CHAPTER-V

GA-NN SELF TUNING TECHNIQUE
CHAPTER-V

GA-NN SELF TUNING TECHNIQUE

5.0. INTRODUCTION

A HVDC transmission system is generally preferred to transfer large chunk power over widely displaced systems or load centers owing to its advantages. Development of fault in a HVDC system results in increase of system current to a value twice that of the system current under normal conditions. This sudden increase in current may results in malfunctioning of the various components of the system or may completely damage them. Proportional plus Integral controllers are generally incorporated in the system to limit the fault current and retain system stability. One of the familiar intelligent techniques was employed in the just preceding chapter to adjust the proportional and integral gain. A significant difference could be observed in system functioning. A different combination of the optimization technique which overweighs the performance of conventional controllers, employing artificial neural networks and genetic algorithms is attempted.

GA is utilized to produce input required to train NN. The training dataset is acquired from different error values of current and their corresponding proportional and integral gains. The so engendered set of values are made use of to tame the neural network using Back propagation algorithm which in turn outputs $K_p$ & $K_i$. 
gains of the PI controller. The proportional and integral gains are applied to the PI controller in order to enable the system to return to its stable state in a remarkably short duration. The process of hybrid technique along with the explanation of the optimization employed is elaborated in following sections

5.1. GENETIC ALGORITHM TO GENERATE TRAINING DATASET

An NN is trained utilising training data generated by employing genetic algorithm. For generating training dataset different values of errors along with their proportional and integral gain values are selected. The process that takes place while generating the training dataset is explained briefly in chapter 5.1.2 and the basic introduction and steps that takes place in genetic algorithm are explained briefly in section 5.1.1.

5.1.1. Brief Description of Genetic Algorithm

GA’s are investigating tools which work on the basis of biological evolution. GA finds the best solution appropriate to the environment by searching the solution space with probabilistic methods and hierarchical exchanges of information’s between individuals. Intelligent search, optimization and machine learning are the crucial areas where Genetic algorithms are applied. Various terminologies associated with GA’s are:

a) Individuals: Every possible solution to problem of optimization is coded in one series called a chromosome.

b) Population: is pool of individuals.
c) Fitness Function: A fitness function is a meticulous type of target function that prescribes the optimality of a solution in a genetic algorithm so that priority ordering of chromosomes can be defined in a collection. A correct selection of the chromosomes can be done by matching their fitness values. Fitness value of a chromosome generally varies as the application varies and is decided by the given input, obtained output and their performance.

The basic flow of above operations for GA is depicted in the Figure 5.1.

![Figure 5.1 Basic Flow Chart of GA](image-url)
The above flow chart shows the basic structure of the genetic algorithm. First process in genetic algorithm is to generate initial chromosome and after generating initial chromosome then calculate fitness function for selecting best chromosome from the generated initial chromosome.

5.1.2. Genetic Operators

i. Selection

Selection means extricating a group of individuals from an existing population depending on definition of quality the consequent process involved in GA.

ii. Cross Over

Cross Over is an operator employed to make a difference in the encoding of chromosome from a stage of the evolution process to the consequent stage. Crossover uses genes of selected parents to produce child chromosome that will form the next generation.

iii. Mutation

Mutation prevents union of the population by snapping few randomly selected bits to include consequent variations. Fig 5.1 explains mutation.
In mutation operation, gene corresponding to the assigned point in the chromosome will only change. No change in the remaining genes.

5.2. GENETIC ALGORITHM APPLIED TO THE PROBLEM

The errors are calculated from different fault currents. The reference current chosen in our proposed method is 1 KA and during a fault current can rise up to a peak value of 2.5 KA. For generating training dataset, different fault current values are taken and corresponding error values are calculated and then the proportional and integral gain values are calculated for each error values using the GA.

The error data is generated within the error limit $[e_{\text{min}}, e_{\text{max}}]$. The elements of error data sets are given by $E = \{e_{\text{min}}, e_{\text{min}} + e_T, e_{\text{min}} + 2e_T, \ldots, e_{\text{max}}\}$ where, $e_T$ is the threshold to
generate elements in the periodic interval. For every element of $E$, optimal $K_P$ and $K_I$ values are determined using the GA. The steps for generating training dataset using genetic algorithm is depicted in the flowchart shown below.

![Flowchart of the proposed GA based system for generating training dataset](image)

**Figure 5.3**: The proposed GA based system for generating training dataset
Various steps involved in producing the required set are:

**5.2.1. Chromosome Production**

The initial part is chromosome generation. Generate a population pool of size $N_p$ with $N_p$ number of arbitrary chromosomes, $X_p = [x_0^{(p)} \ x_1^{(p)}]$; $p = 0, 1, \ldots, N_p - 1$, where, $x_0^{(p)}$ and $x_1^{(p)}$ are the two genes of the $p^{th}$ chromosome that are generated arbitrarily in the interval $[K_p^{\min}, K_p^{\max}]$ and $[K_I^{\min}, K_I^{\max}]$ respectively. i.e $x_0^{(p)} \in [K_p^{\min}, K_p^{\max}]$ and $x_1^{(p)} \in [K_I^{\min}, K_I^{\max}]$. Here, $N_p$ is the size of the population pool, $K_p^{\min}$ and $K_p^{\max}$ are the minimum and maximum range of $K_p$ and $K_I^{\min}$ and $K_I^{\max}$ are the minimum and maximum range of $K_I$ respectively.

**5.2.2. Fitness Function**

After generating the chromosome, the next process is to calculate the fitness for each chromosome in the population pool, using the fitness function shown below.

\[
F_p = \arg \min_{p \in [0, N_p - 1]} \int_0^T |\Delta I| \, dt \quad \text{------------------------(5.1)}
\]

where, $\Delta I = I_{ref} - \left( x_0^{(p)} \cdot e + x_1^{(p)} \int_0^T e \cdot dt \right)$ \quad \text{------------------------.(5.2)}
\( F_p \) is the fitness of the \( p^{th} \) chromosome, \( \Delta I \) is the change in current due to the \( p^{th} \) chromosome, \( T \) is the time maxima, \( I_{ref} \) is the reference current and \( e \) is the error element of \( E \).

By using the above function the fitness value is calculated and based on this fitness value the chromosome is selected. FF works based on the change in current values. For calculating the fitness function, first the change in current value is calculated using equation 5.2. For every change in current value in the system, the value of \( \Delta I \) is calculated and the proportional and integral gain to be selected corresponding to the change in current are calculated. By using the fitness value, lower value of change in current is selected for the subsequent process.

5.2.3. Selection

The best \( N_p/2 \) chromosomes are selected using the above fitness criteria and then the selected fit chromosome is placed in the mating pool. The next process after selecting the best chromosome is the crossover operation.

5.2.4. Crossover

Crossover is performed on the selected chromosome at a crossover rate of \( Cr \) to obtain a child chromosome, \( X^{child} \) for every parent chromosome. After getting the new child chromosome the next process is mutation.
5.2.5. Mutation

Mutation operation is applied to the above set of new child chromosomes. The chromosomes are mutated by randomly selecting the genes at a mutation rate of \( M_r \). The gene values are replaced by arbitrarily selecting the corresponding range of values to obtain \( N_p/2 \) new children for the \( N_p/2 \) parent chromosomes. After applying mutation the next process is termination.

5.2.6. Termination

The population pool is refilled with the \( N_p/2 \) mating pool Chromosomes and new \( N_p/2 \) child chromosomes. The process is iteratively repeated until it reaches a maximum number of iterations \( I_{\text{max}} \). Once the iteration reaches \( I_{\text{max}} \), the process is terminated.

The obtained best chromosome has a proportional gain and an integral gain for all corresponding error values. The dataset generated between the error and the proportional and integral gain values are as follows.

\[
P_{dj} = P_{dij} + P_{dij}^{(set)}
\]

where, \( D_{in} \) is the training dataset generated. By using the above generated dataset the neural network is trained. The next process in our method is to train the neural network to self tune the PI controller.
5.3. **Training the Neural Network Using the Generated Data Set**

The basics of neural network and description of the considered network are explained briefly in Section 4.6.1 and the NN employed is explained in section 4.5 of the thesis.

![NN diagram](image)

**Figure 5.4:** NN utilised in GA-NN technique

The procedure for training an NN is as follows.

Step 1: Generate arbitrary weights within the interval $[0,1]$ and assign it to the hidden layer neurons as well as the output layer neurons. Maintain a unity value weight for all the neurons of the input layer.

Step 2: Input the training dataset $D_{in}$ to the classifier and determine the BP error as follows

\[
BP_{err} = D_{tar} - D_{out}
\]  

\[ (5.4) \]

In equation (5.4), $D_{tar}$ is the target output and $D_{out}$ is the network output, which can be determined as $D_{out} = [y_2^{(1)} y_2^{(2)}]$, where $y_2^{(1)}$ and
$y_2^{(2)}$ are the network outputs which directly represent $K_p$ and $K_i$ respectively. The network outputs can be determined as

$$y_2^{(1)} = \sum_{r=1}^{N_r} w_{2r} y_1(r)$$  \hspace{1cm} \text{(5.5)}$$

$$y_2^{(2)} = \sum_{r=1}^{N_r} w_{2r} y_2(r)$$  \hspace{1cm} \text{(5.6)}$$

where,

$$y(r) = \frac{1}{1 + \exp(-w_{1r} \cdot D_r)}$$  \hspace{1cm} \text{(5.7)}$$

Equation (5.5) and equation (5.6) represents the activation function performed in the output layer and hidden layer respectively.

Step 3: Adjust the weights of all neurons as $w = w + \Delta w$, where, $\Delta w$ is the change in weight which can be determined as

$$\Delta w = \gamma \cdot P \cdot e_{BP}$$  \hspace{1cm} \text{(5.8)}$$

In equation (5.8), $\gamma$ is the learning rate, usually it ranges from 0.2 to 0.5.

Step 4: Repeat the process from step 2, until BP error gets minimized to a least value. Practically, the criterion to be satisfied is $e_{BP} < 0.1$.

Once is network is trained it is ready for practical application. During practical application if any fault occurs in the system, the fault current will increase rapidly. From the fault current value the error is calculated and this calculated error value is given as input to the neural network. The network gives proportional and integral gain
value that corresponds to the error value as output. After training the
next process is to clear the fault.

5.4. FAULT CLEARANCE

In the proposed method PI controller is used for fault clearance. By
adjusting the proportional and integral gain of the PI controller, it
clears the fault. The basic PI controller is explained briefly in chapter
4.7.1.

The error that occurs in the system is calculated by using the
equation given below.

\[ e_{test} = I_{ref} - I_{test} \]  \hspace{1cm} (5.9)

where, \( I_{ref} \) is the reference current which exists in the system when it
is in the normal state and \( I_{test} \) is the measured current from the
system. The error is the difference between the system current and
the reference current. The error \( e_{test} \) is given as an input to the neural
network. The network gives \( K_P \) and \( K_I \) as output for the
corresponding given error value. By using this proportional and
integral gain the PI controller controls the fault current in the HVDC
system and the current value is calculated by using the equation
given below.

\[ I_{ntest} = K_P e_{test} + K_I \int_0^T e_{test} dt \]  \hspace{1cm} (5.10)

where, \( I_{ntest} \) is the output of the PI controller. After calculating the
output of the controller it is checked whether the current matches the
reference. If it does not reach the reference value then the above process of calculating the error values and checking are continued. By using this method the fault that occurs in the system can be cleared within a remarkably small duration. The methodology parameters are tabulated in Table 5.1.

Table 5.1

The parametric values used in the GA-NN technique

<table>
<thead>
<tr>
<th>S.No</th>
<th>Technique Parameters</th>
<th>Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>$e_{\text{min}}/e_{\text{max}}$</td>
<td>0.1/5</td>
</tr>
<tr>
<td>2</td>
<td>$e_T$</td>
<td>0.1</td>
</tr>
<tr>
<td>3</td>
<td>$K_p^{\text{min}}/K_p^{\text{max}}$</td>
<td>0/10</td>
</tr>
<tr>
<td>4</td>
<td>$K_I^{\text{min}}/K_I^{\text{max}}$</td>
<td>0/10</td>
</tr>
<tr>
<td>5</td>
<td>$C_r$</td>
<td>0.5</td>
</tr>
<tr>
<td>6</td>
<td>$M_r$</td>
<td>0.5</td>
</tr>
<tr>
<td>7</td>
<td>$N_p$</td>
<td>10</td>
</tr>
<tr>
<td>8</td>
<td>$I_{\text{max}}$</td>
<td>50</td>
</tr>
<tr>
<td>9</td>
<td>$N_H$</td>
<td>2</td>
</tr>
</tbody>
</table>

5.5. SIMULATION RESULTS

The proposed technique was implemented in the working platform of MATLAB 7.10 and its operation was simulated. The test system, described in section 4.1.1 is considered. Validation of the
The proposed technique is done by subjecting the test system to the following faults:

**5.5.1. Case-I: Line to Ground Fault at the inverter**

The system is simulated with a Single Line to Ground fault at inverter bus ac bus from 0.5 sec to 0.54 sec. There was a rise in the dc current from 1KA to 2 KA as shown in figure 5.5(c) as soon as the fault was initiated. With the conventional PI controller oscillations over a very large scale in the current could be observed for nearly about 4 cycles where it has reduced down to 2 cycles in the case of fuzzy controller. The system current restored to original value in the case of proposed technique in fraction of seconds. That means a significant reduction is observed in the time taken to clear the fault. The rectifier side dc voltage suddenly dropped from 400KV to 0v with the proposed technique but the same voltage reduced to -200V in the case of conventional and fuzzy controllers. Therefore, it can be concluded that the chances of commutation failure are reduced a lot with the proposed hybrid technique with an increment in the commutation margin. A quantitative measure cannot be correctly specified as the work carried out does not focus much on commutation margins. But reduced fault clearance times itself is an indication of improvement of commutation margin.

**5.5.2. Case-II Line –to – Line Fault at the inverter**

The wave forms of voltages at the either ends and the dc link current resulting from a 2.5 cycle dc line to line fault have been
shown in Figures 5.8, 5.9, 5.10 for the three controllers. Several oscillations have been observed in dc link current and voltage with the conventional controller. The fault clearance time was nearly equal to 0.4 sec in the case of conventional but it has come down to 0.2 sec with a fuzzy controller and to a fraction of seconds in the proposed technique. The results depict that the response is very fast and error is minimum in attaining the steady state values. Rectifier side dc voltage dropped from 480KV to -480 KV at the instant of fault. The chances for commutation failures was very high in the case of conventional controllers but those chances got reduced to zero with fuzzy and the implemented hybrid controller.

5.5.3. Case-III  Line –to ground fault at the rectifier

Figures 5.11, 5.12 are the waveforms of current and voltage when a 2.5 cycle SLG fault has been initiated at the rectifier end ac bus. As the strength of the ac system is high, the effect of controllers on the waveforms of currents and voltages was very minimal. Maximum value of fault current was 1.5 KA. So, the rise in the current was reduced compared to the earlier cases. Oscillations in current were also low but system attained its normal state earlier with the proposed scheme compare to conventional and fuzzy schemes. Due to the robustness of the controller the parametric uncertainties in the system could not have an influence on its performance and stability.
5.5.4. Case-IV Line to Line fault at the rectifier

To test the effectiveness of the proposed technique, the system has been subjected to a 2.5 cycle Line to Line fault at the rectifier end ac bus. Figures 5.13, 5.14 are the plots. Due to this variation the scheduled flow of current was affected much with the conventional and fuzzy controllers whereas the disturbance was minimum with the proposed technique. The rectifier end voltage dropped to zero at the instant of the fault. This is in a way similar to the three-phase fault on the ac bus and the dc power flow momentarily reduced to zero followed by oscillations. Owing to the strength of the ac system, the actions could not affect voltages and currents much.

Figure 5.5(a) Dc Link Current for Single Line to Ground Fault at the Inverter for Conventional Controller
Figure 5.5(b)  Dc Link Current for Single Line to Ground Fault at the Inverter for Fuzzy Controller

Figure 5.5(c)  Dc Link Current for Single Line to Ground Fault at the Inverter for GA-NN Controller
Figure 5.5 (a, b, c) represent the current on the rectifier side when the
test system is subjected to a single line to ground fault on the inverter
side.

Figure 5.6(a)    Rectifier end Dc Voltage for Single Line to Ground
Fault at the Inverter for Conventional Controller

Figure 5.6(b)    Rectifier end Dc Voltage for Single Line to Ground
Fault at the Inverter for Fuzzy Controller
Figure 5.6(c) Rectifier end Dc Voltage for Single Line to Ground Fault at the Inverter for GA-NN Technique

Figure 5.7(a) Inverter end Dc Voltage for Single Line To Ground Fault at the Inverter for Conventional Controller
Figure 5.7(b) Inverter end Dc Voltage for Single Line To Ground Fault at the Inverter for Fuzzy Controller

Figure 5.7(c) Inverter end Dc Voltage for Single Line To Ground Fault at the Inverter for GA-NN Controller
Figures 5.7(a, b, c) represent the dc voltage at the inverter side when the system recovers from a single line to ground fault simulated at the inverter end for a) conventional b) Fuzzy c) GA-NN Controllers.

Figure 5.8(a) Dc Link Current for Line to Line Fault at the Inverter for Conventional Controller

Figure 5.8(b) Dc Link Current for Line to Line Fault at the Inverter for Fuzzy Controller
Figures 5.8(a,b,c) depicts how the dc link current varies when the test system is subjected to a line to line fault at the inverter end employing the three a)conventional b)fuzzy c) the proposed GA-NN hybrid tuning methods.
Figure 5.9(a) Rectifier end Dc Voltage for Line to Line Fault at the Inverter for Conventional Controller.

Figure 5.9(b) Rectifier end Dc Voltage for Line to Line Fault at the Inverter for Fuzzy Controller.
Figure 5.9(c) Rectifier end Dc Voltage for Line to Line Fault at the Inverter for GA-NN Controller.

Figure 5.9(a, b, c) how the rectifier side dc voltages varies during the fault clearing process when the test system is subjected to a line to line fault at the inverter for a)conventional b)fuzzy and c)GA-NN Technique’s.
Figure 5.10 (a) Inverter end Dc Voltage for Line to Line Fault at the Inverter for Conventional Controller.

Figure 5.10 (b) Inverter end Dc Voltage for Line to Line Fault at the Inverter for Fuzzy Controller.
Figure 5.10 (c) Inverter end Dc Voltage for Line to Line Fault at the Inverter for GA-NN Controller.

Figures 5.10(a, b, c) represent the variation of inverter side dc voltage when the system is subjected to a dc line to line fault at the inverter end a) conventional b) fuzzy and c) GA-NN Techniques.

Figure 5.11(a) DC link Current for Single Line to Ground Fault at the Rectifier for Conventional Controller.
Figures 5.11(a, b & c) represent the variation of rectifier dc current when the system is subjected to a single line to ground fault at the rectifier end for a) conventional b) fuzzy and c) GA-NN Controllers.
Figure 5.12(a) Rectifier end Dc Voltage Single Line to Ground fault at the Rectifier for Conventional Controller.

Figure 5.12(b) Rectifier end Dc Voltage Single Line to Ground fault at the Rectifier for Fuzzy Controller
Figures 5.12(a, b & c) represent the variation of rectifier side dc voltage when the system is subjected to a single line to ground fault at the rectifier end for a) conventional, b) fuzzy and c) GA-NN Controllers.

Figure 5.13 (a) DC link Current for Line to Line fault at the Rectifier for Conventional Controller
Figure 5.13 (b)   DC link Current for Line to Line fault at the Rectifier for Fuzzy Controller

Figure 5.13 (c)   DC link Current for Line to Line fault at the Rectifier for GA-NN Controller
Figures 5.13(a, b &c) represent the variation of rectifier side dc current when the system is subjected to a line to line fault at the rectifier end for a) conventional b) fuzzy and c) GA-NN Techniques.

Figure 5.14(a) Rectifier end Dc Voltage for Line to Line fault at the Rectifier for Conventional Controller

Figure 5.14(b) Rectifier end Dc Voltage for Line to Line fault at the Rectifier for Fuzzy Controller
Figure 5.14(c)  Rectifier end Dc Voltage for Line to Line fault at the Rectifier for GA-NN Controller

Figures 5.14(a, b &c) represent the variation of rectifier side dc current when the system is subjected to a line to line fault at the rectifier end.
Table 5.2: Comparison of Fault Clearance times for Conventional, Fuzzy and GA-NN Controllers

<table>
<thead>
<tr>
<th>Type of the Fault</th>
<th>Parameters</th>
<th>Fault Clearance Time for Adopted Controller in Sec</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Conventional</td>
</tr>
<tr>
<td>SLG Fault at the Inverter</td>
<td>IDCr</td>
<td>0.4</td>
</tr>
<tr>
<td></td>
<td>VDCr</td>
<td>0.54</td>
</tr>
<tr>
<td></td>
<td>VDCi</td>
<td>0.54</td>
</tr>
<tr>
<td>LL Fault at Inverter</td>
<td>IDCr</td>
<td>0.5</td>
</tr>
<tr>
<td></td>
<td>VDCr</td>
<td>0.49</td>
</tr>
<tr>
<td></td>
<td>VDCi</td>
<td>0.54</td>
</tr>
<tr>
<td>SLG Fault at the Rectifier</td>
<td>IDCr</td>
<td>0.49</td>
</tr>
<tr>
<td></td>
<td>VDCr</td>
<td>0.49</td>
</tr>
<tr>
<td>LL Fault at Rectifier</td>
<td>IDCr</td>
<td>0.47</td>
</tr>
<tr>
<td></td>
<td>VDCr</td>
<td>0.49</td>
</tr>
</tbody>
</table>

5.6. CONCLUSIONS

In this chapter a hybrid technique to self tune the PI Controllers parameters used in HVDC systems was implemented. The technique was proposed with the intention of supporting the PI
controller during the fault clearance process. This has been accomplished by offering optimum PI controller parameters at every instant of time during the fault clearance process which stabilizes the system in a shorter time. The performance of the system has been evaluated from the implementation results. Also, the system was validated by comparing the hybrid technique with the conventional and fuzzy-based self tuning techniques. The comparison results proved that the hybrid technique consumes considerably less time to clear the fault and hence to stabilize the system. The proposed technique is found to be robust, producing significant damping and reductions of overshoots for short circuits at the buses. Therefore, it was evident that the proposed technique makes the controlling of HVDC systems significantly more effective than other conventional self tuning techniques.