Initialization of Recurrent Networks Using Fourier Analysis
Lai-Wan CHAN and Ying-Qian ZHANG
Department of Computer Science and Engineering
The Chinese University of Hong Kong
Shatin, Hong Kong
Email: {lwchan, yqzhang}@cse.cuhk.edu.hk
http://www.cse.cuhk.edu.hk/~lwchan

Abstract
Time-Delayed recurrent neural network models preserve information through time and are more powerful than the static feedforward networks, especially in dynamic problems. At present, recurrent networks are mostly formulated as nonlinear autoregression models [4] when applied to time series prediction problem. In this paper, we use a novel approach to interpret the recurrent networks. We build the linkage between Fourier analysis and recurrent networks. The major advantage of our method is that it provides a means to initialize the weights. This initialization significantly shortens the training time.

1 Introduction
One major application area of recurrent networks is in time series prediction [11, 1, 10, 5, 3], as the feedback mechanism in recurrent networks enables the hidden units to serve as internal memories for historical information. To obtain a good prediction result, we have to find an appropriate architecture, to use an efficient learning algorithm, and to assign an initial weight vector. In the cases of feedforward networks, the randomly assigned initial weight values would not have significant effect on the training time, as long as the weight values are small. However, in recurrent networks, an inappropriate initial weight vector would not only affect the states of other units, but also the states of units in future time steps. Hence, the training time would be substantially affected by the initial weight values. The current approach to initialize the weight values of recurrent networks is to randomly assign small values. The recurrent networks would then rely mainly on first order or second order training methods to adjust the weights, and the states of hidden units are simply set to zero or any random values.

In this paper, we use Discrete Fourier Transform to formulate the prediction task and then apply this formulation to construct a new model of recurrent neural networks, called Fourier Recurrent Network (ForeNet). ForeNet, using the Fourier analysis, provides a good estimation of the network weights. The following sections will describe the derivation of ForeNet and experimental results to compare ForeNet with the existing approach.

2 Recursive form of DFT
Suppose we have a sequence of time series data \(x(1), \ldots, x(t), \ldots, x(T)\). We apply Discrete Fourier Transform (DFT) to the series and obtain the Fourier coefficients of \(c_n\) where \(n = 1, 2, \ldots, T\).

\[
c_n = \sum_{t=1}^{T} x(t) \exp(-j\omega_n t) \quad n = 1, 2, \ldots, T \quad (1)
\]

and

\[
x(t) = \frac{1}{T} \sum_{n=1}^{T} c_n \exp(j\omega_n t) \quad t = 1, 2, \ldots, T \quad (2)
\]

where \(\omega_n = \frac{2\pi n}{T}\). Define

\[
h_n(t) = \frac{1}{T} c_n \exp(j\omega_n t) \quad (3)
\]

From equations 2 and 3, we have

\[
x(t + 1) = \frac{1}{T} \sum_{n=1}^{T} c_n \exp(j\omega_n (t + 1))
\]
for \( t = 1, 2, ..., T - 1 \). Suppose we append the series by an extra value \( x(T + 1) \) and the series becomes \( x(1), x(2), ..., x(T), x(T + 1) \). The new \( T + 1 \) points DFT coefficients would be updated from \( c_n \) to

\[
c'_n = \sum_{t=1}^{T+1} x(t) \exp(-j\omega'_n t) \tag{5}
\]

for \( n = 1, 2, ..., T + 1 \) and \( \omega'_n = \frac{2\pi n}{T + 1} \). If \( T \) is sufficiently large, we approximate \( \omega'_n \) by \( \omega_n \) and

\[
c'_n \approx c_n + x(T + 1) \exp(-j\omega_n(T + 1)) \tag{6}
\]

for \( n = 1, 2, ..., T \). From equation 3, the corresponding \( h'_n(T + 1) \) becomes

\[
h'_n(T + 1) = \frac{1}{T + 1} c'_n \exp(j\omega'_n(T + 1))
= \frac{c_n \exp(j\omega_n(T + 1)) + x(T + 1)}{T + 1}
= \frac{T}{T + 1} h_n(T) \exp(j\omega_n)
+ \frac{x(T + 1)}{T + 1} \tag{7}
\]

for \( n = 1, 2, ..., T \). It suggests that the value of the function \( h'_n \) at time \( T + 1 \) can be derived from its value at previous time step \( T \) and the current time series value.

3 Fourier Recurrent Neural Networks

We have derived the recursive expression of the Discrete Fourier Transformation in the previous section. In this section, we will make use of the recursive formulation to construct our recurrent neural networks for time series prediction.

Here our input to the recurrent network is \( x(t) \) at each time step \( t \). The output units, \( z(t) \), and hidden units, \( y(t + 1) \) are governed by

\[
z(t) = \sum_{i=1}^{N} V_i y_i(t) \tag{8}
\]

\[
y_i(t) = f(\sum_{k=1}^{N} W_{ki} y_k(t - 1) + U_i x(t)) \tag{9}
\]

where \( f(\bullet) \) is the activation function of the hidden units, \( W_{ki} \) is the recurrent weights, \( U_i \) and \( V_i \) are the weights connecting the input to hidden units and the hidden to output units. \( N \) is the number of hidden units in the network.

To predict the future time series \( x(T + 1) \) based on historical observations \( x(1), x(2), ..., x(T) \), we input \( x(t) \) to the network at each time step \( t \). The network is expected to output the next step prediction \( x(t + 1) \), i.e., prediction of \( x(T + 1) \) can be obtained after we have inputted the whole sequence, \( x(t), t = 1, 2, ..., T \) to the network.

Comparing equations (9) and (8) to equations (4) and (7), we can find a direct correspondence between them by putting

- \( h_n(t) \) as \( y_i(t) \),
- \( U_i \) as \( \frac{1}{T + 1} \),
- \( W_{ik} \) as \( \frac{\exp(i\omega_n)}{T + 1} \) for \( k = i \) and
- \( W_{ik} = 0 \) for \( k \neq i \),
- \( V_i \) as \( \exp(j\omega_n) \),
- \( f(\bullet) \) as the linear activation function.

In this way, the output of the network would be \( z(t) = x(t + 1) \) which is the next step prediction of the time series, and this prediction is achieved by extrapolating equation (4) to \( t = T \). One difference of the two forms is that the DFT representation requires a summation of \( T \) terms in equation (2), whereas in a recurrent network, the output unit is evaluated using the summation of \( N \) terms, where \( N \) corresponds to the number of hidden units. It is not practical to have a recurrent network with \( N \) hidden units where \( N \) is comparable to the length of the sequence, which is supposed to be long. Fortunately, in most time-series, the component frequencies are usually with low orders. High order frequencies usually are of small magnitude and they may correspond to the undesirable noise term. Thus, we can assume that the higher frequency terms are negligible and hence we use a small and fixed number of hidden units.

In this way, we can construct a recurrent network and the corresponding weight values can be assigned as described. Another issue related to this weight initialization is the determination of the value of \( T \). In our analysis, \( T \) refers to the period of the time series. However, it is not practical to simply let \( T \) be the length of the time series. We know that the data may contain a few repeating periods of the time series. As the length of the data
gets longer, would be very small and hence $U_t$ and $W_t$. Thus, instead of making $T$ equals to the length of the data, we assign $T$ as $N$, the number of hidden units.

![Figure 1: Time-varying sinusoidal series](image1)

![Figure 2: Noisy sinusoidal series](image2)

**Figure 1:** Time-varying sinusoidal series  
**Figure 2:** Noisy sinusoidal series

A result of using the Fourier Analysis is that the parameters involved are complex number. Thus *ForeNet* is a complex recurrent network. At present, most commonly used neural network learning algorithms assumed real parameters. Previous works have been done to extend neural networks to cope with complex weights. The backpropagation algorithm for training a feedforward neural network with complex weights have been proposed [2] [6]. Georgious and Manolakos [8] used Complex Real Time Recurrent Learning algorithm to train a fully recurrent network. Using the complex parameters to construct neural network avoids the problem of the standstill in learning [9], and can also deal with dynamic time-sequential signal more stably and smoothly than the conventional recurrent networks [7]. In this paper, *ForeNet* was trained by Complex Real Time Recurrent Learning algorithm (CRTRL), which combines [3] and [8].

![Figure 3: Laser Series](image3)

**Figure 3:** Laser Series

### 4 Experimental Results

We have proposed an initialization method which is based on Discrete Fourier Transform. In this section, we demonstrate the properties of *ForeNet* by some experiments. We show the efficiency of such initialization for network training. The architecture of *ForeNet* is a 1-4-1 network, i.e., the network has one input unit, one output unit and 4 hidden units. The learning rate $\eta$ is set to 0.1 and the momentum coefficient $\alpha$ is 0.5. We used the root mean square error (RMS) to monitor prediction performance of the network. The whole data sequence is partitioned into three parts: the first is the training data set; the second part is the validation set, which determines the stopping condition of the training process; the last is the testing data set.

While the optimal network parameters have been determined, the initial error is substantially small and the number of iterations required to achieve the convergence has also been significantly reduced. We perform the prediction tasks using *ForeNet* and compare the result with *ForeNet* with randomly initialized parameters.

### 4.1 The Data Set

Here we have used four prediction tasks in this paper.

- Mackey-Glass series generated from the following equation. The series includes 1000 data. The first 400 data were used for training. The following 300 points were for validation and the remaining points were for test-
\[ \frac{\partial x(t)}{\partial t} = -bx(t) + a \frac{x(t - \tau)}{1 + x(t - \tau)^2} \] (10)

- Time-varying sinusoidal series (Figure 1), which is generated from the equation

\[ y(t) = \sin(w_t t) \quad t = 1, 2, \ldots, n \]

where \( w_t = 0.008 t \) is the frequency varying with time. \( n \) is set to 600, and the first 300 data were used for training. This is a non-stationary time series and is used to test the performance of the network towards gradually changing series.

- The sinusoidal series with noise generated from the equation

\[ y(t) = \sin(w t) + \sigma \]

where a uniformly distributed random value \( \sigma \) in the interval \([-0.5, 0.5]\) is added to the sin function at each time step. \( w = 1 \), \( t \) ranges from 1 to 200. The first 100 points were used as training data. The next 60 data were used for validation. The remaining 40 points were testing data. Figure 2 shows the series.

- The laser series from the Santa Fe time series competition. The series is illustrated in Figure 3. The network was trained to perform a one step ahead prediction. All samples were scaled to zero mean and unit variance. There are 1500 points in the laser series. The first 900 data were the training data and the next 300 were the validation data. After training, we performed the forecasting on the remaining 300 data.

4.2 The Comparison

Tables 1, 4, 3 and 2 show the experimental results for the four data sets respectively. In Mackey-Glass prediction (Table 1), the proposed method generates good initial parameters, which produce smaller initial training error (RMS error = 0.0205) than that using randomly setting initial values (RMS error = 2.2589). The network needs only only 13 iterations to converge and with better generalization. The learning curves of both initialization methods are shown in Figure 4 and Figure 5 respectively. Figure 4 shows that the training error starts with

<table>
<thead>
<tr>
<th>model</th>
<th>Initial training error</th>
<th>Testing RMS error</th>
<th>No. of iterations</th>
<th>Time taken (sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>0.0205</td>
<td>0.0102</td>
<td>13</td>
<td>63</td>
</tr>
<tr>
<td>B</td>
<td>2.2589</td>
<td>0.0558</td>
<td>200</td>
<td>866</td>
</tr>
<tr>
<td>C</td>
<td>0.6455</td>
<td>0.0601</td>
<td>500</td>
<td>2117</td>
</tr>
</tbody>
</table>

Table 1: Mackey-Glass series. A: ForeNet with proposed initialization, B: ForeNet without random initialization, C: real-valued RNN with random initialization.

<table>
<thead>
<tr>
<th>model</th>
<th>Initial training error</th>
<th>Testing RMS error</th>
<th>No. of iterations</th>
<th>Time taken (sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>0.7156</td>
<td>0.0176</td>
<td>19</td>
<td>24</td>
</tr>
<tr>
<td>B</td>
<td>0.6889</td>
<td>0.4285</td>
<td>52</td>
<td>60</td>
</tr>
<tr>
<td>C</td>
<td>0.5450</td>
<td>0.3267</td>
<td>500</td>
<td>575</td>
</tr>
</tbody>
</table>

Table 2: Noisy sinusoidal series. A: ForeNet with proposed initialization, B: ForeNet without random initialization, C: real-valued RNN with random initialization.

<table>
<thead>
<tr>
<th>model</th>
<th>Initial training error</th>
<th>Testing RMS error</th>
<th>No. of iterations</th>
<th>Time taken (sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>0.0762</td>
<td>0.2250</td>
<td>73</td>
<td>180</td>
</tr>
<tr>
<td>B</td>
<td>0.7394</td>
<td>0.3359</td>
<td>71</td>
<td>178</td>
</tr>
<tr>
<td>C</td>
<td>0.4947</td>
<td>0.3180</td>
<td>87</td>
<td>198</td>
</tr>
</tbody>
</table>

Table 3: Time-varying sine series. A: ForeNet with proposed initialization, B: ForeNet without random initialization, C: real-valued RNN with random initialization.

<table>
<thead>
<tr>
<th>model</th>
<th>Initial training error</th>
<th>Testing RMS error</th>
<th>No. of iterations</th>
<th>Time taken (sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>0.7449</td>
<td>0.5049</td>
<td>19</td>
<td>142</td>
</tr>
<tr>
<td>B</td>
<td>1.1961</td>
<td>0.4784</td>
<td>200</td>
<td>1324</td>
</tr>
<tr>
<td>C</td>
<td>0.6680</td>
<td>0.3906</td>
<td>300</td>
<td>1868</td>
</tr>
</tbody>
</table>

Table 4: Laser data. A: ForeNet with proposed initialization, B: ForeNet without random initialization, C: real-valued RNN with random initialization.
a very small value and then further training provides slight improvement. In Figure 5, we see that the initial RMS error is large and long training time was required. The training error was still around 0.05 after 200 iterations.

![Figure 4: The learning curve of ForeNet with proposed initialization using Mackey-Glass data.](image)

Figure 4: The learning curve of ForeNet with proposed initialization using Mackey-Glass data.

![Figure 5: The learning curve of ForeNet with random initialization using Mackey-Glass data.](image)

Figure 5: The learning curve of ForeNet with random initialization using Mackey-Glass data.

The parameters in ForeNet are complex-valued and ForeNet is trained by complex-valued learning algorithm. As we known, a complex number provides not only the magnitude information but the phase state. Therefore a neuron with complex value has more representational power than a real-valued neuron. Moreover, Akira proved the behavior of complex-valued recurrent network is more stable than real-valued recurrent network[7]. For comparison, we train a recurrent network with randomly generated real-valued weights. Comparing to two previous networks, it is found that the complex-valued network has more powerful computational ability in terms of testing error and training time.

Figures 6 and 7 show the network output with proposed initialization on the time-varying sinusoidal series training set, before training and after training respectively. We can see that the network output is very close to the target values even before training has been taken place. Further training only provides small fine tune to the results. Figure 8 shows the corresponding ForeNet's output with random initialization. Although this data set is a non-stationary series, our initialization method is able to deal with this time-varying signals. Table 3 indicates that the training time of the three networks are similar. However, the testing RMS error of ForeNet is the smallest one.

From the prediction of noisy sinusoidal series (Table 2), we find that the proposed initialization method has worse performance in terms of initial root mean square error than random initialization. However, compared with the random initialization method, which takes 52 training iterations to converge, the proposed method uses only 19 iterations to achieve much better generalization according to the final RMS error. It is interesting to see that the model gets fast convergence speed even with large initial error. It has to be noted that RMS error measures network performance, but it shows no indication of the closeness to the optimal point in the weight space. Thus, small initial error does not necessarily imply better initialization. Efficient initialization methods should be those assigning the weights in the region not far away from the minimum. In addition, the region should not be flat, if training is carried out by gradient methods. The path from the region to the minimum should not be blocked by high energy barriers.

![Figure 6: Time-varying sinusoidal series with proposed initialization before training.](image)

Figure 6: Time-varying sinusoidal series with proposed initialization before training.

![Figure 7: Time-varying sinusoidal series with proposed initialization after training.](image)

Figure 7: Time-varying sinusoidal series with proposed initialization after training.

![Figure 8: Time-varying sinusoidal series with random initialization before training.](image)

Figure 8: Time-varying sinusoidal series with random initialization before training.

Table 4 shows the result for laser data and it indicates that the real-valued RNN produces the smallest testing error. By inspecting the network out-
puts of the ForeNet (Figure 10) and real-valued RNN (Figure 9), we are unable to observe very significant discrepancy. However, the training of the ForeNet was stopping after 19 iterations, which is much shorter than the other methods.

![Figure 9: Laser Data predicted by the real-valued RNN.](image1)

![Figure 10: Laser Data predicted by ForeNet.](image2)

5 Conclusions

In this paper we derived the link between Fourier analysis and recurrent networks and proposed the Fourier recurrent networks, ForeNet, for time series prediction. ForeNet has an major advantage. It makes use of the Fourier Transform of the time series to initialize the weight parameters. The initialized configuration is very close to the final solution. We have applied ForeNet to predict some benchmark data, such as Mackey Glass data, time-varying sinusoidal series, noisy sinusoidal series and laser data. ForeNet gives an accurate prediction result, even without the training of the weights. Further improvement can be made if we apply the training algorithm to fine tune the weights.

Acknowledgement The authors would like to thank The Research Grants Council of the Hong Kong Special Administration Region for funding support.

References


