CHAPTER 2

THEORETICAL AND METHODOLOGY OF STUDY

2.1 Power Flow Solutions

Power flow studies are of the great importance in planning and designing the future expansion of power system as well as in determining the best operation of existing system. The principal information obtained from a power-flow study is the magnitude and phase angle of the voltage at each bus and the real and reactive power flowing in each line. However, much additional information of value is provided by the printout of the solution from computer programs used by the electric utility companies.

This model is the power flow equation. In the general case, consider the general bus i shown in Figure 2.1. Power at generation and load are assumed to equal S_{Gi} and S_{Di} respectively. The bus power S_i is thus given by

$$S_i = S_{Gi} - S_{Di} = P_{Gi} - P_{Di} + j(Q_{Gi} - Q_{Di}) \qquad (2.1)$$

Where: Z is line impedance, Ω; Y_{ij} is line charging admittance, ohm
For each line numerical values for the series impedance Z and the total line charging admittance.

Admittance (Y) are necessary so that the computer can determine all elements of the $N \times N$ bus admittance matrix of which the typical element Y_{ij} is:

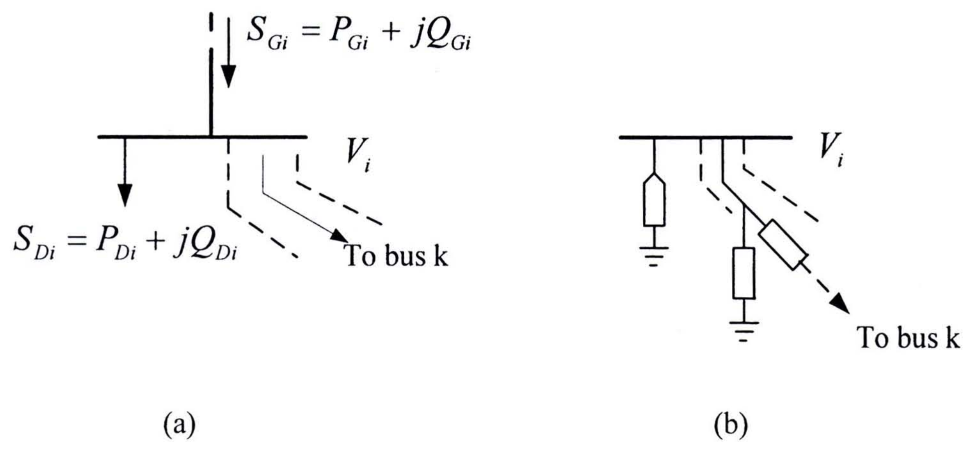


Figure 2.1 General buses with generation and load outgoing line

$$Y_{ij} = |Y_{ij}|\angle\theta_{ij} = |Y_{ij}|\cos\theta_{ij} + j|Y_{ij}|\sin\theta_{ij} = G_{ij} + jB_{ij} \tag{2.2}$$

where:
G_{ij} is called *conductance*, and the B_{ij} is called *susceptance*.
The voltage at a typical bus i - th of the system is given by

$$V_i = |V_i|\angle\delta_i = |V_i|(cos\,\delta_i + jsin\,\delta_i) \tag{2.3}$$

where:
δ_i is phase angle of the voltage bus i - th.

The net current injected into the network at bus i in term of the elements Y

$$I_i = Y_{i1}V_1 + Y_{i2}V_2 + \ldots + Y_{iN}V_N = \sum_{n=1}^{N} Y_{in}V_n \tag{2.4}$$

The net real and reactive power entering the network at the bus i is obtained from

$$P_i - jQ_i = V_i^*I_i = V_i^*\sum_{n=1}^{N} Y_{in}V_n \tag{2.5}$$

Substituting for I_i in Eq. (2.4) into Eq.(2.5)

$$P_i - jQ_i = \sum_{n=1}^{N} |Y_{in}||V_i||V_n|\angle(\theta_{in} + \delta_n - \delta_i) \tag{2.6}$$

Expanding this equation and equating real and reactive parts

$$P_i = \sum_{n=1}^{N} |Y_{in}||V_i||V_n|\cos(\theta_{in} + \delta_n - \delta_i) \tag{2.7}$$

$$Q_i = -\sum_{n=1}^{N} |Y_{in}||V_i||V_n|\sin(\theta_{in} + \delta_n - \delta_i) \tag{2.8}$$

2.2 Modified Y- Bus Matrix

The nodal admittance matrix of the typical power system is large and sparse, and can be constructed in a systematic building-block manner. The building-block approach provides insight for developing algorithms to account for network changes. Because the network matrices are very large, sparsity techniques are needed to enhance the computational efficiency of computer programs employed in solving many of the power system problems [15].

This methodology is based on modified Y-bus matrix proposed by W.C. Chu [3]. This method begins with the system node equation. This can be written in a matrix form as (2.9).

$$I = YV \tag{2.9}$$

For the convenience of explanation, it is assumed that the power system has a total number of n buses, g generators, and l loads. Therefore, the Y bus of $n \times n$ dimension can be divided into four sub matrixes as shown in (2.10).

$$\left[\begin{array}{cc|cc} Y_{1,1} & Y_{1,g} & Y_{1,g+1} & Y_{1,n} \\ \vdots & \vdots & \vdots & \vdots \\ Y_{g,1} & Y_{g,g} & Y_{g,g+1} & Y_{g,n} \\ \hline Y_{g+1,1} & Y_{g+1,g} & Y_{g+1,g+1} & Y_{g+1,n} \\ \vdots & \vdots & \vdots & \vdots \\ Y_{n,1} & Y_{n,g} & Y_{n,g+1} & Y_{n,n} \end{array}\right] \left[\begin{array}{c} V_1 \\ \vdots \\ V_g \\ \hline V_{g+1} \\ \vdots \\ V_n \end{array}\right] = \left[\begin{array}{c} I_1 \\ \vdots \\ I_g \\ \hline I_{g+1} \\ \vdots \\ I_n \end{array}\right] \tag{2.10}$$

From (2.10) can be briefly represented as the following

$$\left[\begin{array}{c|c} YGG & YGL \\ \hline YLG & YLL \end{array}\right] \times \left[\begin{array}{c} VG \\ \hline VL \end{array}\right] = \left[\begin{array}{c} IG \\ \hline IL \end{array}\right] \tag{2.11}$$

A possible way to deduce load node voltage as a function of generator bus voltages is to apply superposition theorem. However, it requires replacing all load bus current injections into equivalent admittances in the circuit. Using a readily available load flow results, the equivalent shunt admittance YL_j of load node j can be calculated using the following

$$YL_j = \frac{1}{VL_j}\left(\frac{SL_j}{VL_j}\right)^{*} \tag{2.12}$$

where (*) means conjugate, SL_j is the apparent power of load on bus j, YL_j is the equivalent admittance of load on bus j, and VL_j is the resultant voltage at bus j of power flow analysis. The modification is executed by adding the corresponding YL_j to the diagonal elements in the [YLL] matrix, so the original matrix [YLL] is replaced by matrix [YLL'] (2.11) can be rewritten as follows

$$\left[\begin{array}{c|c} YGG & YGL \\ \hline YLG & YLL' \end{array}\right] \times \left[\begin{array}{c} VG \\ \hline VL \end{array}\right] = \left[\begin{array}{c} IG \\ \hline 0 \end{array}\right] \tag{2.13}$$

From (2.13) on the lower half part of the matrix is used to arrive.

$$[VL] = -[YLL']^{-1}[YLG][VG] \tag{2.14}$$

Then (2.14) can be simplified as

$$[VL] = [YA][VG] \tag{2.15}$$

where: $[YA] = -[YLL']^{-1}[YLG]$

The voltage of each load bus consisting of the voltage contributed by individual generators is as shown in (2.16).

$$VL_j = \sum_{i=1}^{g} YA_{j,i} \times VG_i \tag{2.16}$$

It is assumed that

$$\Delta VLi, j = YA_{j,i} \times VG_i \tag{2.17}$$

where $\Delta VLi, j$ voltage contribution is that load j acquires from generator i. The reactive power contribution that load j acquires from generator i is as follows:

$$QL_{i,j} = \text{Im}\{\Delta VL_{i,j} \times IL_j^*\} \tag{2.18}$$

where IL_j is the load current which is to divide the power of the load by know load bus voltage and take the conjugate of the complex number on load bus j.

2.3 Backpopagation Neural Network Technical

Neural network has two different phases: the training and learning phase, and the recall phase. In the training and learning phase, the neural nets is trained to return a specific output when give a specific input. This is done by continuous training on a set of training data. In the recall phase, the neural nets return output based on the input [16].

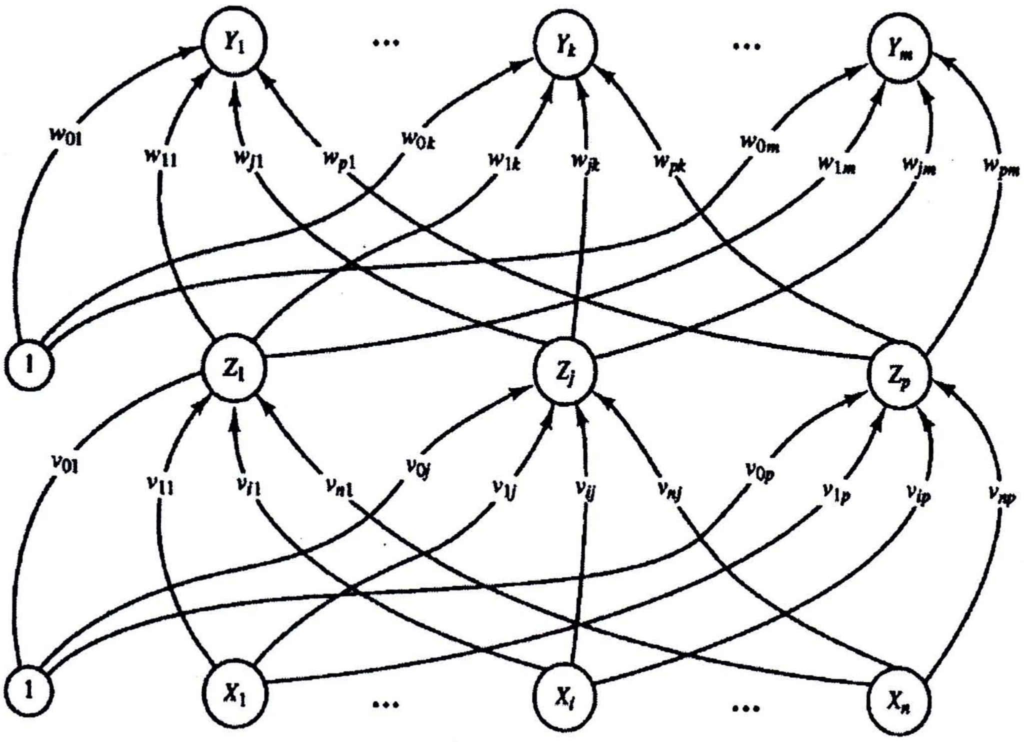


Figure 2.2 Back-propagation neural networks with one hidden layer [15]

2.3.1 Architecture

Neural network with back-propagation is a multilayer neural net with input layer, hidden layer, and output layer. As shown in Figure 2.2, for instances, there are one layer of input units (the X units) and one layer of hidden units (the Z units). The output units (the Y units) and the hidden units are able to have the biases. The bias on a typical output unit Y_k is denoted by w_{ok}, and the bias on a typical hidden unit Z_j is denoted v_{oj}. Such bias terms act like the weights on the connections from the units of which the output is always 1. However, in this Figure 2.2, only the flow direction of information for the feed forward phase of operation is shown. During the back propagation phase of learning, the signal direction is reverse.

2.3.2 Algorithm

As mentioned in 2.4.1, training a network by back propagation involves in three stages: the feed forward of the input training pattern, the back propagation of the associated error, and the adjustment of the weights.

During feed forward, each input unit (X_i) receives an input signal and transmits this signal to each of the hidden units Z_1,..., Z_p. Each hidden unit then computes its activation and sends its signal (z_j) to each output unit. Each output unit (Y_k) computes its activation (y_k) to form the response of the net for the given input pattern.

During training, each output unit compares its computed activation y_k with its target value t_k to determine the associated error. Based on this error, the factor δ_k (k = 1,..., m) is computed. δ_k is used to distribute the error at the output unit Y_k back to all units in the previous layer (the hidden units that are connected to Y_k). It is also used (later) to update the weights between the output layer and the hidden one. In a similar manner, the factor δ_j (j = 1, ..., p) is computed for each hidden unit Z_j. It is not necessary to propagate the error back to the input layer, but δ_j is used to update the weights between the hidden layer and the input one.

After all of the δ factors have been determined, the weights for all layers are adjusted simultaneously. The adjustment to the weight w_{jk} (from hidden unit Z_j to output unit Y_k) is based on the factor δ_k and the activation z_j of the hidden unit Z_j, The adjustment to the weight v_{ij} (from input unit X_i to hidden unit Z_j) is based on the factor δ_j and the activation x_i of the input unit.

2.3.3 Initial weights and bias

<u>Random Initialization</u>

The choice of initial weights will influence whether the net reaches a global (or only a local) minimum of the error arid, if so, how quickly it converges. The update of the weight between two units depends on both the derivatives of the activation functions of the upper unit and the lower unit. Consequently, it is important to avoid choices of the initial weights that would make it likely that either the activations or their derivatives are zero. The values for the initial weights must not be too large, or the initial input signals to each hidden or output unit will be likely to fall in the region in which the derivative of the sigmoid function has a very small value (the so-called saturation region). Alternatively, if the initial weights are too small, the net input to a hidden or output unit will be close to zero, which also causes extremely slow learning.

A common procedure is to initialize the weights (and biases) to random values between -0.5 and 0.5, or -1 and 1, or some other suitable interval. The values may be positive or negative because the final weights after training may also be of either sign. A simple modification of random initialization developed by Nguyen and Windrow [15] is given here.

Nguyen-Widrow Initialization

Typically, this modification of the common random weight initialization gives much faster learning. The approach is based on a geometrical analysis of the response of the hidden neurons to a single input. The analysis is extended to the case of several inputs by using Fourier transforms.

The initialization of the weights from the input units to the hidden units is designed to improve the ability of the hidden units to learn. This is accomplished by distributing the initial weights and biases so that it is likely for each input pattern that the net input to one of the hidden units will be in the range in which the hidden neuron will learn most readily. The expressions we use are as follows:

$$v_{ij}(new) = \frac{\beta \times v_{ij}(old)}{v_{ij}(old)} \tag{2.1}$$

β = scale factor: $\beta = 0.7(p)^{1/n}$

n = number of input units

p = number of hidden units

v_{ij} (old) = random number between - 0.5 and 0.5 (or between $-\gamma$ and γ)

Set bias: v_{oj} = random number between $-\beta$ and β

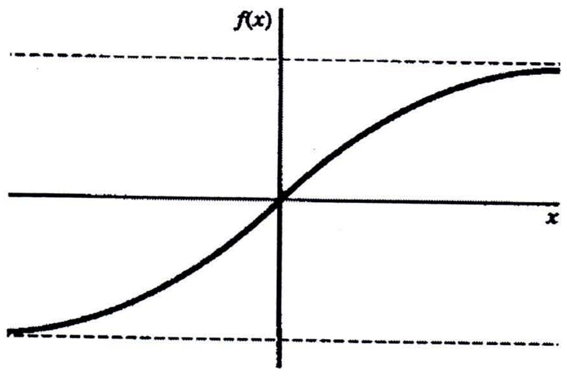
2.3.4 Activation function

An activation function for a backpropagation net should be continuous, differentiable, and monotonically non-decreasing. Moreover, its derivative should be easy to compute. For the most commonly used activation functions, the value of the derivative (at a particular value of the independent variable) can be expressed in terms of the value of the function (at that value of the independent variable)

As illustrated in Figure 2.3, one of the most typical activation functions is the binary sigmoid function of which has range of (0,1), and is defined as

$$f_1(x) = \frac{1}{1+\exp(-x)} \tag{2.2}$$

The other one is bipolar sigmoid function of which has range of (-1, 1) as illustrated in Figure 2.4 and is defined as



(2.3)

Figure 2.3 Binary sigmoid rang (0, 1) [16]

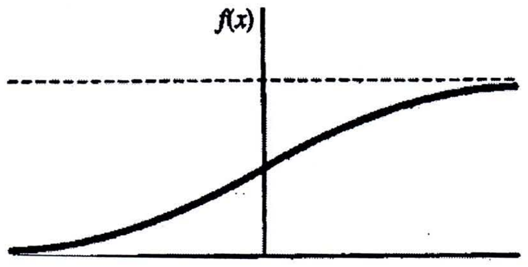


Figure 2.4 Bipolar sigmoid rang (-1, 1) [16]

2.3.5 Momentum and Learning rate

In backpropagation with momentum, the weight change is in a direction that is a combination of the current gradient and the previous gradient. This is a modification of gradient descent of which advantages arise chiefly when some training data are very different from the majority of the data (and possibly even incorrect). It is desirable to use a small learning rate to avoid a major disruption of the direction of learning when a very unusual pair of training patterns is presented. However, it is also preferable to maintain training at a fairly rapid pace as long as the training data are relatively similar.

Convergence is sometimes faster if a momentum term is added to the weight update formulas. In order to use momentum, the weights (or weight updates) from one or more previous training patterns must be saved in which the momentum parameter is constrained in the range of 0 to 1.

2.3.6 Number of hidden layers

For a neural net with more than one layer of hidden units, only minor modifications of the algorithm are required. The calculation is repeated for each additional hidden layer in turn, summing over for the units in the previous layer that feed into the current layer for which is being calculated. With reference to the algorithm, one hidden layer is sufficient for a backpropagation net to estimate any continuous mapping from the input patterns to the output patterns to an arbitrary degree of accuracy. Possibly, two hidden layers make training easier in some situations but there is no general solution for this problem.