Proceedings of the Fourth IASTED International Conference
POWER AND ENERGY SYSTEMS (AsiaPES 2008)
April 2-4, 2008 Langkawi, Malaysia
ISBN CD: 978-0-88986-732-1

DESIGN OF RECURRENT NEURAL NETWORK POWER SYSTEM STABILIZER BASED ON GENETIC ALGORITHM

Chun-Jung Chen and Tien-Chi Chen

Department of Engineering Science
National Cheng Kung University
Tainan, Taiwan
Email: tchichen@mail.ncku.edu.tw

ABSTRACT
This paper presents a novel recurrent neural network power system stabilizer (RNNPSS) based on genetic algorithm (GA) for a multi machines power system. The proposed PSS consists of a recurrent neural network identifier (RNNI) and recurrent neural network controller (RNNC) that identifies the power system and supplies an adaptive signal to the governor and exciter to damp power system oscillations. Both the RNNI and RNNC of the PSS are trained offline by using GA to determine the optimal learning rates. The proposed PSS is simulated for three-generator power system, which results demonstrate the effectiveness and performance of the proposed PSS.

KEY WORDS
genetic algorithm**,** recurrent neural network, power system stabilizer.

1. Introduction

Power systems are complex nonlinear systems that often exhibit low-frequency oscillations due to dynamic loads and/or sudden power system transmission events. Because power systems are highly nonlinear, the PSS provides supplementary control signals to the exciter and governor systems of the generator to dampen these oscillations and improve the generator's dynamic performance. Three basic tuning techniques have been successfully utilized with PSS applications: the phase compensation method, the root-locus method and linear quadratic regulator (LQR) [1-3]. These three methods provide excellent performance at a given operating point. But the gain settings of these stabilizers were solved and based on linear models. Since the power systems are highly nonlinear, the operating ranges of the conventional stabilizer are usually limited.

In recent years, a number of adaptive control approaches have been proposed to improve their performance in nonlinear cases. The artificial intelligent (AI) approaches, including a fuzzy based stabilizer, and a neural network based stabilizer. Simulation results demonstrate that the AI PSS can effectively damp the oscillation of power systems [4]. However, the neural network proposed in the past was at least three layers. The Chaturvedi, D. K. first proposed a generalized neuron-based PSS to reduce the number of neuron [5]. In this research, a new two-layer recurrent neural networks PSS (RNNPSS) was proposed. The RNNPSS lies in updating the weights base on back propagation algorithm. The weights of the neural networks are updated by using a steepest descent algorithm (gradient algorithm). Simulation results demonstrate the proposed RNNPSS have better performance comparing with the LQR method.

However, an RNNPSS cannot be guaranteed to be convergent unless proper learning rate values have been chosen. The PSS is very easy to be uncontrollable if the RNNI cannot identify the power generators due to incorrect choice of learning rates. Moreover, the choices of learning rates for RNNPSS are usually determined by tried and error [6, 7]. Thus, the time period necessary to find the adaptive values for the learning rate is sometimes unacceptably long. To overcome this drawback, an optimal learning rate based on GA for RNNPSS is proposed in this paper. Simulation results demonstrate the GA based on RNNPSS has an effectively performance and wider operating point range [8].

2. Configuration of Power System

The configuration of three machine power system and RNNPSS is shown in Figure 1. One PSS controls one generator. The RNNPSS provides a supplementary control signals to the exciter and governor systems of the generator. The generator states are phase angle deviation $\Delta\delta$, rotation speed deviation $\Delta\omega$, field voltage deviation Δe_{FD} and q-axis voltage deviation Δe_q respectively. The RNNPSS is used to damp these oscillations for power system.

3. The Proposed RNNPSS

Figure 2 shows the architecture of the RNNPSS, which consists of a RNNI and a RNNC. The RNNI contains five inputs, which are the output states of power generators at time (t–1) and the output of the RNNC. The RNNC also contains five inputs, which are the output states of the

power generator and the output of the RNNC at time (*t* -1).

3.1 The RNNI

Figure 3 shows the neural network structure of the RNNI, which consists of a two-layered network structure, input layer and output layer. The input layer and output layer have m_I neurons and n_I neurons, respectively. The mathematical operation of the RNNI can be expressed as follows.

3.1.1 Input layer

The input and output of the *j*th neuron of the input layer can be expressed as:

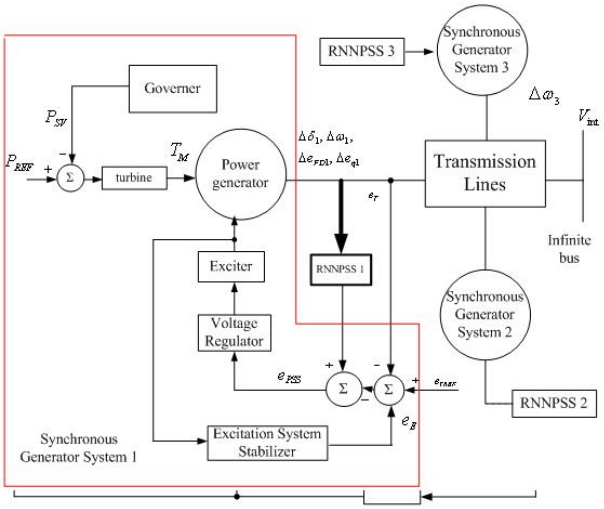


Figure 1. Three machine power system

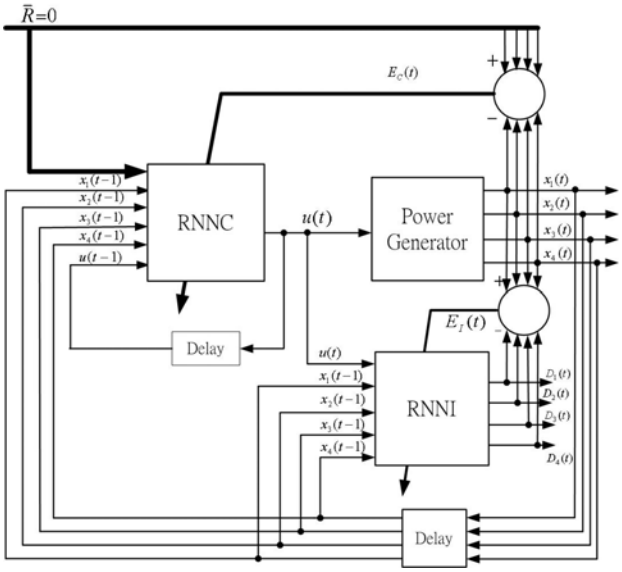


Figure 2. The configuration of RNNPSS and power system

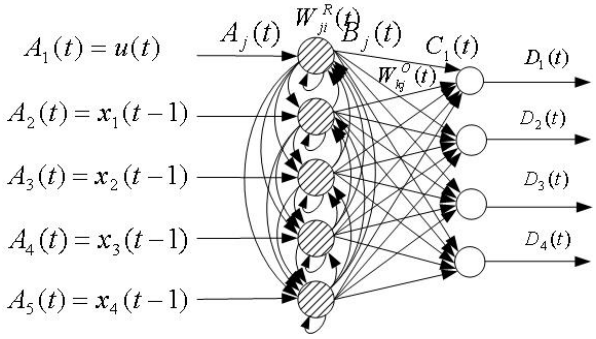


Figure 3. The network structure of the RNNI

$$A_i(t) = u(t) + \sum_{i=1}^{m_I} W_{ji}^R(t) B_i(t-1) \tag{1.a}$$

$$A_j(t) = x_j(t-1) + \sum_{i=1}^{m_I} W_{ji}^R(t) B_i(t-1), \quad j = 2,.....,m_I \tag{1.b}$$

and

$$B_j(t) = f(A_j(t)) = \frac{e^{A_j(t)} - e^{-A_j(t)}}{e^{A_j(t)} + e^{-A_j(t)}}, \quad j = 1,.....,m_I \tag{2}$$

where $A_j(t)$ and $B_j(t)$ are the input and output of the *j*th neuron of the input layer, $W_{ji}^R(t)$ is the recurrent weights.

3.1.2 Output layer

The input and output of the *k*th neuron of the output layer are equal and can be expressed as:

$$D_k(t) = C_k(t) = \sum_{j=1}^{m_I} W_{kj}^O(t) B_j(t), \quad k = 1,.....,n_I \tag{3}$$

where $C_k(t)$ and $D_k(t)$ are the input and output of the *k*th neuron of the output layer, $W_{kj}^O(t)$ is the connected weights.

Due to the properties of guaranteed convergence, and minimizing the performance function, an error function for the RNNI is defined as

$$E_I(t) = \frac{1}{2}\sum_{k=1}^{n_I} \left(x_k(t) - D_k(t)\right)^2 \tag{4}$$

where $x_k(t),\ k = 1,...,n_I$ are the power system state outputs.

Using the steepest descent algorithm and according to (1)-(4), the weights $W_{kj}^O(t)$ and $W_{ji}^R(t)$ can be adjusted as

$$W_{kj}^O(t+1) = W_{kj}^O(t) - \lambda_I \frac{\partial E_I(t)}{\partial W_{kj}^O(t)} = W_{kj}^O(t) + \lambda_I \left(x_k(t) - D_k(t)\right) B_j(t) \tag{5}$$

$$W_{ji}^R(t+1) = W_{ji}^R(t) - \lambda_I \frac{\partial E_I(t)}{\partial W_{ji}^R(t)} = W_{ji}^R(t) + \lambda_I \sum_{k=1}^{n_I} \left[\left(x_k(t) - D_k(t)\right) W_{kj}^O(t) \left(1 - B_j^2(t)\right) B_j(t-1) \right] \tag{6}$$

where λ_I is the learning rate of RNNI.

3.2 The RNNC

The network structure of the RNNC is shown in Figure 4. The RNNC consists of a two-layered network structure, input layer and output layer. The input layer and output layer have m_C neurons and one neuron, respectively. The mathematical operation of the RNNC can be expressed as follows.

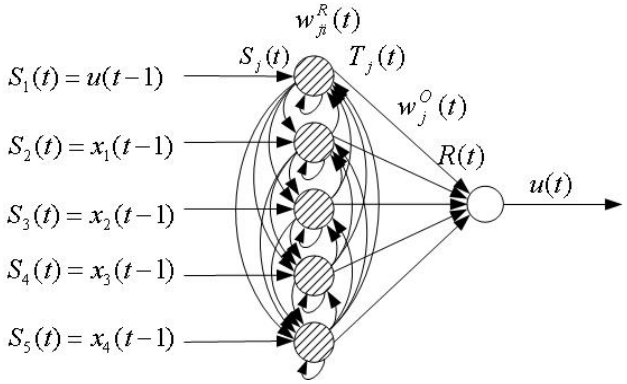


Figure 4. The network structure of the RNNC

3.2.1 Input layer

The input and output of the *j*th neuron of the input layer can be expressed as:

$$S_1(t) = u(t-1) + \sum_{i=1}^{m_C} w_{ji}^R(t) T_i(t-1) \tag{7.a}$$

$$S_j(t) = x_j(t-1) + \sum_{i=1}^{m_C} w_{ji}^R(t) T_i(t-1), \quad j = 2,.....,m_C \tag{7.b}$$

and

$$T_j(t) = f(S_j(t)) = \frac{e^{S_j(t)} - e^{-S_j(t)}}{e^{S_j(t)} + e^{-S_j(t)}}, \quad j = 1,.....,m_C \tag{8}$$

where $S_j(t)$ and $T_j(t)$ are the input and output of the *j*th neuron of the input layer, $w_{ji}^R(t)$ is the recurrent weights.

3.2.2 Output layer

The output layer has one neuron, in which input and output are equal and can be expressed as:

$$u(t) = R(t) = \sum_{j=1}^{m_C} w_j^O(t) T_j(t) \tag{9}$$

where $R(t)$ and $u(t)$ are the input and output of the *k*th neuron of the output layer, $w_j^O(t)$ is the connected weights.

Due to the properties of guaranteed convergence, and minimizing the performance function, an error function for the RNNC is defined as

$$E_C(t) = \frac{1}{2} \sum_{k=1}^{m_C} \left(r_k(t) - x_k(t) \right)^2 \tag{10}$$

where $r_k(t)$ are the power system reference commands.

Using the steepest descent algorithm and according to (7)-(10), the weights $w_j^O(t)$ and $w_{ji}^R(t)$ can be adjusted as

$$w_j^O(t+1) = w_j^O(t) - \lambda_C \frac{\partial E_C(t)}{\partial w_j^O(t)} \tag{11}$$

$$= w_j^O(t) - \lambda_C + \sum_{k=1}^{m_C} \left[\left(r_k(t) - x_k(t) \right) W_{k1}^O(t) \left(1 - B_1^2(t) \right) \right] T_j(t)$$

$$w_{ji}^R(t+1) = w_{ji}^R(t) - \lambda_C \frac{\partial E_C(t)}{\partial w_{ji}^R(t)} \tag{12}$$

$$= w_{ji}^R(t) + \lambda_C \sum_{k=1}^{m_C} \left[\left(r_k(t) - x_k(t) \right) W_{k1}^O(t) \left(1 - B_1^2(t) \right) \right] \cdot w_j^O(t) \left(1 - T_j^2(t) \right) T_i(t-1)$$

where λ_C is the learning rate of RNNC.

3.3 RNNPSS Training Algorithm

According to the above formula, the RNNPSS training algorithm is expressed as follows.

Step 1: For t=1, arbitrarily initialize the RNNI weights $W_{ji}^R(t)$, $W_{kj}^O(t)$ and RNNC weights $w_{ji}^R(t)$, $w_j^O(t)$. Select the RNNI learning rate λ_I and RNNC learning rate λ_C using GA for Optimal Learning Rates of RNNPSS shown in section 5.

Step 2: Calculate RNNI operation $A_j(t)$, $B_j(t)$, $C_j(t)$, and $D_k(t)$ using equations (1.a), (1.b), (2) and (3) in sequence. Also calaulate the power system state outputs.

Step 3: Update RNNI weights $W_{kj}^O(t+1)$ and $W_{ji}^R(t+1)$ using equations (5) and (6) in sequence.

Step 4: Calculate RNNC operation $S_j(t)$, $T_j(t)$ and $u(t)$ using equations (7.a), (7.b), (8) and (9) in sequence.

Step 5: Update RNNC weights $w_j^O(t+1)$ and $w_{ji}^R(t+1)$ using equations (11) and (12) in sequence. Let t=t+1, return step 2.

4. Genetic Algorithm

In recent years, GA has been proposed for finding the optimal value in many applications. In [8], a conventional PSS simulation for multi-machine power systems based on GA was proposed. The PSS was simulated using Matlab toolbox that uses GA to solve the optimal damping coefficients. The simulation results show that the GA allows simultaneous tuning of the power system damping controller under different operating conditions.

The GA is first initialized a population of binary code sets. These binary code sets represent the initial values of the learning rate of the neural networks. This technique provides a powerful tool for finding the optimal learning rates of a neural network by minimizing the error function of the neural network selection and genetics.

The GA algorithm process is shown in Figure 5, which includes reproduction, crossover, mutation, and fitness calculation. The GA is operated using binary codes

on a given population, applying the principle of survival of the fittest to produce better and better approximations to a solution. Competition among individuals in each cycle results in a population in which the fittest individuals are selected over the weaker ones. In each generation, a new set of approximations is created by selecting the individuals according to their level of fitness in the application domain and breeding them together using operators borrowed from natural genetics. Thus, the solutions for the RNNPSS are successively improved with respect to the search objective by replacing the least fit individuals with new ones, better suited to the environment, just as in natural evolution.

In order to find the optimal learning rates for the neural networks, the RNNI and RNNC are first trained off-line using GA. The range of learning rates λ_I and λ_C can be expressed as

$$\lambda_I(t) = \lambda_{I\min}(t) + \frac{S_I}{2^L - 1}\left(\lambda_{I\max}(t) - \lambda_{I\min}(t)\right) \quad (13)$$

and

$$\lambda_C(t) = \lambda_{C\min}(t) + \frac{S_C}{2^L - 1}\left(\lambda_{C\max}(t) - \lambda_{C\min}(t)\right) \quad (14)$$

where λ_{Imin}, λ_{Imax}, λ_{Cmin} and λ_{Cmax} are the maximum and minimum value of λ_I and λ_C, $S_I(t)$ and $S_C(t)$ are the strings of binary code with number L. The fitness for RNNPSS is defined as

$$\text{Fitness} = \sum_{t=1}^{5000}\left(E_I(t) + E_C(t)\right) \quad (15)$$

where $E_I(t)$ and $E_C(t)$ are given in equations (4) and (10).

Based on the foregoing discussion, a GA for optimal learning rates of RNNPSS is expressed as follows.

Step 1. Initially, arbitrarily 30 sets string $S_I(t)$ and $S_C(t)$ and calculate $\lambda_I(t)$ and $\lambda_C(t)$ for individual strings using equations (13) and (14). For individual strings, perform the RNNPSS Training Algorithm shown in section 3.3 and evaluate their fitness using equation (15).

Step 2. A string pool is created by the reproduction, crossover and mutation process or the initial 30 sets string. If the smallest fitness in the string pool is smaller than a preset small value, the optimal string is achieved; otherwise, go to step 3.

Step 3. Perform the reproduction process, 10 newly reproduced strings are copied according to higher fitness in the string pool.

Step 4. Perform the crossover process, 10 sets string are created by swapping the newly reproduced strings. For individual strings, perform the RNNPSS Training Algorithm and evaluate their fitness.

Step 5. Perform the mutation process, 10 sets string are created by mutating the newly reproduced strings. For individual strings, perform the RNNPSS Training Algorithm and evaluate their fitness. Go to step 2.

5. Computer Simulations

In this section, some simulation results are presented to evaluate the effectiveness of the proposed control scheme. The mathematical package MATLAB was used for the simulations. To test the effectiveness of the control scheme, simulations were executed using two types of control schemes, the proposed RNNPSS and the linear quadratic regulator (LQR) stabilizer, described as follows.

The nominal state equation of a three-machine power system shown in Figure 1 is given as [9]:

$$\dot{\mathbf{x}}(\mathbf{t}) = \mathbf{Ax(t)} + \mathbf{Bu(t)} + \mathbf{Tw(t)} \quad (16)$$

where **x**, **u** and **w** are state vector, input vector and disturbance vector, respectively, which can be expressed as follows:

$$\mathbf{x(t)} = [\Delta\delta_1(t), \Delta\omega_1(t), \Delta e_{q1}(t), \Delta e_{FD1}(t), \Delta\delta_{12}(t), \Delta\omega_2(t), \Delta e_{q2}(t), \Delta e_{FD2}(t), \Delta\delta_3(t), \Delta\omega_3(t), \Delta e_{q3}(t), \Delta e_{FD3}(t)]^T$$

$$\mathbf{u} = [u_1, u_2, u_3]^T,$$

$$\mathbf{w} = [\Delta V_{ref1}, \Delta T_{m1}, \Delta V_{ref2}, \Delta T_{m2}, \Delta V_{ref3}, \Delta T_{m3}]^T$$

The system matrix **A**, **B**, and **T** are given as [9].

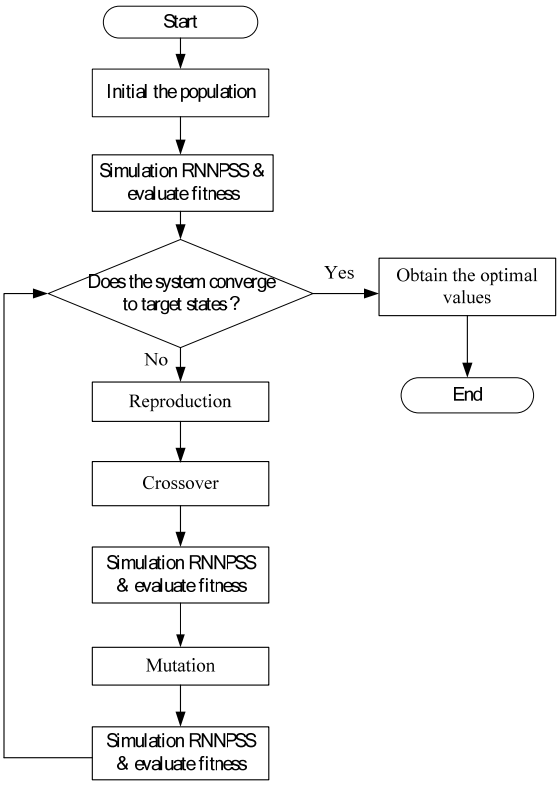


Figure 5. The flowchart of the genetic algorithm

5.1 The proposed RNNPSS

The GA for optimal learning rates of RNNPSS was used to determine the optimal learning rates of λ_I and λ_C, which are 0.0057 and 0.0315 respectively. The initial weight values of RNNPSS, $W_{ji}^R(t)$, $W_{ij}^O(t)$ and $w_{ji}^R(t)$, $w_j^O(t)$, are all chosen arbitrarily to be between 0.1~0.5. The RNNPSS Training Algorithm was used to update weight values of RNNPSS and calculate the power system state outputs.

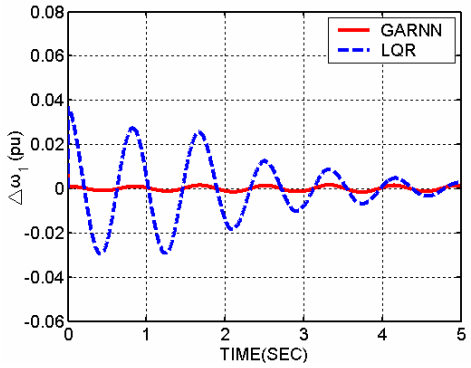
5.2 The linear quadratic regulator (LQR) stabilizer [9]

The control input was **u = - Kx .**

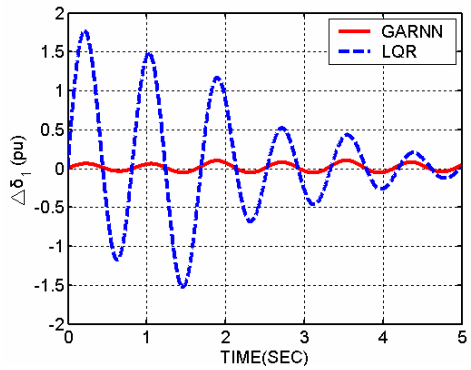
Figures 6(a)-(d) show the dynamic state responses of machines 1, 2 and 3 for a disturbance $\mathbf{w} = [\Delta V_{ref1}, \Delta T_{m1}, \Delta V_{ref2}, \Delta T_{m2}, \Delta V_{ref3}, \Delta T_{m3}]^T$

$= [0 \quad 0.01 \quad 0 \quad 0 \quad 0 \quad 0]^T$ pu

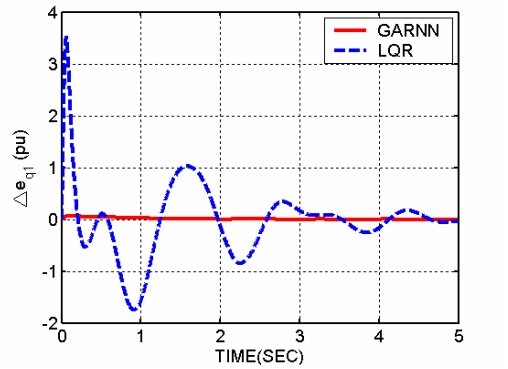
using the proposed RNNPSS and the LQR stabilizer. As these figures reveal, that improvement in dynamic state responses of machines 1, 2 and 3 are achieved by the proposed RNNPSS. However, in spite of a disturbance $\Delta T_{m1} = 0.01$ pu in machine 1, the dynamic state responses of machines 1, 2 and 3 obtained by applying the LQR stabilizer had serious oscillations. These figures indicate the power system oscillations were effectively damped by proposed RNNPSS.



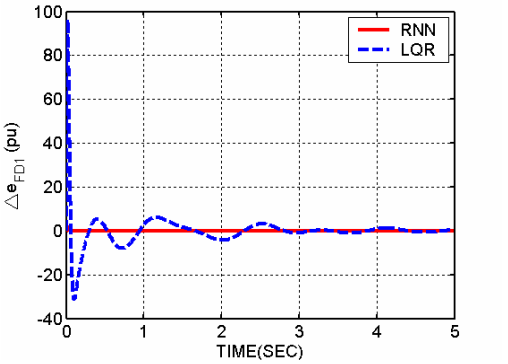
(a)



(b)



(c)



(d)

Figure 6. The state responses of machine 1 for a disturbance $w = [0 \quad 0.01 \quad 0 \quad 0.01 \quad 0 \quad 0.01]^T$ pu using the LQR stabilizer and RNNPSS, (a) $\Delta\omega_1$, (b) $\Delta\delta_1$,(c) Δe_{q1}, (d) Δe_{FD1}.

To further test the performance of the proposed RNNPSS control scheme under different operating conditions, the nominal system parameter matrix A_{11} and A_{12} is changed with 10%.

Figures 7(a)-(d) show the dynamic state responses of machines 1, 2 and 3 for a disturbance $\mathbf{w} = [\Delta V_{ref1}, \Delta T_{m1}, \Delta V_{ref2}, \Delta T_{m2}, \Delta V_{ref3}, \Delta T_{m3}]^T$

$= [0 \quad 0.01 \quad 0 \quad 0.01 \quad 0 \quad 0.01]^T$ pu

using the proposed RNNPSS and the LQR stabilizer. With the proposed RNNPSS being augmented, the dynamic state responses of machines 1, 2 and 3 shown in the figures above indicate that good performance improvement can be achieved for the system parameter variations. By contrast, the dynamic state responses of machines 1, 2 and 3 of the LQR stabilizer are easily affected by a change in the system parameter variations. Therefore, an improvement in the dynamic state responses of multi-machine system using the proposed RNNPSS control scheme can be observed.

6. Conclusion

The RNNPSS proposed in this paper shows faster convergence than the LQR stabilizer in a multi-machine

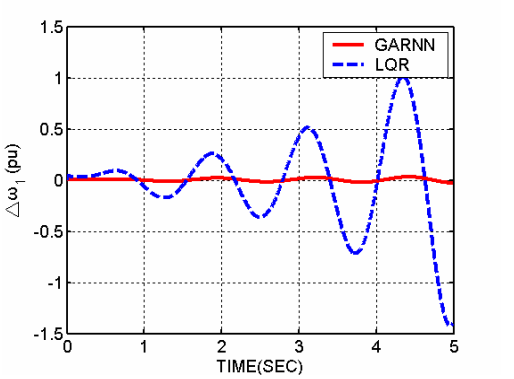
power system, because the proposed GA based neural network was first trained off-line to determine the optimal values of the learning rates. Otherwise, the RNNPSS consists of just two layers. Therefore the time consumption of the damping oscillations is lower than with conventional methods. Moreover, the operating range of the RNNPSS is greater than that of the LQR and conventional three layer neural networks, since the RNNPSS can greatly reduce system complexity and effectively damp system oscillations.

Acknowledgements

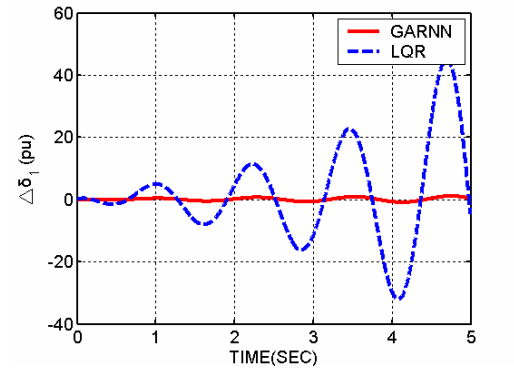
The authors would like to express their appreciation to National Science Council for supporting under contact NSC96-2221-E-006-037.

References

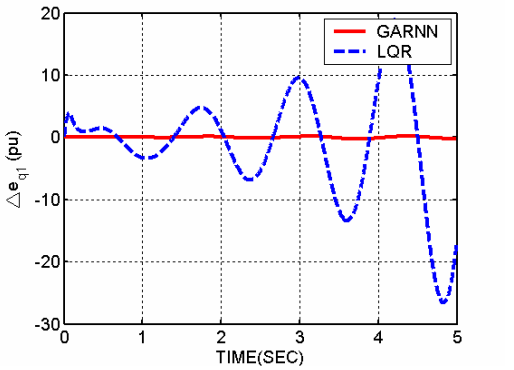
[1] E. V. Larsen & D. A. Swann, Applying power system stabilizers part-I: general concepts, *IEEE Trans. Power Appar. Syst., 100*, 1981, 3017-3024.
[2] W. C. Chan & Y. Y. Hsu, An optimal variable structure stabilizer for power system stabilization, *IEEE Trans. Power Appar. Syst., 102*, 1983, 1738-1746.
[3] T. C. Chen & L. R. Chang-Chien, Optimal pole assignment to stabilize a power system, *Journal of Control Systems and Technology*, *4*, 1996, 119-126.
[4] B. M. Nomikos & C. D. Vournas, Investigation of induction machine contribution to power system oscillations, *IEEE Trans. Power Syst., 20*, 2005, 916-925.
[5] D. K. Chaturvedi, O. P. Malik, & P. K. Kalra, Experimental studies with a generalized neuron-based power system stabilizer, *IEEE Trans. Power Syst., 19*, 2004, 1445-1453.
[6] R. Gupta, B. Bandyopadhyay, & A. M. Kulkarni, Design of power system stabilizer for single-machine system using robust periodic output feedback controller, *IEE Proc. Generation, Transmission and Distribution, 150*, 2003, 211-216.
[7] Y. Toshihiko, N. Kaazushi, K. Akihro, & T. Katsuyuki, Neural network with variable type connection weights for autonomous obstacle avoidance on a prototype of six-wheel type intelligent wheelchair, *International Journal of Innovative Computing, Information and Control* (*IJICIC*), *2*, 2006, 1165-1177.
[8] G. Q. Hu, D. J. Xu, & R. M. He, Genetic algorithm based design of power system stabilizers, *IEEE* International *Conf. on Electric Utility Deregulation, Restructuring and Power Technologies (DRPT2004), 1*, 2004, 167-171.
[9] P. Hoang & K. Tomsovic, Design and analysis of an adaptive fuzzy power system stabilizers, *IEEE Trans. Energy Conversion, 11*, 1996, 455-461.



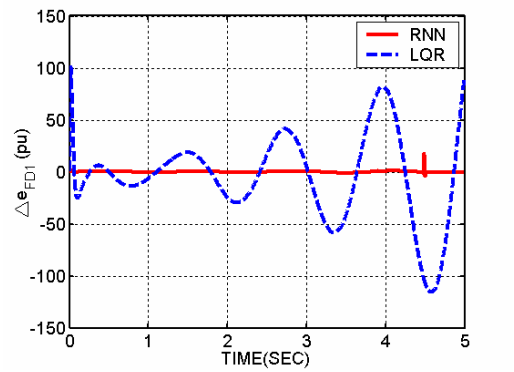
(a)



(b)



(c)



(d)

Figure 7. The state responses of machine 1 for a disturbance $w=[0 \quad 0.01 \quad 0 \quad 0 \quad 0 \quad 0]^T$ pu using the LQR stabilizer and RNNPSS, (a) $\Delta\omega_1$, (b) $\Delta\delta_1$,(c) Δe_{q1}, (d) Δe_{FD1}