CHAPTER 5

DESIGN, TRAINING AND IMPLEMENTATION OF NEURAL NETWORK CONTROLLERS

5.1 SPECIFICATIONS OF PROPOSED NEURAL NETWORK CONTROLLERS

Type of Neural Network: Feed Forward back propagation network
Number of layers: 3
Neurons: 20 in 1st layer (input), 10 in 2nd layer (hidden), 2 or 3 in 3rd layer (output)
Activation functions: Logarithmic Sigmoid for 1st and 2nd layers, Linear for 3rd layer
Training algorithm: Lavenberg-Marquardt (LM) back propagation
Training method: Supervised training

5.2 PROPOSED SCHEME OF TRAINING AND USING THE ANN CONTROLLERS FOR AGC

Neural network with power system in training mode:

As an Illustration, the proposed scheme of training and using the ANN controllers for AGC has been demonstrated here for one of the power system models namely, the two area thermal-thermal (non reheat) power system.

Fig. 5.1 shows interface of neural network with the power system in training mode, for a two area thermal-thermal (non reheat) power system (9 system states).
Trained neural network with power system as controller:

The interface of trained neural network with the power system as a real time controller is shown in Fig. 5.2.
5.3 THE OVERALL PROCEDURE ADOPTED FOR DEVELOPMENT OF ANN CONTROLLERS WITH PROGRAMING IN MATLAB

The programing in MATLAB for developing a neural network controller for one of the power system models, i.e., the thermal-thermal (non reheat) power system is described here. The similar procedure has been adopted for developing the controllers for other power system models under consideration.

**Step 1: Generation of training data**

The system state equations (equations for \( x_1 \) to \( x_9 \)) and control input equations (equations for \( u_1 \) & \( u_2 \)) in discrete form for a two area thermal-thermal (non reheat) power system are given in section 3.4.1. These equations have been used to generate the training data.

A program has been executed in MATLAB which generates values of load disturbances in two areas (\( d_1 \) & \( d_2 \)) randomly in the range of 0 to 1 percent p.u. (i.e., between 0 and 0.01 pu). The program evaluates the equations and stores the values of all the variables after each iteration. Since the time of study and the sampling time have been chosen as 15 sec. and 0.01 sec. respectively, a total of 1501 samples are collected for each variable for one pair of load disturbances. These variables are stored in workspace. All such variables make one data set. Thus one data set comprises of 13 variables (\( x_1 \) to \( x_9 \), \( u_1 \), \( u_2 \), \( d_1 \), \( d_2 \)). For \( d_1 \) or \( d_2 \), all the 1501 values in one data set are equal (since we assume load disturbances of constant magnitude). About 60 to 100 such data sets at different combinations of load disturbances have been collected and saved for each power system model under consideration. Following MATLAB program generates and saves the data sets.

\[
i=1; \\
j=1500; \\
d1(i)=\text{abs}(	ext{randn})/200,\
\]
$$d_2(i) = \text{abs/randn}/200,$$

for n=1:71,

x1(i)=0,

x2(i)=0,

x3(i)=0,

x4(i)=0,

x5(i)=0,

x6(i)=0,

x7(i)=0,

x8(i)=0,

x9(i)=0,

u1(i)=0,

u2(i)=0,

for k=i:j,

x1(k+1) = 0.9995*x1(k) + 0.06*[x2(k) - x7(k) - d1(k)],

x2(k+1) = 0.975*x2(k) + 0.025*x3(k),

x3(k+1) = -0.05208*x1(k) + 0.875*x3(k) + 0.125*u1(k),

x4(k+1) = 0.9995*x4(k) + 0.06*[x5(k) + x7(k) - d2(k)],

x5(k+1) = 0.975*x5(k) + 0.025*x6(k),

x6(k+1) = -0.05208*x4(k) + 0.875*x6(k) + 0.125*u2(k),

x7(k+1) = 0.004442*x1(k) - 0.004442*x4(k) + x7(k),

x8(k+1) = 0.00425*x1(k) + 0.01*x7(k) + x8(k),

x9(k+1) = 0.00425*x4(k) - 0.01*x7(k) + x9(k),

u1(k+1) = -0.4226*x1(k) - 0.8294*x2(k) - 0.1538*x3(k) + 0.063*x4(k) + 0.1156*x5(k) + 0.02*x6(k) + 0.2737*x7(k) - x8(k),

u2(k+1) = 0.063*x1(k) + 0.1156*x2(k) + 0.02*x3(k) - 0.4226*x4(k) - 0.8294*x5(k) - 0.1538*x6(k) - 0.2737*x7(k) - x9(k),

d1(k+1) = d1(k),

d2(k+1) = d2(k),

end

i = i + 1501;
j=j+1501;
d1(i)=abs(randn)/200,
d2(i)=abs(randn)/200,
end

**Step 2: Training of neural network**

An input vector ‘P’ is defined which consists of all the system states ($x_1$ to $x_9$) and load disturbances ($d_1$ & $d_2$). The target vector ‘T’ is defined which consists of the control inputs ($u_1$ & $u_2$) since the supervised training method has been chosen. A feedforward neural network has been defined with three layers. First layer has 20 neurons, second has 10 and third has 2 neurons. The activation functions are chosen as logarithmic sigmoid for first and second layers and linear for third layer. The training algorithm is chosen as Lavenberg-Marquardt (LM) backpropagation. Certain number of epochs (between 100 to 200) has been set for training. One epoch corresponds to scanning of all the training data once. The number of epochs is not fixed one. The specified neural networks have been trained several times with different number of data sets and different number of epochs and the best trained network has been retained and saved. During training, ‘P’ is the input to the network and ‘T’ is the target. During each epoch, the network adjusts the weights of neural connections such that it can give output closer to that of the target ‘T’. The error is backpropagated after each epoch. The error goal has been kept to a very small value of $1 \times 10^{-11}$. For the best trained network obtained so far for this model after many attempts, the error of about $1 \times 10^{-9}$ has been achieved which is quite acceptable. The MATLAB program to get a trained neural network for this model is given below.

```matlab
P=[x1; x2; x3; x4; x5; x6; x7; x8; x9; d1; d2];
T=[u1; u2];
net=newff([minmax(P)],[20,10,2],{'logsig','logsig','purelin'},'trainlm');
```
sim(net,P);
net.trainParam.show = 1;
net.trainParam.epochs = 100;
net.trainParam.goal = 1e-11;
net.trainParam.mem_reduc = 10;
[net,tr]=train(net,P,T);
sim(net,P)

Step 3: Obtaining performance of ANN controller for random load disturbances

The trained network is now ready to work as controller. For any values of load disturbances, the appropriate outputs are given by the network, which act as the control inputs of the power system. The performance of trained neural network has been tested for many pairs of load disturbances and the network has been giving a superior control as compared to integral control and quite matching control as compared to optimal control.

The MATLAB program to obtain the neural network controller performance is given below for sample values of load disturbances, e.g., $d_1 = 0.0081, d_2 = 0.001$. All the variables, viz., system states, control inputs and load disturbances have been suffixed with ‘_N’ to identify them as variables corresponding to ANN control strategy. The time of study is 15 seconds (1501 samples).

x1_N(1)=0,
x2_N(1)=0,
x3_N(1)=0,
x4_N(1)=0,
x5_N(1)=0,
x6_N(1)=0,
x7_N(1)=0,
x8_N(1)=0,
x9_N(1)=0,
\[ u_{1,N}(1) = 0, \]
\[ u_{2,N}(1) = 0, \]
\[ d_{1,N}(1) = 0.0081, \]
\[ d_{2,N}(1) = 0.001, \]

for \( k = 1:1500, \)
\[ x_{1,N}(k+1) = 0.9995 x_{1,N}(k) + 0.06 \cdot (x_{2,N}(k) - x_{7,N}(k) - d_{1,N}(k)), \]
\[ x_{2,N}(k+1) = 0.975 x_{2,N}(k) + 0.025 x_{3,N}(k), \]
\[ x_{3,N}(k+1) = -0.05208 x_{1,N}(k) + 0.875 x_{3,N}(k) + 0.125 u_{1,N}(k), \]
\[ x_{4,N}(k+1) = 0.9995 x_{4,N}(k) + 0.06 \cdot (x_{5,N}(k) + x_{7,N}(k) - d_{2,N}(k)), \]
\[ x_{5,N}(k+1) = 0.975 x_{5,N}(k) + 0.025 x_{6,N}(k), \]
\[ x_{6,N}(k+1) = -0.05208 x_{4,N}(k) + 0.875 x_{6,N}(k) + 0.125 u_{2,N}(k), \]
\[ x_{7,N}(k+1) = 0.0044422 x_{1,N}(k) - 0.0044422 x_{4,N}(k) + x_{7,N}(k), \]
\[ x_{8,N}(k+1) = 0.00425 x_{1,N}(k) + 0.01 x_{7,N}(k) + x_{8,N}(k), \]
\[ x_{9,N}(k+1) = 0.00425 x_{4,N}(k) - 0.01 x_{7,N}(k) + x_{9,N}(k), \]
\[ d_{1,N}(k+1) = d_{1,N}(k), \]
\[ d_{2,N}(k+1) = d_{2,N}(k), \]

\[ R = [x_{1,N}(k+1); x_{2,N}(k+1); x_{3,N}(k+1); x_{4,N}(k+1); x_{5,N}(k+1); x_{6,N}(k+1); x_{7,N}(k+1); x_{8,N}(k+1); x_{9,N}(k+1); d_{1,N}(k+1); d_{2,N}(k+1)]; \]

\[ b = \text{sim}(\text{net},R), \]
\[ u_{1,N}(k+1) = b(1,:), \]
\[ u_{2,N}(k+1) = b(2,:), \]
end

**Step 4: Obtaining performances of optimal and integral controllers**

For the purpose of comparison, the response of system states for the same load disturbances has also been obtained by optimal control strategy and integral control strategy with following MATLAB program. The suffixes ‘_O’ (for optimal) and ‘_I’ (for integral) have been attached to the variables to identify them separately.
Program for obtaining optimal control performance

\[
x_{1,0}(1) = 0, \\
x_{2,0}(1) = 0, \\
x_{3,0}(1) = 0, \\
x_{4,0}(1) = 0, \\
x_{5,0}(1) = 0, \\
x_{6,0}(1) = 0, \\
x_{7,0}(1) = 0, \\
x_{8,0}(1) = 0, \\
x_{9,0}(1) = 0, \\
u_{1,0}(1) = 0, \\
u_{2,0}(1) = 0, \\
d_{1,0}(1) = 0.0081, \\
d_{2,0}(1) = 0.001,
\]

for \( k = 1:1500, \)

\[
x_{1,0}(k+1) = 0.9995x_{1,0}(k) + 0.06(x_{2,0}(k) - x_{7,0}(k) - d_{1,0}(k)), \\
x_{2,0}(k+1) = 0.975x_{2,0}(k) + 0.025x_{3,0}(k), \\
x_{3,0}(k+1) = -0.05208x_{1,0}(k) + 0.875x_{3,0}(k) + 0.125u_{1,0}(k), \\
x_{4,0}(k+1) = 0.9995x_{4,0}(k) + 0.06(x_{5,0}(k) + x_{7,0}(k) - d_{2,0}(k)), \\
x_{5,0}(k+1) = 0.975x_{5,0}(k) + 0.025x_{6,0}(k), \\
x_{6,0}(k+1) = -0.05208x_{4,0}(k) + 0.875x_{6,0}(k) + 0.125u_{2,0}(k), \\
x_{7,0}(k+1) = 0.0044422x_{1,0}(k) - 0.0044422x_{4,0}(k) + x_{7,0}(k), \\
x_{8,0}(k+1) = 0.00425x_{1,0}(k) + 0.01x_{7,0}(k) + x_{8,0}(k), \\
x_{9,0}(k+1) = 0.00425x_{4,0}(k) - 0.01x_{7,0}(k) + x_{9,0}(k), \\
u_{1,0}(k+1) = -0.4226x_{1,0}(k) - 0.8294x_{2,0}(k) - 0.1538x_{3,0}(k) + 0.063x_{4,0}(k) + 0.1156x_{5,0}(k) + 0.02x_{6,0}(k) + 0.2737x_{7,0}(k) - x_{8,0}(k), \\
u_{2,0}(k+1) = 0.063x_{1,0}(k) + 0.1156x_{2,0}(k) + 0.02x_{3,0}(k) - 0.4226x_{4,0}(k) - 0.8294x_{5,0}(k) - 0.1538x_{6,0}(k) - 0.2737x_{7,0}(k) - x_{9,0}(k), \\
d_{1,0}(k+1) = d_{1,0}(k), \\
d_{2,0}(k+1) = d_{2,0}(k),
\]
end
Program for obtaining integral control performance

\[ \begin{align*}
x_1_{(I)}(1) &= 0, \\
x_2_{(I)}(1) &= 0, \\
x_3_{(I)}(1) &= 0, \\
x_4_{(I)}(1) &= 0, \\
x_5_{(I)}(1) &= 0, \\
x_6_{(I)}(1) &= 0, \\
x_7_{(I)}(1) &= 0, \\
u_1_{(I)}(1) &= 0, \\
u_2_{(I)}(1) &= 0, \\
d_1_{(I)}(1) &= 0.0081, \\
d_2_{(I)}(1) &= 0.001, \\
\end{align*} \]

for \( k=1:1500, \)

\[ \begin{align*}
x_1_{(I)}(k+1) &= 0.9995x_1_{(I)}(k)+0.06[x_2_{(I)}(k)-x_7_{(I)}(k)-d_1_{(I)}(k)], \\
x_2_{(I)}(k+1) &= 0.975x_2_{(I)}(k)+0.025x_3_{(I)}(k), \\
x_3_{(I)}(k+1) &= -0.05208x_1_{(I)}(k)+0.875x_3_{(I)}(k)+0.125u_1_{(I)}(k), \\
x_4_{(I)}(k+1) &= 0.9995x_4_{(I)}(k)+0.06[x_5_{(I)}(k)+x_7_{(I)}(k)-d_2_{(I)}(k)], \\
x_5_{(I)}(k+1) &= 0.975x_5_{(I)}(k)+0.025x_6_{(I)}(k), \\
x_6_{(I)}(k+1) &= -0.05208x_4_{(I)}(k)+0.875x_6_{(I)}(k)+0.125u_2_{(I)}(k), \\
x_7_{(I)}(k+1) &= 0.0044422x_1_{(I)}(k)-0.0044422x_4_{(I)}(k)+x_7_{(I)}(k), \\
u_1_{(I)}(k+1) &= -0.00085x_1_{(I)}(k)-0.002x_7_{(I)}(k)+u_1_{(I)}(k), \\
u_2_{(I)}(k+1) &= -0.00085x_4_{(I)}(k)+0.002x_7_{(I)}(k)+u_2_{(I)}(k), \\
d_1_{(I)}(k+1) &= d_1_{(I)}(k), \\
d_2_{(I)}(k+1) &= d_2_{(I)}(k), \\
\end{align*} \]

end

Step 5: Plotting the graphs of system states

The power system performance with neural network controller has been compared with results of optimal control strategy and integral control strategy by plotting graphs of system states on same scale, with the help of following MATLAB program.
epochs=1:1501;
plot(epochs,x1_I,epochs,x1_O,epochs,x1_N);
plot(epochs,x2_I,epochs,x2_O,epochs,x2_N);
plot(epochs,x3_I,epochs,x3_O,epochs,x3_N);
plot(epochs,x4_I,epochs,x4_O,epochs,x4_N);
plot(epochs,x5_I,epochs,x5_O,epochs,x5_N);
plot(epochs,x6_I,epochs,x6_O,epochs,x6_N);
plot(epochs,x7_I,epochs,x7_O,epochs,x7_N);
plot(epochs,x8_O,epochs,x8_N);
plot(epochs,x9_O,epochs,x9_N);
plot(epochs,u1_I,epochs,u1_O,epochs,u1_N);
plot(epochs,u2_I,epochs,u2_O,epochs,u2_N);

As an example, the graph for state $x_1$ (frequency deviation in area 1) is shown in Fig. 5.3 for load disturbances of $d_1 = 0.0081$ and $d_2 = 0.001$. 
It is seen that the ANN controller gives the performance quite closer to that of the optimal control and much superior than that of the integral control.

Similar procedure has been adopted to successfully obtain properly trained neural network controllers for all the other models of power systems under consideration.

5.4 ANN CONTROLLER WITH INCOMPLETE STATE FEEDBACK

Efforts have also been made to obtain ANN controllers trained with feedback of only crucial system states (incomplete state feedback), e.g., for the two area thermal-thermal (non reheat) model, only the states $x_1, x_4, x_7, x_8$ & $x_9$ have been used for training. Such controllers are often preferable in practice since they require information of observable states only.

**Neural network with power system in training mode:**

The interface of neural network with the power system in training mode with incomplete state feedback is shown in Fig. 5.4.

![Fig. 5.4: Neural network while training with incomplete states](image-url)
Trained neural network with power system as controller:

The interface of trained neural network with the power system in controller mode with incomplete state feedback is shown in Fig. 5.5.

![Diagram of neural network with power system as controller](image)

**Fig. 5.5: Neural network with power system as controller**

Performance of neural network trained with incomplete state feedback has also been tested with same load disturbances, i.e., $d_1 = 0.0081$ and $d_2 = 0.001$. The response of state $x_1$ (i.e., the frequency deviation in area 1) is shown in Fig. 5.6.
It is seen that, the ANN controller gives satisfactory performance even with incomplete state feedback.

5.5 CONCLUDING REMARKS

It has been demonstrated that, the neural networks can be successfully trained by using optimal and suboptimal control strategy, which can satisfactorily work as controllers for AGC in interconnected power systems so as to give the performance quite closer to optimal control and much superior than the integral control.

In a similar way, the ANN controllers have been developed for all the other power system models under consideration and all of them have shown satisfactory performance.