Transient Stability Analysis Using MATLAB Simulink and Neural Network

Devender Kumar & Balwinder Singh Surjan
PEC University of Technology, Chandigarh
E-mail: devenderelec@gmail.com, balwindersingh@pec.ac.in

Abstract – In this paper a Neural Networks (NN) is proposed for transient stability prediction. Transient stability of a power system is first determined based on the generator relative rotor angles procured from time domain simulation outputs. Simulations were carried out on a single machine infinite bus system by considering three phase short circuit fault on the system. The data collected from the time domain simulations are then used as inputs to the NN in which NN is used as a classifier to determine whether the power system is stable or unstable. To verify the effectiveness of the proposed NN method, it is compared with the probabilistic Neural Networks (PNN) and the Multi-Layer Perceptrons Neural Networks (MLP). Results show that the NN gives more accurate transient stability assessment compared to the probabilistic neural network and multi-layer perceptrons neural networks in terms of classification results.

Keywords – Transient Stability Assessment (TSA), MATLAB Simulink, Artificial Neural Networks (ANN).

I. INTRODUCTION

Power system stability is the ability of an electric power system for a given initial operating condition, to regain a state of operating equilibrium after being subjected to a physical disturbance, with most system variables bounded so that practically the entire system remains intact[1-2]. Due to the complexity and vastness of this problem, it has been divided to smaller areas including rotor angle, frequency, and voltage stabilities. Rotor angle stability refers to the ability of synchronous machines of an interconnected power system to remain in synchronism after being subjected to a disturbance. Rotor angle stability is divided to two subcategories: small signal and transient stabilities[2-4]. These valuations aim to assess the dynamic behavior of a power system in a fast and accurate way. Methods normally employed to assess TSA are by using time domain simulation, direct and artificial intelligence methods. Time domain simulation method is implemented by solving the state space differential equations of power networks and then determines transient stability. Direct methods such as the transient energy method determine transient stability without solving differential state space equations of power systems[5]. These two methods are considered most accurate but are time consuming and need heavy computational effort. Presently, the use of artificial neural network (ANN) in TSA has gained a lot of interest among researchers due to its ability to do parallel data processing, high accuracy and fast response[9]. Transient stability evaluation usually focuses on the Critical Clearing Time (CCT) of the power system in response to a fault, defined as the maximum time after occurrence of disturbance, during which if the fault is cleared, the power system can save its transient stability[6-8]. The CCT is the maximum time duration that a fault may occur in power systems without failure in the system so as to recover to a steady state operation. Some works have been carried out using the feed forward multilayer perceptrons (MLP) with back propagation learning algorithm to determine the CCT of power systems[10], the use of radial basis function networks to estimate the CCT[11]. Another method to assess power system transient stability using ANN is by means of classifying the system into either stable or unstable states for several contingencies applied to the system[12].

II. SIMULINK DIAGRAM OF SINGLE MACHINE INFINITE BUS SYSTEM

In Fig 1 Simulink diagram output torque angle is measured by applying swing equation into the system. In this system two conditions are applied one is of sustained fault condition and the other is for post fault condition.
We can change from one condition to other condition by using switch, and by varying the clock we can change the fault clearing time of the system. As early we clear the fault of the system the chances of attaining transient stability is more of the system. For stable operation of the system our torque angle should remain with limit.

Fig. 1: Simulink model of single machine infinite bus system

In fig 3 torque angle response of the single machine system with various fault clearing time is shown. The critical clearing time of the system is 0.38sec., line 1 is with fault clearing time of 0.49sec., this is the condition for sustained fault, means our system is suffering from fault, the next line 2 is with fault clearing time of 0.3sec., in response our system is out of the fault but due to large fault clearing time it have more transients, the next line 3 is with fault clearing time 0.2sec,

in this response our torque angle is also above the required limit, the next line 4 is with fault clearing time of 0.1sec, this is the response where transients are very low and our torque angle is within required limit.

III. INTRODUCING DAMPING INTO THE SYSTEM

By introducing Damping into the system we can remove all the transients that are presented into the system after removing the fault of the system. Damping can be introduced into the system by providing a negative gain of very low value less than 1. This negative feedback gain remove all the transients which are present in the system.

<table>
<thead>
<tr>
<th>Accelerating power(Pa)</th>
<th>angle</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.288</td>
<td>21.64</td>
</tr>
<tr>
<td>0.539</td>
<td>24.21</td>
</tr>
<tr>
<td>0.439</td>
<td>31.59</td>
</tr>
<tr>
<td>-4.46</td>
<td>37.65</td>
</tr>
<tr>
<td>-0.63</td>
<td>38.92</td>
</tr>
<tr>
<td>-0.66</td>
<td>38.64</td>
</tr>
<tr>
<td>-0.56</td>
<td>36.40</td>
</tr>
<tr>
<td>-0.327</td>
<td>29.85</td>
</tr>
<tr>
<td>0.04</td>
<td>22.44</td>
</tr>
<tr>
<td>0.435</td>
<td>16.25</td>
</tr>
<tr>
<td>0.50</td>
<td>13.10</td>
</tr>
</tbody>
</table>

TABLE I output with fault clearing time of 0.10 sec.

Fig 2 is when fault is cleared into the system and transients remain within the system after the clearing of the fault. If the fault clearing time is reduced of the system then our system would remain in stable state. If the fault is cleared after critical clearing time them our system would no longer operate in stable state.

Fig 2: Angle v/s time relationship during post fault condition

III. INTRODUCING DAMPING INTO THE SYSTEM

By introducing Damping into the system we can remove all the transients that are presented into the system after removing the fault of the system. Damping can be introduced into the system by providing a negative gain of very low value less than 1. This negative feedback gain remove all the transients which are present in the system.

Fig 3: Angle v/s time response of different fault clearing time

Fig 4 Single machine infinite bus system with damping
Fig 4 shows the SMIB system with damping, as we decrease the value of negative gain it would decrease the transient period, we can adjust gain as per our requirement. The output response with damping into the system is shown in the figure 5, it shows that the transients are reduced to zero after 2-3 cycles, this period can also be reduced by decreasing the value of negative feedback constant.

Advantages:
- A neural network can perform tasks that a linear program can not.
- When an element of the neural network fails, it can continue without any problem by their parallel nature.
- A neural network learns and does not need to be reprogrammed.
- It can be implemented in any application.
- It can be implemented without any problem.

Disadvantages:
- The neural network needs training to operate.
- The architecture of a neural network is different from the architecture of microprocessors therefore needs to be emulated.
- Requires high processing time for large neural networks.

Application of neural network
- CoEvolution of Neural Networks for Control of Pursuit & Evasion
- Learning the Distribution of Object Trajectories for Event Recognition
- Radiosity for Virtual Reality Systems
- Autonomous Walker & Swimming Eel
- Robocup: Robot World Cup
- Using HMM's for Audio-to-Visual Conversion
- Artificial Life: Galapagos
- Speechreading (Lipreading)
- Detection and Tracking of Moving Targets
- Real-time Target Identification for Security Applications
- Facial Animation

Another aspect of the artificial neural networks is that there are different architectures, which consequently requires different types of algorithms, but despite to be an apparently complex system, a neural network is relatively simple. Artificial neural networks are among the newest signal processing technologies nowadays. The field of work is very interdisciplinary, but the explanation I will give you here will be restricted to an engineering perspective. In the world of engineering, neural networks have two main functions: Pattern classifiers and as non-linear adaptive filters. As its biological predecessor, an artificial neural network is an
adaptive system. By adaptive, it means that each parameter is changed during its operation and it is deployed for solving the problem in matter. This is called the training phase.

A artificial neural network is developed with a systematic step-by-step procedure which optimizes a criterion commonly known as the learning rule. The input/output training data is fundamental for these networks as it conveys the information which is necessary to discover the optimal operating point. In addition, a non-linear nature make neural network processing elements a very flexible system.

The neuron computes the weighted sum of the input signals and compares the result with a threshold value, \( \theta \). If the net input is less than the threshold, the neuron output is \(-1\). But if the net input is greater than or equal to the threshold, the neuron becomes activated and its output attains a value \(+1\).

The neuron uses the following transfer or activation function. This type of activation function is called a sign function.

\[
X = \sum_{i=1}^{n} x_i w_i \\
Y = \begin{cases} 
+1, & \text{if } X \geq \theta \\
-1, & \text{if } X < \theta 
\end{cases}
\]

V. TRAINING OF NEURAL NETWORK

Training to neural network small adjustments in the weights to reduce the difference between the actual and desired outputs of the perceptron. The initial weights are randomly assigned, usually in the range \([-0.5, 0.5]\), and then updated to obtain the output consistent with the training examples. It can be done by going to matlab nntoolbox.NN toolbox can be open by entering command >>nntool. It will open NN Network/ Data Manager screen.

Let \( P \) denote the input and \( T \) denote the target/output.

In Matlab as per the guidelines of implementation these are to be expressed in the form of matrices:

\[ T = [0 1 1 0] \]

To use a network first design it, then train it before start simulation. We follow the steps in order to do the above:

Step-1: First we have to enter \( P \) and \( T \) to the NN Network Manager. This is done by clicking New Data once.

Step-2: Type \( P \) as the Name, and corresponding matrix as the Value, select Inputs under Data Type, then confirm by clicking on Create.

Step-3: Similarly, type in \( T \) as the Name, and corresponding matrix as the Value, select Targets, under Data Type, then confirm.

Step-4: Now we try to create a Network. For this click on New Network.

Make Sure the parameters are as follows:

Network Type = Feed forward – Back prop
Train Function = TRAINLM
Adapation Learning Function = LEARNNGDM
Performance Function = MSE
Numbers of Layers = as per the requirement

Step-5: Select Layer 1, type in 2 for the number of neurons, & select TANSIG as Transfer Function.

Select Layer 2, type in 1 for the number of neurons, & select TANSIG as Transfer Function.

Step-6: Then, confirm by hitting the Create button, which concludes the network implementation phase.

Step-7: Now, highlight Network with DOUBLE click, then click on Train button.

Step-8: On Training Info, select \( P \) as Inputs, \( T \) as Targets.
On Training Parameters, specify: epochs = 1000
Goal = 0.000000000000001
Max fail = 50

After, confirming all the parameters have been specified as indented, hit Train Network.

Fig 9 Training process of neural network

Now we can get following plots Performance plot, It should get a decaying plot (since you are trying to minimize the error).

Training State Plot

Regression Plot

Fig 10 Performance plot of neural network

Fig 11 Neural network implemented in smib system.

Now for training this neural network according to the output of our smib system, the output shown in table 1 will be taken as the input to this neural network. For the better performance of neural our reference data should be accurate. Output result with fault clearing 0.1sec is the most accurate result available, so this output result will be used for training the neural network. Neural network will obtain its final output by changing its weight according to the requirement. Neural network is hit and trail method, it will improve its result iteration by iteration and when the final goal is reached it is finally trained. Untrained neural network can-not be used for the simulation purpose, training of the network is compulsory for its use.

Fig 9 shows the training of neural network in matlab, all the parameters should be within precise limit for the better operation of the system, after the training of neural network is done, various plots for the performance of neural network is shown, by seeing these performance plot we are sure that our neural network is trained as per our requirement, shown in fig 10, after verifying all the performance plot now we have to implement this neural network in our smib system, by taking power as input characteristic which control the output torque angle of the system, we can control the output performance of the smib by providing neural network into the system. The output response of the system with neural network and with time domain solution is shown in fig 12 and fig 13.

Fig 12 output response of smib system with neural network
Fig 13 shows the v/s time response of smib system with damping with neural network implemented in system. Fig 12 shows that with same critical clearing time our torque angle is limited to 50 degree in case of neural network as compared to 55 degree in case of time domain solution. So neural network can give better results when used. Fig 13 shows the damping introduced into the system when neural network is implemented into the system. The results are even better in this case as compared to time domain solution.

VI. CONCLUSION

The results shows that transient stability can be calculated with artificial neural network. The output obtained from both the result are almost identical and best suited for transient stability analysis. In this paper smib system is simulated on matlab Simulink and output generated from this Simulink work as the input for neural network training and the result obtained are compared. Transients present in the system are also been removed by providing damping into the system.

VII. REFERENCES

[10] Power system engineering Nagrath & Kothari, page no559-634, page no 981-988