American Journal of Environment and Climate (AJEC)

ISSN: 2832-403X (ONLINE)

VOLUME 4 ISSUE 1 (2025)

PUBLISHED BY
E-PALLI PUBLISHERS, DELAWARE, USA

American Journal of Environment and Climate (AJEC)

Volume 4 Issue 1, Year 2025
ISSN: 2832-403X (Online)
DOI: https://doi.org/10.54536/ajec.v4i1.2932
https://journals.e-palli.com/home/index.php/ajec

Modeling Time Series Global Mean Temperature Index Using SARIMA, NARNN, and SARIMA-NARNN Hybrid Models

Yeong Nain Chi[1*]

Article Information

Received: September 10, 2024

Accepted: October 16, 2024

Published: February 10, 2025

Keywords

Global Mean Temperature Index, Modeling, NARNN, SARIMA, SARIMA-NARNN Hybrid, Time Series

ABSTRACT

This study aimed to demonstrate the efficacy of time series models in both modeling and forecasting processes, leveraging extensive monthly global mean temperature index data spanning from January 1880 to December 2016. Employing the Box–Jenkins methodology, this study identified the SARIMA (2,1,2)(0,0,2)12 model with drift as the most suitable fit for the time series, determined by its lowest AIC value. Utilizing the LM algorithm, the empirical results revealed that the NARNN model, comprising 11 neurons in the hidden layer and 6-time delays, exhibited superior performance among nonlinear autoregressive neural network models, boasting a smaller MSE value. While SARIMA and NARNN models excel in addressing linear and nonlinear challenges within time series data, respectively, this study proposes the use of a SARIMA-NARNN hybrid model. By combining SARIMA and NARNN capabilities, this hybrid approach offers a comprehensive solution, effectively addressing both linear and nonlinear modeling requirements. Comparative analyses underscored the superiority of the hybrid model over standalone SARIMA(2,1,2)(0,0,2)12 with drift model and the NARNN model with 11 neurons in the hidden layer and 6 time delays, showcasing higher accuracy through its lowest MSE in this study. These findings contribute significantly to bridging critical gaps in time series forecasting methodologies by leveraging the strengths of both statistical and machine learning approaches.

INTRODUCTION

Global mean temperature rise denotes the escalation in the Earth's average surface temperature over a specified period, typically spanning decades or centuries. This metric serves as a critical indicator of climate change, predominantly propelled by human activities, notably the release of greenhouse gases such as carbon dioxide (CO_2), methane (CH_4), and nitrous oxide (N_2O). The repercussions of global temperature rise are extensive and profound, encompassing phenomena like melting ice caps and glaciers, rising sea levels, intensified extreme weather events, disruption of ecosystems, ocean acidification, and impacts on agriculture and food security.

Throughout history, Earth's climate has undergone natural temperature fluctuations due to factors such as volcanic eruptions and variations in solar radiation. However, since the Industrial Revolution, human actions, particularly fossil fuel combustion, deforestation, and industrial processes, have substantially elevated greenhouse gas concentrations in the atmosphere. These gases trap solar heat, culminating in a warming phenomenon recognized as the greenhouse effect.

The Intergovernmental Panel on Climate Change (IPCC), a leading global authority on climate change assessment, has amassed compelling evidence illustrating the nexus between human activities and global temperature rise. According to IPCC reports, the global mean surface temperature has ascended by approximately 1.1 degrees Celsius (°C) since the pre-industrial era (late 19th century), with the bulk of this warming occurring in recent decades (Haskett, 2022).

The global mean temperature index is a pivotal metric in assessing Earth's climate dynamics and monitoring long-term temperature trends. Analyzing global mean temperature time series data poses several challenges due to its intrinsic complexity and the presence of diverse influencing factors, including greenhouse gas emissions, natural climate variability, and external forcings like volcanic eruptions and solar radiation.

Furthermore, the imperative of addressing climate change underscores the necessity of developing reliable forecasting models for global mean temperature. Accurate predictions are indispensable for guiding policy decisions, implementing mitigation strategies, and adapting to evolving environmental conditions. As global temperatures continue to escalate, the repercussions of climate change intensify, impacting ecosystems, economies, and societies worldwide. Hence, timely and accurate global mean temperature forecasts are indispensable for policymakers to formulate effective climate policies, allocate resources efficiently, and implement adaptation strategies to mitigate adverse climate change effects.

While traditional statistical methods like Seasonal Autoregressive Integrated Moving Average (SARIMA) have been extensively utilized for modeling and forecasting time series data by capturing linear dependencies, seasonal patterns, and autocorrelation, they may struggle to encapsulate the nonlinear dynamics and intricate interactions inherent in global mean

[1] University of Maryland, Eastern Shore, USA
[*] Corresponding author's e-mail: ychi@umes.edu

temperature data. Consequently, advanced statistical and machine learning techniques have gained prominence for modeling and forecasting global mean temperature time series data. These approaches offer the flexibility to capture nonlinearities and complex interactions, thereby enhancing prediction accuracy and facilitating informed decision-making.

This paper delves into the methodologies and techniques employed in modeling global mean temperature index time series data, elucidating the associated challenges, opportunities, and implications. By integrating SARIMA, Nonlinear Autoregressive Neural Network (NARNN), and SARIMA-NARNN hybrid modeling approaches, this paper aims to provide a comprehensive framework for modeling global mean temperature time series data, enabling informed decision-making and policy development in the face of a rapidly evolving climate. The subsequent sections of the paper are structured as follows: Section 2 presents a literature review of time series analysis of global mean temperature, Section 3 introduces global mean temperature time series data, Section 4 delineates SARIMA, NARNN, SARIMA-NARNN hybrid models, and training algorithms employed in this study, Section 5 presents and analyzes empirical results, and finally, the Discussion and Conclusions section synthesizes the study's findings.

LITERATURE REVIEW

The study of modeling and forecasting global mean temperature time series data has garnered considerable attention in climate science literature, owing to its pivotal role in comprehending climate dynamics and projecting future climate trends. Among the widely utilized techniques for modeling time series data, including global mean temperature index, is the autoregressive integrated moving average (ARIMA) model. ARIMA models are favored for their capability to capture linear trends and autocorrelation (Romilly, 2005; Wang *et al.*, 2023). Specifically, the SARIMA model stands out for its effectiveness in capturing seasonal patterns and trends in the data, rendering it suitable for analyzing global mean temperature indices with cyclical behavior (Ye *et al.*, 2013). In addition to time series modeling techniques, trend analysis (Mudelsee, 2019), and decomposition methods have been employed to scrutinize global temperature trends. The Innovative Trend Template (ITT) approach presents a novel method that divides the time series into two equal sub-segments, arranges them in ascending order, and plots them against each other to derive a scatter of points (Mohorji, 2017). This approach facilitates a refined calculation of global warming trends and can yield insights into underlying patterns and dynamics. Another technique used in trend analysis is the Pairwise Comparison (PAI) method, which relaxes the assumption of a linear relationship between proxy data and temperature (Kaufman *et al.*, 2020). Instead, PAI relies on the relative ranking of data points, acknowledging that proxy data may not perfectly represent temperature due to random and systematic errors.

Various statistical methods have been employed in reconstructing global mean surface temperature using proxy data from sources like tree rings, corals, and ice cores (PAGES 2k Consortium, 2019). One approach is the Composite-Plus-Scaling (CPS) method, which amalgamates proxy records into a single time series (composite) by computing a weighted mean (PAGES 2k Consortium, 2019). Other methods include regression-based techniques like Principal Component Regression (PCR) and the M08 method, as well as newer linear methods like Optimal Information Extraction (OIE) and Bayesian Hierarchical Models (BHM) (PAGES 2k Consortium, 2019). Furthermore, researchers have explored techniques accounting for nonlinear relationships between proxy values and temperature, such as the PAI method mentioned earlier, or methods combining information from proxy data and climate models, like Data Assimilation (DA) techniques (PAGES 2k Consortium, 2019).

This literature review underscores the diverse range of methodologies and techniques employed in modeling global mean temperature indices' time series. As climate change research progresses, these methodologies will play a pivotal role in understanding and predicting global mean temperature trends. Nonetheless, there is a growing interest in utilizing neural networks to model and forecast global mean temperature. Despite global mean temperature 's significance on both global and local scales, there remains a dearth of studies in the technical literature concerning global mean temperature forecasting schemes.

While neural network models offer flexibility and scalability in modeling complex data patterns, they may lack interpretability and necessitate large amounts of training data to attain optimal performance. To mitigate these challenges, hybrid modeling approaches have emerged as promising alternatives for global mean temperature forecasting, amalgamating the strengths of both statistical and machine learning techniques. Hybrid models, such as the SARIMA-NARNN hybrid model proposed in this study, integrate traditional ARIMA modeling with neural network-based forecasting to capture both linear and nonlinear dynamics present in global mean temperature data.

MATERIALS AND METHODS

Data

The long-term records of monthly global mean temperature index from January 1880 to December 2016 (Figure 1) are publicly available through GISTEMP: NASA Goddard Institute for Space Studies (GISS) Surface Temperature Analysis, Global Land-Ocean Temperature Index (https://datahub.io/core/global-temp#readme). This dataset encompasses combined Land-Surface Air and Sea-Surface Water Temperature Anomalies, representing deviations from the corresponding 1951-1980 means. The average global mean temperature index was 0.0244, with a standard deviation of 0.3437 (Minimum: -0.78, Maximum: 1.35, and Median: -0.05).

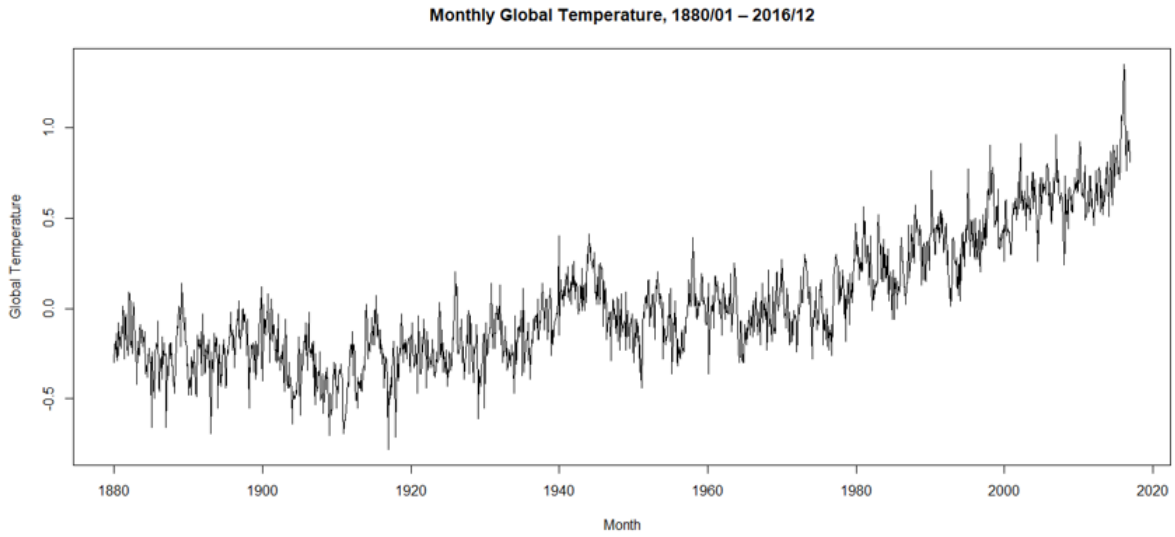

Figure 1: Time Series Plot of Monthly Global Mean Temperature Index, January 1880 - December 2016 (R Output)

Seasonal Autoregressive Integrated Moving Average (SARIMA) Model

A time series involves data collected sequentially over time. In a univariate time-series, it comprises single (scalar) observations recorded at specific time points, denoted as t. The sequence of random variables $\{y_t: t = 1, 2, ..., T\}$, $y_t \in R$ where $t \in T$ represents the time index when the data was observed, constitutes a stochastic process. This stochastic process is commonly employed in modeling time series data to determine its parameters and subsequently utilize it as a model for predicting future values of the time series.

The ARIMA model is a statistical analysis tool for both modeling and forecasting time series data, aiming to predict future trends. Each component of the ARIMA model—autoregressive (AR), integrated (I), and moving average (MA)—serves the purpose of enhancing the model's capability to predict future points in the time series (Montgomery *et al.*, 2008). Statistically, the ARIMA(p, d, q) model can be represented as:

$$y_t = \phi_1 y_{t-1} + \phi_2 y_{t-2} + \cdots + \phi_p y_{t-p} + e_t + \theta_1 e_{t-1} + \theta_2 e_{t-2} + \cdots + \theta_q e_{t-q} = \sum_{i=1}^{p} \phi_i y_{t-i} - \sum_{j=1}^{q} \theta_j e_{t-j} + e_t \quad (1)$$

where p = the order of the autoregressive process (the number of lagged terms),

d = the number of differences required to make the time series stationary,

q = the order of the moving average process (the number of lagged terms),

$\Phi = (\Phi_1, \Phi_2, ..., \Phi_p)$ is the vector of model coefficients for the autoregressive process,

$\theta = (\theta_1, \theta_2, ..., \theta_q)$ is the vector of model coefficients for the moving average process, and

e_t = the residual error (i.e., white noise).

In backshift notation, denoted as B, the "backshift operator" or "lag operator" serves as a valuable notational tool when dealing with time series lags. For instance, $By_t = y_{t-1}$ denotes "backing up by one time unit," while $Bky_t = y_{t-k}$ represents "backshifting k times." Therefore, the ARIMA (p,d,q) model can be expressed in backshift notation as:

$$\phi_p(B)(1 - B)^d y_t = c + \theta_q(B)e_t \quad (2)$$

where $\Phi_p(B) = (1 - \Phi_1 B - ... - \Phi_p B^p) = (1 - \Sigma^p_{i=1} \Phi_i B^i)$, $\theta_q(B) = (1 - \theta_1 B - ... - \theta_q B^q) = (1 - \Sigma^q_{j=1} \theta_j B^j)$, c is a constant, and et is the error term.

The SARIMA model is an extension of the ARIMA model designed specifically for univariate time series with seasonal patterns. In statistical notation, SARIMA(p, d, q)(P, D, Q)S represents the SARIMA model, where p, d, and q are the parameters for the non-seasonal components, and P, D, and Q are the parameters for the seasonal components, with s denoting the seasonal period. This model can be expressed in backshift notation as:

$$\phi_p(B)\Phi_p(B^m)(1 - B)^d(1 - B^m)^D y_t = c + \theta_q(B)\Theta Q(B^m)e_t \quad (3)$$

where P = the order of the seasonal autoregressive process,

D = the number of seasonal differences applied to the time series,

Q = the order of the seasonal moving average process,

s = the seasonality of the model, i.e., the number of time steps for a single seasonal period,

$\Phi_p(B^m) = (1 - \Phi_1 B^m - ... - \Phi_p B^{mP}) = (1 - \Sigma^P_{i=1} \Phi_i B^{mi})$, and

$\Theta_Q(B^m) = (1 - \Theta_1 B^m - ... - \Theta_Q B^{mQ}) = (1 - \Sigma^Q_{j=1} \Theta_j B^j)$.

In time series forecasting, the Box-Jenkins methodology (Box & Jenkins, 1970) outlines a systematic approach to identifying, estimating, verifying, and forecasting ARIMA models (Box *et al.*, 2015). This methodology serves as the foundation for forecasting the SARIMA model in this study, leveraging the autocorrelation function (ACF) and partial autocorrelation function (PACF) to assess the stationarity of the univariate time series and determine appropriate lag lengths.

A fundamental assumption in time series analysis is that the series is stationary, indicating that its statistical properties (e.g., mean and variance) remain constant over time. Hence, the Box-Jenkins methodology initially presumes the time series to be stationary. If not, differencing can be employed to induce stationarity. Through empirical examination, plots and summary

statistics aid in identifying trends and autoregressive components, providing insight into the required degree of differencing and lag size for model identification.

To select optimal parameters for the time series model, criteria such as Akaike's Information Criterion (AIC) or Bayesian Information Criterion (BIC) are employed, aiming to minimize their values in determining the orders of a SARIMA model. In the diagnostic checking phase, residual error plots and statistical tests are utilized to evaluate model fit, assess the model's performance within the context of the time series data, and pinpoint areas for potential improvement.

Nonlinear Autoregressive Neural Network (NARNN) Model

The AR process aims to elucidate the current value of the time series, yt, through a function of its p previous values, (y_{t-1}, y_{t-2}, ..., y_{t-p}). Hence, the AR process of order p, denoted as AR(p), is defined by the equation:

$$y_t = \phi_1 y_{t-1} + \phi_2 y_{t-2} + \cdots + \phi_p y_{t-p} + e_t = \sum_{i=1}^{p} \phi_i y_{t-i} + e_t \quad (4)$$

where,

$\Phi = (\Phi_1, \Phi_2, ...,\Phi_p)$ represents the vector of model coefficients for the autoregressive process, and et denotes white noise, with $e_t \sim N(0, \sigma^2)$ (Montgomery *et al.*, 2008).

The NARNN serves as a natural extension of the traditional linear AR(p) process. A NARNN of order p can be expressed as:

In this equation, $\Phi(\cdot)$ represents an unknown function determined by the neural network structure and connection weights, w denotes a vector of all parameters (weights), and ε_t signifies the error term. Thus, the NARNN performs a nonlinear functional mapping from past observations, (y_{t-1}, y_{t-2}, ..., y_{t-p}), to the future value, y_t, effectively functioning as a nonlinear autoregressive model (Zhang, 2003).

Utilizing lagged values of the time series data as inputs to

$$y_t = \Phi(y_{t-1}, y_{t-2}, \cdots, y_{t-p}, w) + \varepsilon_t \quad (5)$$

a neural network yields the NARNN model.

Mathematically, the NARNN model (Benrhmach *et al.*, 2020) can be expressed in the following form:

$$y_t = a_0 + \sum_{j=1}^{k} w_j \Phi(b_{0j} + \sum_{i=1}^{d} w_{ij} y_{t-i}) + \varepsilon_t \quad (6)$$

where,

d = the number of input units,

k is the number of hidden units,

a_0 is the constant corresponding to the output unit,

b_{0j} is the constant corresponding to the hidden unit j,

w_j is the weight of the connection between the hidden unit j and the output unit,

w_{ij} is the parameter corresponding to the weight of the connection between the input unit i and the hidden unit j, and

$\Phi(\cdot)$ is a nonlinear function, so-called this the transfer (activation) function. The logistic function (i.e., sigmoid) is commonly used as the hidden layer transfer function, that is, $\Phi(y) = 1 / (1 + \exp(-y))$.

SARIMA-NARNN Hybrid Model

The SARIMA and NARNN models excel in modeling linear and nonlinear aspects of time series data, respectively. Combining these models into a hybrid model offers the advantage of leveraging both linear and nonlinear modeling capabilities, making it a potentially superior choice for time series modeling. Assuming that an unknown function can effectively demonstrate the relationship between linear and nonlinear components in the time series, it can be represented as:

$$y_t = f(L_t, N_t) \quad (7)$$

Here, L_t represents the linear component, and N_t signifies the nonlinear component. Assuming a simple additive relationship between the linear and nonlinear components (Zhang, 2003), the time series can be expressed as:

$$y_t = L_t + N_t \quad (8)$$

Initially, the linear component is modeled using the SARIMA model. Subsequently, the residuals from the SARIMA model solely capture the nonlinear relationship, obtained by calculating the difference between observed values and predicted values:

$$e_t = y_t - \hat{y}_t \quad (9)$$

Where et denotes the residual of the linear model at time t, and $\hat{y}_t$ represents the predicted value for time t. To capture the nonlinear relationship, these residuals can be modeled using the NARNN model, expressed as:

$$\hat{y}_t = e_t = f(e_{t-1}, e_{t-2}, \cdots, e_{t-n}) + \varepsilon^t \quad (10)$$

Here, f represents the transformation function modeled by the NARNN, and εt denotes the random error. The forecasts from both the SARIMA and NARNN models are then combined to derive the forecast of the time series, denoted as $\hat{y}_t$:

$$\hat{y}_t = \hat{y}_t \text{ (SARIMA)} + \hat{y}t \text{ (NARNN)} \quad (11)$$

Training Algorithms

Levenberg-Marquardt (LM) Algorithm

The LM algorithm, initially proposed by Levenberg (1944) and later rediscovered by Marquardt (1963), is an iterative optimization technique used for solving nonlinear least squares problems. It combines aspects of gradient descent and Gauss-Newton methods. At each iteration, it computes a search direction by incorporating information from the gradient of the objective function (residuals) and the curvature of the objective function surface (approximated by the Jacobian matrix). The LM algorithm modifies the Gauss-Newton method by introducing a damping parameter that ensures stable convergence, particularly when the curvature of the objective function surface changes significantly (Gavin, 2020).

The LM algorithm, well-suited to neural network training where the performance index is the mean squared error, exhibits a useful feature: as the combination coefficient increases, it approaches the steepest descent algorithm with a small learning rate, while decreasing to zero makes it akin to Gauss-Newton (Hagan *et al.*, 2014). Generally, LM converges faster than gradient descent methods for well-conditioned problems, effectively handles large-scale problems, and is efficient for non-linear least squares optimization problems. However, it necessitates the

computation of the Jacobian matrix and may converge to local minima for poorly conditioned problems.

Bayesian Regularization (BR) Algorithm

The BR algorithm, introduced by MacKay (1992), is a technique for regularizing models by incorporating prior knowledge about the parameters. It formulates the problem within a Bayesian framework, defining prior distributions over the model parameters. The objective is to maximize the posterior probability of the parameters given the data, achieved by combining the likelihood of the data given the parameters with the prior distribution. Regularization penalizes complex models, encouraging simplicity and preventing overfitting (Sariev & Germano, 2020).

Relying on the probabilistic interpretation of network parameters, the BR algorithm optimizes regularization parameters based on the Hessian matrix calculation at the minimum point. It employs a probabilistic framework for network weight and architecture identification, updating parameters iteratively until convergence (Yue *et al.*, 2011). While offering a principled framework for regularization and uncertainty estimation of model parameters, BR can be computationally expensive due to posterior distribution estimation and sensitive to the choice of prior distributions.

Scaled Conjugate Gradient (SCG) Algorithm

The SCG algorithm, developed by Møller (1993), is an iterative optimization method to find function minima. Belonging to the conjugate gradient family, it incorporates adaptations for improved convergence speed and efficiency. Unlike traditional methods, SCG employs an adaptive step-size approach and scales search directions based on error surface curvature, enhancing efficiency, especially for high-dimensional or ill-conditioned problems.

SCG is well-suited for large-scale optimization problems, avoiding second-order derivative computations, and is effective for ill-conditioned or nearly singular Hessians. However, it can be sensitive to step-size parameter choices and less effective for highly nonlinear optimization compared to LM.

Training Algorithm Comparison

Each training algorithm (LM, BR, and SCG) offers unique optimization and regularization approaches, valuable across diverse machine learning and optimization tasks. Their distinct mathematical frameworks and optimization methods suit various problem characteristics and objectives. Understanding their strengths and weaknesses enables their judicious selection for different optimization problems based on factors such as dimensionality, conditioning, and desired solution properties (Table1).

Table 1: Training Algorithm Comparison

Algorithm	Advantages	Disadvantages
LM Algorithm	Fast convergence; Efficient for non-linear least squares problems	Requires Jacobian computation; May converge to local minima
BR Algorithm	Principled framework for regularization; Balances fitting data with preventing overfitting	Computationally expensive; Sensitive to the choice of prior distributions
SCG Algorithm	Well-suited for large-scale optimization; No need for second-order derivatives	Sensitive to step-size parameters; Less effective for highly non-linear problems

RESULTS AND DISCUSSION

Seasonal Autoregressive Integrated Moving Average (SARIMA) Model

Identification of the SARIMA Model

R version 4.3.3 for Windows, an open-source software for statistical computing and graphics supported by the R Foundation for Statistical Computing, was utilized to model and forecast the monthly global mean temperature index from January 1880 to December 2016 in this study. The "decompose()" function in R was employed to estimate the seasonal, trend, and irregular components of the seasonal time series. Figure 2 illustrates plots depicting the original time series (top), the estimated trend component (second from the top), the estimated seasonal component (third from the top), and the estimated irregular component (bottom). Notably, the estimated trend component exhibits a consistent upward trend over time, while the estimated seasonal component clearly demonstrates seasonality, with a recurring pattern observed every 12 months (yearly).

Decomposing a time series involves separating it into three distinct components: a trend component, a seasonal component, and an irregular component. This decomposition facilitates the presentation of seasonally adjusted values. Seasonal adjustment entails estimating and removing seasonal effects from a time series that cannot be attributed to trend or cyclical dynamics, thereby revealing underlying non-seasonal features. This adjustment is achieved by subtracting the estimated seasonal component from the original time series. Consequently, the seasonally adjusted time series solely comprises the trend component and an irregular component (Figure 3), devoid of seasonal variation.

Upon inspecting the time series plot of the data, the Autocorrelation Function (ACF) serves as a valuable tool for discerning between stationary and non-stationary time

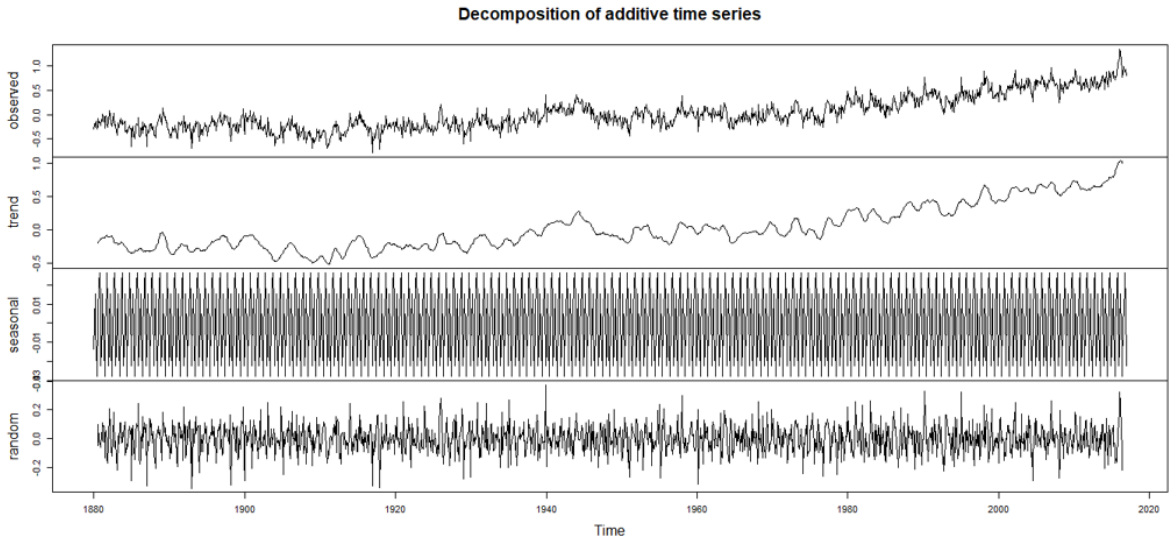

Figure 2: Decomposition of Monthly Global Mean Temperature Index, January 1880 - December 2016 (R Output)

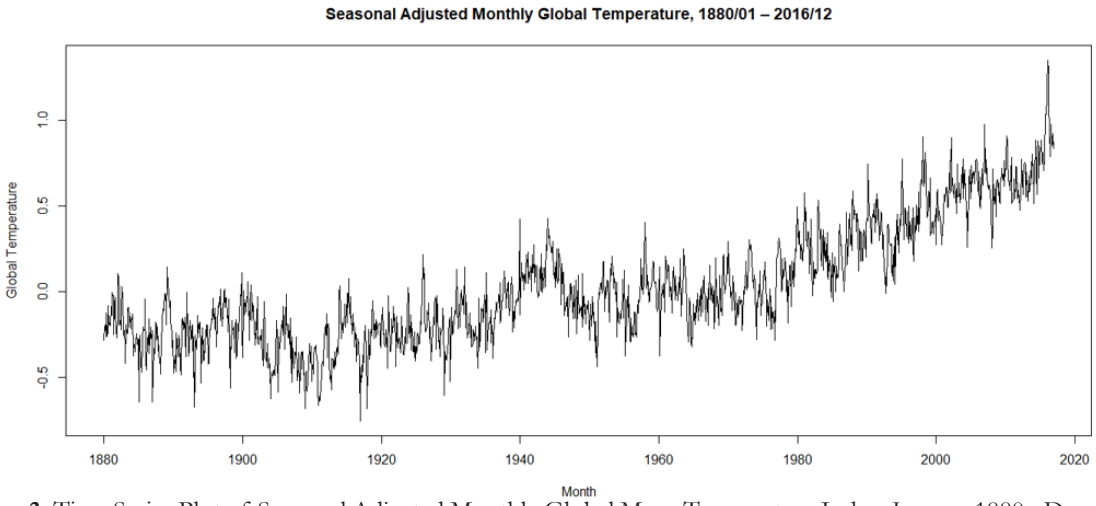

Figure 3: Time Series Plot of Seasonal Adjusted Monthly Global Mean Temperature Index, January 1880 - December 2016 (R Output)

series. The ACF (depicted in Figure 4) of the seasonal adjusted monthly global mean temperature index from January 1880 to December 2016 exhibited robust positive correlations extending up to 33 lags, indicating a lack of decay to zero. This observation suggests that the time series is non-stationary and necessitates differencing. This non-stationarity is further illustrated by a single spike at the first lag in the Partial Autocorrelation Function (PACF) plot (Figure 5), followed by small, seemingly random values after the initial lag. Typically, when both the ACF and PACF exhibit autocorrelation only at the first lag, as shown in Figures 4 and 5, it indicates that the time series follows an AR process. Additionally, the PACF cutoff after the first lag suggests a possible autoregressive behavior in the time series.

In time series analysis, differencing serves as a technique to transform a non-stationary time series into a stationary one. When both trend and seasonality are present, it is advisable to apply both a non-seasonal first difference and a seasonal difference separately. The first difference of a time series represents the changes from one period to the next. Notably, the graph depicting the first difference of the time series appears approximately stationary (Figure 6). Further validation using the Augmented Dickey-Fuller Test yielded a Dickey-Fuller statistic of -15.574 with a lag order of 11, and a p-value smaller than 0.01. This result led to the rejection of the null hypothesis, indicating non-stationarity. Instead, it suggested that the first difference of the time series was indeed stationary.

The ACF of the first differenced time series, as depicted in Figure 7, revealed a prominent positive spike at the first lag, followed by statistically significant correlations. Concurrently, the PACF of the first differenced time series, illustrated in Figure 8, exhibited a gradual decrease after the initial few lags, predominantly in a negative direction. Given that the ACF cutoff after the first lag

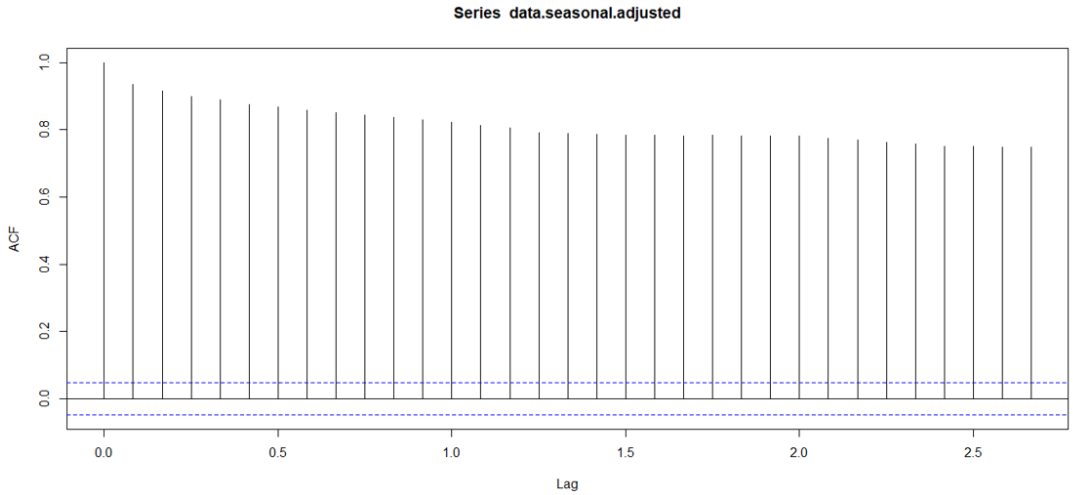

Figure 4: ACF Plot of Seasonal Adjusted Monthly Global Mean Temperature Index, January 1880 - December 2016 (R Output)

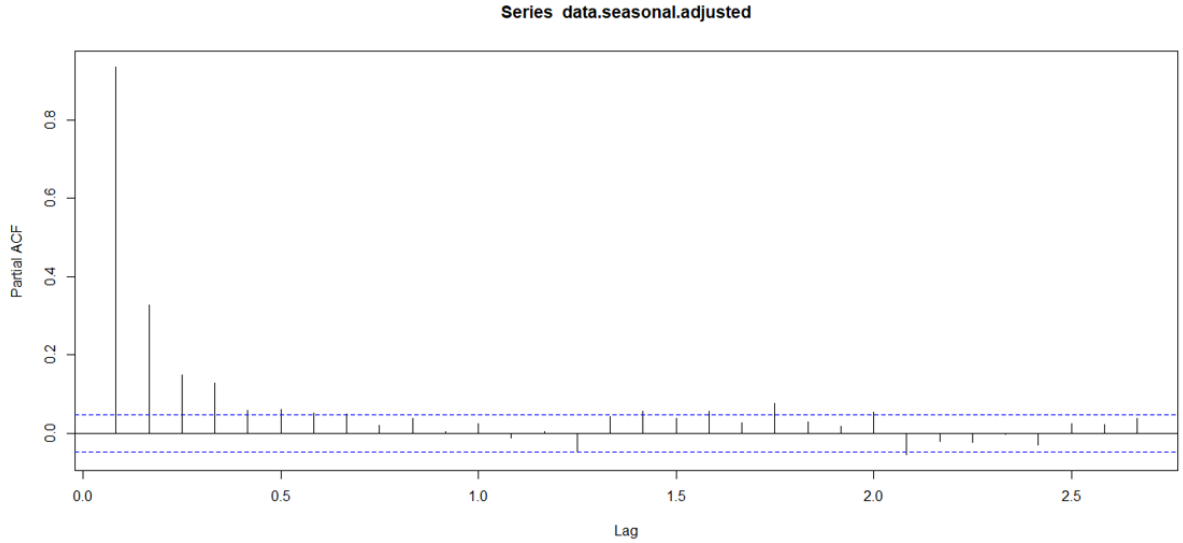

Figure 5: PACF Plot of Seasonal Adjusted Monthly Global Mean Temperature Index, January 1880 - December 2016 (R Output)

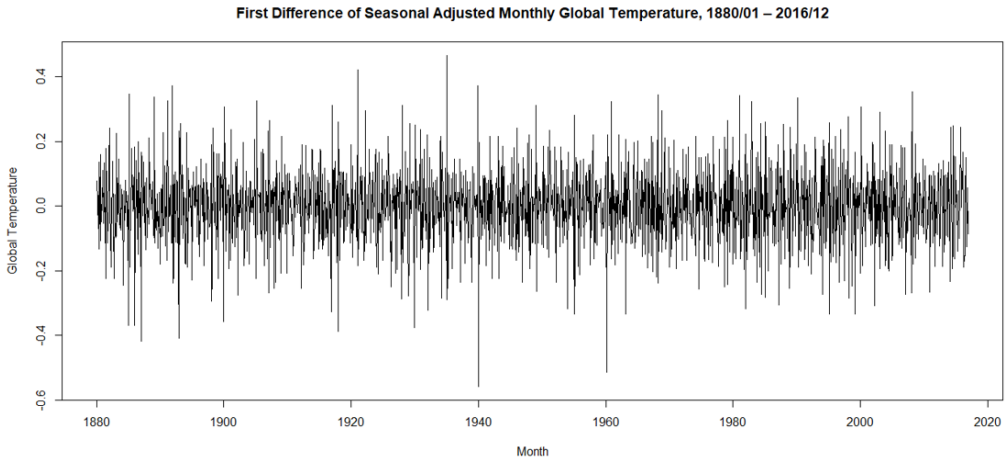

Figure 6: Time Series Plot of First Difference of Seasonal Adjusted Monthly Global Mean Temperature Index, January 1880 - December 2016 (R Output)

and the PACF displayed a gradual decline, a reasonable inference can be made that the first differenced time series follows an MA process. This observation aligns with the characteristics typically associated with an MA process.

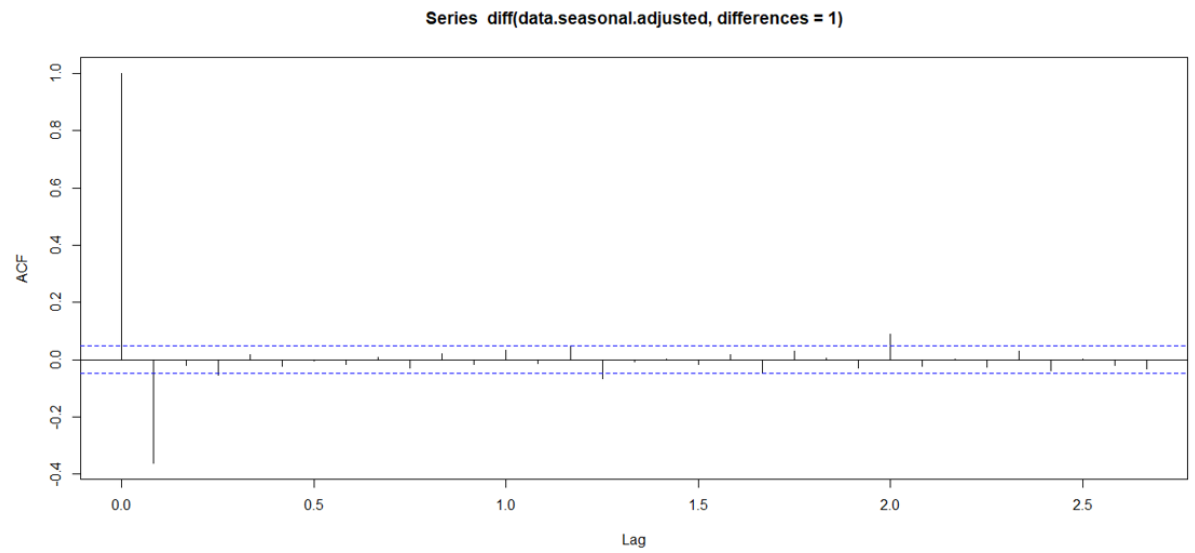

Figure 7: ACF Plot of First Difference of Seasonal Adjusted Monthly Global Mean Temperature Index, January 1880 - December 2016 (R Output)

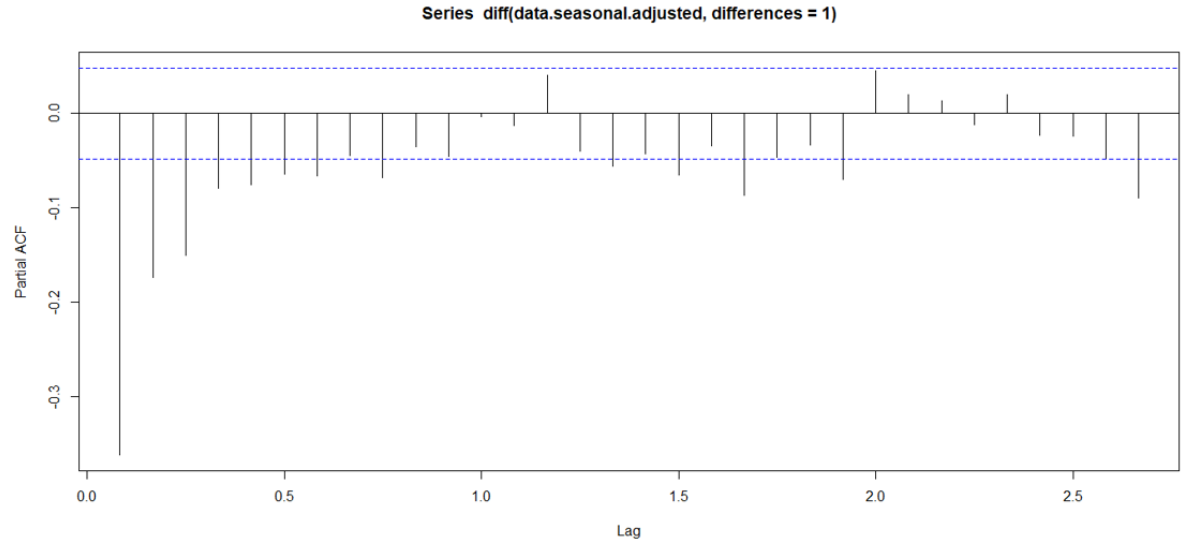

Figure 8: PACF Plot of First Difference of Seasonal Adjusted Monthly Global Mean Temperature Index, January 1880 - December 2016 (R Output)

Seasonal differencing serves to eliminate seasonal trends and can effectively address seasonal random walk-type non-stationarity. It involves taking the difference between a value and a value with a lag that is a multiple of the seasonality. In this instance, where seasonality equals 12 months per year, the span of the periodic seasonal behavior. As illustrated in Figure 9, the graph of the 12th difference of the time series appears approximately stationary. Additionally, the Augmented Dickey-Fuller Test yielded a test statistic of Dickey-Fuller = -57.46 with a lag order of 11 and a p-value smaller than 0.01. This outcome leads to the rejection of the null hypothesis, indicating non-stationarity, and suggests that the 12th first difference of the time series is indeed stationary.

Figure 10 displays an ACF characterized by a steady decay after the initial few lags, with statistically significant fluctuations alternating between positive and negative values. Conversely, Figure 11 illustrates a PACF primarily exhibiting a gradual negative decay in partial correlations toward zero. These observations align with the conventional guideline suggesting that when the PACF displays a gradual decrease, an MA process should be considered. This pattern is consistent with the behavior typically associated with an MA process.

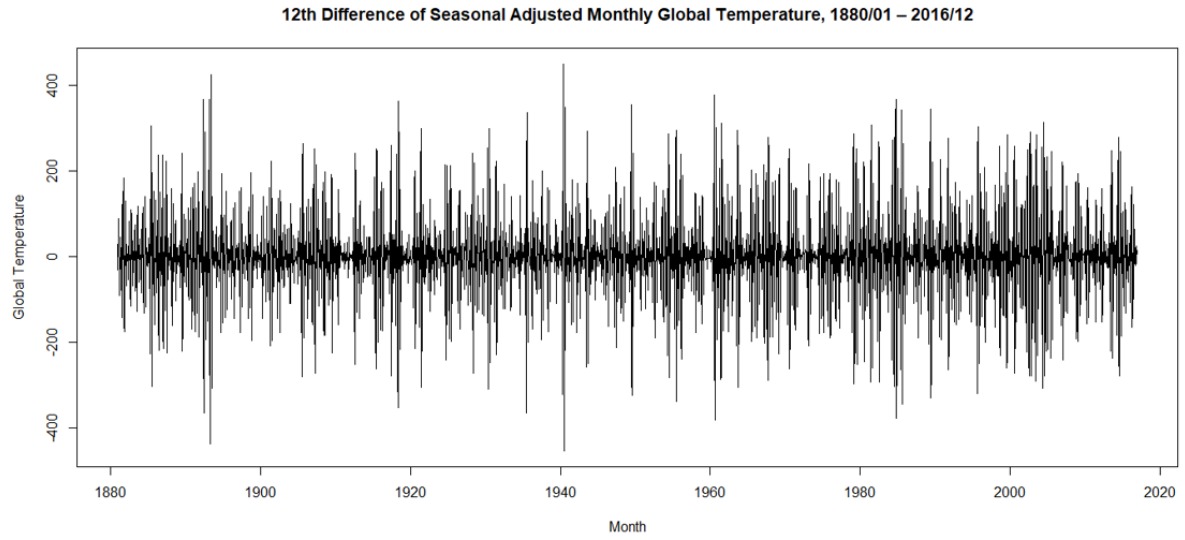

Figure 9: Time Series Plot of 12th Difference of Seasonal Adjusted Monthly Global Mean Temperature Index, January 1880 - December 2016 (R Output)

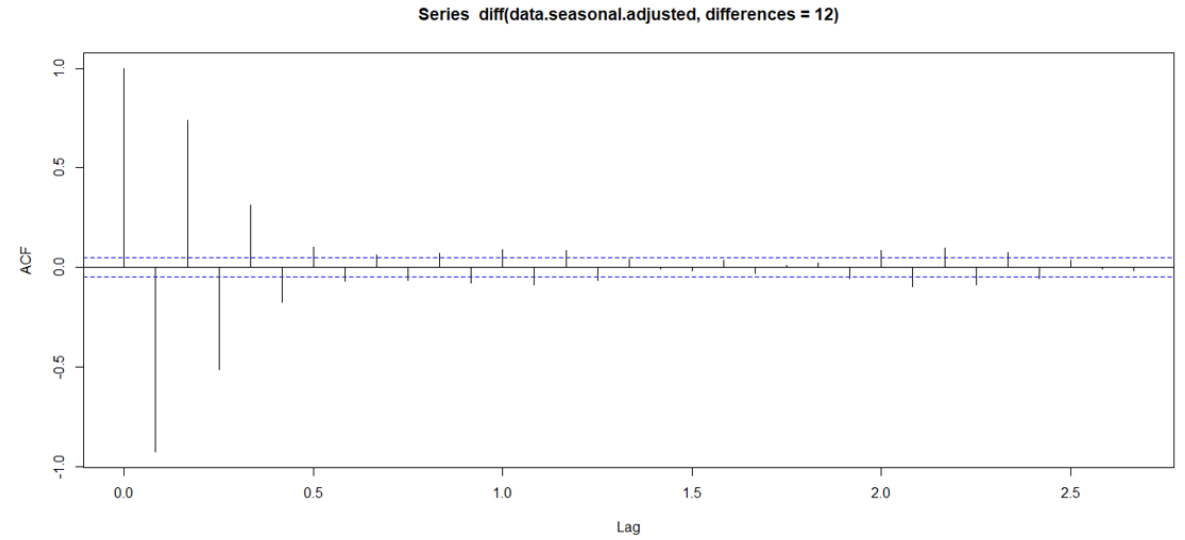

Figure 10: ACF Plot of 12th Difference of Seasonal Adjusted Monthly Global Mean Temperature Index, January 1880 - December 2016 (R Output)

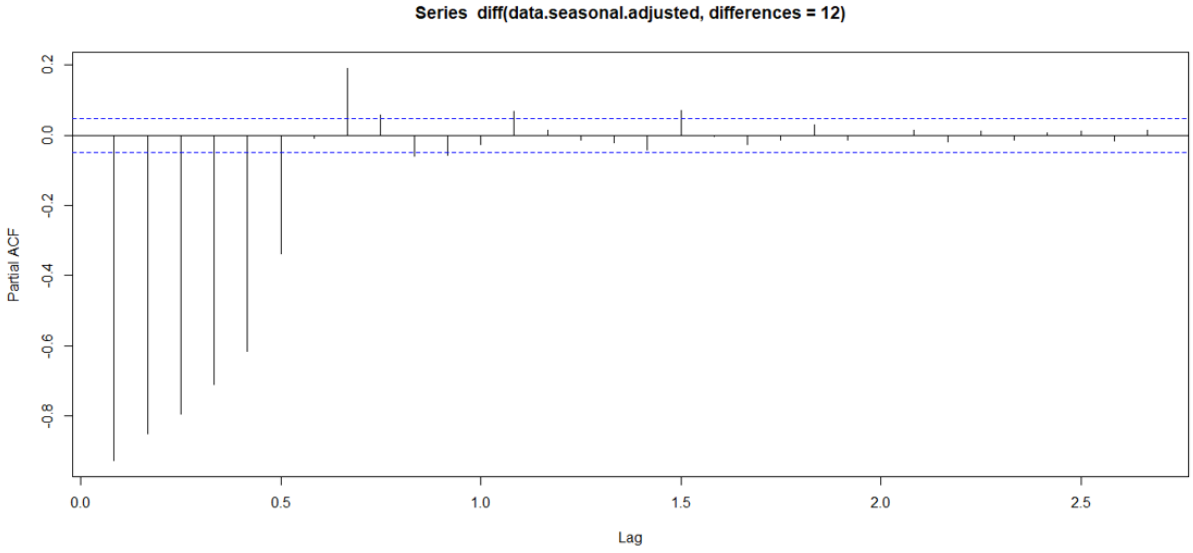

Figure 11: PACF Plot of 12th Difference of Seasonal Adjusted Monthly Global Mean Temperature Index, January 1880 - December 2016 (R Output)

Estimation of the SARIMA Model Parameters

Empirically, selecting the appropriate model order can be somewhat arbitrary. To address this challenge, the "auto.arima()" function from the "forecast" package in R 4.3.3 for Windows was utilized. This function identifies both the structure of the time series (stationary or not) and its type (seasonal or not), while setting the model parameters. It considers the Akaike Information Criterion (AIC), corrected AIC (AICc), or Bayesian Information Criterion (BIC) values to determine the best fit SARIMA model. Following this approach, the SARIMA $(2,1,2)(0,0,2)_{12}$ model with drift was identified as the best fit for the time series, based on the lowest AIC value (-2653.44) observed in this study. Consequently, the SARIMA $(2,1,2)(0,0,2)_{12}$ with drift model was selected for further forecasting. The parameters of this model are presented in Table 2. The forecasting equation for the SARIMA $(2,1,2)(0,0,2)_{12}$ with drift model would be as follows:

$$(1 - \phi_1 B - \phi_2 B^2)(1 - B)\hat{Y}_t = c + (1 - \theta_1 B - \theta_2 B^2)(1 - \Theta_1 B^{12} - \Theta_2 B^{12}) \quad (11)$$

The SARIMA$(2,1,2)(0,0,2)_{12}$ model for the seasonal adjusted monthly global mean temperature index, spanning from January 1880 to December 2016, can be expressed as follows:

$$(1 - 0.9717B + 0.0829B^2)(1 - B)\hat{Y}_t = 0.0006 + (1 + 1.4828B - 0.4920B^2)(1 - 0.0392B^{12} - 0.0922B^{12}) \quad (12)$$

Table 2: Parameters of the SARIMA(2,1,2)(0,0,2)12 with Drift Model

Parameter	Estimate	Standard Error
Constant	6e-04	3e-04
AR Lag 1	0.9717	0.0853
AR Lag 2	-0.0829	0.0619
Difference		
MA Lag 1	-1.4828	0.0799
MA Lag 2	0.4920	0.0776
SMA1	0.0392	0.0251
SMA2	0.0922	0.0239
Sigma^2 estimated as 0.01157, Log Likelihood = 1334.72		
AIC = -2653.44, AICc = -2653.36, BIC = -2610.21		
Training Set Error Measures:		
RMSE = 0.1073, MSE = 0.0115, MAE = 0.0834, MAPE = 206.8279		

Source: own work

Diagnostic Checking of the SARIMA Model

When building forecasting models, it's crucial to verify that the residuals meet certain assumptions, such as being uncorrelated and normally distributed. In Figure 12, the top figure indicates that the residuals from the SARIMA$(2,1,2)(0,0,2)_{12}$ with drift model maintain a consistent location and scale, satisfying the assumption of constant behavior. Likewise, the bottom right figure demonstrates that the residual histogram doesn't reveal any substantial deviation from normality. Examining the ACF plot of the residuals, as shown in the bottom left figure of Figure 12, reveals that all sample

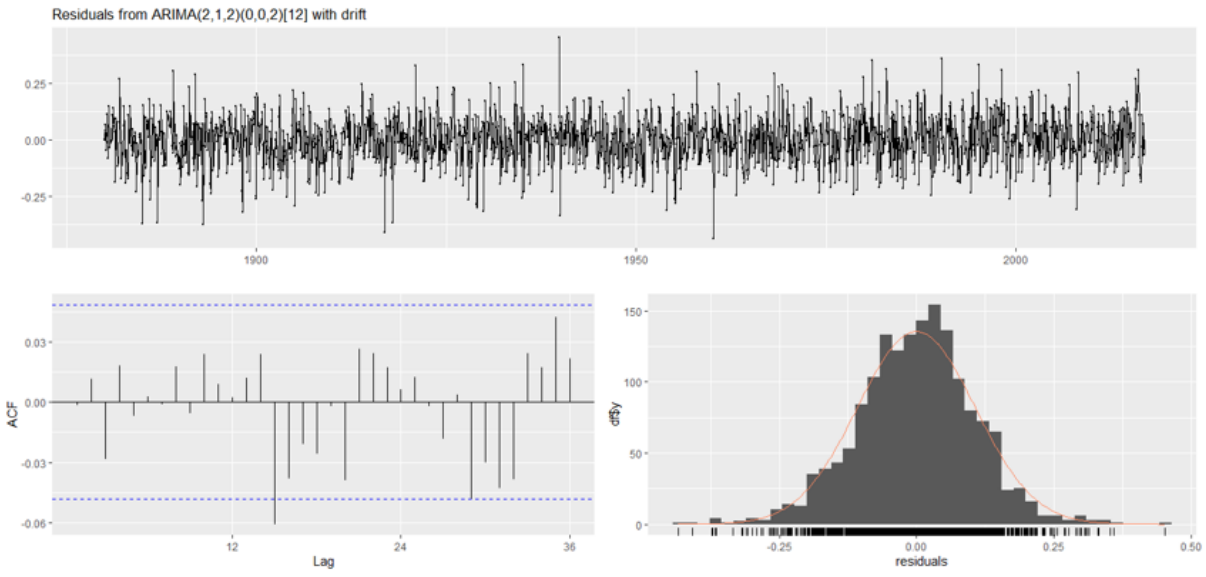

Figure 12: Residuals of ACF and PACF (R Output)

autocorrelations fall within acceptable threshold limits, suggesting randomness in the residuals.

Furthermore, the Ljung-Box Q-test (Ljung & Box, 1978) is a diagnostic tool utilized to assess the lack of fit of a time series model. In this case, the test statistic yielded a value of Q = 20.585 with 17 degrees of freedom, and a p-value of 0.2454. With model degrees of freedom set to 7 and a total of 24 lags used, the p-value indicates that the residuals exhibit randomness, affirming that the model adequately fits the time series. Taken together, the results from the Ljung-Box Q-test suggest that the SARIMA $(2,1,2)(0,0,2)_{12}$ with drift model appropriately captures the dynamics of the time series.

Using the SARIMA Model in Forecasting Process

In Figure 13, forecasts generated from the SARIMA $(2,1,2)(0,0,1)_{12}$ with drift model depict a slight downward trend projected into the near future. Meanwhile, Figure 14 provides a visual representation of the monthly global price of bananas dataset, with the black line indicating the original data and the red line representing the dataset incorporating forecasted values. The forecasting process conducted using the SARIMA $(2,1,2)(0,0,2)_{12}$ with drift model suggests a favorable fit for forecasting, showcasing a consistent declining trend in the short-term.

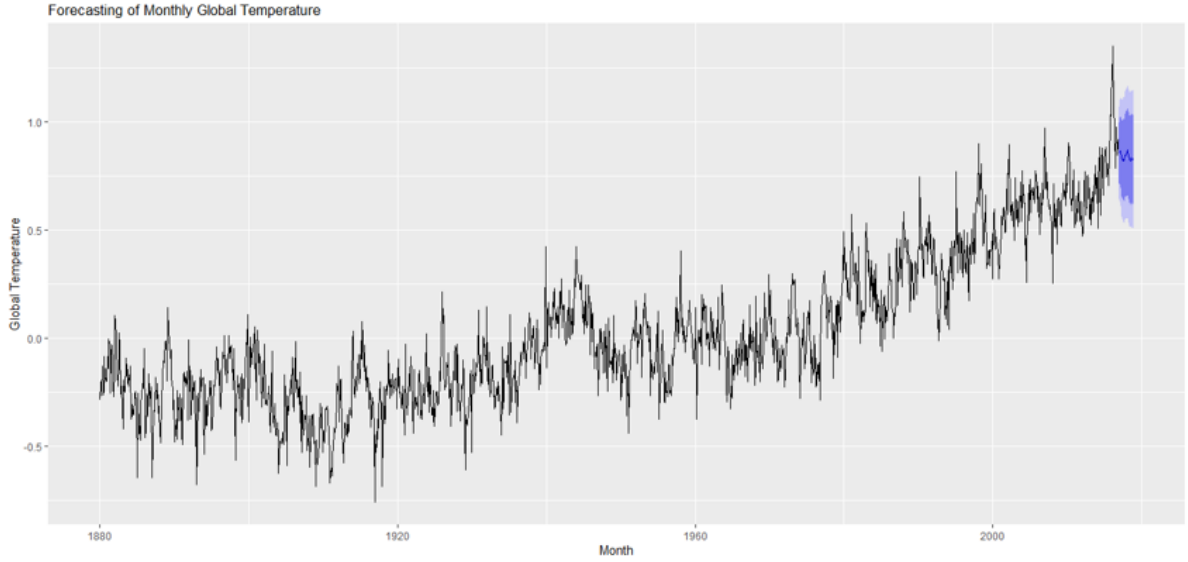

Figure 13: Observed and Forecasted Monthly Global Mean Temperature Index (R Output)

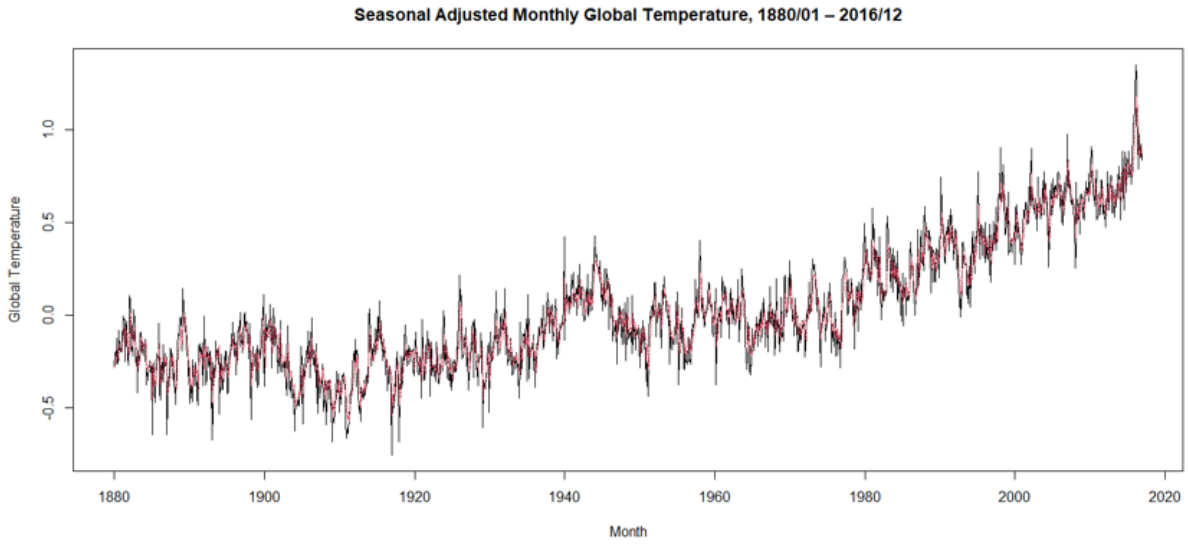

Figure 14: Observed and Forecasted Monthly Global Mean Temperature Index (R Output)

Nonlinear Autoregressive Neural Network (NARNN) Model

In MATLAB, the NARNN model is implemented for time series prediction by leveraging past values of a univariate time series. The model can be represented as:

$$y(t) = \Phi(y(t-1), y(t-2), \cdots, y(t-d)) + e(t) \quad (13)$$

where y(t) denotes the time series value at time t, d represents the time delay, and e(t) stands for the error in approximating the time series at time t. The function $\Phi(\cdot)$ denotes an unknown nonlinear function, and the

neural network training aims to approximate this function by optimizing network weights and neuron biases. The objective is to minimize the sum of squared differences between observed and predicted values, expressed as Mean Squared Error (MSE).

In this study, the NARNN model is applied to model and forecast the seasonal adjusted monthly global mean temperature index from January 1880 to December 2016. The neural network architecture includes logistic sigmoid and linear transfer functions at the hidden and output layers, respectively. The determination of the number of hidden neurons and time delays involves experimental setup following data preprocessing and analysis. The training is conducted using the LM, BR, and SCG training algorithms available in MATLAB (2023a) Neural Network Toolbox. The dataset comprises 1638 timesteps of one element, divided into 70% for training, 15% for validation, and 15% for testing.

The optimal architecture for the NARNN model entails identifying suitable time delays, the number of hidden neurons, and selecting an efficient training algorithm. This process involves a trial-and-error approach to determine the optimal parameters. Subsequently, the LM, BR, and SCG algorithms are utilized for training, and their performance is evaluated under the optimal neural network structure. The prediction accuracy is assessed based on MSE. Error analysis reveals that the NARNN model with 11 neurons in the hidden layer and 6 time delays exhibits the best performance (MSE = 0.0110) using the LM algorithm (refer to Table 3 for detailed results).

Table 3: NARNN Model Selection Using the LM, BR, SCG Algorithms

Layer Size	Time Delay	LM		BR		SCG	
		MSE	R	MSE	R	MSE	R
9	2	0.0130	0.9458	0.0127	0.9444	0.0138	0.9396
	3	0.0119	0.9491	0.0125	0.9460	0.0141	0.9406
	4	0.0124	0.9477	0.0123	0.9475	0.0134	0.9410
	5	0.0122	0.9465	0.0118	0.9481	0.0128	0.9447
	6	0.0114	0.9505	0.0120	0.9472	0.0127	0.9448
10	2	0.0120	0.9474	0.0127	0.9450	0.0138	0.9409
	3	0.0125	0.9458	0.0127	0.9456	0.0132	0.9419
	4	0.0126	0.9465	0.0120	0.9475	0.0141	0.9379
	5	0.0118	0.9503	0.0120	0.9482	0.0129	0.9448
	6	0.0117	0.9514	0.0124	0.9449	0.0133	0.9421
11	2	0.0125	0.9452	0.0127	0.9452	0.0134	0.9431
	3	0.0115	0.9471	0.0125	0.9456	0.0157	0.9316
	4	0.0120	0.9474	0.0122	0.9463	0.0127	0.9458
	5	0.0123	0.9474	0.0122	0.9467	0.0158	0.9305
	6	0.0110	0.9471	0.0126	0.9455	0.0130	0.9441

Source: own work

The open-loop architecture of the NARNN model, as depicted in Figure 15, provides a visual representation of the network structure generated within MATLAB's Neural Network Toolbox illustrated in the diagram:

- Input Series (y(t)): This block represents the input series, which consists of the monthly global mean temperature index observations. The "1" at the bottom signifies that it's a univariate time series.
- Hidden Layer with Delays: The second block represents the hidden layer of the network. Each neuron in this layer receives inputs from the previous time steps, indicated by the "1:3" notation, which means that the neuron considers the input values from the last three time steps. The "w" and "b" inner boxes represent the weights and biases, respectively, for a single neuron in the hidden layer. The larger box after the summation sign denotes the activation function, which is typically a sigmoid function in this layer. The number "11" at the bottom signifies the number of hidden neurons.
- Output Layer: This block represents the output layer of the network. The outputs from the hidden layer are combined using weights and biases represented by the "w" and "b" inner boxes. The transfer function at the output layer is linear. Only one output neuron is depicted, as denoted below the "Output Layer" block.
- Predicted Output (y(t)): The last block represents the predicted output of the network. It's important to note that this output, denoted as y(t), differs from the input series y(t). Since the output of the network is a prediction of the input time series, MATLAB uses the same variable name for both input and predicted output to signify their relationship.

NARNN Training Output

The LM algorithm is known for its efficient use of time during training, although it may require more memory

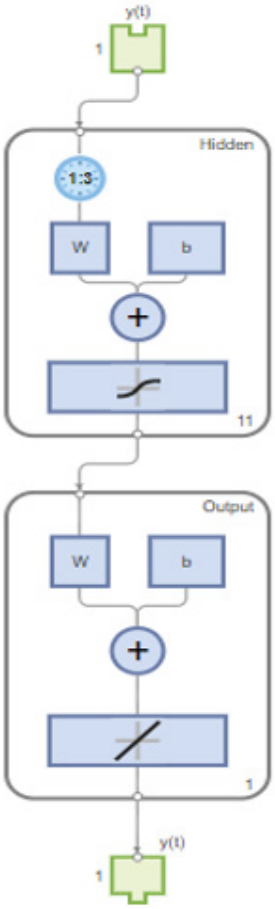

Figure 15: Open Loop Architecture (MATLAB Output)

compared to other algorithms. In the context of neural network training, the algorithm automatically stops when the generalization stops improving, typically indicated by an increase in the mean square error of the validation samples. Figure 16 provides insights into the training progress using the LM algorithm:

• Performance: Refers to the performance of the network during training, often measured by MSE or another relevant metric. In this case, the performance is reported as 0.0109.

• Gradient: Indicates the gradient of the error function, which provides information about the direction and rate of change of the error with respect to the network weights.

• Mu: Represents the damping parameter used in the LM algorithm. This parameter helps control the step size during gradient descent, ensuring stable convergence.

• Epoch 22: Refers to the number of iterations during training. Each epoch represents one complete pass of the training dataset through the network. In this case, training stopped after 22 epochs. In terms of processing time, the LM algorithm took 00:00:00, indicating that the training process completed very quickly. This efficiency makes the LM algorithm a preferred choice for many applications where training time is a critical factor.

Training Progress

Unit	Initial Value	Stopped Value	Target Value
Epoch	0	22	1000
Elapsed Time	-	00:00:00	-
Performance	0.169	0.0109	0
Gradient	0.72	0.00141	1e-07
Mu	0.001	1e-05	1e+10
Validation Checks	0	6	6

Figure 16: NARNN Training Output (MATLAB Output)

NARNN Best Performance

Figure 17 presents the performance plot for forecasting the monthly global mean temperature index using the NARNN model:

• MSE: This metric measures the average of the squares of the errors between the predicted and actual values. Lower MSE values indicate better model performance.

• Epochs: Refers to the number of iterations during training. Each epoch represents one complete pass of the training dataset through the network.

• Training Phase: The blue line represents the MSE for the training dataset over the course of training epochs. It shows how the training error decreases as the model learns from the data.

• Validation Phase: The red line indicates the MSE for the validation dataset. It helps monitor how well the model generalizes to new, unseen data during training.

• Testing Phase: The green line represents the MSE for the testing dataset. It evaluates the model's performance on a completely independent dataset.

• Circled Epoch: Training stopped at this epoch, where the validation error started to increase. This indicates that further training would likely lead to overfitting, where the model performs well on the training data but poorly on new data.

• Best Performance: The lowest MSE for the validation phase occurred at epoch 16, with a value of 0.013832. This suggests that the model achieved its best performance in terms of generalization at this point in training.

Overall, the performance plot helps assess the model's ability to generalize to new data and provides insights into potential overfitting. In this case, the similar characteristics of the validation and testing errors indicate that the model has good generalization performance without significant overfitting.

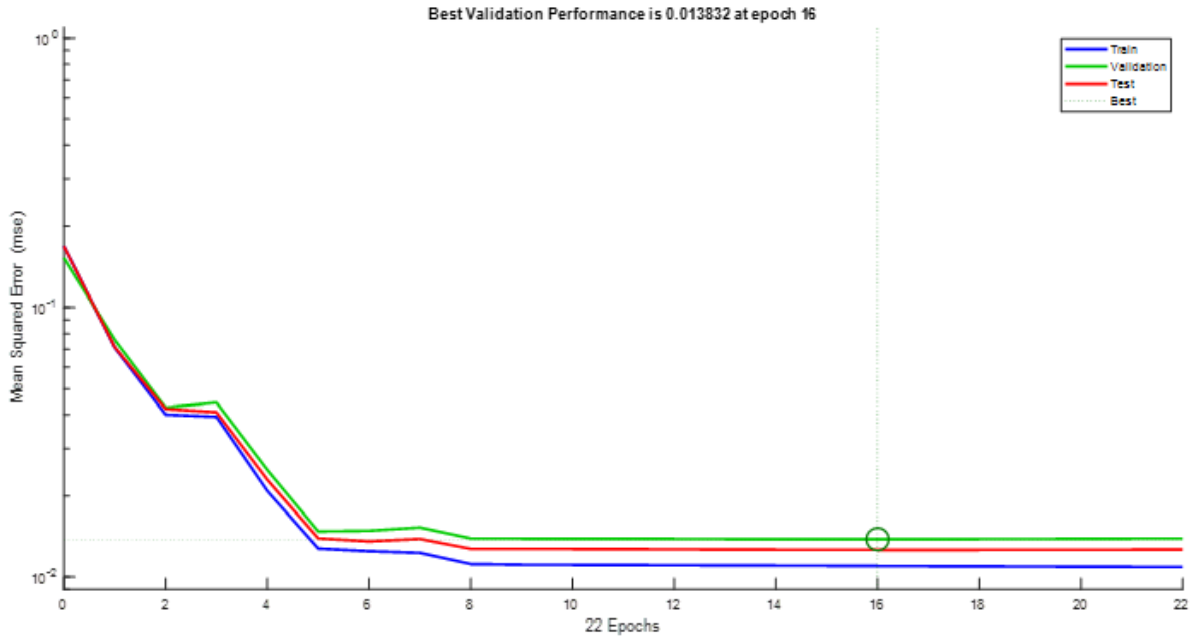

Figure 17: Performance Plot of the NARNN Model (MATLAB Output)

NAR Neural Network Error Histogram

Figure 18 displays error histograms for the training, validation, and testing datasets:

• Error Histograms: Each histogram shows the distribution of errors between the predicted and actual values for the respective dataset.

• Blue Bars (Training Data): Represent the errors for the training dataset. The height of each bar indicates the frequency of errors within a specific range.

• Green Bars (Validation Data): Represent the errors for the validation dataset. Similarly, the height of each bar shows the frequency of errors within a specific range.

• Red Bars (Testing Data): Represent the errors for the testing dataset. Again, the height of each bar denotes the frequency of errors within a specific range.

• Outliers: Points outside of the expected range may indicate outliers in the data. These outliers could be valid data points that are significantly different from the rest of the dataset.

• Interpretation: If outliers are present, it suggests that the model may not generalize well to those particular data points. Retraining the model with additional similar data points or adjusting the model architecture may help improve its performance on outliers.

Overall, the error histograms provide insight into the distribution of errors across different datasets and can help identify potential outliers that may require further investigation or model refinement.

NARNN Time-Series Response

Figure 19 presents dynamic network time-series response plots for the NARNN model:

• Response Plots: These plots show the predicted outputs of the NARNN model over time. The x-axis represents

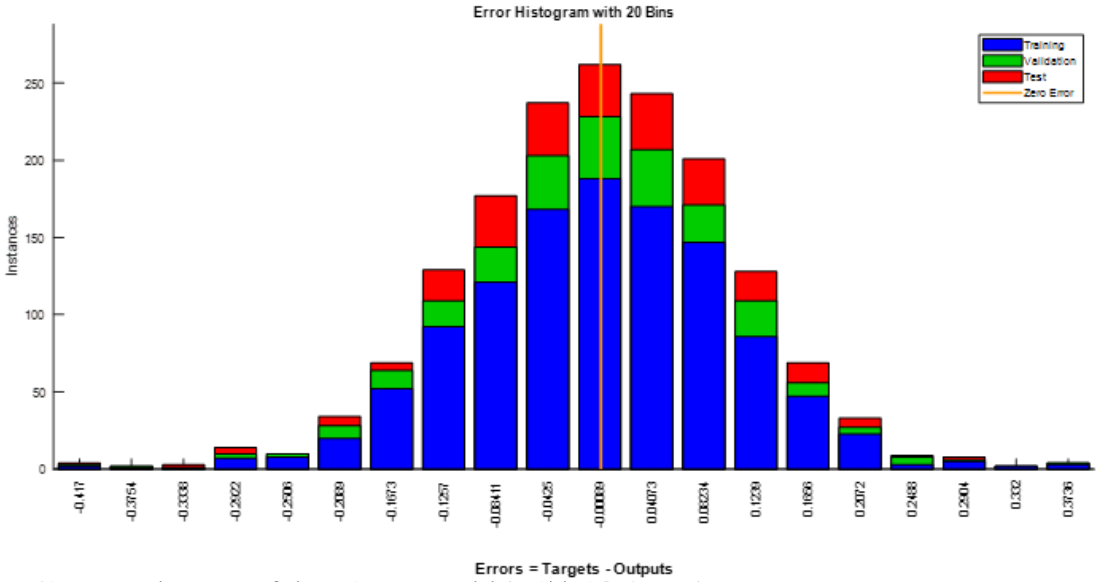

Figure 18: Error Histogram of the NARNN Model (MATLAB Output)

time, while the y-axis represents the predicted values.

• Even Distribution: The outputs are evenly distributed on both sides of the response curve, indicating that the model's predictions are consistent across different time points.

• Error Versus Time: The errors versus time are small for all three phases: training, validation, and testing. This suggests that the model's predictions closely match the actual values throughout the simulation period.

• Efficient Prediction: The results indicate that the NARNN model efficiently predicts the time series over the simulation period. The small errors and even distribution of outputs demonstrate the model's effectiveness in capturing the underlying patterns and dynamics of the time series data.

Overall, the dynamic network time-series response plots provide visual confirmation of the NARNN model's ability to accurately predict the time series data across different phases of training, validation, and testing.

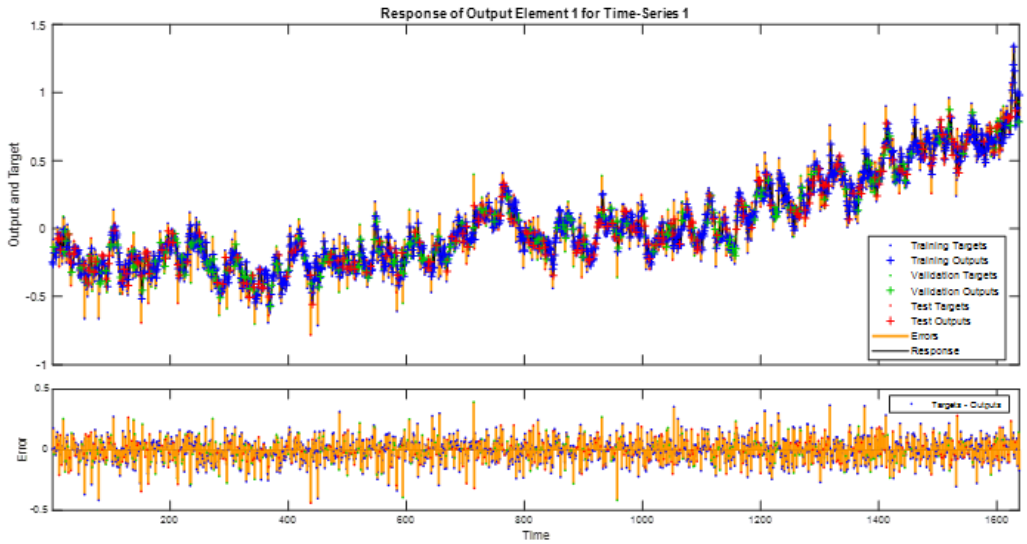

Figure 19: Network Time-Series Response of the NARNN Model (MATLAB Output)

NARNN Error Autocorrelation

Figure 20 displays the error autocorrelation function for the NARNN model:

• Prediction Error Correlation: The autocorrelation function describes how the prediction errors are correlated over time. In an ideal scenario, there should only be one non-zero value at zero lag, indicating that the prediction errors are completely uncorrelated with each other (white noise).

• Adequacy of the Model: The correlations for the NARNN model, except for the one at zero lag, mostly fall within the 95% confidence limits around zero. This suggests that the model is adequate for predicting the time series data.

• Room for Improvement: However, there are some exceptions where the correlations deviate from zero, indicating significant correlation in the prediction errors. This suggests that there is room for improvement in the

model. One approach could be to retrain the network or increase the number of neurons in the hidden layer to potentially achieve more accurate results.

• Retraining the Network: Retraining the network involves changing the initial weights and biases of the network, which may lead to an improved model. This process can help address any issues identified in the error autocorrelation function and improve the overall performance of the NARNN model.

Overall, the error autocorrelation function provides valuable insights into the correlation structure of the prediction errors and highlights areas where the NARNN model can be enhanced to achieve better accuracy in predicting the time series data.

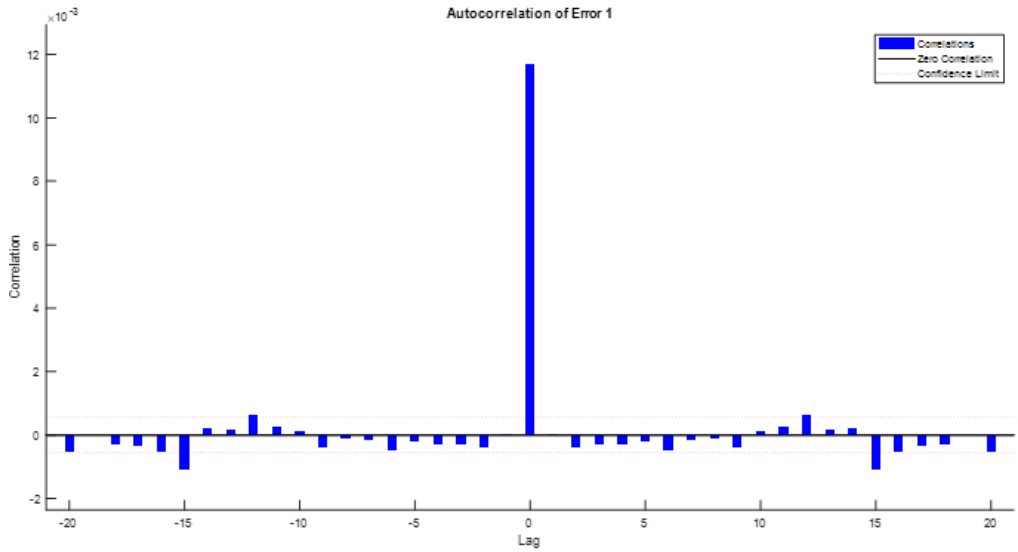

Figure 20: Error Autocorrelation of the NARNN Model (MATLAB Output)

SARIMA-NARNN Hybrid Model

The hybrid model, which combines the SARIMA and NARNN models, plays a crucial role in improving the forecasting accuracy of time series data. The key points regarding the hybrid model in MATLAB as follows:

1. Model Expression: The hybrid model predicts the future residuals of a time series using past residuals. The equation expresses this relationship, where e(t) represents the residual at time t, d denotes the time delay, and Φ represents an unknown function learned by the model. The goal is to predict the future residual term ε(t) based on past residual values.

2. Model Architecture Optimization: Similar to the NARNN model, determining the optimal architecture for the hybrid model involves selecting the appropriate number of time delays and hidden neurons. In this case, the model achieved the best performance with 10 neurons in the hidden layer and 6 time delays. This configuration was determined through a trial and error procedure.

3. Data Division for Training: The time series data was divided into training, validation, and testing sets, with 70% allocated for training, and 15% each for validation and testing. This division ensures that the model is trained on a sufficient amount of data while also validating its performance on unseen data.

4. Training Algorithm and Stopping Criteria: The LM algorithm was used for training the hybrid model. Training stopped when the validation error increased for six iterations, indicating a potential decrease in model performance. At the stopping point, the algorithm achieved a performance of 0.0102 with a gradient of 0.00288.

5. Performance Evaluation: The performance of the hybrid model was evaluated based on MSE, with lower MSE values indicating better performance. In this case, the hybrid model achieved an MSE of 0.0105, indicating its effectiveness in predicting future residuals.

Overall, the hybrid model offers a powerful approach to time series forecasting by combining the strengths of both the SARIMA and NARNN models. By leveraging past residuals from the SARIMA model, the hybrid model enhances the forecasting accuracy and provides valuable insights into future trends in the time series data.

The results for the hybrid model indicate its effectiveness in predicting time series data. A summary of the findings based on the provided figures and descriptions as follows:

1. Validation Performance: The hybrid model achieved its best performance for the validation phase with an MSE of 0.014383 at epoch 7. This indicates that the model performed well in capturing the underlying patterns in the data during the validation phase.

2. Error Histograms: Analysis of the error histograms showed that there were a few training points outside of the expected range. However, the validation and testing errors exhibited similar characteristics, indicating that the model's performance was consistent across different data subsets.

3. Dynamic Network Time-Series Response Plots: The dynamic network time-series response plots displayed in Figure 23 demonstrated that the outputs of the hybrid model were evenly distributed around the response curve. Additionally, the errors versus time were small across all phases of training, validation, and testing, indicating the model's ability to accurately predict the time series data.

Table 4: The Hybrid Model Selection Using the LM, BR, SCG Algorithms

Layer Size	Time Delay	LM	BR	SCG
		MSE	MSE	MSE
9	2	0.0111	0.0117	0.0124
	3	0.0116	0.0114	0.0114
	4	0.0114	0.0116	0.0119
	5	0.0120	0.0115	0.0124
	6	0.0110	0.0113	0.0113
10	2	0.0109	0.0113	0.0116
	3	0.0117	0.0116	0.0115
	4	0.0106	0.0112	0.0119
	5	0.0117	0.0114	0.0116
	6	0.0105	0.0118	0.0113
11	2	0.0111	0.0118	0.0119
	3	0.0113	0.0117	0.0117
	4	0.0116	0.0113	0.0114
	5	0.0114	0.0118	0.0122
	6	0.0109	0.0114	0.0117

Source: own work

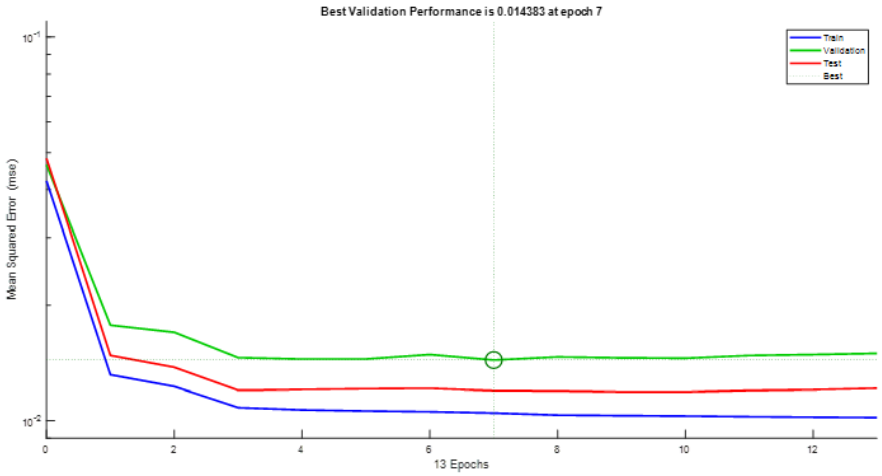

Figure 21: Performance Plot of the Hybrid Model (MATLAB Output)

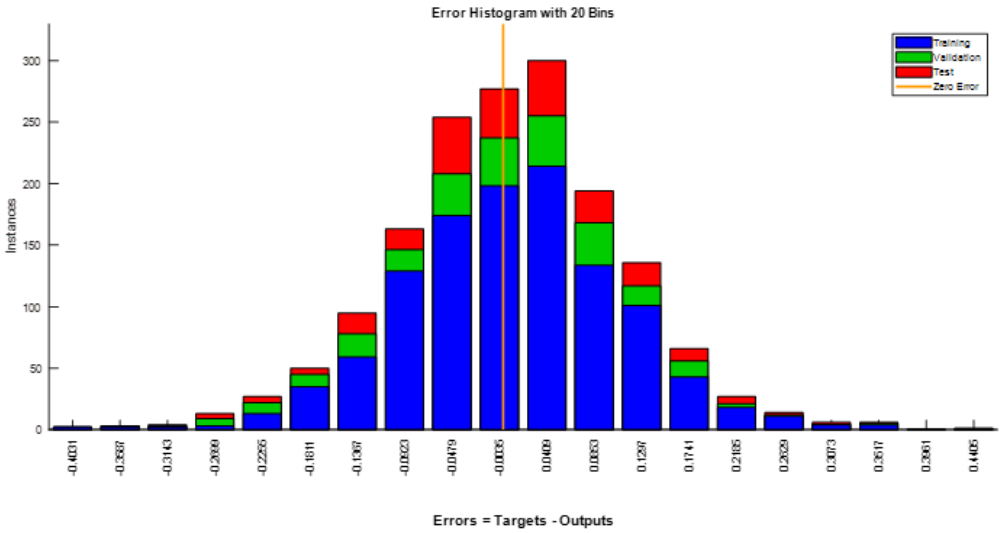

Figure 22: Error Histogram of the Hybrid Model (MATLAB Output)

4. Error Autocorrelation Function: The error autocorrelation function plot (Figure 24) showed that, except for the correlation at zero lag, all correlations fell approximately within the 95% confidence limits around zero. This suggests that the model adequately captured the underlying patterns in the data, as significant correlations in the prediction errors were not observed.

Overall, these results indicate that the hybrid model, which combines the SARIMA and NARNN models, effectively predicts time series data and provides reliable forecasts over the simulation period.

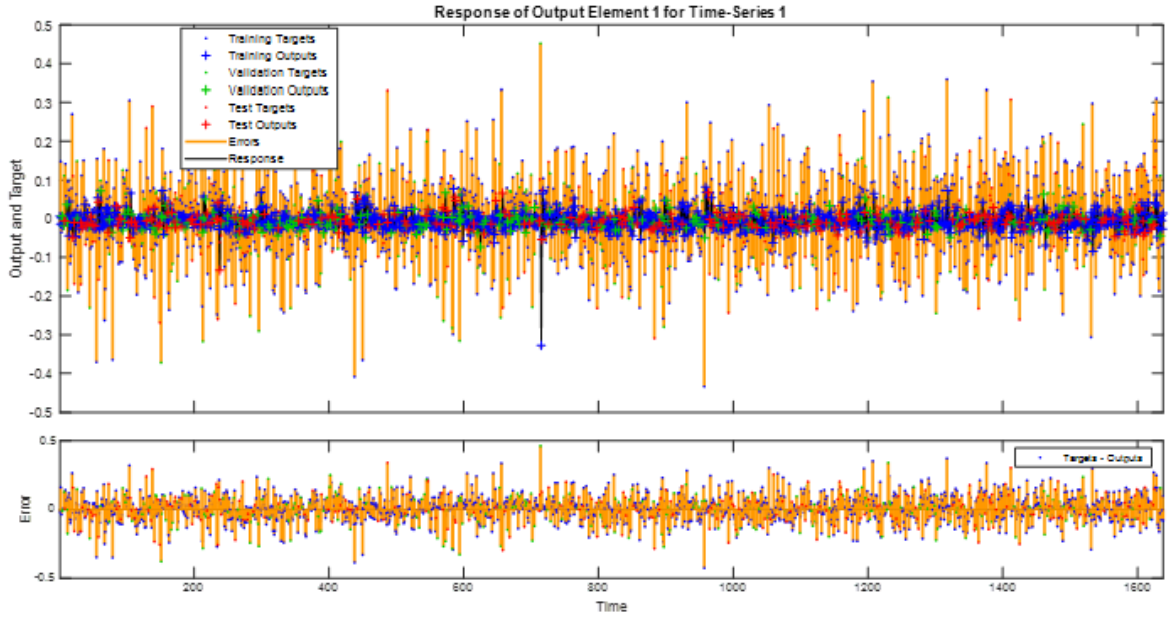

Figure 23: Network Time-Series Response of the Hybrid Model (MATLAB Output)

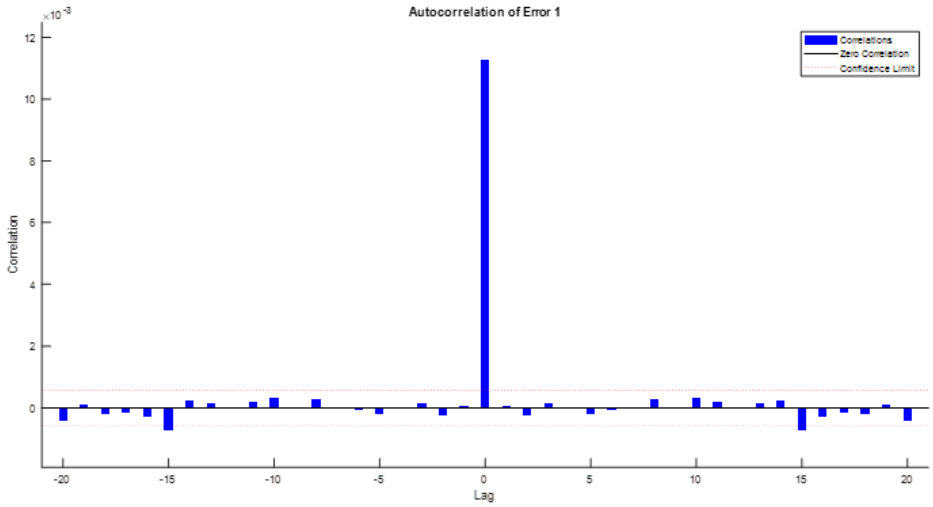

Figure 24: Error Autocorrelation of the Hybrid Model (MATLAB Output)

Discussion

The integration of SARIMA, NARNN, and SARIMA-NARNN hybrid models presents a significant advancement in the field of global mean temperature modeling and forecasting. Each of these modeling approaches offers unique strengths in capturing different aspects of the complex temperature dynamics inherent in global mean temperature time series data. SARIMA models excel at capturing linear trends and seasonal patterns, making them well-suited for modeling the long-term climate variability observed in global mean temperature time series data. On the other hand, NARNN models leverage the nonlinear modeling capabilities of neural networks to capture intricate patterns and temporal dependencies within the data, enhancing forecasting accuracy.

The SARIMA-NARNN hybrid model combines the strengths of both approaches, offering a comprehensive framework for global mean temperature modeling and forecasting that captures both linear and nonlinear dynamics. By integrating SARIMA's ability to capture long-term trends and seasonality with NARNN's capacity to capture complex temporal patterns, the SARIMA-NARNN hybrid model provides a more robust and accurate representation of global mean temperature

time series data. This hybridization approach addresses the limitations of individual modeling techniques and leverages their complementary strengths to improve forecasting performance.

In conclusion, the modeling and forecasting of monthly global mean temperature time series data using SARIMA, NARNN, and the SARIMA-NARNN hybrid models represent a significant advancement in climate science research. These modeling approaches offer valuable insights into the complex dynamics of global temperature variability and provide accurate forecasts of future temperature trends. By integrating traditional statistical methods with advanced machine learning techniques, the SARIMA-NARNN hybrid model presents a promising framework for global mean temperature forecasting that captures both linear and nonlinear dynamics, thereby improving forecasting accuracy and reliability.

Moreover, the incorporation of exogenous variables, such as greenhouse gas concentrations and solar radiation levels, into the modeling process further enhances the predictive capabilities of the SARIMA-NARNN hybrid model. By accounting for external factors that influence global mean temperature variability, the hybrid model can better capture the complex interactions between natural climate variability and anthropogenic influences, thereby improving the accuracy of future temperature projections. This aspect is particularly crucial given the multifaceted nature of climate change and the need for comprehensive modeling frameworks that consider various contributing factors.

CONCLUSIONS

Modeling and forecasting global mean temperature play a crucial role in climate science and policy-making to develop informed strategies for climate adaptation and mitigation efforts. Global mean temperature rise integrates knowledge from diverse disciplines, employs rigorous methodologies, and emphasizes the need for evidence-based policymaking and societal action. It underscores the urgency of addressing climate change as a pressing global challenge with far-reaching implications for human well-being and planetary health.

By integrating traditional time series analysis with cutting-edge machine learning algorithms, this study aims to provide a comprehensive understanding of the complexities inherent in global mean temperature time series data and to offer robust forecasting models for anticipating future temperature trends. In this study the best choice time series model was the SARIMA $(2,1,2)(0,0,2)_{12}$ with drift model as its lowest AIC values among other models. It was noticed that this SARIMA $(2,1,2)(0,0,2)_{12}$ with drift model gave evidence about future monthly global mean temperature index would follow the trends increase over time. In this study, the NARNN model with 11 neurons in the hidden layer and 6 time delays was evaluated as the optimal neural network structure using the LM algorithm, because it works without computing the exact Hessian matrix, increasing the training speed and has stable convergence (Gavin, 2020).

This study not only wanted to find the best fit model for the time series, monthly global mean temperature index from January 1880 to December 2016, but also tried to evaluate the accuracy of the SARIMA, NARNN, and SARIMA-NARNN hybrid models used in the forecast of monthly global mean temperature index. The comparative results revealed that the hybrid model with 10 neurons in the hidden layer and 6 time delays (MSE = 0.0105) yielded higher accuracy than the NARNN model with 11 neurons in the hidden layer and 6 time delays (MSE = 0.0110), and the ARIMA $(2,1,2)(0,0,2)_{12}$ with drift model (MSE = 0.0115) in this study.

Moving forward, further research is needed to explore additional factors that may influence global mean temperature variability and to refine modeling techniques for enhanced forecasting performance. Additionally, the development of ensemble modeling approaches that combine multiple forecasting models could further improve the robustness of global mean temperature predictions and provide more reliable estimates of future temperature trends. Ultimately, the insights gained from modeling and forecasting global mean temperature time series data are invaluable for informing climate policy decisions, guiding mitigation efforts, and adapting to the challenges of a changing climate.

In summary, future global mean temperature rise depends on a complex interplay of greenhouse gas emissions, climate feedbacks, and socio-economic factors. While uncertainty exists in temperature projections, urgent and concerted action is needed to mitigate climate change, adapt to its impacts, and safeguard the planet for future generations.

Acknowledgements

This work is supported by the USDA National Institute of Food and Agriculture, Evans-Allen project [Accession # 7004162].

REFERENCES

Benrhmach, G., Namir, K., Namir, A., & Bouyaghroumni, J. (2020). Nonlinear autoregressive neural network and extended Kalman filters do prediction of financial time series. *Journal of Applied Mathematics, 2020, 5057801*, 1-6. https://doi.org/10.1155/2020/5057801

Box, G. E. P., & Jenkins, G. M. (1970). *Time Series Analysis: Forecasting and Control*. San Francisco: Holden-Day.

Box, G. E. P., Jenkins, G. M., Reinsel, G. C., & Ljung, G. M. (2015). *Time Series Analysis: Forecasting and Control* (5th ed.). Hoboken, N.J.: John Wiley and Sons Inc.

Gavin, H. P. (2020). The Levenberg-Marquardt algorithm for nonlinear least squares curve-fitting problems. *Department of Civil and Environmental Engineering Duke University*, 19. http://people.duke.edu/~hpgavin/ce281/lm.pdf

Hagan, M. T., Demuth, H. B., Beale, M. H., & De Jesus, O. (2014). *Neural network design* (2nd ed.). pp. 12-19~12-

22 and 13-12~13-14, Publisher: Martin Hagan.

Haskett, J. D. (2022). Intergovernmental Panel on Climate Change: Sixth Assessment Report. *Congressional Research Service Report, R47082*, 17. https://crsreports.congress.gov

Kaufman, D., McKay, N., Routson, C., Erb, M., Dätwyler, C., Sommer, P. S., Heiri, O., & Davis, B. (2020). Holocene global mean surface temperature, a multi-method reconstruction approach. *Scientific Data, 7*(201), 13. https://doi.org/10.1038/s41597-020-0530-7

Levenberg, K. (1944). A Method for the Solution of Certain Non-Linear Problems in Least Squares. *Quarterly of Applied Mathematics, 2*(2), 164–168. https://doi.org/10.1090/qam/10666

Ljung, G. M., & Box, G. E. P. (1978). On a measure of lack of fit in time series models. *Biometrika, 65*(2), 297-303. https://doi.org/10.1093/biomet/65.2.297

MacKay, D. J. C. (1992). Bayesian Interpolation. *Neural Computation, 4*(3), 415–447. https://doi.org/10.1162/neco.1992.4.3.415

Marquardt, D. W. (1963). An algorithm for least-squares estimation of nonlinear parameters. *Journal of the Society for Industrial and Applied Mathematics, 11*(2), 431-441. https://www.jstor.org/stable/2098941

Mohorji, A. M., Şen, Z. & Almazroui, M. (2017). Trend Analyses Revision and Global Monthly Temperature Innovative Multi-Duration Analysis. *Earth Systems and Environment, 1*(9), 13 pages. https://doi.org/10.1007/s41748-017-0014-x

Møller, M. F. (1993). A scaled conjugate gradient algorithm for fast supervised learning. *Neural Networks, 6*(4), 525-533. https://doi.org/10.1016/S0893-6080(05)80056-5

Montgomery, D. C., Jennings, C. L., & Kulahci, M. (2008). *Introduction to Time Series Analysis and Forecasting.* Hoboken, N.J.: John Wiley & Sons. Inc.

Mudelsee, M. (2019). Trend analysis of climate time series: A review of methods. *Earth-Science Reviews, 190*, 310-322. https://doi.org/10.1016/j.earscirev.2018.12.005

PAGES 2k Consortium. (2019). Consistent multidecadal variability in global temperature reconstructions and simulations over the Common Era. *Nature Geoscience, 12*(8), 643–649. https://doi.org/10.1038/s41561-019-0400-0

Romilly, P. (2005). Time series modeling of global mean temperature for managerial decision-making. *Journal of Environmental Management, 76*(1), 61-70. https://doi.org/10.1016/j.jenvman.2005.01.008

Sariev, E., & Germano, G. (2020). Bayesian regularized artificial neural networks for the estimation of the probability of default. *Quantitative Finance, 20*(2), 311–328. https://doi.org/10.1080/14697688.2019.1633014

Wang, L., Wang, L., & Wang, J. (2023). A century-long analysis of global warming and earth temperature using a random walk with drift approach. *Decision Analytics Journal, 7*, 100237, 10. https://doi.org/10.1016/j.dajour.2023.100237

Ye, L. M., Yang, G. X., Van Ranst, E., & Tang, H. J. (2013). Time-series modeling and prediction of global monthly absolute temperature for environmental decision making. *Advances in Atmospheric Sciences, 30*(2), 382-396. https://doi.org/10.1007/s00376-012-1252-3

Yue, Z., Songzheng, Z., & Tianshi, L. (2011). Regularization BP neural network model for predicting oil-gas drilling cost. In *International Conference on Business Management and Electronic Information, Guangzhou, China, 2*, 483–487. https://doi.org/10.1109/ICBMEI.2011.5917952

Zhang, G. P. (2003). Time series forecasting using a hybrid ARIMA and neural network model. *Neurocomputing, 50*, 159-175. https://doi.org/10.1016/S0925-2312(01)00702-0