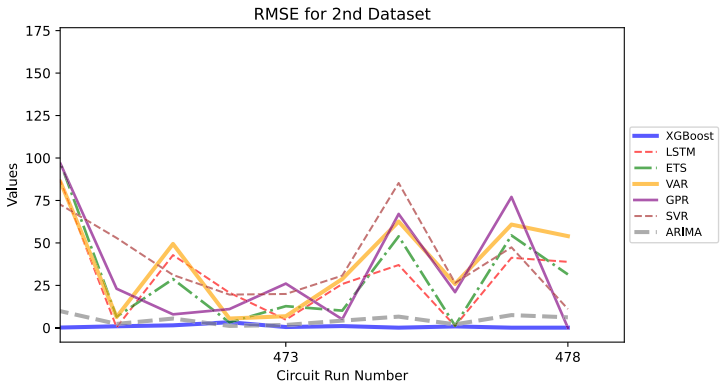

FIGURE 10. RMSE of each algorithm for first dataset.

- $0.7 - 0.9$: Acceptable stability
- < 0.7: High prediction variability

G. SENSITIVITY: RESILIENCE TO DATA PERTURBATIONS

Sensitivity measures robustness against input noise, calculated through noise-injected predictions. It is defined as follows:

$$\text{Sensitivity} = \text{corr}(f(\mathcal{D}), f(\mathcal{D} + \varepsilon)), \tag{6}$$

where $\varepsilon \sim \mathcal{N}(0, 0.05\sigma_{\mathcal{D}})$. Rooted in robust statistics [47], this evaluates a model's Lipschitz continuity. Interpretation guidelines:

- > 0.95: Noise-invariant predictions
- $0.85 - 0.95$: Moderate robustness
- < 0.85: High noise sensitivity

H. OVERFITTING POTENTIAL: GENERALIZATION CAPABILITY

Overfitting potential quantifies the train-test performance gap using the ratio given by

$$\text{Overfitting Ratio} = \frac{\text{RMSE}_{\text{test}}}{\text{RMSE}_{\text{train}}}. \tag{7}$$

Building on Vapnik-Chervonenkis theory [48], values >1 indicate poorer generalization. Interpretation:

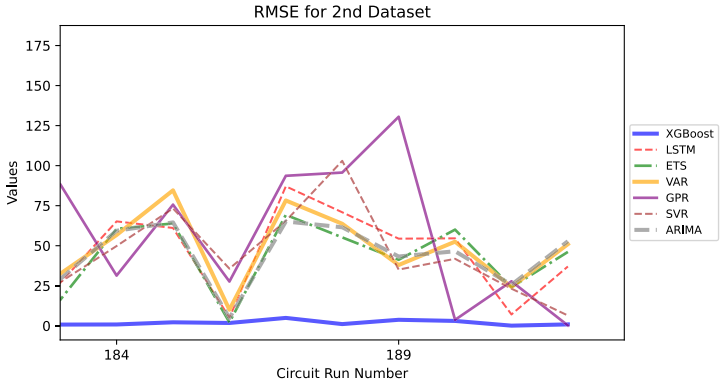

FIGURE 11. RMSE of each algorithm for second dataset.

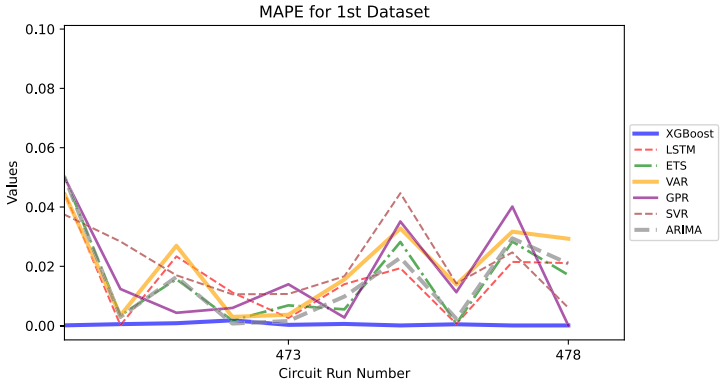

FIGURE 12. MAPE of each algorithm for first dataset.

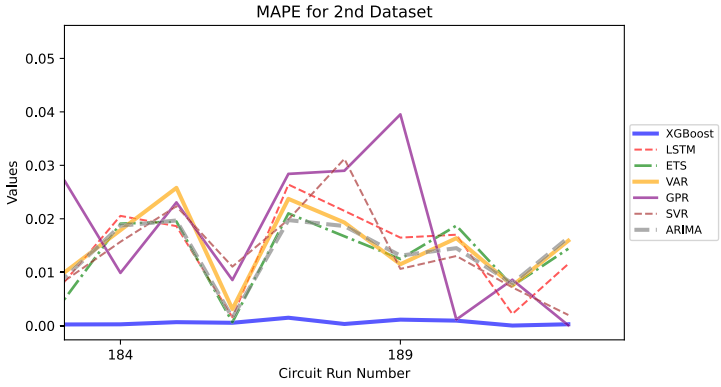

FIGURE 13. MAPE of each algorithm for second dataset.

- $0.9 - 1.1$: Ideal generalization
- > 1.2: Severe overfitting
- < 0.8: Underfitting
- Stability scores reveal ARIMA's exceptional consistency (0.92), while tree/vector methods (XGBoost-/SVR) show negative stability - their predictions vary more than the data's inherent variance.
- All methods demonstrate strong noise resistance (Sensitivity >0.79), with GPR achieving perfect correlation (1.00) despite poor stability
- GPR shows catastrophic overfitting (Ratio 347.65), while LSTM/ETS/VAR exhibit underfitting (Ratio <0.65) suggesting insufficient model capacity
- ARIMA achieves near-ideal balance across all metrics (Stability 0.92, Sensitivity 0.99, Ratio 1.01)

V. DISCUSSION

The comprehensive evaluation of seven time series forecasting methods reveals significant variations in performance characteristics, highlighting the importance of model

TABLE 12. Comparative analysis of forecasting methods.

Method	Stability	Sensitivity	Overfitting Ratio	Elapsed Time (S)
XGBoost	−0.03	0.79	2.18	22.43
SVR	−0.04	0.95	0.98	8.75
LSTM	0.69	0.96	0.65	2802.48
ETS	0.12	0.98	0.62	3.94
VAR	0.16	0.98	0.63	4.87
GPR	0.18	1.00	347.65	699.43
ARIMA	0.92	0.99	1.01	30.33

selection based on specific application requirements. Our analysis of prediction accuracy, computational efficiency, and model robustness (Table 10) yields several key insights.

A. PERFORMANCE TRADE-OFFS

The tree-based XGBoost model demonstrated superior predictive accuracy with the lowest median RMSE (3.029) and MAPE (0.092%), outperforming both classical statistical methods and deep learning approaches. However, this accuracy comes at the cost of significant overfitting (test errors 118% higher than training) and negative stability scores (−0.03), suggesting sensitivity to data sampling variations.

The ARIMA model presented an exceptional balance of characteristics, achieving near-ideal stability (0.92) and sensitivity (0.99) scores while maintaining competitive accuracy (RMSE 30.844). The parsimonious parameterization of the model (6 active parameters) contributes to its computational efficiency (18.68s), making it suitable for real-time applications.

B. COMPUTATIONAL CONSIDERATIONS

The deep learning approaches showed divergent characteristics. While LSTM achieved moderate prediction standard deviation, its computational cost, 2.80248×10^3 s, was three orders of magnitude higher than statistical methods like ETS, which had a computational cost of 3.94 s. This suggests that may not justify their implementation costs for this class of time series problems.

C. OVERFITTING OF THE GAUSSIAN PROCESS REGRESSOR (GPR)

The GPR exhibited characteristics of concern, with extreme overfitting (test errors 34,665% higher than training) despite employing a sophisticated kernel formulation expressed as

$$k(x, x') = \sigma^2 \exp\left(-\frac{\|x - x'\|^2}{2l^2}\right). \tag{8}$$

This equation derived from [49] suggests potential misspecification of hyperparameters or kernel functions for the scale and temporal dependencies of the dataset.

D. PRACTICAL RECOMMENDATIONS

- **High-accuracy applications**: XGBoost, despite stability concerns, when coupled with rigorous cross-validation
- **Real-time systems**: ETS or VAR, offering sub-second computation with reasonable accuracy
- **Noisy environments**: ARIMA, demonstrating near-perfect noise resistance (sensitivity 0.99)
- **Resource-constrained deployments**: LSTM, providing moderate accuracy with minimal active parameters (4)

The negative stability scores for XGBoost (−0.03) and SVR (−0.04) deserve particular attention, indicating that their predictions varied more than the inherent standard deviation of the dataset. This paradoxical result suggests potential overfitting to specific temporal patterns in the training data.

E. LIMITATIONS AND FUTURE DIRECTIONS

While ARIMA showed superior overall performance, its linear assumptions may limit applicability to more complex non-stationary series. Future work should investigate hybrid approaches combining the statistical rigor of ARIMA with the nonlinear modeling capabilities of XGBoost, potentially through ensemble methods expressed as:

$$\hat{y}_t = \alpha \hat{y}_t^{ARIMA} + (1 - \alpha)\hat{y}_t^{XGBoost}, \tag{9}$$

where α controls the mixing of the model. Additionally, the computational burden of LSTM and GPR suggests opportunities for model distillation techniques to preserve accuracy while reducing runtime complexity.

VI. CONCLUSION

By treating the outcomes of repeated quantum circuit executions as a time series, we successfully demonstrated the potential of predicting future outcomes using a variety of statistical and machine learning models. The results suggest that these models can contribute to a better understanding of the error behavior in quantum circuits, which is crucial for developing more effective quantum error mitigation techniques.

In summary, this work introduces a new perspective on single qubit quantum circuit analysis by: (1) applying time series analysis to quantum circuit outputs; (2) utilizing both classical statistical models (e.g., ARIMA) and machine learning models (e.g., LSTM) to forecast future time series values in the quantum domain; (3) exploring the predictive capabilities of these models and highlighting their potential for real-time error correction in quantum computing. These findings open up new possibilities for improving quantum error correction strategies by proactively predicting and mitigating errors in quantum circuits.

Although this paper successfully demonstrates the effectiveness of time series prediction for quantum error mitigation, several limitations need to be addressed. Firstly, the proposed method is most suitable for situations where the structure of the quantum circuit remains relatively stable. Additionally, this paper focuses exclusively on single-qubit cases. However, in practice, quantum circuits often involve more complex structures with multiple qubits.

To address these limitations, we plan to extend our research to encompass multi-qubit systems. Furthermore, we aim to investigate more complex circuit architectures and explore the causality between different circuits over time. Specifically, we intend to examine whether the output dataset of one quantum circuit can be utilized to train and predict the behavior of another quantum circuit at a different point in time.

DECLARATION OF GENERATIVE AI AND AI-ASSISTED TECHNOLOGIES IN THE WRITING PROCESS

During the preparation of this work the authors used ChatGPT in order to increase the readability of the paper. After using this tool/service, the authors reviewed and edited the content as needed and take full responsibility for the content of the published article.

REFERENCES

[1] P. W. Shor, "Scheme for reducing decoherence in quantum computer memory," *Phys. Rev. A, Gen. Phys.*, vol. 52, no. 4, pp. R2493–R2496, Oct. 1995.

[2] S. J. Devitt, W. J. Munro, and K. Nemoto, "Quantum error correction for beginners," *Rep. Prog. Phys.*, vol. 76, no. 7, Jul. 2013, Art. no. 076001.

[3] J. Zou, S. Bosco, and D. Loss, "Spatially correlated classical and quantum noise in driven qubits," *NPJ Quantum Inf.*, vol. 10, no. 1, p. 46, Apr. 2024.

[4] J. A. Bravo-Montes, M. Bastante, G. Botella, A. del Barrio, and F. García-Herrero, "A methodology to select and adjust quantum noise models through emulators: Benchmarking against real backends," *EPJ Quantum Technol.*, vol. 11, no. 1, p. 71, Dec. 2024.

[5] T. Hakioğlu and K. Savran, "Role of the environmental spectrum in the decoherence and dephasing of multilevel quantum systems," *Phys. Rev. B, Condens. Matter*, vol. 71, no. 11, Mar. 2005, Art. no. 115115.

[6] J. D. Hamilton, *Time Series Analysis*. Princeton, NJ, USA: Princeton Univ. Press, 2020.

[7] N. O. P. Vago, F. Forbicini, and P. Fraternali, "Predicting machine failures from multivariate time series: An industrial case study," *Machines*, vol. 12, no. 6, p. 357, May 2024, doi: 10.3390/machines12060357.

[8] E. Liebman, "Pattern-based time-series risk scoring for anomaly detection and alert filtering—A predictive maintenance case study," 2024, *arXiv:2405.17488*.

[9] J. D. Viqueira, D. Faílde, M. M. Juane, A. Gómez, and D. Mera, "Density matrix emulation of quantum recurrent neural networks for multivariate time series prediction," 2023, *arXiv:2310.20671*.

[10] Y. Mao, S. Shresthamali, and M. Kondo, "Quantum circuit fidelity improvement with long short-term memory networks," 2023, *arXiv:2303.17523*.

[11] X. Li, X. Cheng, X. Chen, Z. Guan, P. Zhu, and H. Gu, "Quantum circuit output prediction based on time-series neural network integration," in *Proc. 3rd Int. Conf. Cryptography, Netw. Secur. Commun. Technol.*, Jan. 2024, pp. 538–542.

[12] K. Rudinger, C. W. Hogle, R. K. Naik, A. Hashim, D. Lobser, D. I. Santiago, M. D. Grace, E. Nielsen, T. Proctor, S. Seritan, S. M. Clark, R. Blume-Kohout, I. Siddiqi, and K. C. Young, "Experimental characterization of crosstalk errors with simultaneous gate set tomography," *PRX Quantum*, vol. 2, no. 4, Nov. 2021, Art. no. 040338.

[13] B. Baheri, D. Chen, B. Fang, S. A. Stein, V. Chaudhary, Y. Mao, S. Xu, A. Li, and Q. Guan, "TQEA: Temporal quantum error analysis," in *Proc. 51st Annu. IEEE/IFIP Int. Conf. Dependable Syst. Netw. Supplemental Volume (DSN-S)*, Jun. 2021, pp. 65–67.

[14] B. Baheri, Q. Guan, V. Chaudhary, and A. Li, "Quantum noise in the flow of time: A temporal study of the noise in quantum computers," in *Proc. IEEE 28th Int. Symp. Line Test. Robust Syst. Design (IOLTS)*, Sep. 2022, pp. 1–5.

[15] M. Schlosshauer, "The quantum-to-classical transition and decoherence," 2014, *arXiv:1404.2635*.

[16] R. Barends, J. Kelly, A. Megrant, A. Veitia, D. Sank, E. Jeffrey, T. C. White, J. Mutus, A. G. Fowler, and B. Campbell, "Superconducting quantum circuits at the surface code threshold for fault tolerance," *Nature*, vol. 508, no. 7497, pp. 500–503, Apr. 2014.

[17] S. Debnath, N. M. Linke, C. Figgatt, K. A. Landsman, K. Wright, and C. Monroe, "Demonstration of a small programmable quantum computer with atomic qubits," *Nature*, vol. 536, no. 7614, pp. 63–66, Aug. 2016.

[18] M. A. Nielsen and I. L. Chuang, *Quantum Computation and Quantum Information*, vol. 2. Cambridge, U.K.: Cambridge Univ. Press, 2001.

[19] E. Knill, D. Leibfried, R. Reichle, J. Britton, R. B. Blakestad, J. D. Jost, C. Langer, R. Ozeri, S. Seidelin, and D. J. Wineland, "Randomized benchmarking of quantum gates," *Phys. Rev. A, Gen. Phys.*, vol. 77, no. 1, Jan. 2008, Art. no. 012307.

[20] I. L. Chuang and M. A. Nielsen, "Prescription for experimental determination of the dynamics of a quantum black box," *J. Modern Opt.*, vol. 44, nos. 11–12, pp. 2455–2467, Nov. 1997.

[21] Y. Li and S. C. Benjamin, "Efficient variational quantum simulator incorporating active error minimization," *Phys. Rev. X*, vol. 7, no. 2, Jun. 2017, Art. no. 021050.

[22] K. Temme, S. Bravyi, and J. M. Gambetta, "Error mitigation for short-depth quantum circuits," *Phys. Rev. Lett.*, vol. 119, no. 18, Nov. 2017, Art. no. 180509.

[23] A. G. Fowler, M. Mariantoni, J. M. Martinis, and A. N. Cleland, "Surface codes: Towards practical large-scale quantum computation," *Phys. Rev. A, Gen. Phys.*, vol. 86, no. 3, Sep. 2012, Art. no. 032324.

[24] Y. Hirasaki, S. Daimon, T. Itoko, N. Kanazawa, and E. Saitoh, "Detection of temporal fluctuation in superconducting qubits for quantum error mitigation," *Appl. Phys. Lett.*, vol. 123, no. 18, 2023, Art. no. 184002, doi: 10.1063/5.0166739.

[25] A. Seif, H. Liao, V. Tripathi, K. Krsulich, M. Malekakhlagh, M. Amico, P. Jurcevic, and A. Javadi-Abhari, "Suppressing correlated noise in quantum computers via context-aware compiling," 2024, *arXiv:2403.06852*.

[26] Z. Wang and H. Tang, "Artificial intelligence for quantum error correction: A comprehensive review," 2024, *arXiv:2412.20380*.

[27] A. Zlokapa and A. Gheorghiu, "A deep learning model for noise prediction on near-term quantum devices," 2020, *arXiv:2005.10811*.

[28] J. Bausch, A. W. Senior, F. J. H. Heras, T. Edlich, A. Davies, M. Newman, C. Jones, K. Satzinger, M. Y. Niu, S. Blackwell, G. Holland, D. Kafri, J. Atalaya, C. Gidney, D. Hassabis, S. Boixo, H. Neven, and P. Kohli, "Learning high-accuracy error decoding for quantum processors," *Nature*, vol. 635, no. 8040, pp. 834–840, Nov. 2024.

[29] L. Gong, W. Ding, Z. Li, Y. Wang, and N. Zhou, "Quantum K-nearest neighbor classification algorithm via a divide-and-conquer strategy," *Adv. Quantum Technol.*, vol. 7, no. 6, Jun. 2024, Art. no. 2300221.

[30] J. Lee, N. Kang, S.-H. Lee, H. Jeong, L. Jiang, and S.-W. Lee, "Fault-tolerant quantum computation by hybrid qubits with bosonic cat code and single photons," *PRX Quantum*, vol. 5, no. 3, Aug. 2024, Art. no. 030322.

[31] A. Strikis, D. Qin, Y. Chen, S. C. Benjamin, and Y. Li, "Learning-based quantum error mitigation," *PRX Quantum*, vol. 2, no. 4, Nov. 2021, Art. no. 040330.

[32] A. Muqeet, S. Ali, T. Yue, and P. Arcaini, "A machine learning-based error mitigation approach for reliable software development on IBM's quantum computers," in *Proc. Companion 32nd ACM Int. Conf. Found. Softw. Eng.*, Jul. 2024, pp. 80–91.

[33] Z. Han, J. Zhao, H. Leung, K. F. Ma, and W. Wang, "A review of deep learning models for time series prediction," *IEEE Sensors J.*, vol. 21, no. 6, pp. 7833–7848, Mar. 2021.

[34] C. Deb, F. Zhang, J. Yang, S. E. Lee, and K. W. Shah, "A review on time series forecasting techniques for building energy consumption," *Renew. Sustain. Energy Rev.*, vol. 74, pp. 902–924, Jul. 2017.

[35] H. Oukhouya and K. El Himdi, "Comparing machine learning methods—SVR, XGBoost, LSTM, and MLP—For forecasting the Moroccan stock market," *Comput. Sci. Math. Forum*, vol. 7, no. 1, p. 39, 2023.

[36] E. Zivot and J. Wang. (2003). *Vector Autoregressive Models for Multivariate Time Series*. [Online]. Available: https://api.semanticscholar.org/CorpusID

[37] C. K. I. Williams and C. E. Rasmussen, "Gaussian processes for regression," in *Proc. 9th Int. Conf. Neural Inf. Process. Syst. (NIPS)*, Denver, CO, USA. Cambridge, MA, USA: MIT Press, 1995, pp. 514–520.

[38] D. W. Scott, *Multivariate Density Estimation: Theory, Practice, and Visualization*. Hoboken, NJ, USA: Wiley, 2015.
[39] C. Tofallis, "A better measure of relative prediction accuracy for model selection and model estimation," *J. Oper. Res. Soc.*, vol. 66, no. 8, pp. 1352–1362, Aug. 2015.
[40] H. Akaike, "A new look at the statistical model identification," *IEEE Trans. Autom. Control*, vol. AC-19, no. 6, pp. 716–723, Dec. 1974.
[41] G. Schwarz, "Estimating the dimension of a model," *Ann. Statist.*, vol. 6, no. 2, pp. 461–464, Mar. 1978.
[42] W. A. Broock, J. A. Scheinkman, W. D. Dechert, and B. LeBaron, "A test for independence based on the correlation dimension," *Econ. Rev.*, vol. 15, no. 3, pp. 197–235, 1996.
[43] R. W. Wanzala and L. O. Obokoh, "Sustainability implications of commodity price shocks and commodity dependence in selected sub-saharan countries," *Sustainability*, vol. 16, no. 20, p. 8928, 2024.
[44] C. W. J. Granger, "Investigating causal relations by econometric models and cross-spectral methods," *Econometrica*, vol. 37, no. 3, p. 424, Aug. 1969.
[45] M. A. Saleh and H. M. Rasel, "Machine learning for groundwater levels: Uncovering the best predictors," *Sustain. Water Resour. Manage.*, vol. 10, no. 5, pp. 1–23, Oct. 2024.
[46] L. Breiman, "Bias, variance, and arcing classifiers," Dept. Statist., Univ. California, Berkeley, CA, USA, Tech. Rep. 460, 1996.
[47] P. J. Huber and E. M. Ronchetti, *Robust Statistics*. Hoboken, NJ, USA: Wiley, 2011.
[48] V. Vapnik, *The Nature of Statistical Learning Theory*. Cham, Switzerland: Springer, 2013.
[49] G. Nicodemo, D. Pizzocri, L. Luzzi, C. Guéneau, and P. Van Uffelen, "Data assimilation for fuel performance code development: Application to oxide fuel thermal properties," in *Proc. Top Fuel Eur. Nucl. Soc.*, 2024, pp. 160–170.

MOHAMMADREZA SAGHAFI received their B.S. and M.S. degrees in electrical engineering from the University of Tehran, Iran, in 2017 and 2020, respectively. They are currently pursuing the Ph.D. degree with the Department of Electrical and Computer Engineering, Northern Virginia Center, Virginia Tech. Their research interests include developing innovative solutions for emerging challenges in power systems, with a focus on quantum noise mitigation and robust estimation techniques to improve system reliability and efficiency.

LAMINE MILI (Life Fellow, IEEE) is currently a Professor and the Program Director of the Electrical and Computer Engineering Department, Northern Virginia Center, Virginia Tech. He was a Visiting Professor with the Swiss Federal Institute of Technology, Lausanne, Switzerland; the Grenoble Institute of Technology and the École Supérieure d'Électricité, France; the École Polytechnique de Tunisie, Tunisia; and did consulting work for Hydro-Quebec and French Power Transmission Company, RTE. He has published over 150 technical articles and edited two books. His research interests include robust estimation and control, robust Kalman filtering, copula indices, risk management of complex systems to catastrophic failures, non-linear dynamics, and bifurcation theory.