Time Series Analysis of Global Average Absolute Sea Level Change Using Nonlinear Autoregressive Neural Network Model

Yeong Nain Chi[1]
[1]University of Maryland Eastern Shore
[1]ychi@umes.edu

Corresponding author email: ychi@umes.edu

***Abstract*- This study tried to pursue analysis of time series data using long-term records of global average absolute sea level change from 1880 to 2014 extracted from the U.S. Environmental Protection Agency using data from Commonwealth Scientific and Industrial Research Organization. Using the LM algorithm, the results revealed that the nonlinear autoregressive neural network model with 7 neurons in the hidden layer and 7 time delays provided the best performance at its smaller MSE value. The findings in this study may be able to bridge an important gap in time series forecasting by combining the best statistical and machine learning methods. In order to sustain these observations, research programs utilizing the resulting data should be able to significantly improve our understanding and narrow projections of future sea level rise and variability.**

***Keywords*-Global Average Absolute Sea Level Change, Time Series Analysis, Nonlinear Autoregressive Neural Network Model, Levenberg-Marquardt Algorithm, Bayesian Regularization Algorithm, Scaled Conjugate Gradient Algorithm**.

1. Introduction

Climate change is the long-term change that occurs in the average weather patterns of the earth. Sea level rise is caused primarily by two factors related to climate change: the added water from melting of land ice (ice sheets and glaciers) to the world's oceans and the thermal expansion of seawater temperature rise. The potential impacts of sea level rise include, but not limited to, increasing coastal flooding and erosion, damages on agricultural land cover and crops, damages on coastal/urban settlements and infrastructures, damages on coastal flora and fauna ecosystems, increasing environmental sanitation problem, and increasing public health problem.

The Intergovernmental Panel on Climate Change (IPCC) [14] estimated that the sea level has risen by 26–55 cm (10–22 inches) with a 67% confidence interval. In its Fourth National Climate Assessment Report [24] the U.S. Global Change Research Program (USGCRP) estimated that sea level has risen by about 7–8 inches (about 16–21 cm) since 1900, with about 3 of those inches (about 7 cm) occurring since 1993. National Oceanic and Atmospheric Administration (NOAA) 2019 Global Climate Annual Report summarized that the global annual temperature has increased at an average rate of 0.07°C (0.13°F) per decade since 1880 (https://www.ncdc.noaa.gov/sotc/global/201913).

There had many studies pointed out that sea level has risen at an increasing rate [7] [5] [6] [13] [15] [10. Thus, understanding past sea level is important for the analysis of current and future sea level changes. Modeling sea level change and understanding its causes has considerably improved in the recent years, essentially because new in situ and remote sensing observations have become available [8] [25] [4] [22]. Despite the importance of sea level rise and its consequences, there is a lack of studies in the technical literature available on prediction schemes.

Time series forecasting is the use of a model to predict future values based on previously observed values, that is one of the most applied data science techniques in many disciplines. Neural networks have become one of the most popular trends in machine learning for time series modeling and forecasting. Neural networks can be stated as nonlinear nonparametric statistical method [27]. Given its advantages such as no assumptions, alternative solutions, and goal-driven characteristics, neural networks can be used as an alternative to traditional time series models to solve complex forecasting problems. Empirically, neural network model is chosen because it has good forecasting performance than the conventional ones [12], but it might not completely satisfy a systematic procedure in the construction of the model [23]. Therefore, how to formulate new models of neural networks is important task for future research.

Hence, the primary purpose of this study was to apply the nonlinear autoregressive neural network (NARNN) model to analyze the long-term records of global average absolute sea level change from 1880 to 2016. The reason why employed the NARNN model in this study because it converges much faster and performs better in comparison with the conventional neural works [1[. Specifically, the NARNN model were trained with the Levenberg–Marquardt (LM), Bayesian Regularization (BR),

and Scaled Conjugate Gradient (SCG) training algorithms in this study. This novel experiment could be a pilot study that applied NARNN model for evaluating global average absolute sea level change. The findings in this study may be able to bridge an important gap in time series forecasting by combining the best statistical and machine learning methods.

2. Materials

The data used for this study is available to the general public from the US Environmental Protection Agency (http://www3.epa.gov/climatechange/images/indicator_downloads/sea-level_fig-1.csv) using data from CSIRO (Commonwealth Scientific and Industrial Research Organization), 2015 (http://www.cmar.csiro.au/sealevel/GMSL_SG_2011_up.html). In order to analyze accurately, the data, global average absolute sea level change from 1880 to 2014 (Figure 1), has merged two adjust sea level information from CSIRO (1880 – 1992) and NOAA (1993 – 2014) for this study. Mean global average absolute sea level change was 3.6408 mm with a standard deviation of 2.4297 mm (Minimum: -0.4409 mm in 1882, Maximum: 8.6637 mm in 2014, and Median: 3.3740 mm in 1947).

The data contains "*cumulative changes in sea level for the world's oceans since 1880, based on a combination of long-term tide gauge measurements and recent satellite measurements. It shows average absolute sea level change, which refers to the height of the ocean surface, regardless of whether nearby land is rising or falling. Satellite data are based solely on measured sea level, while the long-term tide gauge data include a small correction factor because the size and shape of the oceans are changing slowly over time. (On average, the ocean floor has been gradually sinking since the last Ice Age peak, 20,000 years ago.)*" (Quoted from https://datahub.io/core/sea-level-rise#readme).

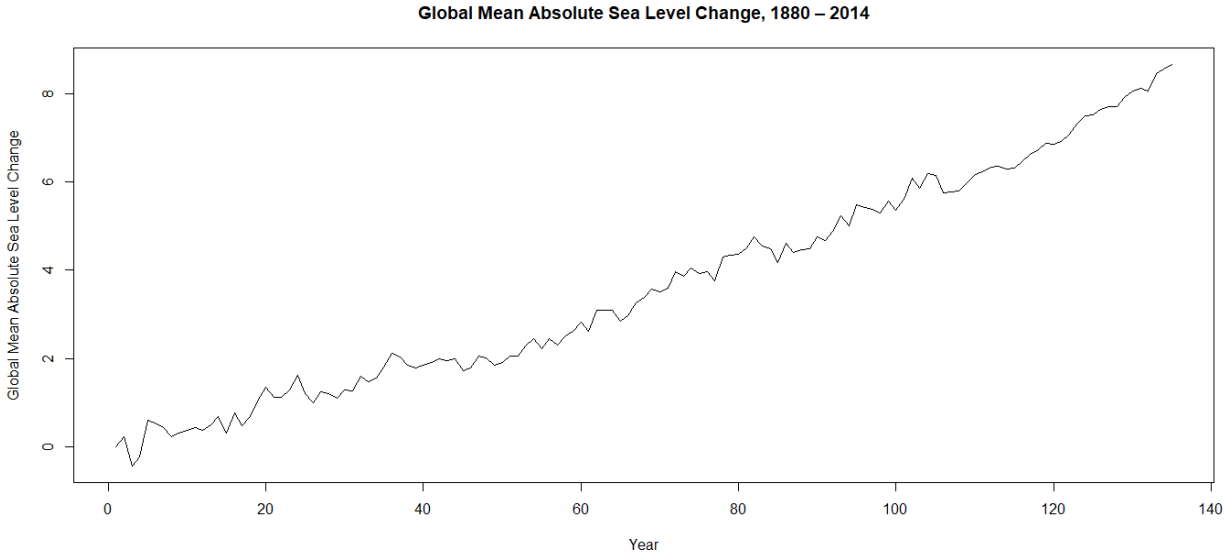

Figure 1. Time Series Plot of Global Average Absolute Sea Level Change, 1880 - 2014 (R Output)

3. Methods

3.1 Nonlinear Autoregressive Neural Network (NARNN) Model

The idea behind the autoregressive (AR) process is to explain the present value of the time series, y_t, by a function of p past values, (y_{t-1}, y_{t-2}, ⋯, y_{t-p}). Thus, the AR process of order p, AR(p), is defined by the equation:

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

where $\phi = (\phi_1, \phi_2, \cdots, \phi_p)$ is the vector of model coefficients for the autoregressive process, and e_t is white noise, $e_t \sim N(0, \sigma^2)$ [20].

The NARNN is a natural generalization of the classic linear AR(p) process. The NARNN of order p can be expressed as:

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

where $\Phi(\cdot)$ is an unknown function determined by the neural network structure and connection weights, w is a vector of all parameters (weights), and ε_t is the error term. Thus, it performs a nonlinear functional mapping from the past observations, (y_{t-1}, y_{t-2}, ⋯, y_{t-p}), to the future value, y_t, which is equivalent to a nonlinear autoregressive model [28]. With the time series data, lagged values of the time series can be used as inputs to a neural network, so-called this the NARNN model. Mathematically, the NARNN model [3] can be written by the equation of the form as:

$$y_t = a_0 + \sum_{j=1}^{k} w_j \Phi(b_{0j} + \sum_{i=1}^{d} w_{ij} y_{t-i}) + \varepsilon_t \quad (3)$$

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, $\Phi(y) = 1 / (1 + \exp(-y))$.

3.2 Training Algorithms

The most common learning rules for the NARNN model are the Levenberg-Marquardt, Bayesian Regularization, and Scaled Conjugate Gradient training algorithms. Training is the process of determining the optimal network weights and bias points of the *multilayer feedforward* neural network. This is done by defining the total error function between the network's output and the desired target and then minimizing it with respect to the weights.

3.2.1 Levenberg-Marquardt (LM) Algorithm

The LM algorithm, first published by Levenberg [16] and then rediscovered by Marquardt [18], is a commonly used iterative algorithm to solve non-linear minimization problems. These minimization problems arise especially in least squares curve fitting. This curve-fitting method is a combination of the gradient descent and the Gauss-Newton. It works without computing the exact Hessian matrix. Instead, it works with the gradient vector and the Jacobian matrix, thereby increasing the training speed while having stable convergence [9].

The LM algorithm is a variation of Newton's method that is very well suited to neural network training where the performance index is the mean squared error. When the performance function (also known as network error function) has the form of a sum of squares, then the Hessian matrix can be approximated and the gradient can be computer as:

$$H = J^T J \quad (4)$$

$$G = J^T e \quad (5)$$

where J is a Jacobian matrix, which contains first order derivatives of the network errors with respect to the weights and biases, e is a vector of network errors. The Jacobian matrix can be computed through a standard backpropagation technique that is much less complex than computing the Hessian matrix. The following is the relation for LM algorithm computation:

$$x_{k+1} = x_k - [J^T J + \mu I]^{-1} J^T e \quad (6)$$

where x_k = the current connection weight, x_{k+1} = the next connection weight, I = the identity matrix, and the scalar μ is combination coefficient. This algorithm has the very useful feature when μ is increased it approaches the steepest descent algorithm with small learning rate, while μ is decreased to zero the algorithm becomes Gauss-Newton [11].

3.2.2 Bayesian Regularization (BR) Algorithm

The BR algorithm was introduced by MacKay [17] automatically sets the best possible performance function to accomplish the excellent generalization on the basis of Bayesian inference approach. The BR algorithm is based on the probabilistic interpretation of network parameters. Bayesian optimization of regularization parameters depends upon the calculation of the Hessian matrix at the minimum point. Therefore, the BR algorithm includes a probability distribution of network weights and the network architecture can be identified as a probabilistic framework [21].

Like the LM algorithm, the BR algorithm is also used to optimize weights and bias and minimize squares of errors. It introduces network weights into the training objective function which is denoted as F(W) as follows:

$$F(W) = \alpha E_W + \beta E_D \quad (7)$$

where E_D is the sum of network errors, E_W is the sum of the squared network weights, α and β are the objective function parameters. The α and β parameters are determined using the Bayes' theorem. Moreover, the Gaussian distribution is employed to develop both network weight and training sets. These parameters are updated and repeated procedure until convergence achieved [26]. In order to find the optimal weight space, the objective function needs to be minimized, which is the equivalent of maximizing the posterior probability function given as follows:

$$P(x|D, \alpha, \beta, M) = P(D|x, \beta, M)\, P(x|\alpha, M) / P(D|\, \alpha, \beta, M) \quad (8)$$

where x is the vector containing all of the weights and biases in the network, D represents the training data set, and α and β are parameters associated with the density functions $P(D|x, \beta, M)$ and $P(x|\alpha, M)$, and M is the selected model - the architecture of the network we have chosen [11].

As a result of this process, optimum values for α and β for a given weight space are found. Later, algorithm moves into LM phase where Hessian matrix calculations take place and updates the weight space in order to minimize the objective function. Then, if the convergence is not met, algorithm estimates new values for α and β and the whole procedure repeats itself until convergence is reached [26].

3.2.3 Scaled Conjugate Gradient (SCG) Algorithm

The SCG algorithm, developed by Møller [19], is based on the Conjugate Gradient Method, but this algorithm does not perform a line search at each iteration. Unlike many other standard backward

propagation algorithms, the SCG algorithm is fully-automated, includes no critical user-specific parameters, and avoids a time-consuming line search. By integrating the model trust region way known from the LM algorithm with Conjugate Gradient, the SCG algorithm can be shown as [19]:

$$s_k = [E`(w_k + \sigma_k p_k) - E`(w_k) / \sigma_k] + \lambda_k p_k \quad (9)$$

where s is the Hessian matrix approximation, E is the total error function and $E`$ is the gradient of E, scaling factors λ_k and σ_k are acquainted with approximate the Hessian matrix and initialized by user at the starting of the algorithm such that $0 < \lambda_k < 10^{-6}$ and $0 < \sigma_k < 10^{-4}$. For the SCG algorithm, factor β_k calculation and direction of the new search as follows [19]:

$$\beta_k = (|g_{k+1}|^2 - g_{k+1}^T g_k) / g_k^T g_k \quad (10)$$

$$p_{k+1} = -g_{k+1} + \beta_k p_k \quad (11)$$

For the success of the algorithm, it is very important to update the design parameters independently at each iteration user. This is a major advantage compared to the line search-based algorithms.

4. Results

4.1 Nonlinear Autoregressive Neural Network (NARNN) Model

In MATLAB, the NARNN model applied to time series prediction using its past values of a univariate time series can be expressed as follows:

$$y(t) = \Phi(y(t-1), y(t-2), \cdots, y(t-d)) + e(t) \quad (12)$$

where y(t) is the time series value at time t, d is the time delay, and e(t) is the error of the approximation of the time series at time t. This equation describes how the NARNN model is used to predict the future value of a time series, y(t), using the past values of the time series, (y(t-1), y(t-2), ⋯, y(t-d)). The function $\Phi(\cdot)$ is an unknown nonlinear function, and the training of the neural network aims to approximate the function by means of the optimization of the network weights and neuron bias. This tends to minimize the sum of the squared differences between the observed (y_i) and predicted ($\hat{y}_i$) values (i.e., MSE = $(1/n) \sum_{i=1}^{n} (y_i - \hat{y}_i)^2$) [2].

In this study, the NARNN model was applied to model and predict the time series, global average absolute sea level change from 1880 to 2014. Furthermore, the logistic sigmoid and linear transfer functions at the hidden and output layers were used respectively. The number of hidden neurons and the number of delays was set experimentally after a data pre-processing and analysis stage. The extracted features were trained using the LM, BR, SCG training algorithms, respectively, for the target time series in the MATLAB (2022a) Neural Network Toolbox: 135 timesteps of one element, global average absolute sea level change from 1880 to 2014.

The training target timesteps are presented to the network during training, and the network is adjusted according to its error. The validation target timesteps are used to measure network generalization, and to halt training when generalization stops improving. The testing target timesteps have no effect on training and so provide an independent measure of network performance during and after training [2]. The division of the time series in this analytical work was 70% for the training, 15% for the validation, and 15% for the testing. Randomly, 135 data samples were divided into 90 data for the training, 19 data for the validation, and 19 data for the testing.

The development of the optimal architecture for the NARNN model requires determination of time delays, the number of hidden neurons, and an efficient training algorithm. The optimum number of time delays and hidden neurons were obtained through a trial and error procedure. Furthermore, the LM, BR, SCG algorithms were employed for training of the NARNN model, respectively, and their performance were evaluated under the optimal neural network structure. The prediction performance of the models was evaluated by its MSE. The error analysis showed that the NARNN model with 7 neurons in the hidden layer and 7 time delays provided the best performance (MSE = 0.0227) using the LM algorithm (Table 1).

Table 1. NARNN Model Selection Using the LM, BR, SCG Algorithms

Layer Size	Time Delay	LM		BR		SCG	
		MSE	R	MSE	R	MSE	R
6	5	0.0339	0.9976	0.0330	0.9970	0.0578	0.9950
	6	0.0272	0.9975	0.0338	0.9968	0.0468	0.9961
	7	0.0269	0.9976	0.0335	0.9970	0.0323	0.9968
	8	0.0235	0.9978	0.0322	0.9971	0.0432	0.9961
	9	0.0248	0.9978	0.0301	0.9971	0.0592	0.9945
7	5	0.0299	0.9975	0.0356	0.9968	0.0409	0.9963
	6	0.0282	0.9973	0.0278	0.9975	0.0579	0.9951
	7	0.0227	0.9978	0.0353	0.9967	0.0581	0.9948
	8	0.0336	0.9969	0.0307	0.9973	0.0445	0.9962
	9	0.0274	0.9973	0.0344	0.9968	0.0404	0.9964
8	5	0.0263	0.9974	0.0335	0.9971	0.0349	0.9970

	6	0.0291	0.9976	0.0321	0.9970	0.0415	0.9960
	7	0.0257	0.9979	0.0321	0.9970	0.0550	0.9951
	8	0.0379	0.9972	0.0322	0.9970	0.0475	0.9956
	9	0.0255	0.9978	0.0339	0.9967	0.0604	0.9944
9	5	0.0231	0.9979	0.0339	0.9968	0.0492	0.9953
	6	0.0270	0.9977	0.0306	0.9974	0.0479	0.9955
	7	0.0239	0.9977	0.0346	0.9965	0.0507	0.9951
	8	0.0239	0.9978	0.0306	0.9971	0.0570	0.9939
	9	0.0232	0.9979	0.0344	0.9969	0.0451	0.9959

Source: own work

In order to train the NARNN, open loop architecture (Figure 2) shows a block diagram of the NARNN generated during MATLAB processing in the MATLAB (2022a) Neural Network Toolbox. In Figure 2, the block y(t) is the input series consisting of global mean absolute sea level change observations. The number "1" at the bottom of the block indicates univariate time series.

The hidden layer of the network is illustrated in the second block, namely "Hidden Layer with Delays". The inner boxes "w" and "b" represent input-hidden weights and bias respectively for a single neuron in the hidden layer. The term "1:3" denotes the number of delays used. The larger box after the summation sign indicates the sigmoid transfer function of each neuron. The number "11" at the bottom of the "Hidden Layer with Delays" block denotes the number of hidden neurons.

The "Output Layer" block represents the output layer of the network. The inner boxes "w" and "b" represent the hidden-output weights and biases respectively. The transfer function of the output layer is linear. There is only one output neuron, which is denoted below the "Output Layer" block. The last block y(t) represents the predicted output. This output y(t) is different from the input y(t). Since the output of the network is a prediction of the input time series, MATLAB signifies both with the same variable [2].

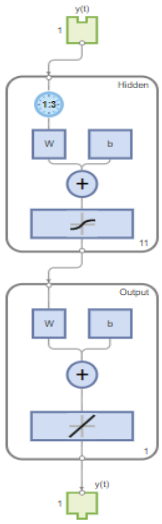

Figure 2. Open Loop Architecture (MATLAB Output)

4.2 NARNN Training Output

The LM algorithm typically requires more memory but less time. Training automatically stop when generalization stop improving, as indicated by an increase in the mean square error of the validation samples [2]. Table 2 displayed the training progress using the LM algorithm, stopped when the validation error increased for six iterations with Performance = 0.0204, Gradient = 0.00151, and Mu = 0.001 at epoch 13. In terms of processing time, the LM algorithm took 00:00:00 during training. The term epoch represents the number of iterations during training in which it is attempted to minimize the error function.

Table 2. NARNN Training Output

Unit	Initial Value	Stopped Value	Target Value
Epoch	0	13	1000
Elapsed Time	---	00:00:00	---
Performance	85.2	0.0204	0
Gradient	172	0.0151	1e-07
Mu	0.001	0.001	1e+10
Validation Checks	0	6	6

(MATLAB Output)

4.2.1 NARNN Best Performance

The performance plot illustrated the relationship between the training, validation, and testing phases in forecasting global mean absolute sea level change, in terms of MSE versus the number of epochs. The performance was evaluated by taking MSE and epochs after the training was completed, and then the values were generated. The performance plot is a useful diagnostic tool to plot the training, validation, and testing errors to check the progress of training. It also illustrated that the training stopped when the validation error increased at the circled epoch. As illustrated in Figure 3, the best performance for the validation phase was 0.060945 at epoch 7 for the NARNN model. The results showed a good network performance because the validation error and testing error have similar characteristics, and it did not appear that any significant overfitting has occurred.

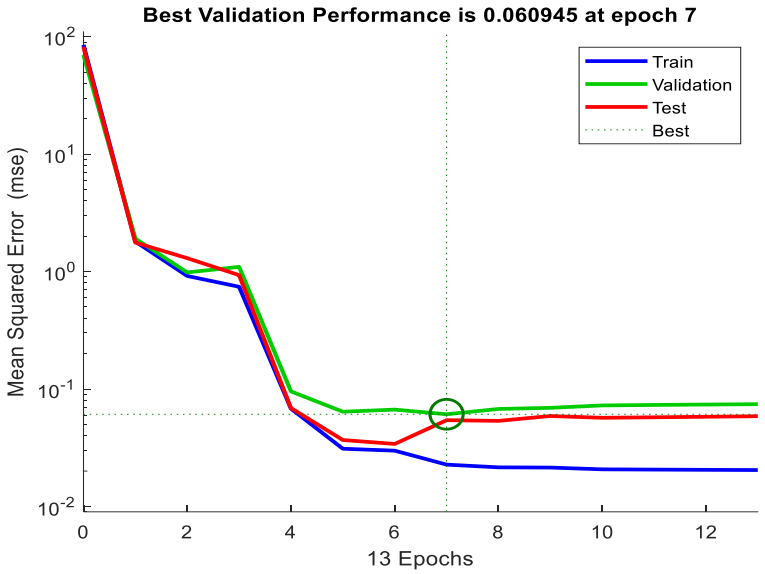

Figure 3. Performance Plot of the NARNN Model (MATLAB Output)

4.2.2 NAR Neural Network Error Histogram

The error histogram can give an indication of outliers, which are data points where the fit is significantly worse than that of most of the data. In the error histograms (Figure 4), the blue bars represent the training data, the green bars represent the validation data, and the red bars represent the testing data. The results showed that there had a few training points and testing points outside of the range. If the outliers are valid data points but are unlike the rest of the data, then the network is extrapolating for these points. It means more data similar to the outlier points should be considered in training analysis and that the network should be retrained.

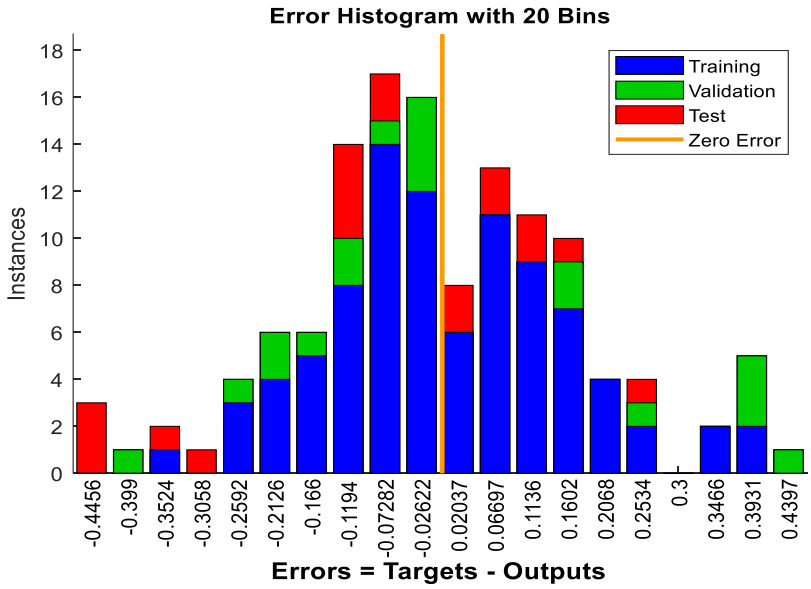

Figure 4. Error Histogram of the NARNN Model (MATLAB Output)

4.2.3 NARNN Time-Series Response

The dynamic network time-series response plots were displayed in Figure 5 for the NARNN model, showing that the outputs were distributed evenly on both sides of the response curve, and the errors versus time were small in the training, validation, and testing phases. The results indicated that the model was able to predict the time series over the simulation period efficiently.

Figure 5. Network Time-Series Response of the NARNN Model (MATLAB Output)

4.2.4 NARNN Error Autocorrelation

The error autocorrelation function describes how the prediction errors are related in time. For a perfect prediction model, there should only be one nonzero value of the autocorrelation function, and it should occur at zero lag (this is the MSE). This would mean that the prediction errors are completely uncorrelated with each other (white noise). If there is significant correlation in the prediction errors, then it should be possible to improve the prediction - perhaps by increasing the number of delays in the tapped delay lines.

The correlations for the NARNN model (Figure 6), except for the one at zero lag, all fell approximately within the 99.5% confidence limits around zero, so the models seemed to be adequate. If there are some exceptions which suggest that the created network can be improved by retraining it or by increasing the number of neurons in the hidden layer. If even more accurate results are required, retrain the network will change the initial weights and biases of the network, and may produce an improved network after retraining.

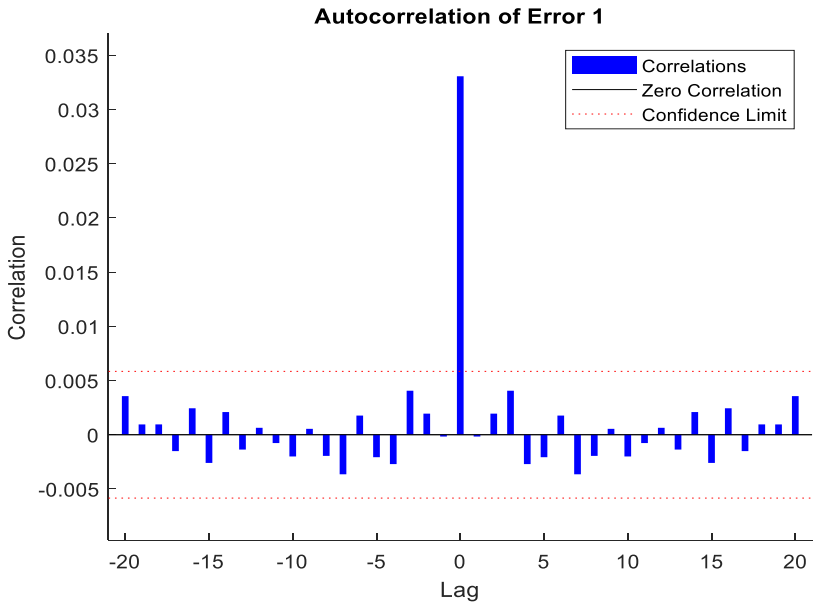

Figure 6. Error Autocorrelation of the NARNN Model (MATLAB Output)

5. Conclusion

The future is more intensive in knowledge, more understanding of the natural complexities of living systems. In order to bring together a wide variety of perspectives and concepts, it requires holistic solutions that involve working across disciplines, principles and methods to support interdisciplinarity and transdisciplinarity, to explore and formalize systems concepts, and to develop systemic methods for learning and change.

Understanding past sea level is important for the analysis of current and future sea level changes. Sea level rise is a relatively slow process, and the majority of impacts, with the exception of seasonally flooded low-lying coastal areas, are predicted and modeled for the future. In order to sustain these observations, research programs utilizing the resulting data should be able to significantly improve our understanding and narrow projections of future sea level rise and variability.

Prediction is a kind of dynamic filtering, in which past values of the time series can be used to predict future values. Empirically, the NARNN model is good at modelling nonlinear problems for the time series. In this study, the NARNN model with 7 neurons in the hidden layer and 7 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 [7].

According to the results of this study, this NARNN model not only can provided richer information which are important in decision making process related to the future global sea level rise impacts, but also can be employed in forecasting the future performance for global average sea level change outcomes. Thus, this study may provide an integrated modelling approach as a decision-making supportive method for formulating global average sea level change prediction in advance.

Acknowledgement

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

References

[1] Adamowski, J., Chan, H. F., Prasher, S. O., Ozga-Zielinski, B., and Sliusarieva, A. 2012. Comparison of multiple linear and nonlinear regression, autoregressive integrated moving average, artificial neural network, and wavelet artificial neural network methods for urban water demand forecasting in Montreal, Canada, *Water Resources Research*, 48, 1–14.

[2] Beale, Mark Hudson, Hagan, Martin T., & Demuth, Howard B. 2019. Deep Learning Toolbox™: Getting Started Guide. Natick, MA: The MathWorks, Inc.

[3] Benrhmach, G., Namir, K., Namir, A., and Bouyaghroumni, J. 2020. Nonlinear autoregressive neural network and extended Kalman filters do prediction of financial time series. *Journal of Applied Mathematics*, Vol. 2020, Article ID 5057801, 1-6.

[4] Bolin, D., Guttorp, P., Januzzi1, A., Jones1, D., Novak1, M., Podschwit, H., Richardson, L., S¨arkk¨a, A., Sowder, C., & Zimmerman, A. (2015). Statistical prediction of global sea level from global temperature. *Statistica Sinica*, 25, 351-367. doi: 10.5705/ss.2013.222w

[5] Cazenave, A., & Llovel, W. (2010). Contemporary sea level rise. *Annual Review of Marine Science*, 2, 145-173. doi: 10.1146/annurev-marine-120308-081105.

[6] Cazenave, A., & Cozannet, G. Le. (2013). Sea level rise and its coastal impacts. *Earth's Future*, *2*, 15–34, doi:10.1002/2013EF000188.

[7] Church, J. A., White, N. J., Aarup, T., Wilson, W. S., Woodworth, P. L., Domingues, C. M., Hunter, J. R., &d Lambeck, K. (2008). Understanding global sea levels: past, present and future. *Sustainability Science*, 3, 9-22. doi: 10.1007/s11625-008-0042-4

[8] Foster, G., & Brown, P. T. (2014). Time and tide: analysis of sea level time series. *Climate Dynamics*, 45(1-2), 291-308. doi:10.1007/s00382-014-2224-3

[9] Gavin, Henri P. 2020. The Levenberg-Marquardt algorithm for nonlinear least squares curve-fitting problems. Department of Civil and Environmental Engineering Duke University. 19 pages. Retrieved from: http://people.duke.edu/~hpgavin/ce281/lm.pdf

[10] Haasnoot, M., Kwadijk, J., Alphen, J. van, Bars, D. Le, Hurk, B. van den, Diermanse, F., Spek, A. van der, Essink, G. O., Delsman, J., & Mens, M. (2020). Adaptation to uncertain sea-level rise; how uncertainty in Antarctic mass-loss impacts the coastal adaptation strategy of the Netherlands. *Environmental Research Letters*, 15, 034007, 1-15. doi.org/10.1088/1748-9326/ab666c

[11] Hagan, M. T., Demuth, H. B., Beale, M. H., and De Jesus, O. 2014. Neural network design, 2nd Edition, pp. 12-19~12-22 and 13-12~13-14, Publisher: Martin Hagan.

[12] Hill, T., O'Connor, M., and Remus, W. 1996. Neural network models for time series

forecasts, *Management Science*, 42(7), 1082-1092.

[13] Horton, B. P., Kopp, R. E., Garner, A. J., Hay, C. C., Khan, N. S., Roy, K., & Shaw, T. A. (2018). Mapping sea-level change in time, space, and probability. *Annual Review of Environment and Resources*, 43, 481-521. doi.org/10.1146/annurev-environ-102017-025826

[14] IPCC, (2014). *Climate Change 2014: Synthesis Report. Contribution of Working Groups I, II and III to the Fifth Assessment Report of the Intergovernmental Panel on Climate Change* [Core Writing Team, R.K. Pachauri and L.A. Meyer (eds.)]. IPCC, Geneva, Switzerland, 151 pp. Retrieved from: https://www.ipcc.ch/report/ar5/syr/

[15] Kulp, S. A., & Strauss, B. H. (2019). New elevation data triple estimates of global vulnerability to sea-level rise and coastal flooding. *Nature Communications*, 10:4844, 1-12. doi.org/10.1038/s41467-019-12808-z

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

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

[18] Marquardt, Donald W. 1963. An algorithm for least-squares estimation of nonlinear parameters. *Journal of the Society for Industrial and Applied Mathematics*, 11(2), 431-441.

[19] Møller, Martin Fodslette. 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

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

[21] Sariev, Eduard, & Germano, Guido. 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

[22] Srivastava, P. K., Islam, T., Singh, S. K., Petropoulos, G. P., Gupta, M., & Dai, Q. (2016). Forecasting Arabian sea level rise using exponential smoothing state space models and ARIMA from TOPEX and Jason satellite radar altimeter data. *Meteorological Applications*, 23, 633-639. doi:10.1002/met.1585

[23] Tealab, A. 2018. Time series forecasting using artificial neural networks methodologies: A systematic review, *Future Computing and Informatics Journal*, 3(2), 334-340.

[24] U.S. Global Change Research Program (USGCRP), (2017). *Climate science special report: Fourth National Climate Assessment*, Volume I. [Wuebbles, D.J., D.W. Fahey, K.A. Hibbard, D.J. Dokken, B.C. Stewart, and T.K. Maycock (eds.)]. U.S. Global Change Research Program, Washington, DC, USA, 470 pp. Retrieved from: https://science2017.globalchange.gov/

[25] Visser, H., Dangendorf, S., & Petersen, A. C. (2015). A review of trend models applied to sea level data with reference to the "acceleration-deceleration debate". *Journal of Geophysical Research: Oceans*, 120(6), 3873-3895. doi:10.1002/2015JC010716

[26] Yue, Z., Songzheng, Z., and Tianshi, L. 2011. Regularization BP neural network model for predicting oil-gas drilling cost, In Proceedings of the 2011 International Conference on Business Management and Electronic Information, Guangzhou, China, Volume 2, 483–487.

[27] Zhang, G. Peter, Patuwo, B. Eddy, and Hu, Michael Y. 1998. Forecasting with artificial neural networks: The state of the art. *International Journal of Forecasting*, 14(1), 35-62.

[28] Zhang, G. Peter. 2003. Time series forecasting using a hybrid ARIMA and neural network model. *Neurocomputing*, 50, 159-175.