System Identification and Prediction of Dynamic System and Microwave Thermal Process Using a Recurrent Fuzzy Quantum Neural Network

Chaoyin Li[1,2], Qingyu Xiong[1,3], Kai Wang[1,2],Yang Yu[1,3] and Tong Liu[1,2]

[1]Key Laboratory of Dependable Service Computing in Cyber Physical Society (Chongqing University), Ministry of Education, China
[2]School of Automation, Chongqing University, Chongqing 400044, China
[3]School of Big Data & Software, Chongqing University, Chongqing 401331, China
Email: cquxqy@163.com

Abstract. Microwave heating is a time-varying, non-linear process. Mechanism modeling of the microwave thermal process is extremely difficult because of the complex microwave heating environment. This paper presents a recurrent fuzzy quantum neural network with full feedbacks (RFQNN) for prediction and identification of dynamic systems and the actual microwave heating process. In the RFQNN, a quantum neural network is introduced to the consequent part of the fuzzy rules to improve the mapping ability and the identification precision. All of the rules are generated and learned online through a simultaneous structure and parameter learning. During the structure learning, an online clustering algorithm combined with Mahalanobis distance elimination algorithm perform effectively in generating or removing fuzzy rules. And then a gradient descent algorithm is introduced to update the parameters during the parameter learning process. And finally, we test the RFQNN by dynamic plants and the microwave thermal process. The results show that it performs well in dynamic system processing compared with other recurrent fuzzy neural networks.

1. Introduction

Microwave is a novel, green and efficient energy source, which is widely used in both domestic and industry as a source of heat energy, such as food heating, sintering, drying, etc. However, there is a big challenge in the mechanism modeling of the dynamic microwave thermal process because of the non-uniformity of the electric field and the complex microwave heating environment. Researchers proposed some models for microwave heating process. Akkari et al.[1] proposed a model of heat transfer with a source term in microwave-assisted food thawing process. Yuan, Liang, etc.[2] proposed a one-dimensional heat and mass transports model in microwave drying process of unsaturated porous materials. Zhou and Puri et al.[3] proposed a three-dimensional finite element modeling of heat and mass transfer method in food materials microwave heating process. Nevertheless, almost all the models can only be applied to specific microwave heating equipment, specific heating media and specific application environment. Model parameter data obtained in one separate experiment cannot be applied directly to different media or the same media in different application environment. Therefore, the proposed microwave heating model must have the ability to adapt to different application environments, i.e., possesses the capacity of self-learning and generalizing.

In recent years, some researchers, for example, Murthy and Manohar et al.[4], and Chen et al.[5], Poonnoy et al.[6], Yousefi, Zahra et al.[7]have employed multilayer feed-forward neural networks to predict heating temperature, moisture content, drying rate and so on in microwave thermal process.

However, the multilayer feed-forward perceptron neural networks (MFNN) those studies used have many kinds of shortcomings, such as large numbers of hidden neurons, Inappropriate network structure and the limitation of generalizing ability. And also, MFNN cannot work well for dynamic systems because of the unknown dependency of past inputs and past outputs.

For dynamic system processing, recurrent fuzzy neural networks (RFNNs) has been widely researched in areas such as time-varying system identification, unknown dependency time-series prediction, pattern recognition, and signal processing. Recent studies show that some RFNNs outperform feed-forward fuzzy neural network (FNN) and recurrent neural network (RNN) in solving the time-varying characteristics of dynamic systems[8]. Roh, Wang and Ahn[9] optimized the conventional RFNN and applied it in time series prediction of dynamic systems. Results show that the prediction performance is better than general RFNN. In order to overcome the uncertainties of SMC, Lin, Chen and Sun[10] employed recurrent wavelet fuzzy neural network (FWRNN) to an intelligent sliding mode position control as a uncertainty estimator of electrical power steering system. Although those proposed RFNNs can get good performance and accuracy in some specific systems. However, the fixed structure of these recurrent FNNs limits the flexibility and generalization ability compared with self-evolving recurrent fuzzy neural networks (RSFNN). Juang, Lin and Tu[11] proposed a recurrent self-evolving fuzzy neural network with local feedbacks (RSEFNN), which feeds the firing strength of a fuzzy rule to itself. Lin and Chang[12] improved RSEFNN by introducing a functional-link neural network to the consequent part of the network and feeding firing strength of each recurrent node output to itself and others. Pratama, Lu et al.[13] introduced an evolving type-2 recurrent fuzzy neural network to solve the data uncertainty and temporal behavior problems of the concept drifts by an incremental learning algorithm. However, those neural networks do not take into account the removing of redundant structures and a functional expansion of the input vector leads to a large number of consequent parameters that limits the parameter learning rate of the network. To overcome these disadvantages, some researchers combined recurrent neural network with quantum neural network (QNN) to enhance the computing ability and learning rate of the network. Gandhi, Arora et al.[14] presented a recurrent quantum neural network (RQNN) for a brain-computer interface to enhance the EEG signal. Ganjefar, Tofighi and Karami[15] combined fuzzy wavelet neural network with QNN in a control design to enhance the stability of multi-machine power system. The above studies indicate that the combination of QNN and other neural network can enhance the performance of the network in some degrees.

In this paper, a quantum neural network is introduced to the consequent part of the RFNN with less consequent parameters to be updated and superfast calculation ability. The proposed RFQNN has stronger generalization ability and higher identification accuracy under the same training epochs. Similar to the self-evolving RFNNs, the RFQNN also contains two learning steps: structure learning and parameters learning. For structure learning, the RSFNN uses an efficient rule and fuzzy set generation algorithm, and takes into account of deleting redundant fuzzy rules by considering the distance between two rules and persistently insignificant time. For parameters learning, a fast and effective gradient descent algorithm is introduced to update the parameters online. And then we conduct several simulations to assess the RFQNN's performance and compare the RFQNN with some other existing models.

2. RFQNN Structure

Suppose that the dynamic system to be processed is a multi-input multi-output (MIMO) system. Figure 1 shows a seven-layered RFQNN with n input and n_o output variables. We will discuss the function of each layer in detail next. To make more understanding, we will use some symbols to replace the relationship between each layer and the mathematical function of each node. The input and the output to the i-th node in layer l of the networkis represented as $u_i^{(l)}$ and $O_i^{(l)}$, respectively.

*Layer1 (Input Layer):*The training data $x=(x_1,x_2,\ldots,x_n)$ are fed as inputs to the first layer. Although there is no need to include past states in the training inputs on account of the recurrent structure, it is allowed in this paper. Weights to be adjusted in this layer are absent.

Layer2 (Fuzzification Layer): Fuzzification is performed in this layer, and we define a Gaussian membership function in each node of this layer. For the *i*-th fuzzy set A_j^i on the input variable x_j, *j*=1, 2, …, *n*, a Gaussian membership function is computed by equation (1)

$$\varpi_j^i(x_j) = O_{ij}^2 = \exp\left\{-\frac{1}{2}\left(\frac{u_j^2 - c_j^i}{\sigma_j^i}\right)^2\right\} \text{ and } u_j^2 = O_j^1 \tag{1}$$

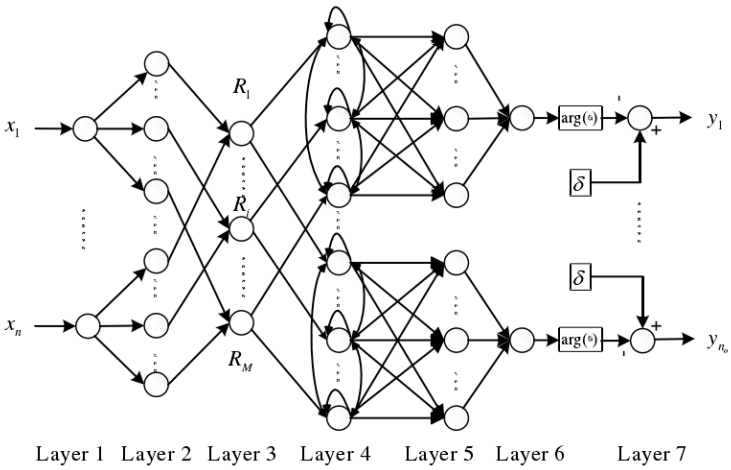

Figure 1. RFQNN structure

Layer3 (Spatial firing Layer): In this layer, each node corresponds to one fuzzy rule that represents the firing strength. To obtain the spatial firing strength, each node of this layer performs a fuzzy meet operation on inputs it receives from layer 2 using an algebraic product operation. There are M nodes that is M fuzzy rules generated in this layer and the spatial firing strength is computed by

$$\varphi_i = O_i^3 = \prod_{j=1}^{n} u_j^{(3)} \text{ , and } u_j^{(3)} = O_j^2 \tag{2}$$

where $i = 1, 2, \ldots, M$, and *M* is the total number of fuzzy rules.

Layer4 (Temporal firing Layer): In this layer, each node equipped with some recurrent structures, which forms a self-feedback and mutual feedback loop. The temporal firing strength of a recurrent rule node, $\phi^i(t)$, which depends both on the current spatial firing strength, $\varphi^i(t)$, and on the previous temporal firing strength, $\phi^k(t-1)$. The temporal firing strength is a linear combination function that can be expressed as

$$\phi_i^{\mathrm{q}}(t) = \sum_{k=1}^{M} (\lambda_{ik}^q \bullet \phi_k^q(t-1)) + (1-\gamma_i^q) \bullet \varphi^i(t) \tag{3}$$

where $\gamma_i^q = \sum_{k=1}^{M} \lambda_{ik}^q$, $q = 1, \ldots, n_o$ and $\lambda_{ik}^q = (\Gamma_{ik}^q)/M$ ($0 \le \Gamma_{ik}^q \le 1$), is the rule interaction coefficients between itself and other rules. The recurrent weights λ_{ik}^q initialized to zeros and determine the ratio that the current inputs and previous inputs contribute to the network outputs, respectively.

Layer5 (Quantization Layer): Each node in this layer performs a quantization operation. The inputs of the layer are real values normalized in the range [0,1]. The input layer then converts the real values into the phase $[0, \pi/2]$ in quantum states. The conversion process that called quantization characteristic representation can be expressed as the following equation:

$$\mathrm{N}_i^q = |O_i^{(5)}\rangle = \cos\left(\frac{\pi}{2} \bullet O_i^{(4)} \Big/ \left(\sum_{k=1}^{M} O_k^{(4)}\right)\right) |0\rangle + \sin\left(\frac{\pi}{2} \bullet O_i^{(4)} \Big/ \left(\sum_{k=1}^{M} O_k^{(4)}\right)\right) |1\rangle = \cos(\vartheta_i |0\rangle + \sin(\vartheta_i) |1\rangle \tag{4}$$

Layer6 (Rotation Layer): In this layer, the quantum phase state is initialized to $F(\rho)$, and then each node performs a rotation operation on the input quantum phase state. The rotation process is computed by

$$R^q = |O^{(6)}\rangle = F(\rho) + \sum_{i=1}^{M} P(\theta_i(t)) \,|\, O_i^{(5)}\rangle = \begin{bmatrix} \cos p + \sum_{i=1}^{M} \cos(\vartheta_i(t) + \theta_i(\mathrm{t})) \\ \sin p + \sum_{i=1}^{M} \sin(\vartheta_i(t) + \theta_i(\mathrm{t})) \end{bmatrix} = \begin{bmatrix} \cos \alpha^q(t) \\ \sin \alpha^q(t) \end{bmatrix} \tag{5}$$

where $\alpha^q(t) = \arctan\left(\dfrac{\sin p + \sum_{i=1}^{M} \sin(\vartheta_i(t) + \theta_i(\mathrm{t}))}{\cos p + \sum_{i=1}^{M} \cos(\vartheta_i(t) + \theta_i(\mathrm{t}))} \right)$

Layer 7 (Reversal and Output Layer): Each node in this layer corresponds to a reversal of the output quantum state. The initialization state of quantum reversal is given by δ, and the output of the quantization layer is square root of probability value of quantum state $|1\rangle$

$$\mathrm{y}_q = O_q^{(7)} = \sin\left(\frac{\pi}{2} \bullet \frac{1}{1+\exp(-\delta^q)} - \alpha^q(\mathrm{t})\right) \tag{6}$$

3. RFQNN Learning Method

The learning process of a RFQNN evolves from simultaneous structure learning and parameter learning. Initially, no rules are in a RFQNN. All of the fuzzy rules generate from the simultaneous structure and parameter learning after receiving each piece of training data. The proposed model uses efficient structure learning algorithm to evolve fuzzy rules and fuzzy sets on-line. A gradient descent algorithm is performed in the parameter learning process.

3.1. Structure Learning

The structure learning can be divided into two parts: fuzzy rule generation part and redundant rule elimination part. It determines whether a new rule should be generated from the training data or an old rule should be eliminated for its little contribution to the neural network outputs.

3.1.1. Fuzzy Rule Generation Part

The spatial firing strength φ^i is used to determine whether a new rule should be extracted from the training data. The first piece of incoming data point *X* is used to generate the first fuzzy rule, and the mean and width of the Gaussian membership functions related to this rule are set as:

$$m_j^1 = x_j \text{ and } \sigma_j^1 = \sigma_{initial}, \; j = 1, 2, ..., n \tag{7}$$

where $\sigma_{initial} = 0.3$ is a pre-specified value. For subsequent new incoming data *X*, we find the maximum spatial firing strength by

$$I = \arg \max_{1<i<M(t)} \varphi^i(t) \tag{8}$$

where *M(t)* is the number of existing rules at time *t*. If $\varphi_I(t) \le f_{th}$ (f_{th} is a pre-specified threshold), then a new fuzzy rule is generated and $M(t+1) = M(t)+1$. For the new generated rule, the mean and width of corresponding fuzzy sets are defined as:

$$m_j^{M(t)+1} = x_j \text{ and } \sigma_j^{M(t)+1} = \varsigma \bullet \left| x_j - m_j^I \right| \tag{9}$$

where ς is an overlap coefficient and we set $\varsigma = 0.5$ in this paper.

3.1.2. Fuzzy Rule Generation Part

An appropriate number of fuzzy rules can efficiently avoid over-fitting and persistently insignificant node. in this paper Mahalanobis distance (M-distance) is used to eliminate the redundant fuzzy rules. M-distance is given by:

$$D_k = \sqrt{\sum_{j=1}^{n} \frac{(x_j - m_j^k)^2}{2(\sigma_j^k)^2}}, \; k = 1, ..., M \tag{10}$$

For the *t*-th observed dataset (x_t, y_t), define a discrete set:

$$\Gamma_t := \{D_k \mid D_k \le D_{set}, k = 1,...,M(t)\} \tag{11}$$

where $D_{set} > 0$ is a pre-specified threshold. If the number of collection element in Γ_t satisfies size $(\Gamma_t) > N_{\max}$, that means too many rules near x_t that would result in an over-fitting problem, then the rules with the smallest M-distance should be eliminated. Else if D_k satisfies $D_k \ge D_{th}$ (D_{th} is a pre-defined upper limit value) for $T_{\max}$ consequent times, then the *k*-th rule should be pruned for its little contribution to the output.

3.2. Parameter Learning

For each piece of incoming data, the parameter learning phase occurs simultaneously with the structure learning phase.For the sake of clarity, we consider the single-output system case, then the objective parameter learning is to minimize the following error cost function

$$E = \frac{1}{2}[y_q(t) - y_d(t)]^2 \tag{12}$$

where $y_q(t)$ represents the RFQNN output and $y_d(t)$ represents the desired output. The parameters are updated by a self-adaptive gradient descent algorithm. And we will show the detailed parameter update steps next.

The inversion layer parameter δ^q is updated by:

$$\delta^q(t+1) = \delta^q(t) - \eta \frac{\partial E}{\partial y_q}\frac{\partial y_q}{\partial \delta} = \delta^q(t) - \eta(y_q - y_d)\frac{\cos(z^o)\bullet\pi\bullet\exp(-\delta^q)}{2\bullet(1+\exp(-\delta^q))^2} \tag{13}$$

The rotation layer parameter ρ^q is updated by:

$$\rho^q(t+1) = \rho^q(t) - \eta \frac{\partial E}{\partial y_q}\frac{\partial y_q}{\partial \rho^q} = \rho^q(t) + \eta(y_q - y_d)\frac{\cos(z^o)}{v}\bullet\begin{pmatrix} 1+\cos p\bullet\sum_{i=1}^{M}\cos(\vartheta_i^q(t)+\theta_i^q(t)) \\ +\sin p\bullet\sum_{i=1}^{M}\sin(\vartheta_i^q(t)+\theta_i^q(t)) \end{pmatrix} \tag{14}$$

where $v = \left(\cos p + \sum_{i=1}^{M}\cos(\vartheta_i^q(t)+\theta_i^q(t))\right)^2 + \left(\sin p + \sum_{i=1}^{M}\sin(\vartheta_i^q(t)+\theta_i^q(t))\right)^2$

The rotation layer parameter θ_i^q is updated as:

$$\theta_i^q(t+1) = \theta_i^q(t) - \eta \frac{\partial E}{\partial y_q}\frac{\partial y_q}{\partial \theta_i^q} = \theta_i^q(t) - \eta(y_q - y_d)\frac{\partial y_q}{\partial \alpha^q}\frac{\partial \alpha^q}{\partial \theta_i^q} = \theta_i^q(t) + \frac{\eta(y_q - y_d)\bullet\cos(z^o)\bullet(\xi_i^q + \zeta_i^q)}{v} \tag{15}$$

where $\xi_i^q = \cos\left(\vartheta_i^q(t)+\theta_i^q(t)\right)\bullet\left(\cos(\rho) + \sum_{k=1}^{M}\cos(\vartheta_k^q(t)+\theta_k^q(t))\right)$, and

$\zeta_i^q = \sin\left(\vartheta_i^q(t)+\theta_i^q(t)\right)\bullet\left(\sin(\rho) + \sum_{k=1}^{M}\sin(\vartheta_k^q(t)+\theta_k^q(t))\right)$

The temporal firing parameter λ_{ij}^q is updated as:

$$\lambda_{ij}^q(t+1) = \lambda_{ij}^q(t) - \eta \frac{\partial E}{\partial y_q}\frac{\partial y_q}{\partial \lambda_{ij}^q} = \lambda_{ij}^q(t) + \eta(y_q - y_d)\frac{\cos(z^o)\bullet\left(\xi_i^q + \zeta_i^q\right)\bullet\pi\bullet(1-\tilde{\phi}_i^q(t))\bullet(\phi_i^q(t-1) - \varphi_i(t))}{2\upsilon\bullet\sum_{k=1}^{M}\phi_k^q(t)} \tag{16}$$

where $\tilde{\phi}_i^q(t) = \phi_i^q(t) / (\sum_{i=1}^{M}\phi_i^q(t))$.

The antecedent part of parameter c_j^i is updated as:

$$c_j^i(t+1) = c_j^i(t) - \eta \frac{\partial E}{\partial y_q}\frac{\partial y_q}{\partial c_j^i} = c_j^i(t) + \eta(y_q - y_d)\frac{\cos(z^o)\bullet\left(\xi_i^q + \zeta_i^q\right)\bullet\pi\bullet(1-\tilde{\phi}_i^q)\bullet(1-\gamma_i^q)\bullet\varphi_i\bullet(x_j - c_j^i)}{2\upsilon\bullet\sum_{k=1}^{M}\phi_k^q(t)\bullet\left(\sigma_j^i\right)^2} \tag{17}$$

The antecedent part of parameter σ_j^i is updated as:

$$\sigma_j^i(t+1)=\sigma_j^i(t)-\eta\frac{\partial E}{\partial y_q}\frac{\partial y_q}{\partial \sigma_j^i}=\sigma_j^i(t)+\eta(y_q-y_d)\frac{\cos(z^o)\bullet\left(\xi_i^q+\zeta_i^q\right)\bullet\pi\bullet(1-\tilde{\phi}_i^q)\bullet(1-\gamma_i^q)\bullet\varphi_i\bullet(x_j-c_j^i)^2}{2\upsilon\bullet\sum_{k=1}^{M}\phi_k^q(t)\bullet\left(\sigma_j^i\right)^3} \quad (18)$$

The learning rate parameter α too big or too small, both can influence the convergence of the training and test process. To keep fast convergence and stability, therefore, this paper uses a self-adaptive learning rate. As in [8] the learning rate can be chosen by the asymptotic stability of Lyapunov function:

$$\eta=\frac{L}{\left(\frac{\partial y_1}{\partial W},\cdots,\frac{\partial y_q}{\partial W}\right)^T\bullet\left(\frac{\partial y_1}{\partial W},\cdots,\frac{\partial y_q}{\partial W}\right)} \quad (19)$$

where $0<L<2$, the parameter L is chosen according to actual applications, and

$$\frac{\partial y_q}{\partial W}=\left[\frac{\partial y_q}{\partial \delta^q},\frac{\partial y_q}{\partial \rho^q},\frac{\partial y_q}{\partial \theta_1^q}\cdots\frac{\partial y_q}{\partial \theta_M^q},\frac{\partial y_q}{\partial \lambda_{11}^q}\cdots\frac{\partial y_q}{\partial \lambda_{MM}^q},\frac{\partial y_q}{\partial c_{11}^q}\cdots\frac{\partial y_q}{\partial c_{nM}^q},\frac{\partial y_q}{\partial \sigma_{11}^q}\cdots\frac{\partial y_q}{\partial \sigma_{nM}^q}\right]^T \quad (20)$$

4. Simulations

In this section, we apply the RFQNN in the identification and prediction of some well-known dynamic systems. We take three examples that include dynamic system identification, chaotic sequence prediction and microwave thermal process identification. These examples also compare the performance of the RFQNN with some other recurrent FNNs. In the follow examples, some initial parameters are set as: δ=0 , ρ=0.2π , θ_1=0.2π , L=1 .

Example 1 (Dynamic System Identification)

This example uses the RFQNN to identify a nonlinear dynamic system with multiple time delays that has been studied in [16]. The dynamic system is described by the following difference equation:

$$y_p(t+1)=0.72y_p(t)+0.025y_p(t-1)u(t-1)+0.01u^2(t-2)+0.2u(t-3). \quad (21)$$

This plant output depends on four previous inputs and two previous outputs. In this example we take the current state $y_p(t)$ and $u(t)$ as the inputs of the RFQNN, and $y_p(t+1)$ as the desired output. As in previous studies for training different recurrent models, we use only 10 epochs, with 900 pieces of training data each epoch. In each epoch, the input sequence $u(t)$ is generated by an independent and identically distributed uniformly random sequence over [-2, 2] for first 350 time steps, and a sinusoid given by $1.05\sin(\pi k/45)$ for the remaining time. There is no repetition in these 900 training data, i.e., each epoch uses a different training set. The structure learning threshold f_{th} decides the number of rules to be generated and the number and the value of M-distance determine the number of rules to be pruned. The parameters used in the training process are set as: $f_{th}=0.005$, $\sigma_{set}=0.3$, $D_{set}=0.005$, $D_{th}=15$, $N_{max}=3$, $T_{max}=1000$. Three rules are generated during the training process and the test input signal is given by

$$u(t)=\begin{cases}\sin(\pi t/25), & t<250,\\ 1.0, & 250\le t<500,\\ -1.0, & 500\le t<750,\\ 0.3\sin(\pi t/25)+0.1\sin(\pi t/32)+0.6\sin(\pi t/10), & 750\le t<1000.\end{cases} \quad (22)$$

Table 1 shows the root-mean-squared error (RMSE) of the training data. Figure 2 shows a comparison of the actual output with the output produced by the RSFNN. The performance of the RFQNN is compared with that of some other recurrent models, including a RSONFIN[16], a TRFN[17], a WRFNN[18], a RSFNN-TSK[8] and a RSEFNN-LF[11]. These models use a same training condition, i.e., include identical numbers of input variables, training data, test data, and training epochs as

designed by the RFQNN.From the results of table 1, RFQNN achieves the lowest test RMSE and almost the same training RSME to RSEFNN-LF. The fuzzy rules generated by RFQNN are three, so that less number of parameters requires to be updated. The above analysis indicates that RFQNN can achieve better identification performance than the other recurrent networks.

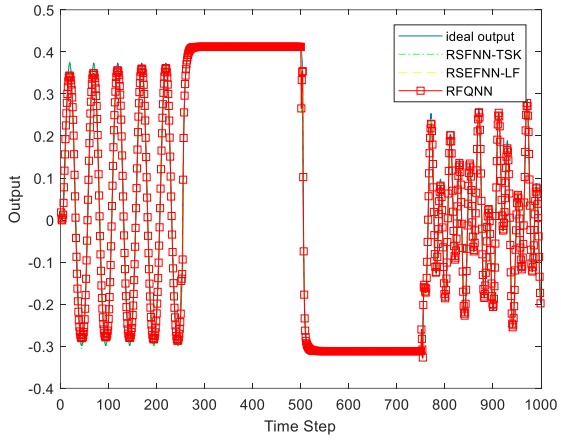

Figure 2. Outputs of the dynamic plant, RSFNN-TSK, RSEFNN-LF and RFQNN in Example 1.

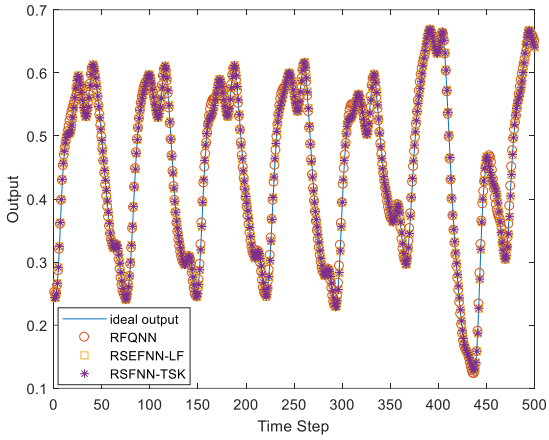

Figure 3. Test results of MG chaotic series prediction using RFQNN, RSEFNN-LF, RSFNN-TSK in Example 2.

Table 1. Performance of RFQNN and other recurrent models for dynamic system identification in example 1.

Models	Rules	Parameters	Training RMSE	Test RMSE
RSONFN [16]	6	36	0.03	0.06
TRFN [17]	3	33	0.007	0.031
WRFNN [18]	5	55	0.057	0.083
RSEFNN-LF [11]	4	32	0.016	0.028
RSFNN-TSK [8]	7	98	0.017	0.0134
RFQNN	3	26	0.017	0.0126

Example 2 (Mackey-Glass Chaotic Series Prediction)

In this example, RFQNN is employed to predict the Mackey-Glass (MG) chaotic series. The time series of the prediction problem is generated by the following delay differential equation:

$$\frac{dx(t)}{dt} = \frac{0.2x(t-\tau)}{1+x^{10}(t-\tau)} - 0.1x(t) \tag{23}$$

In this study, the time delay parameter τ is set to 30, and the initial value $x(0)=1.2$. As the same as in the previous study, four past values are fed as the input and the current state as the label, and the input-output data pattern is given by

$$[x(t-24), x(t-18), x(\mathrm{t}-12), x(t-6); x(t)]. \tag{24}$$

1000 patterns are generated from t=124 to t=1123, where the first 500 patterns are taken as training set and the remaining 500 patterns as the test set. The training epoch is set to 500 and the parameters during this training process are set as: $f_{th}=0.0005$, $\sigma_{set}=2$, $D_{set}=0.0001$, $D_{th}=10$, $N_{max}=3$, $T_{max}=500$. After training, three rules are generated. Table 2 shows the training performance of the RFQNN and some other recurrent FNNs presented in this paper. Figure3 shows the time series prediction results of the RFQNN, RSFNN-TSK and RSEFNN-LF.

The performance of RFQNN is compared with some recently developed fuzzy designed by genetic algorithms or neural learning. For a fair comparison, all the models take the same condition as the

RFQNN. The prediction errors of the test set are mainly fall in the interval [-0.005,0.005]. And from the results of Table 2, FWNN achieves the lowest prediction error, but it may achieve a highest training rate because of the generated 16 rules. Although RFQNN achieves relatively lower prediction performance in this example, it gets a higher training rate. That is because the consequent part is absent in presented neural network and parameters requires to be updated is minimum.

Table 2. Performance of RSFNN and other recurrent models for MG chaotic series prediction in Example 2

Models	Rules	Parameters	Training RMSE	Test RMSE
FWNN[19]	16	128	0.0023	0.0023
G-FNN [20]	10	90	-	0.0056
TRFN [17]	5	95	-	0.0124
RSEFNN-LF [11]	9	126	0.0032	0.0031
RSFNN-TSK [8]	3	48	0.0042	0.0036
RFQNN	3	38	0.0027	0.0026

Example 3 (The Actual Microwave Heating Process Identification)
In this section, we will apply RFQNN to learn the actual microwave heating process. Figure 4 shows the tunnel-based microwave heating system. The system mainly includes five parts, i.e., microwave power source, microwave waveguide, microwave cavity, sensing and acquisition subsystem, and the control system. Each of the first two microwave heating cavity has two microwave power sources and the last heating cavity only one. To detect the temperature of the heated medium, each microwave cavity equipped with an optical fibre temperature sensor and an infrared temperature sensor. The sampling temperature data are directly delivered to the computer, where the control algorithm runs. The output power of each microwave power source ranges from 0 to 3000W.

Figure 4. Tunnel-based microwave heating system

During the microwave heating process, rice is served as the heating medium. In order to agree with the actual heating process, five heating processes under different input powers are used as the training data, i.e., 500, 1000, 1500, 2000, 2500, 3000W. For simplicity, we representatively choose the second and the third microwave heating cavity as the identification object. The microwave thermal process of the second cavity is chosen as the identification object. Throughout the microwave heating process, we collected 2140 patterns of microwave heating data in the second cavity, among which 1712 patterns are used as the training data and the remaining as the test data. The input-output format which contains four input variables and one label is given by

$$[u_1(t-1), u_2(t-1), T(t-1), u_1(t), u_2(t), T(t); T(t+1)] \tag{25}$$

where $u_1(t), u_2(t), \text{and } T(t)$ represent the microwave power of source one and source two and temperature at the present moment, respectively. Before training, we have performed normalization operations of the training data. And the training epoch is set to 300 and the parameters during this training process are set as: $f_{th} = 5\times10^{-8}$, $\sigma_{set} = 0.6$, $D_{set} = 0.0003$, $D_{th} = 15$, $N_{max} = 3$, $T_{max} = 1000$ .For clarity, the first 100 prediction outputs of the test data are printed out in figure 5. Figure 5 Shows the

prediction results of RFQNN, RSFNN-TSK and RSEFNN-LF for microwave thermal process. The temperature test error variation is between $\pm 0.5\mathrm{K}$. Table 3 shows the performance of these RFNNs in microwave thermal process prediction. From Table 3 we can see RFQNN achieves the lowest training RSME and test RSME. i.e. RFQNN get better performance than RSEFNN-LF in microwave thermal process identification.

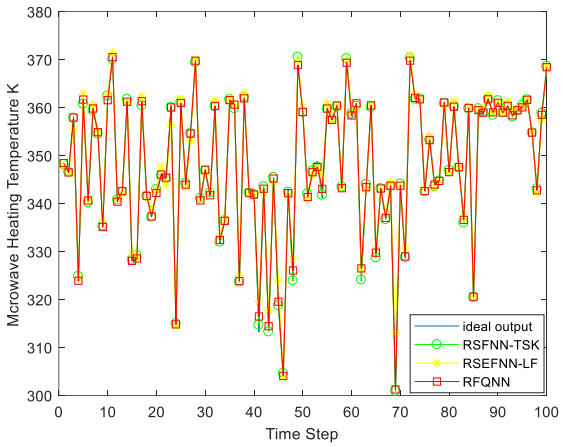

Figure 5. Results of Microwave heating process prediction using RSFNN-TSK, RSEFNN-LF, and RFQNN.

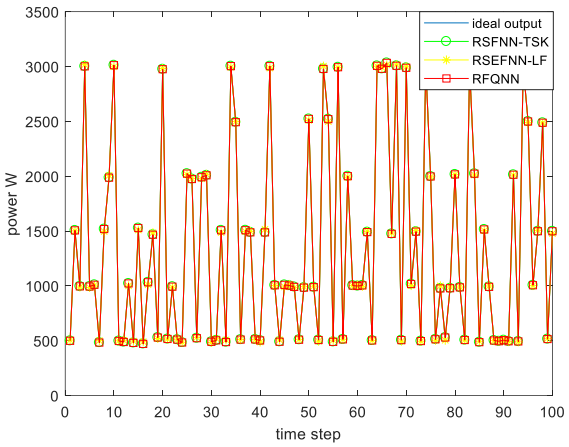

Figure 6. The learning result of the reverse control in microwave thermal process using RSFNN-TSK, RSEFNN-LF and RFQNN

Table 3. Performance of RFQNN, RSEFNN-LF and RSFNN-TSK for microwave thermal process identification in Example 3.

Models	Rules	Parameters	Training RMSE	Test RMSE
RSEFNN-LF[11]	4	128	0.2301	0.5431
RSFNN-TSK[8]	4	92	0.2273	0.5034
RFQNN	5	92	0.1926	0.4812

Table 4. Performance of RFQNN, RSEFNN-LF and RSFNN-TSK in reverse control learning of microwave thermal process in Example 3.

Models	Rules	Parameters	Training RMSE	Test RMSE
RSEFNN-LF[11]	4	51	62.9731	64.1516
RSFNN-TSK[8]	3	39	62.6112	64.0924
RFQNN	3	32	61.5344	62.3812

For temperature control during microwave thermal process, RFQNN can be used as a controller and we've constructed a reverse control block diagram as shown in figure 7. In a reverse control, the first step of our objective is to train the RFQNN to emulate the inverse of the controlled plant input-output mapping relation-ship[24], thus to enhance the performance of RFQNN controller. In this section, the microwave thermal process of the third cavity is taken as learning object. During the control, the reference temperature $y_{ref}(k+1)$, current temperature $y_p(k)$, and last control input $u(k)$, are fed as the RFQNN inputs.

To obtain the training samples, sampling data during five heating processes under different input powers are used as the training data, i.e., 500, 1000, 1500, 2000, 2500, 3000W. Throughout the processes, 1900 patterns of microwave heating data were collected, among which 1500 patterns used for training and the remaining used for testing. The training pattern of data is given by equation (26)

and the training epoch is set to 300 and the parameters during this training process are set as: $f_{th}=0.1$, $\sigma_{set}=0.3$, $D_{set}=0.003$, $D_{th}=5$, $N_{max}=3$, $T_{max}=500$.

$$[u(k), y_p(k), y_{ref}(k+1); u(k+1)] \tag{26}$$

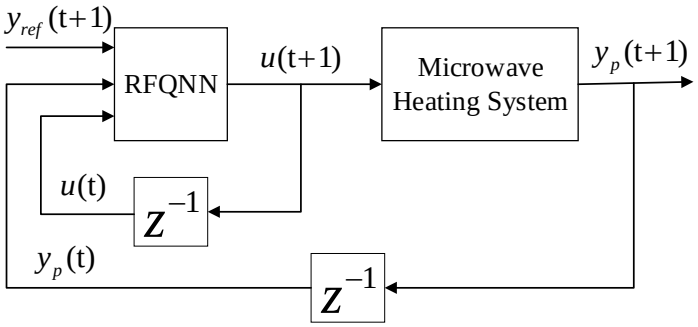

Figure7. The learning process of RFQNN-based reverse control.

Figure 6 shows the learning result of the reverse control in microwave thermal process. The result demonstrated that RFQNN can make better prediction of the reverse control output. The test errors of the input power are between ± 10W, which perfectly reflects its predictive performance as a reverse controller. Table 4 shows the performance of the models in reverse control learning. From table 4, RFQNN achieves the lowest training errors and test errors.

5. Conclusion

This paper presented a recurrent fuzzy quantum neural network (RFQNN), for handling dynamic system identification and time series prediction problems of time-varying systems. The proposed recurrent FNN is effective in modelling dynamic systems because of the online learning manner and its interactive recurrent structure. The online structure learning enables the network to find the best hidden layer structure of the neural network automatically. Instead of simply assigning the number of fuzzy sets being equal to that of rules in each input dimension, the number of rules may increase or prune through the fuzzy rule generation algorithm or elimination algorithm. The full feedback recurrent structure in the neural network not only stores the local information but also collects critical global information. In this study, a quantum computing layer is introduced, which makes full use of characteristics of quantum computing to help improve the performance of the network. The demonstrated performance of RFQNN, RSFNN-TSK, and RSEFNN-LF show that RFQNN can get excellent results in the complex microwave thermal process.

6. Acknowledgements

This research work is supported by The Major Science & Technology Program of Guangxi (Grant NO. GKAA 17129002), The Key Research Program of Chongqing Science & Technology Commission (Grant No. CSTC 2017 jcyjBX0025).

7. References

[1] Akkari E, Chevallier S and Boillereaux L 2009 Global linearizing control of MIMO microwave-assisted thawing *Control. Eng. Pract* **17** 39-47.
[2] Yuan Y, Liang S, Zhong J, Xiong Q and Gao M 2015 Temperature control strategy based on a heat and mass transports model in microwave drying *CCC2015* 1924-28.
[3] Zhou L, Anantheswaran R C 1995 Finite element modeling of heat and mass transfer in food materials during microwave heating - model development and validation *J. Food. Eng* **25** 509-29.

[4] Murthy TPK, Manohar B 2012 Microwave drying of mango ginger (Curcuma amada Roxb): prediction of drying kinetics by mathematical modelling and artificial neural network *Int. J. Food. Sci. Tech* **47** 1229–36.
[5] Chen C, Zhou XZ, Tang ZY and Lei YJ 2015 A neural network based model for temperature prediction in high power microwave heating system *ICEAME2015* 590-8.
[6] Poonnoy P, Tansakul A and Chinnan M 2007 Artificial neural network modeling for temperature and moisture content prediction in tomato slices undergoing microwave-vacuum drying *J. Food. Sci* **72** 042-7.
[7] Yousefi G, Emam-Djomeh PZ, Omid M and Askari GR 2014 Prediction of Physicochemical Properties of Raspberry Dried by Microwave-Assisted Fluidized Bed Dryer Using Artificial Neural Network *Dry. Technol* **32** 4-12
[8] Li J, Xiong Q, Wang K, Shi X and Liang S 2016 A recurrent self-evolving fuzzy neural network predictive control for microwave drying process *Dry. Technol* **34** 1434-44.
[9] Roh SB, Wang J and Ahn TC 2016 Optimization of Recurrent Fuzzy Neural Networks for Time Series Prediction *EASAC2016* 1442-43.
[10] Lin F-J, Chen S-G and Sun IF 2017 Intelligent Sliding-Mode Position Control Using Recurrent Wavelet Fuzzy Neural Network for Electrical Power Steering System *Int. J. Fuzzy. Syst* **19** 1344-61.
[11] Juang CF, Lin YY and Tu CC 2010 A recurrent self-evolving fuzzy neural network with local feedbacks and its application to dynamic system processing *Fuzzy. Set. Syst* **161** 2552-68.
[12] Lin YY, Chang JY and Lin CT 2013 Identification and prediction of dynamic systems using an interactively recurrent self-evolving fuzzy neural network *IEEE. T. Neur. Net. Lear* **24** 310-21.
[13] Mahardhika Pratama JL, Edwin L, Guangquan Z and E aMJ 2017 An Incremental Learning of Concept Drifts Using Evolving Type-2 Recurrent Fuzzy Neural Networks *IEEE. T. Fuzzy. Syst* **25** 1175-92.
[14] Gandhi V, Arora V, Behera L, Prasad G, Coyle DH and Mcginnity TM 2011 A Recurrent Quantum Neural Network model enhances the EEG signal for an improved brain-computer interface *Semi. Assist. Liv* 1-6.
[15] Ganjefar S, Tofighi M and Karami H 2015 Fuzzy wavelet plus a quantum neural network as a design base for power system stability enhancement *Neural. Networks* **71** 172-81.
[16] Juang CF 2002 A TSK-type recurrent fuzzy network for dynamic systems processing by neural network and genetic algorithms *IEEE. T. Fuzzy. Syst* **10** 155-70.
[17] Juang CF and Lin CT 1999 A recurrent self-organizing neural fuzzy inference network *IEEE. T. Neural. Networ* **10** 828-45.
[18] Lin CJ and Chin CC 2004 Prediction and identification using wavelet-based recurrent fuzzy neural networks *IEEE. T. Syst. Man. Cy. B* **34** 2144-58.
[19] Yilmaz S and Oysal Y 2010 Fuzzy wavelet neural network models for prediction and identification of dynamical systems *IEEE. T. Neural. Networ* **21** 1599-609.
[20] Gao Y and Meng JE 2005 NARMAX time series model prediction: Feedforward and recurrent fuzzy neural network approaches *Fuzzy. Set. Syst* **150** 331-50.
[21] Juang C-F, Huang S-T and Duh F-B 2006 Mold temperature control of a rubber injection molding machine by TSK-type recurrent neural fuzzy network *Neurocomputing* **70** 559-67.