Noname manuscript No.
(will be inserted by the editor)

Dynamic ridge polynomial neural network with Lyapunov function for time series forecasting

Waddah Waheeb · Rozaida Ghazali · Abir Jaafar Hussain

Received: date / Accepted: date

Abstract The ability to model the behaviour of arbitrary dynamic system is one of the most useful properties of recurrent networks. Dynamic ridge polynomial neural network (DRPNN) is a recurrent neural network used for time series forecasting. Despite the potential and capability of the DRPNN, stability problems could occur in the DRPNN due to the existence of the recurrent feedback. Therefore, in this study, a sufficient condition based on an approach that uses adaptive learning rate is developed by introducing a Lyapunov function. To compare the performance of the proposed solution with the existing solution, which is derived based on the stability theorem for a feedback network, we used six time series, namely Darwin sea level pressure, monthly smoothed sunspot numbers, Lorenz, Santa Fe laser, daily Euro/Dollar exchange rate and Mackey-Glass time-delay differential equation. Simulation results proved the stability of the proposed solution and showed an average 21.45% improvement in Root Mean Square Error (RMSE) with respect to the existing solution. Furthermore, the proposed solution is faster than the existing solution. This is due to the fact that the proposed solution solves network size restriction found in the existing solution and takes advantage of the calculated dynamic system variable to check the stability, unlike the existing solution that needs more calculation steps.

Waddah Waheeb
Faculty of Computer Science and Information Technology, Universiti Tun Hussein Onn Malaysia, Batu Pahat, Johor, Malaysia
E-mail: waddah.waheeb@gmail.com

Rozaida Ghazali
Faculty of Computer Science and Information Technology, Universiti Tun Hussein Onn Malaysia, Batu Pahat, Johor, Malaysia
E-mail: rozaida@uthm.edu.my

Abir Jaafar Hussain
Liverpool John Moores University, Byrom Street, Liverpool L3 3AF, United Kingdom
E-mail: a.hussain@ljmu.ac.uk

Keywords Dynamic ridge polynomial neural network · Recurrent neural network · Higher order neural network · Time series forecasting · Adaptive learning rate · Lyapunov function

1 Introduction

Time series is a sequence of observations for a variable of interest made over time. Time series is used in many disciplines for things such as daily exchange rate price, quarterly sales and annually sunspots numbers. Time series forecasting is defined as an estimation of the future behaviour of a time series using current and past observations [30].

Various methods for time series forecasting have been developed. From statistics-based to intelligence-based, there are a range of methods available to make a forecast. Intelligent methods such as Artificial Neural Networks (ANNs) have been successfully used in time series forecasting [9, 22, 34, 52]. During training, ANNs use historical data to build a model that has the ability to forecast future observations.

ANNs have a non-linear input-output mapping nature that allow them to approximate any continuous function with an arbitrary degree of accuracy. ANNs are less susceptible to model misspecification than other parametric non-linear methods because the non-linear input-output mapping in ANNs is generated with little priori knowledge about the non-linearity in the series [37, 52].

Higher order neural networks (HONNs) have been used successfully for time series forecasting [9, 16, 22, 23, 39, 41]. They have a single layer of trainable weights which enables faster network training. Ridge Polynomial Neural Network (RPNN) [43] is a HONN that maintains fast learning and powerful mapping properties, which makes it suitable for solving complex problems [23].

For time series forecasting, an explicit treatment for the dynamics involved is needed for neural network models because the behaviour of some time series signals are related to past inputs that present inputs depend on [24]. For that, a recurrent version of the RPNN was proposed and named Dynamic Ridge Polynomial Neural Network (DRPNN) by [24]. Recurrent networks learn the dynamics of the series over time and store them in their memory, and then use these memories when forecasting [40]. RPNN and DRPNN have been successfully applied to forecast time series [9, 22–24], with DRPNN the most suitable for time series forecasting.

Despite the potential and capability of the DRPNN, the problems of complexity and difficulty of training could occur in the DRPNN [22]. To tackle these problems, a sufficient condition for the convergence of the DRPNN was derived based on the stability theorem for a feedback network proposed by Atiya [11]. This solution adjusts the weights of the network to generate network outputs that get as close as possible to the desired output [22]. However, this solution could be too restrictive in some cases where a large network is necessary [11]. Therefore, feedback network stability theorem is restrictive and causes lower forecasting accuracy with many time series where more accurate forecasts are needed such as financial or disaster forecasting.

In an attempt to overcome the stability problems for DRPNN, in this work a sufficient condition based on an approach that uses adaptive learning rate is developed by introducing a Lyapunov function. Such approach has been used effectively with different recurrent ANNs models such as fully connected recurrent networks [31], recurrent wavelet elman neural network [33] and self-recurrent wavelet neural network [54].

The contributions made by this study are as follows:

- To tackle the problems of complexity and difficulty of training for DRPNN, a sufficient condition based on an approach that uses adaptive learning rate is developed by introducing a Lyapunov function. Then, we applied it for time series forecasting.
- A comparative analysis of the proposed solution with the existing solution was completed using six time series, namely Darwin sea level pressure, monthly smoothed sunspot numbers, Lorenz, Santa Fe laser, daily Euro/Dollar exchange rate and Mackey-Glass time-delay differential equation.
- The forecasting performance of DRPNN with the proposed solution was compared with other models in the literature.

The remainder of this study is organized as follows. In Section 2, we review the dynamic ridge polynomial neural network and the existing solution for its stability. In Section 3, we present the proposed solution. Section 4 describes the experimental design. Section 5 presents results and discussion. The conclusion is given in Section 6.

2 Related works

The following subsections describe the dynamic ridge polynomial neural network and stability issue found in it.

2.1 Dynamic ridge polynomial neural network (DRPNN)

Time series forecasting requires explicit treatment of dynamics because present inputs of the time series depend on some past inputs [22]. Neural networks with recurrent connections are dynamic systems with temporal state representations. Due to their dynamic structure, they have been successfully used for time series forecasting [14, 22, 36]. Dynamic ridge polynomial neural network (DRPNN) is a recurrent neural network. It has the extension architecture and functionality of the feedforward ridge polynomial neural network (RPNN). By incorporating the recurrent connection, DRPNN is better able to model the dynamics of time series as compared to RPNN and many other higher order neural networks techniques as found in [9, 21, 22, 24].

The structure of DRPNN is shown in Fig. 1. DRPNN is constructed from a number of increasing order of Pi-Sigma units [42] with the addition of a recurrent connection from the output layer to the input layer. This recurrent connection feeds the network output to the summing nodes in each Pi-Sigma units, thus allowing them to see the resulting output of the previous sample. All weights are fixed to unity except the weights link the inputs with the first summing layer as shown in Fig. 1.

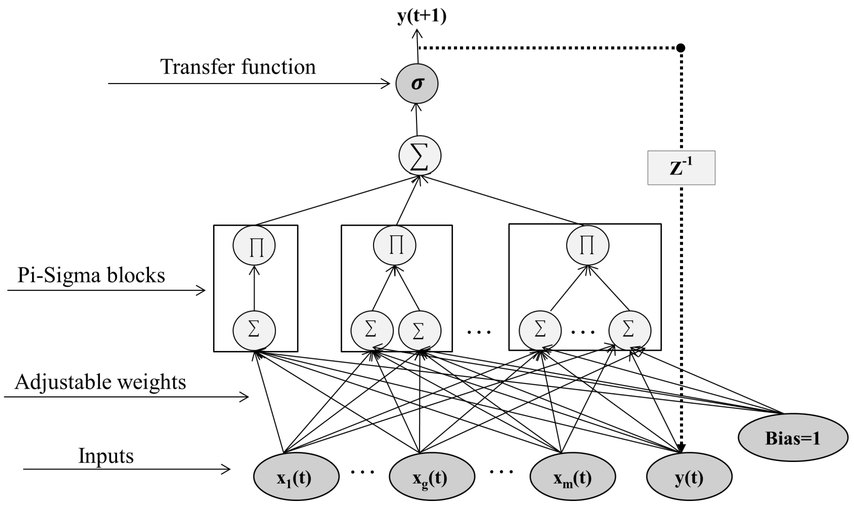


Fig. 1: Dynamic Ridge Polynomial Neural Network. Z^1 denotes the time delay operator.

DRPNN uses a constructive learning algorithm based on the asynchronous updating rule of the Pi-Sigma unit. That means that DRPNN starts with a small basic structure, then grows by adding Pi-Sigma unit of increasing order to its structure as the learning proceeds until the desired mapping task is carried out with the required degree of accuracy. Real Time Recurrent Learning algorithm [51] is used to update network weights. The output of the DRPNN network are given by:

$$y(t+1) \approx \sigma\left(\sum_{i=1}^{k} P_i(t+1)\right) \tag{1}$$

$$P_i(t+1) = \prod_{j=1}^{i} (h_j(t+1)) \tag{2}$$

$$h_j(t+1) = \sum_{g=1}^{m+1} w_{gj} Z_g(t) \tag{3}$$

$$Z_g(t) = \begin{cases} x_g(t) & 1 \leq g \leq m \\ y(t) & g = m+1 \end{cases} \tag{4}$$

where k is the number of PSNN blocks, σ is a non-linear transfer function, m is input vector dimension size, w is a trainable weight, x is an input and $y(t)$ is network output at previous time step.

Sum squared error is used as a standard error measure for training the network as follows [22, 24]:

$$E(t+1) = \frac{1}{2} \sum e(t+1)^2 \tag{5}$$

where

$$e(t+1) = d(t+1) - y(t+1) \tag{6}$$

where $d(t+1)$ is the desired output and $y(t+1)$ is network output. At every time, the weights between inputs g and sigma l are updated as follows:

$$\triangle w_{gl} = -\eta * \left(\frac{\partial E(t+1)}{\partial w_{gl}} \right) \tag{7}$$

where η is the learning rate. The value of $\frac{\partial E(t+1)}{\partial w_{gl}}$ is determined as:

$$\frac{\partial E(t+1)}{\partial w_{gl}} = e(t+1) * \frac{\partial e(t+1)}{\partial w_{gl}} \tag{8}$$

$$\frac{\partial e(t+1)}{\partial w_{gl}} = -\frac{\partial y(t+1)}{\partial w_{gl}} \tag{9}$$

$$\frac{\partial E(t+1)}{\partial w_{gl}} = -e(t+1) * \frac{\partial y(t+1)}{\partial w_{gl}} \tag{10}$$

$$\frac{\partial y(t+1)}{\partial w_{gl}} = \frac{\partial y(t+1)}{\partial P_k(t+1)} * \frac{\partial P_k(t+1)}{\partial w_{gl}} \tag{11}$$

From (1) to (4), we have:

$$\frac{\partial y(t+1)}{\partial w_{gl}} = (y(t{+}1))' * \left(\prod_{j=1, j\neq l}^{k} h_j(t+1) \right) * \left(Z_g(t) + w_{(m+1)l} * \frac{\partial y(t)}{\partial w_{gl}} \right) \tag{12}$$

Assume D^Y as the dynamic system variable, which is defined as a set of quantities that summarizes all the information about the past behavior of the system that is needed to uniquely describe its future behavior [26], where D^Y is:

$$D^Y_{gl}(t+1) = \frac{\partial y(t+1)}{\partial w_{gl}} \tag{13}$$

Substituting (13) into (12), we have:

$$D_{gl}^{Y}(t+1) = \frac{\partial y(t+1)}{\partial w_{gl}} = (y(t+1))' * \left(\prod_{j=1, j \neq l}^{k} h_j(t+1) \right) * \big(Z_g(t) + w_{(m+1)l} * D_{gl}^{Y}(t) \big) \tag{14}$$

The initial values are $D_{gl}^{Y}(t) = 0$, and $y(t) = 0.5$ to avoid zero value of $D_{gl}^{Y}(t) = 0$ [22, 24].

Weights updating rule is derived by substituting (14) into (8) then (7), such that

$$\triangle w_{gl} = \eta * e(t+1) * D_{gl}^{Y}(t+1) \tag{15}$$

Finally,

$$w_{gl}^{new} = w_{gl}^{old} + \triangle w_{gl} \tag{16}$$

2.2 Stability issue for DRPNN

The ability to model the behaviour of arbitrary dynamic system is one of the most useful properties of recurrent networks. Hence, the recurrent feedback in DRPNN enhances its forecasting performance as found in [9, 22, 24]. Despite the potential and capability of the DRPNN, the problems of complexity and difficulty of training could occur in the DRPNN [22]. These problems can be summarized in two main points. First, calculating the gradients and updating the weights of the DRPNN is difficult because the dynamic system variables affect both the gradient and the output. Second, the learning error may not be monotonically decreasing which could lead to long convergence time. To tackle these problems, a sufficient condition for the convergence of the DRPNN was derived based on the stability theorem for a feedback network proposed by Atiya [11].

This stability theorem adjusts the weights of the network to generate network outputs that get as close as possible to the desired output [22]. According to [22], the condition for DRPNN to converge based on the feedback network stability theorem is described by:

$$\left(max \left(\sum_{k=1}^{A} \sum_{L=1}^{k} |W_{L(M+2)}| * \prod_{S=1, S \neq L}^{k} \sum_{m=1}^{M+2} |W_{Sm}| \right) \right) < \frac{1}{(max|f'|)} \tag{17}$$

where A is the number of PSNN blocks, $W_{L(M+2)}$ is the weights that link the recurrent node with hidden layer nodes, and f is a nonlinear transfer function.

The pseudo code used to update the weights of DRPNN based on the feedback network stability theorem is as follows:

Algorithm 1 Constructive learning algorithm for the DRPNN with feedback network stability theorem

```
Set $Epoch_{ID} = 0$.
Assign suitable values to $\epsilon_{threshold}$, $\eta$, $r$, $\delta_r$, $\delta_\eta$ and $Epoch_{threshold}$.
loop
    Calculate $P_i$ using Equation (2)
A:
    for all training samples do
        Calculate actual network output using Equation (1)
        Calculate the dynamic system variable using Equation (14)
        Update weights by applying the asynchronous update rule in Equation (16)
                                        ▷ START STABILITY CALCULATION
        Network stability calculation for both sides in Equation (17)
                                        ▷ END STABILITY CALCULATION
    end for
    Calculate current epoch's error ($\epsilon_c$)
    if $\epsilon_c < \epsilon_{threshold}$ or $Epoch_{ID} > Epoch_{threshold}$ then
        Stop learning
    end if
    $Epoch_{ID} \leftarrow Epoch_{ID} + 1$
    $\epsilon_p \leftarrow \epsilon_c$
    if $|(\epsilon_c - \epsilon_p)/\epsilon_p| \geq r$ then
        Go to Step A
    else
                                        ▷ START STABILITY CHECKING
        if Stability condition in Equation (17) is not satisfied then
            Stop learning
        end if
                                        ▷ END STABILITY CHECKING
        $\widehat{P_k} \leftarrow P_k$
        $r \leftarrow r * \delta_r$
        $\eta \leftarrow \eta * \delta_\eta$
        $k \leftarrow k + 1$
    end if
end loop
```

where $\epsilon_{threshold}$: Mean Squared Error (MSE) threshold for the training phase; ϵ_c, ϵ_p: the training MSE's for the current epoch and previous epoch, respectively; r: threshold for successive addition of new PSNN blocks; η: initial learning rate; δ_r, δ_η: decreasing factors for r and η, respectively; k: degree of PSNN, as well as $Epoch_{ID}$, and $Epoch_{threshold}$: number of training epochs and maximum number of epochs to finish training, respectively.

The feedback network stability theorem was used with recurrent networks for problems such as pattern recognition and time series forecasting as shown in Table 1. However, this solution is restrictive in some cases where a large network is necessary [11] or when working with constructive learning because it stops the learning with small number of hidden units. Furthermore, when we are working with time series forecasting, we are considering only the problem of learning trajectories, not learning fixed points [12]. Therefore, another solution is needed to solve network size restriction and overcome stability issue.

Table 1: Studies used the feedback network stability theorem

Model name	Problem	Stability theorem role
RBP [13]	Time independent pattern recognition	Used to guarantee a solution for the network
RPSN [28]	Differential pulse code modulation image coding	Used to derive a condition for RPSN to converge
RPSN [29]	Time series forecasting	Used to derive a condition for RPSN to converge
DRPNN [22]	Time series forecasting	Used to derive a condition for DRPNN to converge
Hybrid neural network [25]	Classification of graph structured data	Used to guarantee the convergence of the state vector

DRPNN = Dynamic ridge polynomial neural network
RBP = Recurrent backpropagation
RPSN = Recurrent pi-sigma neural network

3 Proposed solution for DRPNN stability issue based on Lyapunov function

In an attempt to overcome the stability problems for DRPNN, in this work a sufficient condition based on an approach that uses adaptive learning rate is developed by introducing a Lyapunov function. Such approach has been used effictively with different models such as fully connected recurrent networks [31], recurrent wavelet elman neural network [33], Adaptive Network based Fuzzy Inference System [44], dynamic neural network [53] and self-recurrent wavelet neural network [54].

First, let us define a Lyapunov function as follows:

$$V(t) = \frac{1}{2}e^2(t) \tag{18}$$

where $e(t)$ represents the error that is calculated by differencing the desired value from the predicted value. We use this error function because the DRPNN model is used to minimize it.

According to Equation (18), the change in the Lyapunov function can be determined by:

$$\triangle V(t) = V(t+1) - V(t) = \frac{1}{2}\left[e^2(t+1) - e^2(t)\right] \tag{19}$$

The error difference can be represented by [33, 53]:

$$e(t+1) = e(t) + \triangle e(t) \tag{20}$$

$$e(t+1) \cong e(t) + \left[\frac{\partial e(t)}{\partial w}\right]^T \triangle w \tag{21}$$

where $\triangle w$ represents the weight change. Based on Equations (9), (13) and (15), we have:

$$e(t+1) \cong e(t) - \eta * e(t) * \left[D^Y(t)\right]^T * D^Y(t) \tag{22}$$

$$e(t+1) \cong e(t)(1 - \eta * \left[D^Y(t)\right]^T * D^Y(t)) \tag{23}$$

From Equations (19) and (23), $\triangle V(t)$ can be represented as:

$$\triangle V(t) = \frac{1}{2}\eta * e^2(t) * \left[D^Y(t)\right]^T * D^Y(t)(\eta * \left[D^Y(t)\right]^T * D^Y(t) - 2) \tag{24}$$

$$\triangle V(t) = \frac{1}{2}\eta * e^2(t) * \left(\| D^Y(t) \|_F\right)^2 (\eta * \left(\| D^Y(t) \|_F\right)^2 - 2) \tag{25}$$

where $\| \, . \, \|_F$ is the Frobenius norm which is calculated by using trace function [44].

A sufficient condition to ensure stability is $\triangle V(t) < 0$. Therefore, Equation (25) leads to:

$$0 < \eta < \frac{2}{\left(\| D^Y(t) \|_F\right)^2} \tag{26}$$

Notice that Equation (26) suggests an upper bound of η for a sufficient condition to ensure stability in DRPNN. If the learning rate is controlled to be within the bounds in Equation (26), the convergence and the stability are guaranteed based on the analysis of a Lyapunov function, as derived in Appendix A.

The pseudo code that we will use for DRPNN to update its weights based on Lyapunov function is as follows:

Algorithm 2 Constructive learning algorithm for the DRPNN with Lyapunov function

```
Set $Epoch_{ID} = 0$.
Assign suitable values to $\epsilon_{threshold}$, $\eta$, $r$, $\delta_r$, $\delta_\eta$ and $Epoch_{threshold}$.
loop
    Calculate $P_i$ using Equation (2)
A:
    for all training samples do
        Calculate actual network output using Equation (1)
        Calculate the dynamic system variable using Equation (14)
                                                        ▷ START STABILITY CHECKING
        if $\eta$ outside the bounds in Equation (26) then
            Stop learning
        end if
                                                          ▷ END STABILITY CHECKING
        Update weights by applying the asynchronous update rule in Equation (16)
    end for
    Calculate current epoch's error ($\epsilon_c$)
    if $\epsilon_c < \epsilon_{threshold}$ or $Epoch_{ID} > Epoch_{threshold}$ then
        Stop learning
    end if
```

```
        $Epoch_{ID} \leftarrow Epoch_{ID} + 1$
        $\epsilon_p \leftarrow \epsilon_c$
        **if** $|(\epsilon_c - \epsilon_p)/\epsilon_p| \geq r$ **then**
            Go to Step A
        **else**
            $\widehat{P_k} \leftarrow P_k$
            $r \leftarrow r * \delta_r$
            $\eta \leftarrow \eta * \delta_\eta$
            $k \leftarrow k + 1$
        **end if**
    **end loop**
```

4 Experimental design

The aim of this section is to provide a step by step methodology that we will use to compare the performance between the proposed solution and the existing solution to forecast time series.

Time series used in the experiments

Six time series have been used in our work, namely Darwin sea level pressure (Darwin SLP), monthly smoothed sunspot numbers (Sunspot), Lorenz (Lorenz), Santa Fe laser (Laser), daily Euro/Dollar exchange rate (EUR/USD) and Mackey-Glass time-delay differential equation (Mackey-Glass).

The first time series is the Darwin sea level pressure time series which consists of monthly values of the Darwin sea level pressure for the years 1882-1998. The dataset can be downloaded from [1].

A sub-series of the monthly smoothed sunspot time series from November 1834 to June 2001 was downloaded from [4]. This time series is sensitive to initial conditions because it can be seen as chaotic systems with noise [8].

The third time series is Lorenz time series obtained from [2]. It is a long synthetic chaotic time series of 16384 samples. The last 4000 samples are considered in this work.

Santa Fe laser time series obtained from a far-infrared laser. This time series has periods of oscillations with increasing amplitude, followed by sudden, difficult to predict, activity collapses [20]. Santa Fe laser time series can be downloaded from [6]. Six inputs were used for one-step-ahead forecasting as reported in [46].

The daily Euro/Dollar (EUR/USD) exchange rate contains 781 observations covering the period from January 3, 2005 to December 31, 2007 [27]. The data can be collected from [5, 7].

The last time series is the well-known Mackey-Glass time series which is defined as follows:

$$\frac{dx}{dt} = \beta x(t) + \frac{\alpha x(t-\tau)}{1 + x^{10}(t-\tau)} \tag{27}$$

where t is a variable, x is a function of t, and τ is the time delay. The initial values of the series are $\alpha = 0.2$, $\beta = -0.1$, $x(0) = 1.2$, and $\tau = 17$. It is known that with this setting the series shows chaotic behaviour. From the generated time series, 1000 points were extracted as explained in [35]. This series can be found in the file mgdata.dat in MATLAB [35] or in [3].

The time series settings used in this research are shown in Table 2. The used intervals for training and out-of-sample sets for all used time series are shown in Fig. 2.

Table 2: Time series information.

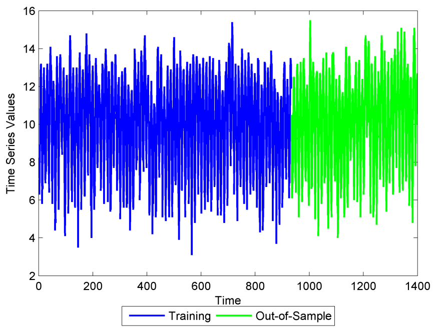
Time series	Input-output data pairs	Training samples#	Out-of-sample samples#
Darwin SLP	$[x(t-11), x(t-6), x(t-3), x(t-2), x(t-1); x(t+1)]$	933	467
Sunspot	$[x(t-4), x(t-3), x(t-2), x(t-1), x(t); x(t+1)]$	1000	1000
Lorenz	$[x(t-3), x(t-2), x(t-1), x(t); x(t+1)]$	2000	2000
Laser	$[x(t-19), x(t-10), x(t-9), x(t-7), x(t-1), x(t); x(t+1)]$	1000	9093
EUR/USD	$[x(t-10), x(t-5), x(t); x(t+5)]$	625	156
Mackey-Glass	$[x(t-18), x(t-12), x(t-6), x(t); x(t+6)]$	500	500

Network topology and training

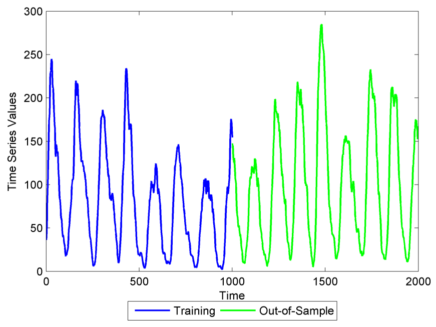
Table 3 shows network topology and training parameters used in this paper. These settings are based on [9,22,24] or by trial and error.

Table 3: Network topology and training.

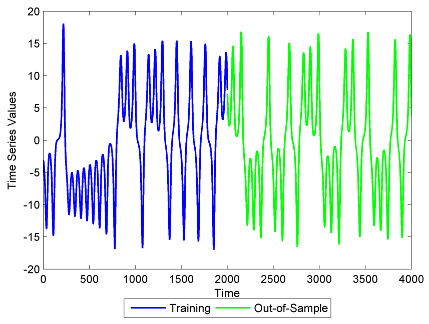
Setting	Value
Learning rate (η) range	[0.01-1]
Momentum range	[0.4-0.8]
Initial weights range	[-0.5,0.5]
Number of input units	Input points as given in Table 2
Transfer function	Sigmoid function
Number of output units	One unit
Stopping criteria for DRPNN	− Maximum number of epochs =3000 or, − After accomplishing the 5th order network learning or, − Network learning becomes unstable.
Threshold for successive addition of a new PSNN (r)	[0.00001,0.1]
Decreasing factors for n (δ_η)	0.8
Decreasing factors for threshold (δ_r)	[0.05,0.2]



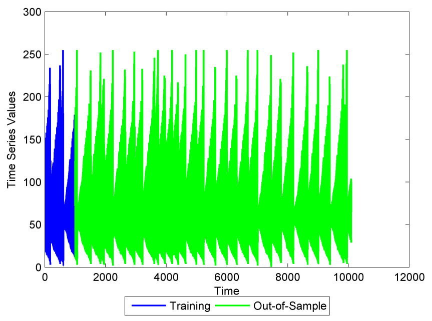
(a) Darwin sea level pressure



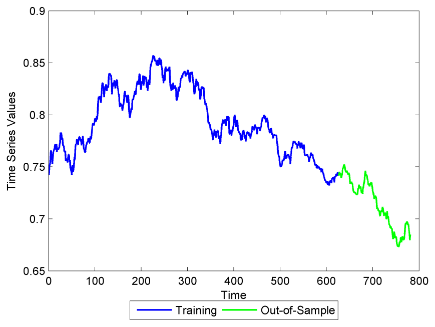
(b) Monthly smoothed sunspot numbers



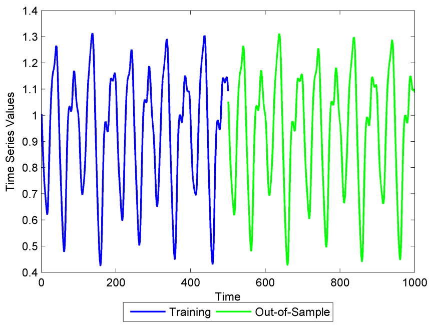
(c) Lorenz



(d) Santa Fe laser



(e) Daily Euro/Dollar exchange rate



(f) Mackey-Glass

Fig. 2: Time series used

Since we used sigmoid transfer function and to follow [22, 24], we scaled the data in the range [0.2, 0.8]. The equation to scale the data is given by:

$$\dot{x} = (max_{new} - min_{new}) * \left(\frac{x - min_{old}}{max_{old} - min_{old}} \right) + min_{new} \tag{28}$$

where $\dot{x}$ refers to the normalized value, x refers to the observation value, min_{old} and max_{old} are the minimum and maximum values of all observations, respectively. min_{new} and max_{new} refer to the minimum and maximum of the new scaled series.

Performance metrics

In this research work, the performance of the networks was evaluated using Root Mean Squared Error (RMSE), Normalized Mean Squared Error (NMSE), training time and network size. Furthermore, we carried out t-test with a significance level of 0.05 to highlight the significant performance. The equation for RMSE and NMSE metrics are given by:

Root Mean Squared Error (RMSE):

$$RMSE = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y_i})^2} \tag{29}$$

Normalized Mean Squared Error (NMSE):

$$NMSE = \frac{1}{N\sigma^2}\sum_{i=1}^{N}(y_i - \hat{y_i})^2 \tag{30}$$

$$\sigma^2 = \frac{1}{N-1}\sum_{i=1}^{N}(y_i - \bar{y})^2 \tag{31}$$

$$\bar{y} = \frac{1}{N-1}\sum_{i=1}^{N} y_i \tag{32}$$

where N, y and $\hat{y}$ represent the number of out-of-sample data, actual output and network output, respectively.

Results and discussion

In this section, the simulation results for the forecasting of the six time series are presented and discussed. We carried out all simulations on a machine with Intel Core i7-3770XPU@3.40GHz, and 4GB of RAM.

Table 4: Root Mean Squared Error (RMSE) improvement of DRPNN$_{Lyapunov}$ to DRPNN$_{feedback}$.

Time series	DRPNN$_{Lyapunov}$	DRPNN$_{feedback}$	Improvement to DRPNN$_{feedback}$ %
Darwin SLP	1.1521	1.2405*	7.13%
Sunspot	2.7781	2.7781	≈ 0%
Lorenz	0.0876	0.1161*	24.55%
EUR/USD	0.007	0.0079*	11.39%
Mackey-Glass	0.0131	0.0252*	48.02%
Laser	9.3152	14.9324*	37.62%
Average of improvement %			21.45%

These are the de-normalized results.
*, means the DRPNN$_{Lyapunov}$ has significant performance than DRPNN$_{feedback}$ using t-test.

Table 5: Normalized Mean Squared Error (NMSE) improvement of DRPNN$_{Lyapunov}$ to DRPNN$_{feedback}$.

Time series	DRPNN$_{Lyapunov}$	DRPNN$_{feedback}$	Improvement to DRPNN$_{feedback}$ %
Darwin SLP	0.1932	0.2239*	13.71%
Sunspot	0.0016	0.0016	≈ 0%
Lorenz	0.00015	0.00028*	46.43%
EUR/USD	0.0866	0.1092*	20.70%
Mackey-Glass	0.0034	0.0173*	80.35%
Laser	0.0440	0.1007*	56.31%
Average of improvement %			36.25%

These are the de-normalized results.
*, means the DRPNN$_{Lyapunov}$ has significant performance than DRPNN$_{feedback}$ using t-test.

Best average simulation results

For fair and more robust comparative evaluation, an average of 30 independent runs are performed for all the neural networks. This was done to avoid any influence due to initial internal state such as random weights initialization. The average performance for the two neural networks using RMSE and NMSE metrics is shown in Table 4 and Table 5, respectively. DRPNN$_{feedback}$ and DRPNN$_{Lyapunov}$ refer to the existing DRPNN with feedback network stability theorem and the proposed DRPNN with Lyapunov function, respectively. Note that in these two tables we de-normalized the forecasted value and compared it with the original desired value.

As seen from Table 4 and Table 5, the forecasting performance of the DRPNN$_{Lyapunov}$ network is significantly better than the DRPNN$_{feedback}$ network in all time series except Sunspot time series. The reason for the absence of any significance in Sunspot is because both networks found best average

Table 6: Network size and average training time for DRPNN$_{Lyapunov}$ and DRPNN$_{feedback}$.

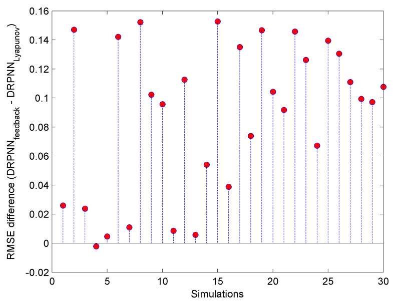
	Network size		Avg. training time in seconds	
Time series	DRPNN$_{Lyapunov}$	DRPNN$_{feedback}$	DRPNN$_{Lyapunov}$	DRPNN$_{feedback}$
Darwin SLP	70	42	199	421
Sunspot	7	7	299	405
Lorenz	36	36	538	463
Laser	80	24	308	412
EUR/USD	50	5	137	234
Mackey-Glass	90	36	65	89
Improvement to DRPNN$_{feedback}$	-122%		23.62%	

Network size equals the number of weights and biases.

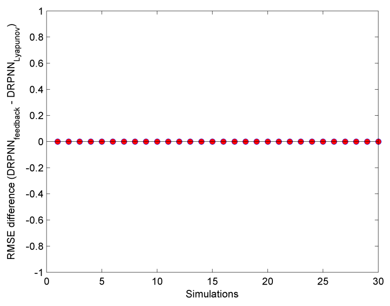
simulations with same parameters setting and network size. Although the network size for both networks are equal with Lorenz time series as shown in Table 6, there is a significance in the performance with DRPNN$_{Lyapunov}$. This is because the parameter settings for the best average for both networks are different.

Results in Table 4 and Table 5 show an average 21.45% improvement in RMSE and an average 36.25% improvement in NMSE with respect to DRPNN$_{feedback}$ network. That means that using the proposed solution helps DRPNN to grow and find more suitable parameter settings during training, thus helping to enhance the forecasting performance for the network. The summary of Table 6 shows that the proposed solution needs more size than the existing solution. However, network size does not increase training time. This is because the proposed solution takes advantage of the calculated dynamic system variable to check the stability, unlike the existing solution that needs more calculation steps. The training time with Lorenz time series for DRPNN$_{Lyapunov}$ is bigger than DRPNN$_{feedback}$ because it needs more epochs to learn from this long time series.

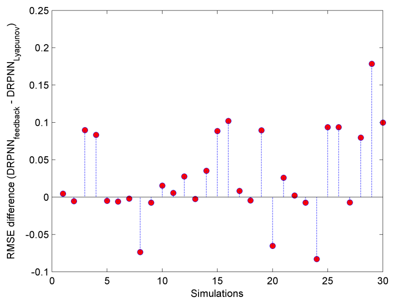
Fig. 3 shows the subtraction of the RMSE of DRPNN$_{feedback}$ from the RMSE of DRPNN$_{Lyapunov}$ with all time series in the 30 simulations. And for NMSE, the subtraction results were plotted in Fig. 4. In these stem plots, a stem in the positive y axis means DRPNN$_{Lyapunov}$ has smaller error than DRPNN$_{feedback}$ in that simulation experiment, but if the stem is in the negative y axis, DRPNN$_{feedback}$ has smaller error than DRPNN$_{Lyapunov}$. As seen from these stem plots, the majority of the stems are in the positive y axis. We can also notice the absence of any stem in some simulations especially with Sunspot time series. This is because there is no difference in the performance.



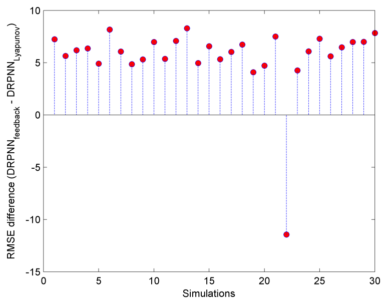
(a) Darwin sea level pressure



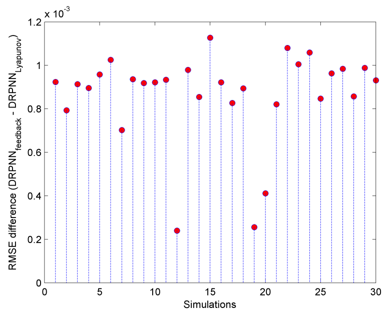
(b) Monthly smoothed sunspot numbers



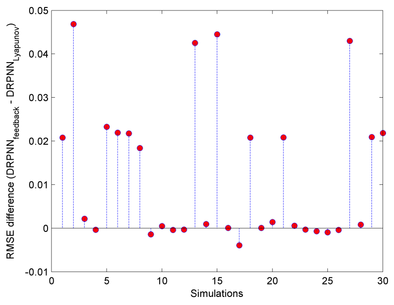
(c) Lorenz



(d) Santa Fe laser

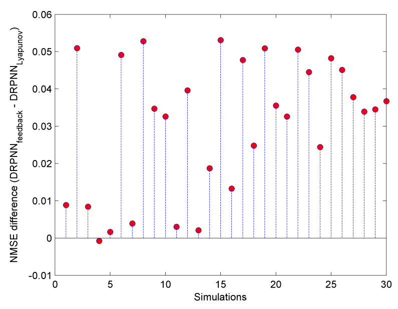


(e) Daily Euro/Dollar exchange rate

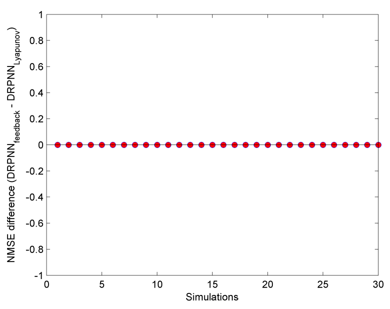


(f) Mackey-Glass

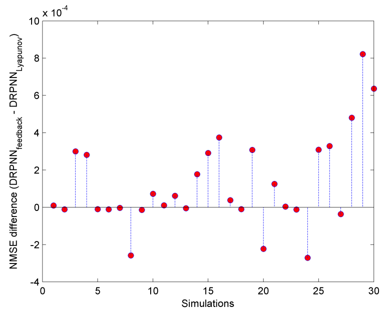
Fig. 3: RMSE difference between DRPNN$_{feedback}$ and DRPNN$_{Lyapunov}$ with time series used



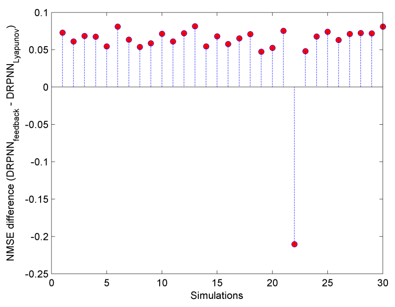
(a) Darwin sea level pressure



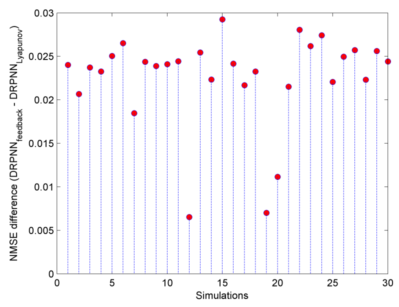
(b) Monthly smoothed sunspot numbers



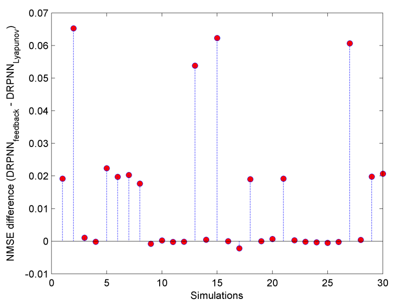
(c) Lorenz



(d) Santa Fe laser



(e) Daily Euro/Dollar exchange rate

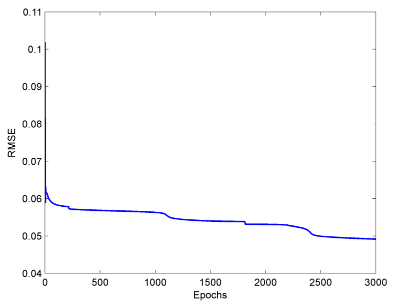


(f) Mackey-Glass

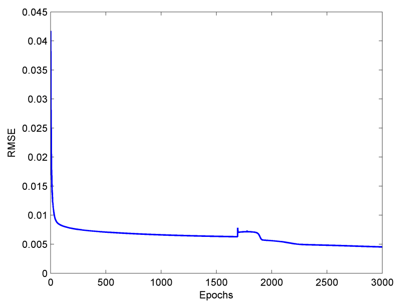
Fig. 4: NMSE difference between DRPNN$_{feedback}$ and DRPNN$_{Lyapunov}$ with time series used

Learning analysis for DRPNN$_{Lyapunov}$

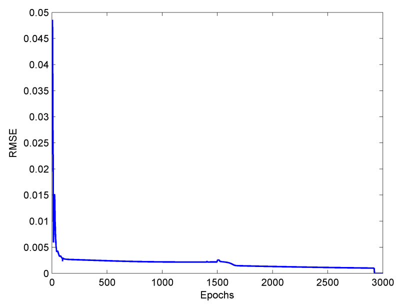
Fig. 5 shows the evolution of RMSE during the learning of DRPNN$_{Lyapunov}$. Each spike shown in the figures comes from the introduction of a new Pi-Sigma block to DRPNN$_{Lyapunov}$. It can be seen from these figures that the learning curves for DRPNN$_{Lyapunov}$ are remarkably stable and RMSE continuously reduced every time Pi-Sigma block is added to the network. Note that the RMSE values in these subfigures are normalized values.



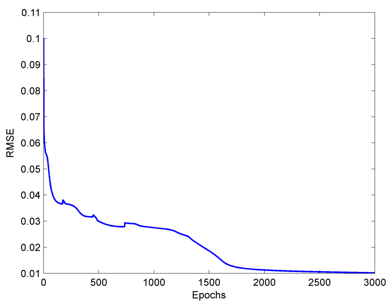
(a) Darwin sea level pressure



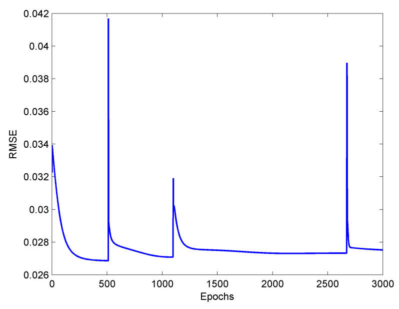
(b) Monthly smoothed sunspot numbers



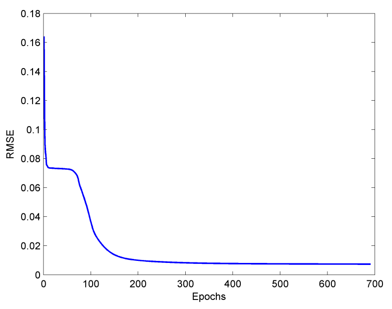
(c) Lorenz



(d) Santa Fe laser



(e) Daily Euro/Dollar exchange rate



(f) Mackey-Glass

Fig. 5: Learning curves based on the best DRPNN$_{Lyapunov}$ simulation with time series used

Comparison of the performance of various existing models

In this section, we compare the performance of DRPNN$_{Lyapunov}$ with other models in the literature. For a fair comparison with recent studies, this study compared the de-normalized results for the DRPNN$_{Lyapunov}$ with de-normalized published results in the literature.

Tables 7 and 8 show the comparison results for generalization capabilities using different methods for the Sunspot and Mackey-Glass time series, respectively. As was observed, DRPNN$_{Lyapunov}$ alone outperforms many hybrid methods. Therefore, hybridizing DRPNN$_{Lyapunov}$ with other models could produce higher forecasting accuracy. The best RMSE for Darwin SLP, Lorenz, Laser and EUR/USD time series using DRPNN$_{Lyapunov}$ is 1.1144, 0.04667, 6.54212 and 0.0068, respectively. The best forecasting for DRPNN$_{Lyapunov}$ using out-of-sample data are shown in Figs. 6- 11. The forecast values were plotted with respect to observed values, as shown in Fig. 12, which show a strong relationship between forecasted and observed values with most time series.

Table 7: Comparison of the performance of various existing models on Sunspot time series.

Model	RMSE	NMSE
FNN [19]	6.4905	0.0174
ART-FNN [19]	6.2204	0.0160
Modified-ART-FNN [19]	5.7173	0.0135
PSNN [47]	-	0.0044
FLNN [47]	-	0.0015
DRPNN$_{Lyapunov}$ (proposed)	**1.9542**	**0.0008**
Brain emotional learning-based RFS [38]	-	0.000664

ART, adaBoost.regression and threshold; FLNN, functional link neural network; FNN, fuzzy neural networks; DRPNN, dynamic ridge polynomial neural network; PSNN, pi-sigma neural network; RFS, recurrent fuzzy system.

Table 8: Comparison of the performance of various existing models on Mackey-Glass time series.

Model	RMSE
Fuzzy modeling method with SVD [50]	0.0894
Gustafson-Kessel fuzzy clustering method + KFA with SVD [49]	0.0748
Orthogonal function neural network + recursive KFA based on SVD [48]	0.05099
Adaptive fuzzy inference system with local research [55]	0.045465
FLNN [47]	0.03656
Beta basis function neural networks + DE algorithm [17]	0.030
Dynamic evolving computation system [15]	0.0289
Backpropagation Network Optimized by Hybrid K-means-Greedy [45]	0.015
Modified DE and the radial basis function [18]	0.013
PSNN [47]	0.0118
DRPNN$_{Lyapunov}$ (proposed)	**0.0105**
Takagi-Sugeno fuzzy system-singleton + simulated annealing [10]	0.00898
Functional-link-based neural fuzzy network-cultural cooperative PSO [32]	0.008424

DE, differential evolution; FLNN, functional link neural network; DRPNN, dynamic ridge polynomial neural network; KFA, kalman filtering algorithm; PSO, particle swarm optimization; PSNN, pi-sigma neural network; SVD, singular value decomposition.

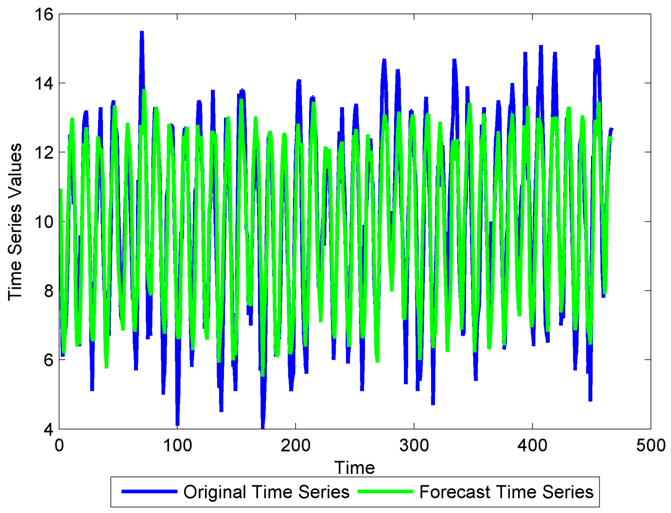


Fig. 6: Out-of-sample forecasting for Darwin sea level pressure time series based on the best DRPNN$_{Lyapunov}$ simulation

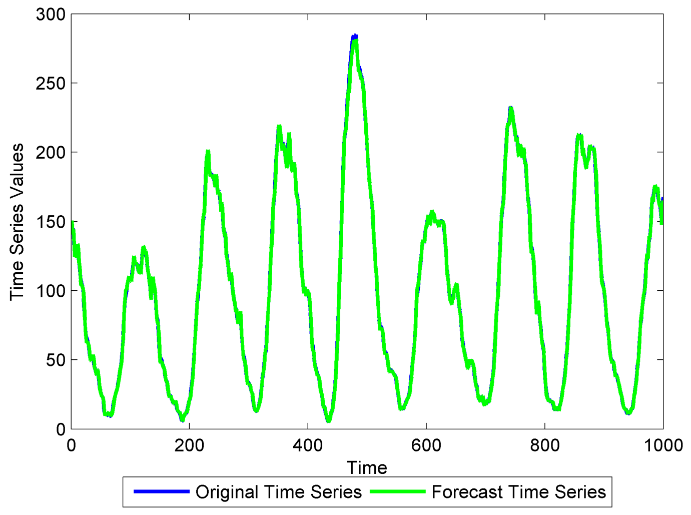


Fig. 7: Out-of-sample forecasting for Sunspot time series based on the best DRPNN$_{Lyapunov}$ simulation

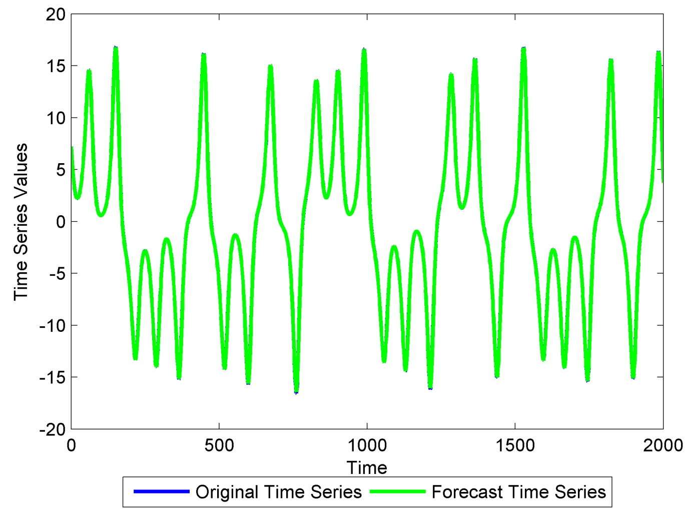


Fig. 8: Out-of-sample forecasting for Lorenz time series based on the best DRPNN$_{Lyapunov}$ simulation

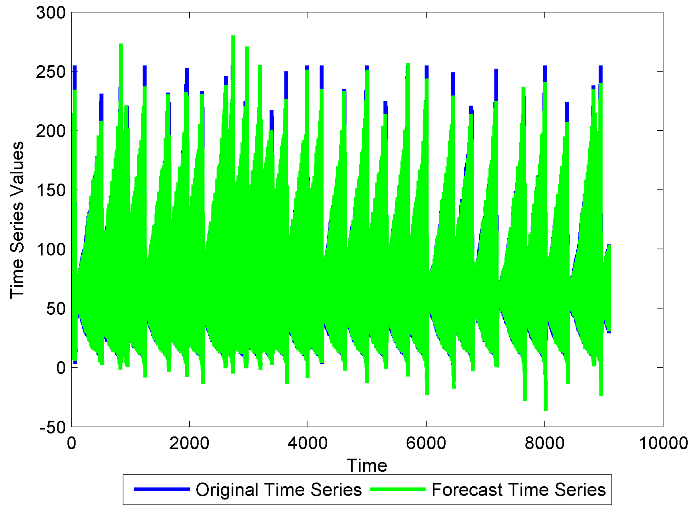


Fig. 9: Out-of-sample forecasting for Laser time series based on the best DRPNN$_{Lyapunov}$ simulation

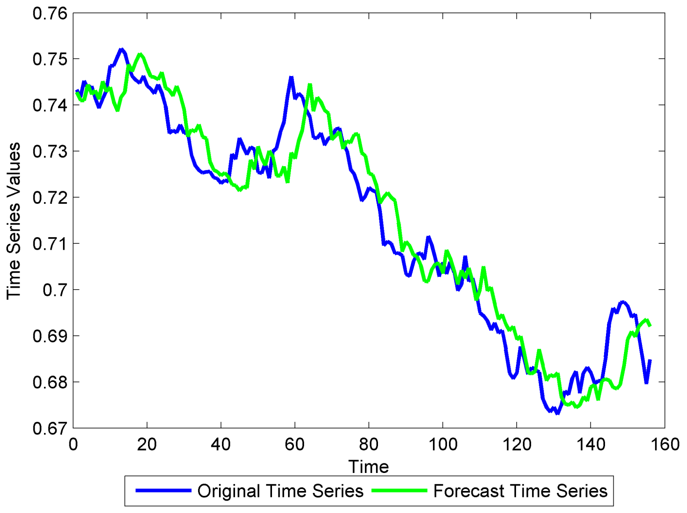


Fig. 10: Out-of-sample forecasting for EUR/USD time series based on the best DRPNN$_{Lyapunov}$ simulation

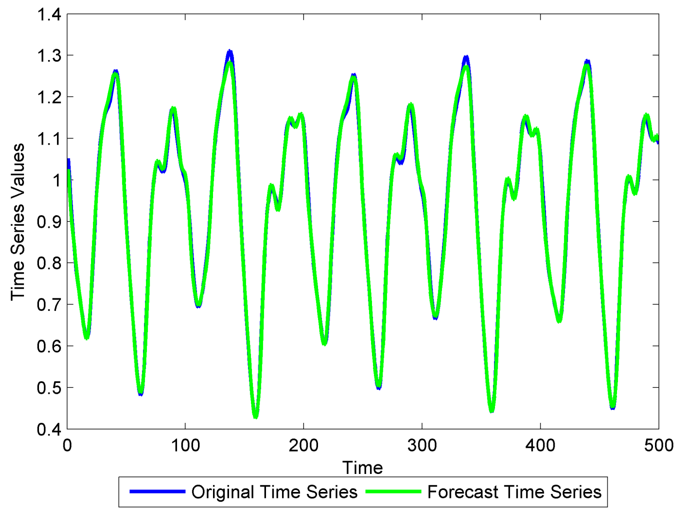


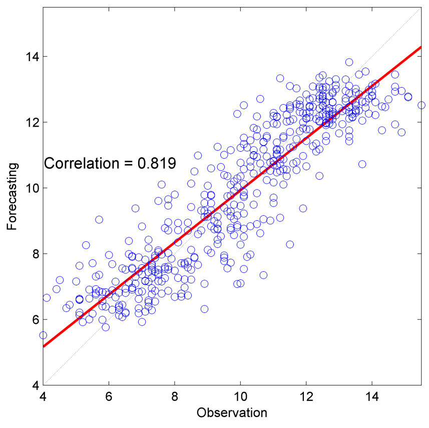
Fig. 11: Out-of-sample forecasting for Mackey-Glass time series based on the best DRPNN$_{Lyapunov}$ simulation

Limitations of DRPNN$_{Lyapunov}$

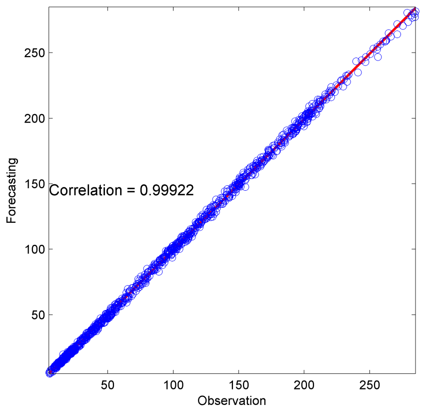
Apart from improved forecasting accuracy and training speed, there is one main limitation of DRPNN$_{Lyapunov}$ compared to DRPNN$_{feedback}$, which is network size. This is because the proposed solution solves network size restriction found in DRPNN$_{feedback}$, which stops the network from growing. This implies a trade-off between forecasting accuracy and the amount of memory. Overall, we would favor DRPNN$_{Lyapunov}$ over DRPNN$_{feedback}$ for its more accurate and faster training, even while sacrificing network size, especially for some applications such as financial or disaster forecasting that require more accurate forecasts.

Conclusion

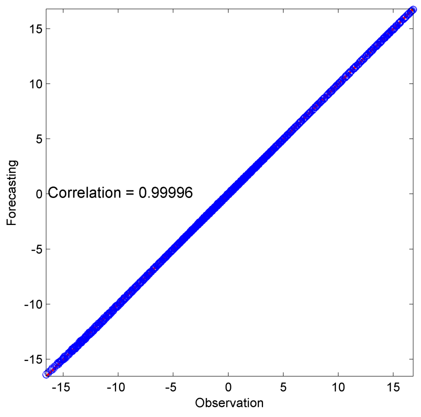
In this study, a sufficient condition based on an approach that uses adaptive learning rate was developed by introducing a Lyapunov function. This solution was proposed to tackle the problems of complexity and difficulty of training for DRPNN. This study demonstrated the effectiveness of the proposed solution by testing it on six time series. This study compared the proposed solution with the existing solution which is derived based on the stability theorem for a feedback network. Simulation results showed that the DRPNN with the



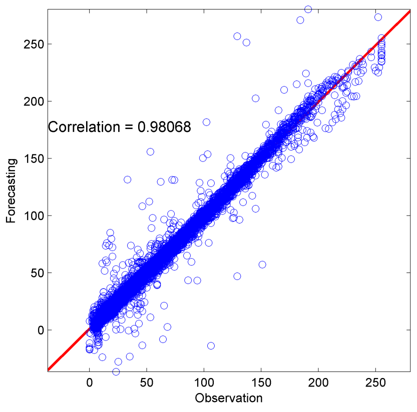
(a) Darwin sea level pressure



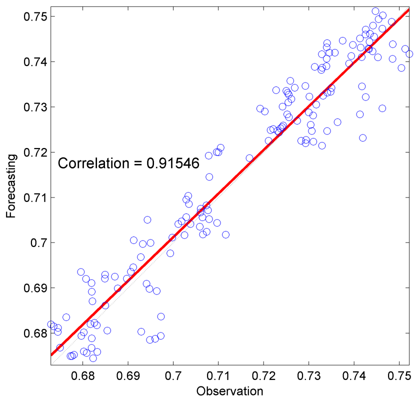
(b) Monthly smoothed sunspot numbers



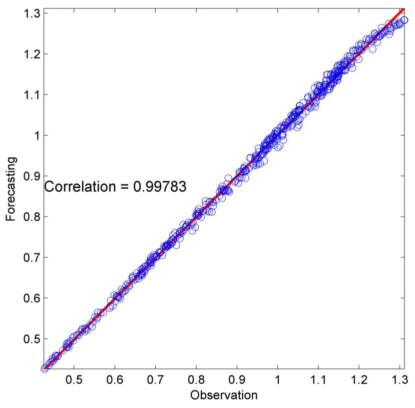
(c) Lorenz



(d) Santa Fe laser



(e) Daily Euro/Dollar exchange rate



(f) Mackey-Glass

Fig. 12: Correlation function between forecast and observed values for time series used based on the best DRPNN$_{Lyapunov}$ simulation

proposed solution improves the forecasting accuracy as well as the training speed as compared with the existing solution.

Appendix A

Theorem 1 *Let η be the learning rate parameter of the connecting weights of the $DRPNN_{Lyapunov}$. Then the convergence and stability are guaranteed if η is chosen as*

$$0 < \eta < \frac{2}{(\parallel D^Y(t) \parallel_F)^2} \tag{A.1}$$

Proof Let us define a Lyapunov function as follows:

$$V(t) = \frac{1}{2}e^2(t) \tag{A.2}$$

According to Equation (A.1), the change in the Lyapunov function can be determined by:

$$\triangle V(t) = V(t+1) - V(t) = \frac{1}{2}\left[e^2(t+1) - e^2(t)\right] \tag{A.3}$$

The error difference can be represented by [33, 53]:

$$e(t+1) = e(t) + \triangle e(t) \tag{A.4}$$

$$e(t+1) \cong e(t) + \left[\frac{\partial e(t)}{\partial w}\right]^T \triangle w \tag{A.5}$$

where $\triangle w$ represents the weight change. Since,

$$\frac{\partial e(t)}{\partial w} = -\frac{\partial y(t)}{\partial w} \tag{A.6}$$

$$D^Y(t) = \frac{\partial y(t)}{\partial w} \tag{A.7}$$

$$\triangle w = \eta * e(t) * D^Y(t) \tag{A.8}$$

Thus,

$$e(t+1) \cong e(t) - \eta * e(t) * \left[D^Y(t)\right]^T * D^Y(t) \tag{A.9}$$

$$e(t+1) \cong e(t)(1 - \eta * \left[D^Y(t)\right]^T * D^Y(t)) \tag{A.10}$$

$$e(t+1) \leq \|e(t)\|\|(1 - \eta * \left[D^Y(t)\right]^T * D^Y(t))\| \tag{A.11}$$

$$e(t+1) \leq \|e(t)\|\|(1 - \eta * \left(\parallel D^Y(t) \parallel_F\right)^2 \| \tag{A.12}$$

if η is between the bounds in Equation (A.1), the term $\|(1-\eta*\left(\parallel D^Y(t) \parallel_F\right)^2\|$ in Equation (A.12) is less than 1. Therefore, the sufficient condition to ensure stability, which is $\triangle V(t) < 0$, is guaranteed. The error will converge to zero as $t \to \infty$, which lead to a stable learning. This completes the proof of the theorem.

References

1. Darwin sea level pressure time series. URL `http://research.ics.aalto.fi/eiml/datasets/darwin.dat`. Last Accessed: 2017-01-07
2. Lorenz time series. URL `http://www.physics.emory.edu/faculty/weeks//research/tseries1.html`. Last Accessed: 2017-01-07
3. Mackey-glass time series. URL `https://raw.githubusercontent.com/dodikk/neuro-mut/master/src/NetworkConverter/Samples/mgdata.dat`. Last Accessed: 2017-01-07
4. Monthly smoothed sunspot time series. URL `http://www.sidc.be/silso/datafiles`. Last Accessed: 2017-01-07
5. Pacific exchange rate service. URL `http://fx.sauder.ubc.ca/data.html`. Last Accessed: 2017-01-07
6. Santa Fe laser time series. URL `http://www.comp-engine.org/timeseries/time-series_data_source/source-151/`. Last Accessed: 2017-01-07
7. XE currency converter. URL `http://www.xe.com/currencytables/`. Last Accessed: 2017-01-07
8. Abarbanel, H.: Analysis of observed chaotic data. Springer Science & Business Media (1996)
9. Al-Jumeily, D., Ghazali, R., Hussain, A.: Predicting physical time series using dynamic ridge polynomial neural networks. PLoS ONE **9**(8), 1–15 (2014). DOI 10.1371/journal.pone.0105766
10. Almaraashi, M., John, R.: Tuning of type-2 fuzzy systems by simulated annealing to predict time series. In: Proceedings of the World Congress on Engineering, vol. 2, pp. 976–980 (2011)
11. Atiya, A.F.: Learning on a general network. In: Neural information processing systems, pp. 22–30. American Institute of Physics (1988)
12. Atiya, A.F., Parlos, A.G.: New results on recurrent network training: unifying the algorithms and accelerating convergence. IEEE Transactions on Neural Networks **11**(3), 697–709 (2000). DOI 10.1109/72.846741
13. Behrens, H., Gawronska, D., Hollatz, J., Schurmann, B.: Recurrent and feedforward backpropagation for time independent pattern recognition. In: Neural Networks, 1991., IJCNN-91-Seattle International Joint Conference on, vol. ii, pp. 591–596 vol.2 (1991). DOI 10.1109/IJCNN.1991.155401
14. Chatterjee, S., Nigam, S., Singh, J.B., Upadhyaya, L.N.: Software fault prediction using nonlinear autoregressive with exogenous inputs (narx) network. Applied Intelligence **37**(1), 121–129 (2012). DOI 10.1007/s10489-011-0316-x
15. Chen, Y., Lin, C.T.: Dynamic parameter optimization of evolutionary computation for on-line prediction of time series with changing dynamics. Applied Soft Computing **7**(4), 1170 – 1176 (2007). DOI http://dx.doi.org/10.1016/j.asoc.2006.01.004
16. Dash, S.K., Bisoi, R., Dash, P.K.: A hybrid functional link dynamic neural network and evolutionary unscented kalman filter for short-term electricity price forecasting. Neural Computing and Applications **27**(7), 2123–2140 (2016). DOI 10.1007/s00521-015-2011-z
17. Dhahri, H., Alimi, A.: Automatic Selection for the Beta Basis Function Neural Networks, pp. 461–474. Springer Berlin Heidelberg, Berlin, Heidelberg (2008). DOI 10.1007/978-3-540-78987-1_42
18. Dhahri, H., Alimi, A.M.: The modified differential evolution and the rbf (mde-rbf) neural network for time series prediction. In: The 2006 IEEE International Joint Conference on Neural Network Proceedings, pp. 2938–2943 (2006). DOI 10.1109/IJCNN.2006.247227
19. Dong, Y., Zhang, J.: An improved boosting scheme based ensemble of fuzzy neural networks for nonlinear time series prediction. In: 2014 International Joint Conference on Neural Networks (IJCNN), pp. 157–164 (2014). DOI 10.1109/IJCNN.2014.6889431
20. Fouad, S., Tino, P.: Ordinal-based metric learning for learning using privileged information. In: The 2013 International Joint Conference on Neural Networks (IJCNN), pp. 1–8 (2013). DOI 10.1109/IJCNN.2013.6706799
21. Ghazali, R., Hussain, A.J., Al-Jumeily, D., Lisboa, P.: Time series prediction using dynamic ridge polynomial neural networks. In: 2009 Second International Conference on Developments in eSystems Engineering, pp. 354–363 (2009). DOI 10.1109/DeSE.2009.35

22. Ghazali, R., Hussain, A.J., Liatsis, P.: Dynamic ridge polynomial neural network: Forecasting the univariate non-stationary and stationary trading signals. Expert Systems with Applications **38**(4), 3765 – 3776 (2011). DOI http://dx.doi.org/10.1016/j.eswa.2010.09.037
23. Ghazali, R., Hussain, A.J., Liatsis, P., Tawfik, H.: The application of ridge polynomial neural network to multi-step ahead financial time series prediction. Neural Computing and Applications **17**(3), 311–323 (2008). DOI 10.1007/s00521-007-0132-8
24. Ghazali, R., Hussain, A.J., Nawi, N.M., Mohamad, B.: Non-stationary and stationary prediction of financial time series using dynamic ridge polynomial neural network. Neurocomputing **72**(1012), 2359 – 2367 (2009). DOI http://dx.doi.org/10.1016/j.neucom.2008.12.005
25. Gnana Jothi, R.B., Meena Rani, S.M.: Hybrid neural network for classification of graph structured data. International Journal of Machine Learning and Cybernetics **6**(3), 465–474 (2015). DOI 10.1007/s13042-014-0230-8
26. Haykin, S.: Neural networks and learning machines, vol. 3. Pearson Education Upper Saddle River (2009)
27. Huang, S.C., Chuang, P.J., Wu, C.F., Lai, H.J.: Chaos-based support vector regressions for exchange rate forecasting. Expert Systems with Applications **37**(12), 8590 – 8598 (2010). DOI http://dx.doi.org/10.1016/j.eswa.2010.06.001
28. Hussain, A., Liatsis, P.: Recurrent pi-sigma networks for DPCM image coding. Neurocomputing **55**(12), 363 – 382 (2003). DOI 10.1016/S0925-2312(02)00629-X
29. Hussain, A.J., Liatsis, P., Tawfik, H., Nagar, A.K., Al-Jumeily, D.: Physical time series prediction using recurrent pi-sigma neural networks. International Journal of Artificial Intelligence and Soft Computing **1**(1), 130–145 (2008)
30. Kitagawa, G.: Introduction to time series modeling. CRC press (2010)
31. Leclercq, E., Druaux, F., Lefebvre, D., Zerkaoui, S.: Autonomous learning algorithm for fully connected recurrent networks. Neurocomputing **63**, 25 – 44 (2005). DOI http://dx.doi.org/10.1016/j.neucom.2004.04.007. New Aspects in Neurocomputing: 11th European Symposium on Artificial Neural Networks
32. Lin, C.J., Chen, C.H., Lin, C.T.: A hybrid of cooperative particle swarm optimization and cultural algorithm for neural fuzzy networks and its prediction applications. IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews) **39**(1), 55–68 (2009). DOI 10.1109/TSMCC.2008.2002333
33. Lin, C.M., Boldbaatar, E.A.: Autolanding control using recurrent wavelet elman neural network. IEEE Transactions on Systems, Man, and Cybernetics: Systems **45**(9), 1281–1291 (2015). DOI 10.1109/TSMC.2015.2389752
34. Madhiarasan, M., Deepa, S.N.: A novel criterion to select hidden neuron numbers in improved back propagation networks for wind speed forecasting. Applied Intelligence **44**(4), 878–893 (2016). DOI 10.1007/s10489-015-0737-z
35. MATLAB: Mackey-glass time-delay differential equation (2016). URL `http://www.mathworks.com/examples/fuzzy-logic/mw/fuzzy-ex38166291-predict-chaotic-time-series`
36. Najibi, E., Rostami, H.: Scesn, spesn, swesn: Three recurrent neural echo state networks with clustered reservoirs for prediction of nonlinear and chaotic time series. Applied Intelligence **43**(2), 460–472 (2015). DOI 10.1007/s10489-015-0652-3
37. Panda, C., Narasimhan, V.: Forecasting exchange rate better with artificial neural network. Journal of Policy Modeling **29**(2), 227 – 236 (2007). DOI http://dx.doi.org/10.1016/j.jpolmod.2006.01.005
38. Parsapoor, M., Bilstrup, U.: Chaotic time series prediction using brain emotional learning–based recurrent fuzzy system (belrfs). International Journal of Reasoning-based Intelligent Systems **5**(2), 113–126 (2013)
39. Ren, Y., Suganthan, P., Srikanth, N., Amaratunga, G.: Random vector functional link network for short-term electricity load demand forecasting. Information Sciences **367368**, 1078 – 1093 (2016). DOI http://dx.doi.org/10.1016/j.ins.2015.11.039
40. Samarasinghe, S.: Neural networks for applied sciences and engineering: from fundamentals to complex pattern recognition. CRC Press (2006)
41. Sermpinis, G., Laws, J., Dunis, C.L.: Modelling commodity value at risk with psi sigma neural networks using openhighlowclose data. The European Journal of Finance **21**(4), 316–336 (2015). DOI 10.1080/1351847X.2012.744763

42. Shin, Y., Ghosh, J.: The pi-sigma network: an efficient higher-order neural network for pattern classification and function approximation. In: Neural Networks, 1991., IJCNN-91-Seattle International Joint Conference on, vol. i, pp. 13–18 vol.1 (1991). DOI 10.1109/IJCNN.1991.155142
43. Shin, Y., Ghosh, J.: Ridge polynomial networks. IEEE Transactions on Neural Networks **6**(3), 610–622 (1995). DOI 10.1109/72.377967
44. Shoorehdeli, M.A., Teshnehlab, M., Sedigh, A.K., Khanesar, M.A.: Identification using ANFIS with intelligent hybrid stable learning algorithm approaches and stability analysis of training methods. Applied Soft Computing **9**(2), 833 – 850 (2009). DOI http://dx.doi.org/10.1016/j.asoc.2008.11.001
45. Tan, J., Bong, D., Rigit, A.: Time series prediction using backpropagation network optimized by hybrid k-means-greedy algorithm. Engineering Letters **20**(3), 203–210 (2012)
46. Tikka, J., Hollmn, J.: Sequential input selection algorithm for long-term prediction of time series. Neurocomputing **71**(1315), 2604 – 2615 (2008). DOI http://dx.doi.org/10.1016/j.neucom.2007.11.037
47. Waheeb, W., Ghazali, R.: Chaotic time series forecasting using higher order neural networks. International Journal on Advanced Science, Engineering and Information Technology **6**(5) (2016)
48. Wang, H.: Modeling of nonlinear systems based on orthogonal neural network with matrix value decomposition. In: 2012 Third International Conference on Intelligent Control and Information Processing, pp. 298–301 (2012). DOI 10.1109/ICICIP.2012.6391564
49. Wang, H., Lian, J.: Fuzzy prediction of chaotic time series based on fuzzy clustering. Asian Journal of Control **13**(4), 576–581 (2011)
50. Wen, Y., Wang, H.: Fuzzy prediction of time series based on kalman filter with svd decomposition. In: 2009 Sixth International Conference on Fuzzy Systems and Knowledge Discovery, vol. 4, pp. 458–462 (2009). DOI 10.1109/FSKD.2009.133
51. Williams, R.J., Zipser, D.: A learning algorithm for continually running fully recurrent neural networks. Neural Comput. **1**(2), 270–280 (1989). DOI 10.1162/neco.1989.1.2.270
52. Wong, W., Xia, M., Chu, W.: Adaptive neural network model for time-series forecasting. European Journal of Operational Research **207**(2), 807 – 816 (2010). DOI http://dx.doi.org/10.1016/j.ejor.2010.05.022
53. Yabuta, T., Yamada, T.: Learning control using neural networks. In: Robotics and Automation, 1991. Proceedings., 1991 IEEE International Conference on, pp. 740–745 vol.1 (1991). DOI 10.1109/ROBOT.1991.131673
54. Yoo, S.J., Choi, Y.H., Park, J.B.: Generalized predictive control based on self-recurrent wavelet neural network for stable path tracking of mobile robots: adaptive learning rates approach. IEEE Transactions on Circuits and Systems I: Regular Papers **53**(6), 1381–1394 (2006). DOI 10.1109/TCSI.2006.875166
55. Zhang, H., Liu, X.N.: Local search for learning algorithm in adaptive fuzzy inference system. In: 2012 9th International Conference on Fuzzy Systems and Knowledge Discovery, pp. 93–96 (2012). DOI 10.1109/FSKD.2012.6233957