System Modeling by Ultra High Frequency Sigmoid and Sine Artificial Higher Order Neural Networks

Ming Zhang
Department of Physics, Computer Science and engineering
Christopher Newport University
Newport News, VA, USA
Email: mzhang@cnu.edu

***Abstract* — New open box and nonlinear system model of Ultra High Frequency Sigmoid and Sine Artificial Higher Order Neural Network (UGS-HONN) is presented in this paper. A new learning algorithm for UGS-HONN is also developed from this study. A time series data modeling system, UGS-HONN Simulator, is built based on the UGS-HONN system models too. Test results show that average error of UGS-HONN system models are closing to zero (10-6). The average errors of Polynomial Higher Order Neural Network (PHONN), Trigonometric Higher Order Neural Network (THONN), and Sigmoid polynomial Higher Order Neural Network (SPHONN) models are from 2.8128% to 4.9076%. It means that UGS-HONN system models are 2.8128% to 4.9076% better than PHONN, THONN, and SPHONN models.**

Keywords-HONN; PHONN; THONN; SPHONN;UGS-HONN

Full/Regular Research Papers

CSIS-ISCI

I. Introduction and Motivations

Artificial Higher Order Neural Network (HONN) has a lot of applications in different areas. Barron, Gilstrap, and Shrier [1] develop polynomial and neural networks for analogies and engineering system modeling applications. An, Mniszewski, Lee, Papcun, and Doolen [2] test a learning procedure, based on a default hierarchy of high-order neural networks, which exhibited an enhanced capability of generalization and a good efficiency to learn to read English. Mao, Selviah, Tao, and Midwinter [3] design a holographic high order associative memory system in holographic area. However, all studies above use traditional artificial neural network models - black box models that did not provide users with a function that describe the relationship between the input and output. The first motivation of this paper is to develop nonlinear "open box" neural network system models that will provide rationale for network's decisions, also provide better results.

Lopez-Franco, Alanis, Arana-Daniel, and Lopez-Franco [4] study a recurrent higher order neural network (RHONN) to identify the plant model of discrete time nonlinear systems, under the assumption that all the state is available for measurement. Then the Extended Kalman Filter (EKF) is used to train the RHONN. The applicability of this scheme is illustrated by identification for an electrically driven nonholonomic mobile robot. Traditionally, system modeling of mobile robots only considers its kinematics. It has been well-known that the actuator dynamics is an important part of the design of the complete robot dynamics. However, most of the reported results in literature do not consider all parametric uncertainties for mobile robots at the actuator level. This is due to the system modeling problem would become extremely difficult as the complexity of the system dynamics increases and when the mobile robot model includes the uncertainties of the actuator dynamics as well as the uncertainties of the robot kinematics and dynamics.

Ding [5] uses high order Hopfiled network, as an expansion of traditional Hopfield network, to solve combinatorial optimization problems. In theory, compared with low order network, high order network has better properties, such as stronger approximation property and faster convergence rate. In this paper we focus on how to use high order network to system model combinatorial optimization problems. Firstly, the high order discrete Hopfield Network is introduced, then we discuss how to find the high order inputs of a neuron. Finally, the construction system method of energy function and the neural computing algorithm are presented

Fallahnezhad and Yousefi [6] suggest that precise insertion of medical needle as end-effecter of robotic or computer-aided system model into the biological tissue is an important issue which should be considered in different operations such as brain biopsy, prostate brachytherapy and percutaneous therapies. Proper understanding of the whole procedure leads to a better performance by operator or system modeling. In this study, the authors use a 0.98mm diameter needle with a real-time recording of force, displacement, and velocity of needle through biological tissue during in-vitro insertions. Using constant velocity experiments from 5mm/min up to 300mm/min the data set for the force-displacement graph of insertion gathered. Tissue deformation with a small puncture and a constant velocity penetration are the two first phases in needle insertion process. Direct effects of different parameters and their correlations during the process, is being modeled using a polynomial neural network. The authors develop different networks in 2nd and 3rd orders to model two first phases of insertion, separately. Modeling accuracies were 98%, and 86% in phase 1 and 2, respectively.

Gupta, Bukovsky, Homma, Solo, and Hou [7] provide fundamental principles of higher order neural units (HONUs) and higher order neural networks (HONNs) for system modeling and simulation. An essential core of HONNs can be found in higher order weighted combinations or correlations between the input variables and HONU. Except the high quality of nonlinear approximation of static HONUs, the capability of dynamic HONUs for modeling of dynamic systems is shown and compared to conventional recurrent neural networks when a practical learning algorithm is used. Also, the potential of continuous dynamic HONUs to approximate high dynamic-order systems is discussed as adaptable time delays can be implemented.

Granger and Bates [8] researched the combination of forecasts. Granger and Weiss [9] showed the importance of cointegration in the modeling of nonstationary economic series. Granger and Lee [10] studied multi-cointegration. Granger and Swanson [11] further developed multi-cointegration in studying of cointegrated variables. The second motivation of this paper is to develop a new nonlinear HONN model, which average error should reach to 0.0000% (10^{-6}).

Psaltis, Park, and Hong [12] studied higher order associative memories and their optical implementations. Redding, Kowalczyk, Downs [13] developed constructive high-order network algorithm. Zhang, Murugesan, and Sadeghi [14] developed a Polynomial Higher Order Neural Network (PHONN) model for data simulation. The idea first extended to PHONN Group models (Zhang, Fulcher, and Scofield, [15]), then to Trigonometric Higher Order Neural Network (THONN) models for data simulation (Zhang, Zhang, and Keen, [16]). Zhang, Zhang, and Fulcher [17] studied HONN group model for data simulation. By utilizing adaptive neuron activation functions, Zhang, Xu, and Fulcher [18] developed a neuron adaptive HONN. Zhang and Fulcher [19] provide detail mathematics for THONN models. Zhang [20] published a HONN book, where all 22 chapters focused on artificial higher order neural networks for economics and business. Zhang [21] found that HONN can simulate non-continuous data with better accuracy than SAS NLIN (non-linear) models. Zhang [22] developed Ultra High Frequency Trigonometric Higher Order Neural Networks, in which model details of UCSHONN (Ultra High Frequency Cosine and Sine Higher Order Neural Network) was given. Two other books of HONN have been edited by Zhang [23] [24], in which the HONN application examples in computer science, computer engineering, modeling, and simulation areas are collected. The third motivation of this paper is to develop a new nonstationary data modeling system by using new generation computer techniques that will improve the accuracy of the data simulation.

The contributions of this paper will be:

- Review the history of how open box and nonlinear HONN models have been developed (section I).
- Present a new open box and nonlinear model – UGS-HONN (Section II), which average error could be close to zero (10^{-6}).
- Based on the UGS-HONN models, build a time series modeling system – UGS-HONN simulator (Section III).
- Develop the UGS-HONN learning algorithm and weight update formulae (Section IV).
- Shows that UGS-HONN can do better than PHONN, THONN, and SPHONN models. (Section V). UGS-HONN average error can reach to 0.0000% (10^{-6}).

II. Models of UGS-HONN

The Nyquist–Shannon sampling theorem, after Harry Nyquist and Claude Shannon, in the literature more commonly referred to as the Nyquist sampling theorem or simply as the sampling theorem, is a fundamental result in the field of information theory, telecommunications, and signal processing. Shannon's version of the theorem states:[25]

If a function x(t) contains no frequencies higher than B hertz, it is completely determined by giving its ordinates at a series of points spaced 1/(2B) seconds apart. In other words, a band limited function can be perfectly reconstructed from a countable sequence of samples if the band limit, B, is no greater than ½ the sampling rate (samples per second).

Modeling and predicting time series data, the new nonlinear models of UGS-HONN should have twice as high frequency as that of the ultra-high frequency of the time series data. To achieve this purpose, a new model should be developed to enforce high frequency of HONN in order to make the simulation and prediction error close to zero.

Figure 1 shows the UGS-HONN Architecture. This model structure is used to develop the model learning algorithm, which make sure the convergence of learning. This allows the deference between desired output and real output of UGS-HONN close to zero. Formula 1, 2, and 3 are for UGS-HONN model 2, 1 and 0 respectively. Model 2 has three layers of weights changeable. Model 1 has two layers of weights changeable. And model 0 has one layer of weights changeable.

For models 2, 1 and 0, Z is the output while x and y are the inputs of UGS-HONN. $c_{kj}{}^{o}$ is the weight for the output layer, $c_{kj}{}^{hx}$ and $c_{kj}{}^{hy}$ are the weights for the second hidden layer, and $c_k{}^{x}$ and $c_j{}^{y}$ are the weights for the first hidden layer. Functions sigmoid and sine are the first hidden layer nodes of UGS-HONN. The nodes of the second hidden layer are multiplication neurons. The output layer node of UGS-HONN is a linear function of $f^o(net^o) = net^o$, where net^o equals the input of output layer node. UGS-HONN is an open neural network model, each weight of HONN has its corresponding coefficient in the model formula, and each node of UGS-HONN has its corresponding function in the model formula. The structure of UGS-HONN is built by a nonlinear formula. It means, after training, there is rationale for each component of UGS-HONN in the nonlinear formula.

$$UGS-HONN \quad Model \quad 2:$$

$$Z = \sum_{k,j=0}^{n} (c_{kj}{}^{o})\{c_{kj}{}^{hx}(1/(1+\exp(c_k{}^{x}(-x))))^k\}\{c_{kj}{}^{hy}(\sin^j(c_j{}^{y} * j * y))\}$$

Formula 2:

$$UGS-HONN \quad Model \quad 1:$$

$$z = \sum_{k,j=0}^{n} (c_{kj}{}^{o} \quad (1/(1+\exp(c_k{}^{x}(-x))))^k (\sin^j(c_j{}^{y} * j * y)))$$

$$where: \quad (c_{kj}{}^{hx}) = (c_{kj}{}^{hy}) = 1$$

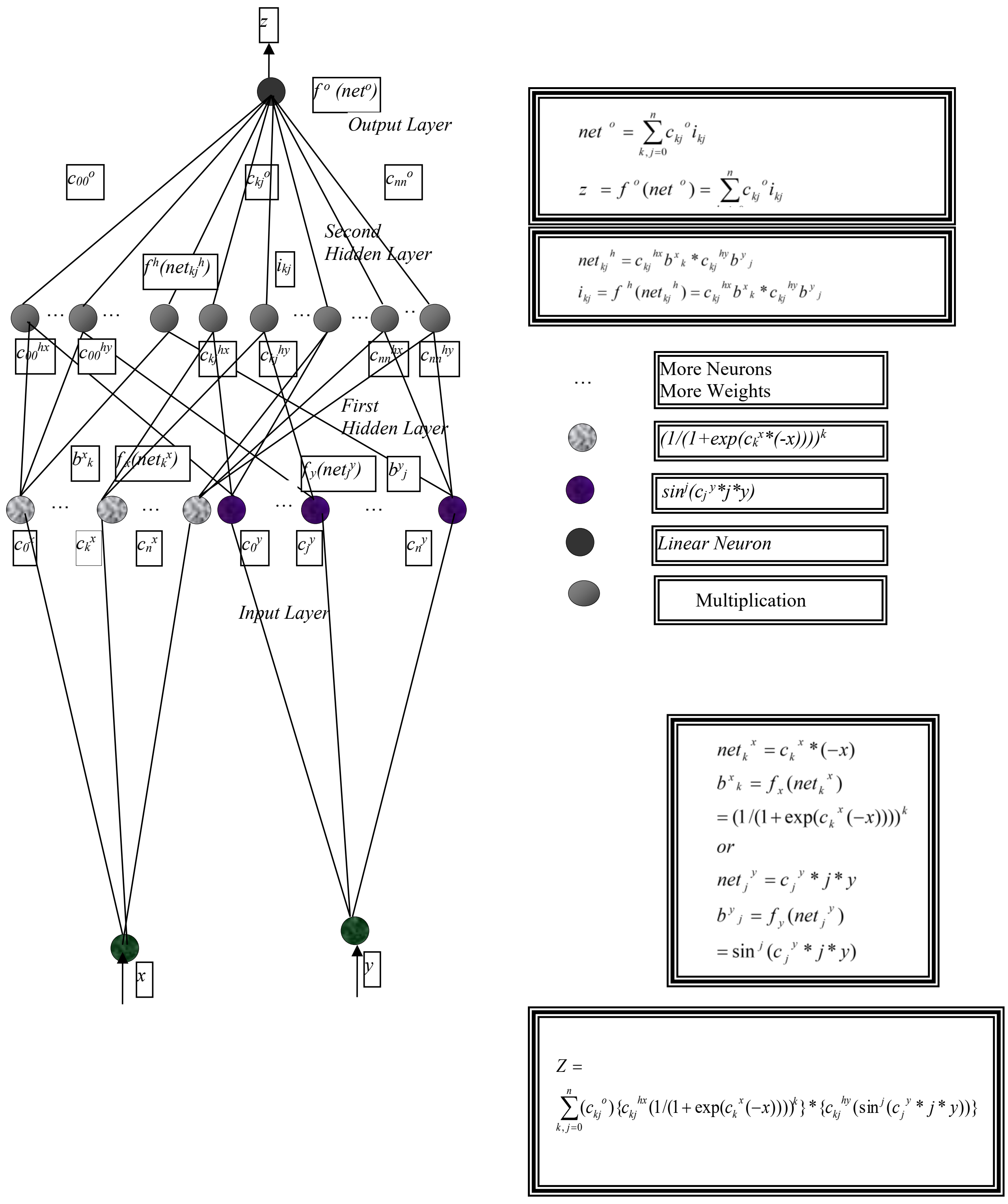
Formula 3:

$$UGS-HONN \quad Model \quad 0:$$

$$z = \sum_{k,j=0}^{n} (c_{kj}{}^{o} \quad (1/(1+\exp(-x)))^k (\sin^j(j * y)))$$

$$where: \quad (c_{kj}{}^{hx}) = (c_{kj}{}^{hy}) = 1$$

$$and \quad c_k{}^{x} = c_j{}^{y} = 1$$



$$net^o = \sum_{k,j=0}^{n} c_{kj}{}^o i_{kj}$$

$$z = f^o(net^o) = \sum^{n} c_{kj}{}^o i_{kj}$$

$$net_{kj}{}^h = c_{kj}{}^{hx} b^x{}_k * c_{kj}{}^{hy} b^y{}_j$$

$$i_{kj} = f^h(net_{kj}{}^h) = c_{kj}{}^{hx} b^x{}_k * c_{kj}{}^{hy} b^y{}_j$$

$$net_k{}^x = c_k{}^x * (-x)$$
$$b^x{}_k = f_x(net_k{}^x)$$
$$= (1/(1+\exp(c_k{}^x(-x))))^k$$
$$or$$
$$net_j{}^y = c_j{}^y * j * y$$
$$b^y{}_j = f_y(net_j{}^y)$$
$$= \sin^j(c_j{}^y * j * y)$$

$$Z = \sum_{k,j=0}^{n}(c_{kj}{}^o)\{c_{kj}{}^{hx}(1/(1+\exp(c_k{}^x(-x))))^k\} * \{c_{kj}{}^{hy}(\sin^j(c_j{}^y * j * y))\}$$

Figure 1. UGS-HONN Architecture

Formula 1:

Formula 4:
Sigmoid and Sine HONN Model:

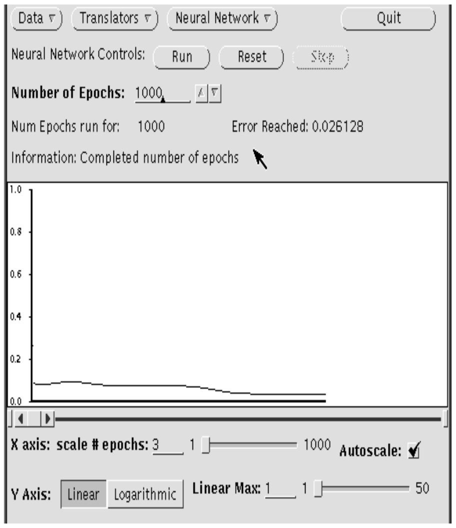
$$Z=\sum_{k,j=0}^{n}(c_{kj}{}^{o})\{c_{kj}{}^{hx}(1/(1+\exp(c_k{}^{x}(-x))))^{k}\}\{c_{kj}{}^{hy}(\sin^{j}(c_j{}^{y}*y))\}$$

Formula 4 is Sigmoid and Sine Artificial Higher Order Neural Network model. The only different between Formula 1 and Formula 4 is that, in Formula 1, the Sine function has j integer in it. It is $\sin^j(c_j{}^y*j*y)$. But in Formula 4, the sine function has no j integer in it. That is $\sin^j(c_j{}^y*y)$.

For formula 1, 2, and 3, values of k and j ranges from 0 to n, where n is an integer. Based on the Nyquist–Shannon sampling theorem, the UGS-HONN model can simulate high frequency data, when n increases to a big number. This property of the model allows it to easily simulate and predicate high frequency time series data, since both k and j increase when there is an increase in n. Because of the integer j, Formula 1 of UGS-HONN can simulate ultra-high frequency data and let average error of UGS-HONN reaches to 0.0000%. This is the key reason why UGS-HONN is better than other tradition HONN models of PHONN, THONN, and SPHONN. This is also the reason why average errors of tradition HONN usually are 2 to 5 %. It is very hard to reach zero of average error for PHONN, THONN, and SPHONN.

III. UGS-HONN Time Series Modeling System

The UGS-HONN simulator is written in C language, runs under X window on Sun workstation, based on previous work by Zhang, Fulcher, Scofield [11]. A user-friendly *GUI* (Graphical User Interface) system has also been incorporated. When you run the system, any step, data or calculation can be reviewed and modified from different windows during processing. Hence, changing data, network models and comparing results can be done very easily and efficiently. UGS-HONN simulator GUI is shown in Figure 2.



Number of Epochs: 1000

Num Epochs run for: 1000 Error Reached: 0.026128

Information: Completed number of epochs

X axis: scale # epochs: 3 1 1000 Autoscale: ✓

Linear Max: 1 1 50

Figure 2. UGS-HONN Simulator

IV. Learning Algorithm of UGS-HONN

Learning Algorithm of UGS-HONN Model can be described as followings. The 1st hidden layer weights are updated according to:

$$c_k{}^{x}(t+1)=c_k{}^{x}(t)-\eta(\partial E_p/\partial c_k{}^{x})$$

$$c_j{}^{y}(t+1)=c_j{}^{y}(t)-\eta(\partial E_p/\partial c_j{}^{y})$$

Where:
$c_k{}^x$ = 1st hidden layer weight for input x
k = kth neuron of first hidden layer
$c_j{}^y$ = 1st hidden layer weight for input y
j = jth neuron of first hidden layer
η = learning rate (positive & usually < 1)
E_p = error
t = training time

The equations for the kth or jth node in the first hidden layer are:

$$net_k{}^{x}=c_k{}^{x}*(-x)$$
$$b^{x}{}_{k}=f_x(net_k{}^{x})=(1/(1+\exp(c_k{}^{x}*(-x))))^{k}$$
$$or$$
$$net_j{}^{y}=c_j{}^{y}*j*y$$
$$b^{y}{}_{j}=f_y(net_j{}^{y})=\sin^{j}(c_j{}^{y}*j*y)$$

Where:
$b^x{}_k$ and $b^y{}_j$= output from the 1st hidden layer neuron (= input to 2nd hidden layer neuron)
f_x and f_y= 1st hidden layer neuron activation function
x and y = input to 1st hidden layer

$$net_{kj}{}^{h}=c_{kj}{}^{hx}b^{x}{}_{k}*c_{kj}{}^{hy}b^{y}{}_{j}$$
$$i_{kj}=f^{h}(net_{kj}{}^{h})=c_{kj}{}^{hx}b^{x}{}_{k}*c_{kj}{}^{hy}b^{y}{}_{j}$$

Where:
i_{kj} = output from 2nd hidden layer (= input to the output neuron)

$$net^{o}=\sum_{k,j=0}^{n}c_{kj}{}^{o}i_{kj}$$

$$z=f^{o}(net^{o})=\sum_{k,j=0}^{n}c_{kj}{}^{o}i_{kj}$$

Where:
Z is output of HONN.

The total error is the sum of the squared errors across all hidden units, namely:

$$
\begin{aligned}
E_p &= 0.5*\delta^2 = 0.5*(d-z)^2 \\
&= 0.5*(d - f^o(net^o))^2 \\
&= 0.5*(d - f^o(\sum_j c_{kj}{}^o i_{kj}))^2
\end{aligned}
$$

Where:

d is actual output of HONN

For sigmoid function and sine function as in the first layer of HONN:

$$
\begin{aligned}
b^x{}_k &= f_x\ (net_k{}^x) = (1/(1+\exp(net_k{}^x)))^k \\
f_x\ '(net_k{}^x) &= \partial b^x{}_k / \partial(net_k{}^x) \\
&= \partial((1/(1+\exp(net_k{}^x)))^k)/\partial(net_k{}^x) \\
&= -k*\exp(net_k{}^x)*(1/(1+\exp(net_k{}^x)))^{k+1}
\end{aligned}
$$

$$
\begin{aligned}
b^y{}_j &= f_y\ (net_j{}^y) = \sin^j(net_j{}^y) \\
f_y\ '(net_j{}^y) &= \partial b^y{}_j / \partial(net_j{}^y) \\
&= \partial(\sin^j(net_j{}^y))/\partial(net_j{}^y) \\
&= j*\sin^{j-1}(net_j{}^y)*\cos(net_j{}^y)
\end{aligned}
$$

$$
\begin{aligned}
&-\partial E_p/\partial c_k{}^x \\
&= (d-z) f^o{}'(net^o) c_{kj}{}^o * f^h{}'(net_{kj}{}^h)\delta_{kj}{}^{hx} c_{kj}{}^{hx} f_x{}'(net_k{}^x) x
\end{aligned}
$$

The gradient ($\partial E_p / \partial c_k{}^x$) is given by:

$$
\begin{aligned}
\partial E_p/\partial c_k{}^x &= \partial(0.5*(d-z)^2)/\partial c_k{}^x \\
&= (\partial(0.5*(d-z)^2)/\partial z)(\partial z/\partial(net^o)) \\
&(\partial(net^o)/\partial i_{kj})(\partial i_{kj}/\partial(net_{kj}{}^h))(\partial(net_{kj}{}^h)/\partial b^x{}_k) \\
&(\partial b^x{}_k/\partial(net_k{}^x))(\partial(net_k{}^x)/\partial c_k{}^x)
\end{aligned}
$$

$$
\partial(0.5*(d-z)^2/\partial z \ = -(d-z)
$$

$$
\partial z/\partial(net^o) = \partial f^o/\partial(net^o) = f^o{}'(net^o)
$$

$$
\partial(net^o)/\partial i_{kj} = \partial(\sum_{k,j=1}^{L}(c_{kj}{}^o i_{kj}))/\partial i_{kj} = c_{kj}{}^o
$$

$$
\begin{aligned}
&\partial i_{kj}/\partial(net_{kj}{}^h) \\
&= \partial(f^h(net_{kj}{}^h))/\partial(net_{kj}{}^h) = f^h{}'(net_{kj}{}^h)
\end{aligned}
$$

$$
\begin{aligned}
&\partial net_{kj}{}^h / \partial b^x{}_k \\
&= \partial((c_{kj}{}^{hx} * b^x{}_k)*(c_{kj}{}^{hy} * b^y{}_j))/\partial b^x{}_k = c_{kj}{}^{hx} * c_{kj}{}^{hy} * b^y{}_j \\
&= \delta_{kj}{}^{hx} c_{kj}{}^{hx} \\
&where: \delta_{kj}{}^{hx} = c_{kj}{}^{hy} * b^y{}_j
\end{aligned}
$$

$$
\partial b^x{}_k / \partial(net_k{}^x) = f_x{}'(net_k{}^x) = k(c_k{}^x * x)^{k-1}
$$

$$
\partial(net_k{}^x)/\partial c_k{}^x = \partial(c_k{}^x * x)/\partial c_k{}^x = x
$$

By combining above formulae, the learning algorithm for the 1st hidden layer weights of x input neuron are:

$$
\begin{aligned}
c_k{}^x(t+1) &= c_k{}^x(t) - \eta(\partial E_p/\partial c_k{}^x) \\
&= c_k{}^x(t) + \eta(d-z) f^o{}'(net^o) c_{kj}{}^o * f^h{}'(net_{kj}{}^h)\delta_{kj}{}^{hx} c_{kj}{}^{hx} f_x{}'(net_k{}^x) x \\
&= c_k{}^x(t) + \eta * \delta^{ol} * c_{kj}{}^o * \delta^{hx} * c_{kj}{}^{hx} \\
&\qquad *(-k)*\exp(net_k{}^x)*(1/(1+\exp(net_k{}^x)))^{k+1} * x \\
&= c_k{}^x(t) + \eta * \delta^{ol} * c_{kj}{}^o * \delta^{hx} * c_{kj}{}^{hx} * \delta^x * x
\end{aligned}
$$

$$
\begin{aligned}
&where: \\
&\quad \delta^{ol} = (d-z) f^o{}'(net^o) = d-z \qquad (linear) \\
&\quad \delta^{hx} = f^h{}'(net_{kj}{}^h)\delta_{kj}{}^{hx} \qquad = c_{kj}{}^{hy} b^y{}_j \qquad (linear) \\
&\quad \delta^x = f_x{}'(net_k{}^x) = (-k)*\exp(net_k{}^x)*(1/(1+\exp(net_k{}^k)))^{k+1}
\end{aligned}
$$

Using the above procedure, the learning algorithm for the 1st hidden layer weights of y input neuron is:

$$
\partial(net_j{}^y)/\partial c_j{}^y = \partial(c_j{}^y * j * y)/\partial c_j{}^y = j*y
$$

$$
\begin{aligned}
c_j{}^y(t+1) &= c_j{}^y(t) - \eta(\partial E_p/\partial c_j{}^y) \\
&= c_j{}^y(t) + \eta(d-z) f^o{}'(net^o) c_{kj}{}^o * f^h{}'(net_{kj}{}^h)\delta_{kj}{}^{hy} c_{kj}{}^{hy} f_y{}'(net_j{}^y) jy \\
&= c_j{}^y(t) + \\
&\quad \eta * \delta^{ol} * c_{kj}{}^o * \delta^{hy} * c_{kj}{}^{hy} * (j*\sin^{j-1}(net_j{}^y)*\cos(net_j{}^y)) * j * y \\
&= c_j{}^y(t) + \eta * \delta^{ol} * c_{kj}{}^o * \delta^{hy} * c_{kj}{}^{hy} * \delta^y * j * y
\end{aligned}
$$

$$
\begin{aligned}
&where: \\
&\quad \delta^{ol} = (d-z) f^o{}'(net^o) = d-z \qquad (linear) \\
&\quad \delta^{hy} = f^h{}'(net_{kj}{}^{hy}) c_{kj}{}^{hx} b^x{}_k \qquad = c_{kj}{}^{hx} b^x{}_k \qquad (linear) \\
&\quad \delta^y = f_y{}'(net_j{}^y) = j*\sin^{j-1}(net_j{}^y)*\cos(net_j{}^y) \\
&\qquad = j*\sin^{j-1}(c_j{}^y * j * y)*\cos(c_j{}^y * j * y)
\end{aligned}
$$

V. TIME SERIES DATE MODELING USING UGS-HONN

This paper uses the monthly Canadian dollar and USA dollar exchange rate from November 2008 to December 2009 as the test data for UGS-HONN models. This paper also uses the monthly Australian dollar and USA dollar exchange rate from November 2008 to December 2009 as the test data for UGS-HONN models. Rate and desired output data, R_t, are from USA Federal Reserve Bank Data bank. Input1, R_{t-2}, are the data at time t-2. Input 2, R_{t-1} are the data at time t-1. The values of R_{t-2}, R_{t-1}, and R_t are converted to a range from 0 to 1 and then used as inputs and output in the UGS-HONN model. UGS-HONN model 1b is used. The test data of UGS-HONN orders 6 for using 10,000 epochs are shown on the tables.

VI. CONCLUSION

This paper develops the details of a open box and nonlinear higher order neural network models of UGS-HONN. This paper also provides the learning algorithm formulae for UGS-HONN, based on the structures of UGS-HONN. This paper uses UGS-HONN simulator and tests the UGS-HONN models using high frequency data and the running results are compared with Polynomial Higher Order Neural Network (PHONN), Trigonometric Higher Order Neural Network (THONN), and Sigmoid polynomial Higher Order Neural Network (SPHONN) models. Test results show that average error of UGS-HONN models are 0.0000% and the average error of Polynomial Higher Order Neural Network (PHONN), Trigonometric Higher Order Neural Network (THONN), and Sigmoid polynomial Higher Order Neural Network (SPHONN) models are from 2.8128% to 4.9076%. It means that UGS-HONN models are 2.8128% to 4.9076% better than PHONN, THONN, and SPHONN models.

REFERENCES

[1] Barron, R., Gilstrap, L. & Shrier, S. (1987). Polynomial and Neural Networks: Analogies and Engineering Applications, Proceedings of International Conference of Neural Networks, Vol. II. (pp. 431-439). New York.

[2] An, Z.G., Mniszewski, S.M., Lee, Y.C., Papcun, G., & Doolen, G.D. (1988). HIERtalker: a default hierarchy of high order neural networks that learns to read English aloud. Proceedings of the Fourth Conference on Artificial Intelligence Applications (pp.388). 14-18 March, 1988.

[3] Mao, Z. Q., Selviah, D. R., Tao, S., & Midwinter, J. E. (1991). Holographic high order associative memory system. Third IEE International Conference on "Holographic Systems, Components and Applications, Heriot Watt University, Edinburgh, Scotland, 342, 132-136

[4] Michel Lopez-Franco, Alma Y. Alanis, Nancy Arana-Daniel, and Carlos Lopez-Franco, Artificial Higher Order Neural Networks for Modeling MIMO Discrete-Time Nonlinear. In Ming Zhang (Ed.), Artificial Higher Order Neural Networks for Modeling and Simulation, Premier Reference Source, Information Science Reference (an imprint of IGI Global), Hershey, PA, USA, 2013, 30-43.

[5] Yuxin Ding, Artificial Higher Order Neural Networks for Modeling Combinatorial Optimization Problems. In Ming Zhang (Ed.), Artificial Higher Order Neural Networks for Modeling and Simulation, Premier Reference Source, Information Science Reference (an imprint of IGI Global), Hershey, PA, USA, 2013, 44-57.

[6] Mehdi Fallahnezhad, and Hashem Yousefi, Needle Insertion Force Modeling using Genetic Programming Polynomial Higher Order Neural Network, In Ming Zhang (Ed.), Artificial Higher Order Neural Networks for Modeling and Simulation, Premier Reference Source, Information Science Reference (an imprint of IGI Global), Hershey, PA, USA, 2013, 58-77.

[7] Madan M. Gupta, Ivo Bukovsky, Noriyasu, Ashu M. G. Solo, and Zeng-Guang Hou, Fundamentals of Higher Order Neural Networks for Modeling and Simulation, In Ming Zhang (Ed.), Artificial Higher Order Neural Networks for Modeling and Simulation, Premier Reference Source, Information Science Reference (an imprint of IGI Global), Hershey, PA, USA, 2013, 103-132.

[8] C. W. J. Granger, Some properties of time series data and their use in econometric model specification, Journal of Econometrics, vol. 16, 1981,121-130.

[9] C. W. J. Granger, and A. A. Weiss, Time series analysis of error-correction models, In S. Karlin, T. Amemiya and L. A. Goodman (Eds), Studies in Econometrics, Time Series and Multivariate Statistics, In Honor of T. W. Anderson. San Diego: Academic Press, 1983, 255-278.

[10] C. W. J. Granger, and T. H. Lee, Multicointegration, In G. F. Rhodes, Jr and T. B. Fomby (Eds.), Advances in Econometrics: Cointegration, Spurious Regressions and Unit Roots, New York: JAI Press, 1990, 17-84.

[11] C. W. J. Granger, and N. R. Swanson, Further developments in study of cointegrated variables, Oxford Bulletin of Economics and Statistics, vol. 58, 1996, 374-386.

[12] D. Psaltis, C. Park, and J. Hong, Higher Order Associative Memories and their Optical Implementations, Neural Networks, vol. 1, 1988, 149-163.

[13] N. Redding, A. Kowalczyk, and T. Downs, Constructive high-order network algorithm that is polynomial time, Neural Networks, vol. 6, 1993, 997-1010.

[14] M. Zhang, S. Murugesan, and M. Sadeghi, Polynomial higher order neural network for economic data simulation, Proceedings of International Conference on Neural Information Processing, Beijing, China, 1995, 493-496.

[15] M. Zhang, J. Fulcher, and R. A. Scofield, Neural network group models for estimating rainfall from satellite images, Proceedings of World Congress on Neural Networks, San Diego, CA, 1996, 897-900.

[16] M. Zhang, J. C. Zhang, and S. Keen, Using THONN system for higher frequency non-linear data simulation & prediction, Proceedings of IASTED, International Conference on Artificial Intelligence and Soft Computing, Honolulu, Hawaii, USA, 1999, 320-323.

[17] M. Zhang, J. C. Zhang, & J. Fulcher, Higher order neural network group models for data approximation, International Journal of Neural Systems, vol. 10(2), 2000, 123-142.

[18] M. Zhang, S. Xu, and J. Fulcher, Neuron-Adaptive Higher Order Neural Network Models for Automated Financial Data Modeling, IEEE Transactions on Neural Networks, vol. 13(1), 2002, 188-204.

[19] M. Zhang, and J. Fulcher, Higher Order Neural Networks for Satellite Weather Prediction, In J. Fulcher and L. C. Jain (Eds.), Applied Intelligent Systems, Springer, Vol. 153, 2004, 17-57.

[20] M. Zhang, Artificial Higher Order Neural Networks for Economics and Business. IGI-Global Publisher, 2009.

[21] M. Zhang, Artificial Higher Order Neural Network Nonlinear Models: SAS NLIN or HONNs? Book chapter in Ming Zhang (Editor), Artificial Higher Order Neural Networks for Economics and Business, IGI-Global Publisher, 2009. 1-47.

[22] M. Zhang, Ultra High Frequency Trigonometric Higher Order Neural Networks, Book chapter in Ming Zhang (Editor), Artificial Higher Order Neural Networks for Economics and Business, IGI-Global Publisher, 2009, 133-163.

[23] M. Zhang, Artificial Higher Order Neural Networks for computer Science and Engineering – Trends for Emerging Applications, IGI-Global Publisher, 2010.

[24] M. Zhang, Artificial Higher Order Neural Networks for Modeling and Simulation. IGI-Global Publisher, 2013.

[25] C. E. Shannon, Communication in the presence of noise, Proceedings of Institute of Radio Engineers, vol. 37, no. 1, pp. 10–21, Jan. 1949. Reprint as classic paper in: Proc. IEEE, vol. 86, no. 2, Feb. 1998.