CHAPTER 5

SIMULTANEOUS FAULT SECTION ESTIMATION AND PROTECTIVE DEVICE FAILURE DETECTION

Abstract

The main aim of the chapter is to diagnose a fault with respect to its fault section and device failure in the Power System under study. An IEEE 14 bus system is considered for diagnosis of fault section. The strategy involves dividing the system into sections involving analysis of internal and external faults. All the sections are analyzed as per the Directional Comparison Blocking scheme. Thereafter, the states of the protective devices are converted from its binary state to percentage values which when fed to a set of three neural networks trained using the evolutionary algorithms namely PSO, CMAES and CMAES-PSO, determines the fault at hand. The three neural network involved are used to estimate the fault section, to estimate the failure in protection arrangement (primary or secondary) and to estimate the failure of the device respectively. The efficiency of the PSO algorithm and CMAES algorithm is further increased by improving the algorithm by the use of selective operators. The main advantage of extremely high speeds of fault detection seeds from the use of well-trained neural networks. The efficiency and time required for implementation or testing is considered for choosing the best suited architecture for the application.

5.1 INTRODUCTION

In electrical power systems a large number of messages and alarms are transmitted to the control center after the occurrence of disturbances. The role of the fault diagnosis function is to estimate the power system section where the outage was originated, and identify relays and circuit breakers improper operations. Thus, based on the identified fault section, a rapid
and accurate restoration action can be taken to minimize service interruption and limit damage to equipments. The simultaneous fault section estimation and protective device failure detection is proposed to be executed by considering the large number of messages and alarms transmitted to the control center after the occurrence of disturbances.

This work involves using the information about the states of protective devices and analyzing it with the protection philosophy and hence designing a system that simultaneously estimates the fault section and also the protective device at fault. Directional Comparison Blocking Scheme was used to analyze the fault in the IEEE 14 bus system after dividing this into sections.

### 5.2 PROPOSED SYSTEM DESCRIPTION

The proposed strategy involves the fault diagnosis using the information about the states of protective devices and the protection philosophy where a set of neural networks have been used to diagnose the fault in an IEEE-14 bus system. Use of available power has been the key to promote the efficient working of any system. The strategy involves conversion of data from binary state (0&1 of protective equipment’s) to percentage values which when fed to a set of three MLP neural networks trained using PSO and CMAES determines the fault at hand. The efficiency of the PSO algorithm and CMAES algorithm is increased further by the use of selective operators.

![Figure 5.1: Proposed Alarm System – Alarm classification](image-url)
Alarms organization and grouping

The proposed approach makes use of the protective relays and circuit breakers operation states represented by 0 and 1 as well as information related to the protection philosophy. Initially, these data undergo a preprocessing stage which groups the operation states and converts the 0 and 1 format to a format expressed by percentage values.

Fig. 5.1 shows the structure of the intelligent alarm processing system to accomplish the fault diagnosis. The upper portion of this Figure shows the alarms acquisition from Digital Fault Recorders (DFRs) and Digital Protection Units (DPUs). To illustrate the alarms grouping, it is used as an example the Directional Comparison Blocking scheme applied to a portion of the IEEE 14-bus systemas shown in Fig. 5.2.

5.3 TRAINING OF NEURAL NETWORK USING EVOLUTIONARY ALGORITHMS

5.3.1: Particle Swarm Optimization

PSO is one the most powerful and versatile tools of optimization in terms of speed of optimization, ideology of the optimization process and computational burden [84-85].

PSO Algorithm
PSO consists of the following main steps:

Step 1: The particles were initially placed randomly in the entire search space which included all dimensions \([d_1, d_2, d_3, d_4 \ldots]\). The particles indicated by \(\lambda = 1, 2, 3, 4\ldots\) were given positions in each dimension as \([x_{\lambda, d1}, x_{\lambda, d2}, x_{\lambda, d3}, x_{\lambda, d4} \ldots]\). The global best and the population size were initialized.

Step 2: Position update of the particles were done using the following Equation (5.1)

\[
    x_{\lambda, d}^i = x_{\lambda, d}^{i-1} + v_{\lambda, d}^{i-1}
\]

Step 3: Function evaluation with each particle was performed by minimization of the objective function (Equation (5.2))

\[
    f = \min f(x)
\]

Step 4: \(G_{\text{best}}\) and \(P_{\text{best}}\) update was performed after evaluating the function.

Step 5: Velocity is updated using the following (Equation (5.3))

\[
    v_{\lambda, d}^i = (w v_{\lambda, d}^{i-1}) + (c_1(p_{\text{best}} - x_{\lambda, d}^i)) + (c_1(g_{\text{best}} - x_{\lambda, d}^i))
\]

Where,

\(c_1\) and \(c_2\) are personal and global cognitive factors and \(w\) is inertia.

Step 6: Inertia is updated using the following (Equation (5.4))

\[
    w = w_{\text{max}} - (i(w_{\text{max}} - w_{\text{min}})/N)
\]

Where,

\(N\) is the maximum number of iterations, and \(i\) is the value of current iteration.
Table 5.1: Particle Swarm Optimization

<table>
<thead>
<tr>
<th>Name of vector</th>
<th>Particle number</th>
<th>Particle property</th>
<th>Particle dimension</th>
</tr>
</thead>
<tbody>
<tr>
<td>Swarm</td>
<td>1,2,3,4……</td>
<td>1 – position</td>
<td>1 – \text{w}_{11}</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2 – velocity</td>
<td>2 - \text{w}_{12}</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3 – \text{p}_{\text{best}}</td>
<td>3 - \text{b}_1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>4 – \text{g}_{\text{best}}</td>
<td>..................</td>
</tr>
</tbody>
</table>

Table 5.1 displays the treatment of the swarm vector for training a neural network. Swarm is a 3 dimensional matrix where the first dimensions records the particle number, the second dimension records the property of the particle and the final dimension records the property of the particle i.e. the value of that property which may be a weight or a bias.

The weights given by each particle are used to evaluate the MSE for a fixed number of patterns and then returned to PSO to update the $\text{p}_{\text{best}}$ and $\text{g}_{\text{best}}$ and velocity of the particles. However, the learning rates can also be taken as the dimensions of search for PSO.

5.3.2: Covariance matrix adaptation – evolution strategy

CMA-ES stands for Covariance Matrix Adaptation Evolution Strategy. Evolution strategies (ES) are stochastic, derivative-free methods for numerical optimization of non-linear or non-convex continuous optimization problems [86-88]. They belong to the class of evolutionary algorithms and evolutionary computation. An evolutionary algorithm is broadly based on the principle of biological evolution, namely the repeated interplay of variation (via recombination and mutation) and selection: in each generation (iteration) new individuals (candidate solutions, denoted as $x$) are generated by variation, usually in a stochastic way, of the current parental individuals. Then, some individuals are selected to become the parents in the next generation based on their fitness or objective function value $f(x)$. Like this, over the generation sequence, individuals with better and better $f$-values are generated.
CMAES Algorithm

Set parameters
Set parameters \( \lambda, \mu, \omega_{i=1,...,\mu}, \sigma, c, c_1, c_\mu \) to their default values according to Table 5.2.

Initialization
Set evolution paths \( p_\sigma = 0, p_c = 0 \), covariance matrix \( C = I \), and \( g = 0 \).
Choose distribution mean \( m \in \mathbb{R}^n \) and step-size \( \sigma \in \mathbb{R}^\sigma \), problem dependent.
Until termination criterion met, \( g \leftarrow g + 1 \)

Sample new population of search points, for \( k = 1, \ldots, \lambda \)

\[ z_k \sim \mathcal{N}(0,1) \] (5.5)

\[ y_k = BDz_k \sim \mathcal{N}(0,C) \] (5.6)

\[ x_k = m + \sigma y_k \sim \mathcal{N}(m,\sigma^2C) \] (5.7)

Selection and recombination

\[ \langle y \rangle_w = \sum_{i=1}^{\mu} \omega_i y_i \lambda \text{where } \sum_{i=1}^{\mu} \omega_i = 1, \omega_i > 0 \] (5.8)

\[ \sum_{i=1}^{\mu} \omega_i x_i \lambda = m + \sigma \langle y \rangle_w \rightarrow m \] (5.9)

Step-size control

\[ (1 - c_\sigma)p_\sigma + \sqrt{c_\sigma(2 - c_\sigma)c_{\mu_{	ext{eff}}}C^{-\frac{1}{2}}} \langle y \rangle_w \rightarrow p_\sigma \] (5.10)

\[ \sigma \exp \left( \frac{c_\sigma \left( \frac{||p_\sigma||}{E||N(0,I)||} - 1 \right) }{d_\sigma} \right) \rightarrow \sigma \] (5.11)

Covariance matrix adaptation

\[ (1 - c_c)p_c + h_\sigma \sqrt{c_c(2 - c_c)c_{\mu_{	ext{eff}}}\langle y \rangle_w} \rightarrow p_c \] (5.12)

\[ (1 - c_1 - c_\mu)C + c_1 (p_c p_c^T + \delta(h_\sigma)C) + c_\mu \sum_{i=1}^{\mu} \omega_i y_{1:2,2:1} \rightarrow C \] (5.13)
Where,
\( \lambda \) – Population size ,
\( \mu \) – Parent Number ,
\( \mu_{\text{eff}} \) – the variance effective selection mass ,
\( \sigma \) – Step – size
\( B \) – Orthogonal Matrix,
\( C^{(g)} \) – Covariance Matrix at generation \( g \),
\( D \) – Diagonal Matrix,
\( c_c \) – learning rate for cumulation for
the rank one update of the covariance matrix
\( c_1 \) – learning rate for the rank – one
update of the covariance matrix update
\( c_\mu \) – learning rate for the rank – update
of the covariance matrix update
\( c_\sigma \) – learning rate for the cumulation for the
step – size control
\( d_1 \) – diagonal elements of diagonal matrix \( D \),
\( d_\sigma \) – damping parameter for step-size update
\( E \) – Expected value
Table 5.2: Default values of control parameters of CMAES

<table>
<thead>
<tr>
<th>Selection and Recombination</th>
<th>( \lambda = 4 + [3\ln(n)] )</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>( \mu = [\mu'] )</td>
</tr>
<tr>
<td></td>
<td>( \mu' = \frac{\lambda}{2} )</td>
</tr>
<tr>
<td></td>
<td>( \omega_i = \frac{\omega_i'}{\sum_{j=1}^{\mu} \omega_j} )</td>
</tr>
<tr>
<td></td>
<td>( \omega_i' = \ln(\mu + 0.5) - \ln(i) ) for ( i = 1, \ldots, \mu )</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Step size control</th>
<th>( c_\sigma = \frac{\mu_{\text{eff}} + 2}{n + \mu_{\text{eff}} + 5} )</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>( d_\sigma = 1 + 2 \max \left( 0, \frac{\mu_{\text{eff}} - 1}{\sqrt{n + 1}} - 1 \right) + c_\sigma )</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Covariance Matrix Adaptation</th>
<th>( c_c = \frac{4 + \mu_{\text{eff}}/n}{n + 4 + 2\mu_{\text{eff}}/n} )</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>( c_1 = \frac{2}{(n + 4)^2 + \mu_{\text{eff}}} )</td>
</tr>
<tr>
<td></td>
<td>( c_\mu = \min[1 - c_1, c_\mu (\mu_{\text{eff}} - 2 + 1/\mu_{\text{eff}})/(n + 2)^2 + \alpha_\mu \mu_{\text{eff}}/2] )</td>
</tr>
<tr>
<td></td>
<td>With ( \alpha_\mu = 2 )</td>
</tr>
</tbody>
</table>
5.3.3: Training and testing benchmark functions using PSO and CMAES

Minimization of the function was considered as the objective function and results were considered and noted as in Table 5.3, Table 5.4, Table 5.5 and Table 5.6.

5.3.3.1: Neural network for Rosenbrock’s functions

The Rosenbrock’s function was optimized using PSO and CMAES. Minimization of the function was considered as the objective function and results were considered and noted as below.

\[ f(x) = \sum_{i=1}^{n-1} [100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2] \quad (5.14) \]

Table 5.3: Optimization of Rosenbrock’s Function

<table>
<thead>
<tr>
<th>Values Obtained for ( f(x)=0 )</th>
<th>Global Minimum</th>
<th>PSO ( x )</th>
<th>CMAES ( x )</th>
</tr>
</thead>
<tbody>
<tr>
<td>X</td>
<td>1</td>
<td>0.14064</td>
<td>1.0000</td>
</tr>
<tr>
<td>Y</td>
<td>1</td>
<td>0.9365</td>
<td>1.0000</td>
</tr>
<tr>
<td>Z</td>
<td>1</td>
<td>0.15105</td>
<td>1.0000</td>
</tr>
</tbody>
</table>

Neural network for Beale’s functions

The Beale’s function was optimized using PSO and CMAES. Minimization of the function was considered as the objective function the results were considered and noted as below.

\[ f(x, y) = (1.5 - x + xy)^2 + (2.25 - x + xy^2)^2 + (2.625 - x + xy^3)^2 \quad (5.15) \]

Table 5.4: Optimization of Beale’s Function

<table>
<thead>
<tr>
<th>Values Obtained for ( f(x)=0 )</th>
<th>Global Minimum</th>
<th>PSO ( x )</th>
<th>CMAES ( x )</th>
</tr>
</thead>
<tbody>
<tr>
<td>X</td>
<td>3</td>
<td>3.0000</td>
<td>3.0000</td>
</tr>
<tr>
<td>Y</td>
<td>0.5</td>
<td>0.5000</td>
<td>0.5000</td>
</tr>
<tr>
<td>Z</td>
<td>0</td>
<td>4.02904</td>
<td>0.094006</td>
</tr>
</tbody>
</table>
Neural network for Ackley’s function

The Ackley’s function was optimized using PSO and CMAES. Minimization of the function was considered as the objective function and results were considered and noted as below.

$$f(x, y) = -20 \exp \left(-0.2\sqrt{0.5(x^2 + y^2)}\right)$$
$$- \exp(0.5(\cos(2 * \pi * x) + \cos(2 * \pi * x))) + 20 + e$$

(5.16)

Table 5.5: Optimization of Ackley’s Function

<table>
<thead>
<tr>
<th>Values Obtained for $f(x)=0$</th>
<th>Global Minimum</th>
<th>PSO</th>
<th>CMAES</th>
</tr>
</thead>
<tbody>
<tr>
<td>X</td>
<td>0</td>
<td>0.04446</td>
<td>0.1580</td>
</tr>
<tr>
<td>Y</td>
<td>0</td>
<td>0.1359</td>
<td>0.07464</td>
</tr>
<tr>
<td>Z</td>
<td>0</td>
<td>3.884</td>
<td>0.3391</td>
</tr>
</tbody>
</table>

3.3.3: Neural network for Rastrigin function

The Rastrigin’s function was optimized using PSO and CMAES. Minimization of the function was considered as the objective function and results were considered and noted as below.

$$f(x) = An + \sum_{i=1}^{n} [x_i^2 - \cos(2 * \pi * x_i)]$$

(5.17)

Table 5.6: Optimization of Rastrigin’s Function

<table>
<thead>
<tr>
<th>Values Obtained for $f(x)=0$</th>
<th>Global Minimum</th>
<th>PSO</th>
<th>CMAES</th>
</tr>
</thead>
<tbody>
<tr>
<td>X</td>
<td>0</td>
<td>0.99496</td>
<td>-3.62029E-07</td>
</tr>
<tr>
<td>Y</td>
<td>0</td>
<td>6.4992e-10</td>
<td>-3.15867E-07</td>
</tr>
<tr>
<td>Z</td>
<td>0</td>
<td>2.62198e-09</td>
<td>2.23677E-07</td>
</tr>
</tbody>
</table>
5.4 SIMULTANEOUS FAULT SECTION ESTIMATION AND PROTECTIVE DEVICE FAILURE DETECTION

5.4.1: Training and Testing of Neural Network Using PSO for Real Time Data

An MLP (Multi Layer Perceptron) neural network was designed with 4 input neurons and 5 output neurons. The number of hidden neurons was considered to be 7. The activation functions used were tansig and logsig for the hidden layer and output layer respectively. The optimization function to be minimized for the training algorithm (PSO) was the Mean Square Error (MSE). A real time data consisting of three classes was used to train the neural network. Training was carried out for 80% of the training data following a random characteristic and testing was done for 20% of the training data following the same pattern as training. The MSE obtained using the designed neural network architecture was noted as in Fig. 5.3. The MSE obtained in such a case was noted as 0.0433.

![Figure 5.3: MSE of Neural Network trained using PSO for real time data](image)

The output obtained for an MSE of 0.0433 was tested and observed for 9 patterns. Fig. 5.4 shows the output of the 3 output neurons for 9 different patterns.
5.4.2: Training and testing of neural network using CMAES for simultaneous fault section estimation and protective device failure detection

Fault section estimation using CMAES

An MLP (Multilayer perceptron) neural network was designed with 10 input neurons, 7 hidden neurons and 1 output neuron. A single neural network was applied to the analysis of each protection arrangement. This neural network has the task of providing a value that indicates whether the section is a fault section or not.

The activation functions used were tansig and logsig for the hidden layer and output layer respectively. The input weights, output weights and the biases were optimized using Covariance Matrix Adaptation Evolution Strategy considering 78 dimensions. The optimized weights and biases were used in the neural network architecture to obtain the MSE. The optimization function to be minimized for the training algorithm (CMAES) was the Mean Square Error (MSE). The inputs to the neural network are $F_{\text{int}}$, $F_{\text{ext}}$, $P_{\text{CRM}}$, $P_{\text{BFP}}$ and $P_{\text{CBS}}$. $Y_{\text{FS}}$ is the output obtained from the neural network. Depending on the output obtained ($Y_{\text{FS}}$) to be 1 or 0, it indicates whether the section is a fault section or not. Fault section depending on the output of the neural network was obtained for both primary and secondary protection arrangements.

Training was carried out for 80% of the training data following a random characteristic. The MSE obtained using the designed neural network architecture was noted as in Fig. 5.5 The MSE obtained in such a case was noted as 0.0864.
Testing was done for 20% of the input data following the same pattern as training. Fig 6.4 displays the testing of Neural Network trained using CMAES for fault section estimation without any improvement in the algorithm.

**Figure 5.6:** Testing of Neural Network trained using CMAES for fault section estimation without selective operator
5.5 SIMULTANEOUS FAULT SECTION ESTIMATION AND PROTECTIVE DEVICE FAILURE DETECTION USING IMPROVED SCHEME

5.5.1: Fault section estimation using improved PSO

An MLP (Multilayer perceptron) neural network was designed with 10 input neurons, 7 hidden neurons and 1 output neuron. A single neural network was applied to the analysis of each protection arrangement. This neural network has the task of providing a value that indicates whether the section is a fault section or not.

The activation functions used were ‘tansig and ‘logsig’ for the hidden layer and output layer respectively. The input weights, output weights and the biases were optimized using Covariance Matrix Adaptation Evolution Strategy. The optimized weights and biases were used in the neural network architecture to obtain the MSE. The optimization function to be minimized for the training algorithm (improved PSO) was the Mean Square Error (MSE). The inputs to the neural network are $F_{int}$, $F_{ext}$, $P_{CBm}$, $P_{BFP}$, $P_{CBs}$. $Y_{FS}$ is the output obtained from the neural network. Depending on the output obtained ($Y_{FS}$) to be 1 or 0, it indicates whether the section is a fault section or not. Fault section depending on the output of the neural network was obtained for both primary and secondary protection arrangements.

Training was carried out for 80% of the training data following a random characteristic. The MSE obtained using the designed neural network architecture was noted as in Fig.5.7. The MSE obtained in such a case was noted as $1.4099 \times 10^{-8}$. Testing was done for 20% of the input data following the same pattern as training. Fig.5.8 displays the testing of Neural Network trained using improved PSO for fault section estimation.
5.5.2: Detection of failure in Protection Arrangement using improved PSO

An MLP (Multi layer perceptron) neural network was designed with 11 input neurons, 7 hidden neurons and 2 output neurons. A neural network was used to identify which protection arrangements have failed. This neural network generates output values that identify which protection arrangements have failed.

The activation functions used are ‘tansig and ‘logsig’ for the hidden layer and output layer respectively. The input weights, output weights and the biases were optimized using Covariance Matrix Adaptation Evolution Strategy. The optimized weights and biases were used in the neural network architecture to obtain the MSE. The optimization function to be minimized for the training algorithm (improved PSO) was the Mean Square Error (MSE).
The inputs to the neural network are $Y_{FS}$, $F_{intF_{ext}}$, $P_{CBm}$, $P_{BFP}$ and $P_{CBS}$. $Y_{FP}$ is the output obtained from the neural network. Depending on the output obtained ($Y_{FP}$) to be 1 or 0, it indicates which protection arrangements have failed.

Training was carried out for 80% of the training data following a random characteristic. The MSE obtained using the designed neural network architecture was noted as in Fig.5.8. The MSE obtained in such a case was noted as $3.87 \times 10^{-4}$. Testing was done for 20% of the input data following the same pattern as training. Fig.5.9 displays the testing of Neural Network trained using improved PSO for detection of failure in protection arrangement.

![Figure 5.9: MSE of Neural Network trained using improved PSO to detect which protection arrangements have failed](image)

![Figure 5.10: Testing of Neural Network trained using improved PSO to detect which protection arrangements have failed](image)

Training was carried out for 80% of the training data following a random characteristic. The MSE obtained using the designed neural network architecture was noted as in Fig.5.10
and Fig.5.11. The MSE obtained in such a case for Protection Arrangement I was 1.025 \( \times 10^{-8} \) and that for Protection Arrangement II is 1.8\( \times 10^{-9} \).

**Figure 5.11:** MSE of Neural Network trained using improved PSO for device failure detection for protection arrangement I

**Figure 5.12:** MSE of Neural Network trained using improved PSO for device failure detection for protection arrangement II

### 5.6 RESULTS AND DISCUSSIONS

#### 5.6.1: Particle Swarm Optimization

The Neural Network was trained using the PSO algorithm without any selective operators or further modification. It was observed that the MSE was within the range of 1 to 9 for the
neural networks used for fault section estimation, detection of failure in protection arrangement and fault device detection. The improvement attained was the reinitializing the population size in Table.5.7.

<table>
<thead>
<tr>
<th>Neural network for fault section estimation</th>
<th>Neural network for detection of failure in protection arrangement</th>
<th>Neural network for fault device detection</th>
</tr>
</thead>
<tbody>
<tr>
<td>MSE without the selective operators</td>
<td>1.4099E-08</td>
<td>Protection arrangement I-1.025E-08</td>
</tr>
<tr>
<td></td>
<td>3.87E-04</td>
<td>Protection arrangement II-1.8E-09</td>
</tr>
</tbody>
</table>

5.6.2: Covariance Matrix Adaptation Evolution Strategy

Covariance Matrix Adaptation Evolution Strategy was used to train the neural network for Simultaneous fault section estimation and protective device failure detection and the results were as follows. It was noted that errors reduced on introduction of the selective operator. The selective operator used was the reinitializing the population size in Table.5.8.
**Table 5.8:** MSE of Neural Network trained using CMAES

<table>
<thead>
<tr>
<th></th>
<th>Neural network for fault section estimation</th>
<th>Neural network for detection of failure in protection arrangement</th>
<th>Neural network for fault device detection</th>
</tr>
</thead>
<tbody>
<tr>
<td>MSE without the selective operators</td>
<td>8.64E-02</td>
<td>1.124E-01</td>
<td>4.21E-02</td>
</tr>
<tr>
<td>MSE with the selective operators</td>
<td>2.2E-03</td>
<td>3.3067E-03</td>
<td>9.3E-05</td>
</tr>
</tbody>
</table>

**5.6.3: Hybrid Covariance Matrix Evolution Strategy- Particle Swarm Optimization**

Covariance Matrix Adaptation Evolution Strategy – Particle Swarm Optimization was used to train the neural network for Simultaneous fault section estimation and protective device failure detection and the results were as follows. It was noted that errors reduced at a very tremendous rate on introduction of Particle Swarm Optimization for the STEP-SIZE control of Covariance Matrix Adaptation in Table.5.9.

**Table 5.9:** MSE of Neural Network trained using HYBRID CMAES-PSO

<table>
<thead>
<tr>
<th></th>
<th>Neural network for fault section estimation</th>
<th>Neural network for detection of failure in protection arrangement</th>
<th>Neural network for fault device detection</th>
</tr>
</thead>
<tbody>
<tr>
<td>MSE</td>
<td>1.2E-12</td>
<td>9.3E-08</td>
<td>4.34E-13</td>
</tr>
</tbody>
</table>
5.6.4: Evolution in Relay System In Terms of Relay Operation Time

The proposed alarm system design on execution provides a high improvement in the relay operation time from 20ms to about 53.77μs.

Table 5.10: Relay Time for the proposed Alarm system

<table>
<thead>
<tr>
<th>RELAY TIME FOR THE PROPOSED ALARM SYSTEM</th>
</tr>
</thead>
<tbody>
<tr>
<td>MSE</td>
</tr>
<tr>
<td>YS-1.2E-12</td>
</tr>
<tr>
<td>YP-9.3E-08</td>
</tr>
<tr>
<td>YPD-4.34E-13</td>
</tr>
</tbody>
</table>

The results in two different processors (CORE i3 and CORE i5) are noted as in Table.5.6. It was observed that the faster the processor, the shorter is the system operation time.

5.7 CONCLUSION

The proposed approach makes use of the protective relays and circuit breakers operation states represented by 0, 1 and information related to the protection philosophy. Initially, these data undergo a preprocessing stage which groups the operation states and converts the 0 and 1 format to a format expressed by percentage values. These values were submitted as inputs of the artificial neural networks that compose the computational intelligent alarms processing system. A new algorithm CMAES-PSO was formulated to train the neural networks to further increase the efficiency of computational intelligent alarms processing system. The same artificial neural networks trained from an electrical power system can be applied in other power system. The amount of neurons used in the diagnosis system allows performing fault diagnosis in about 53.77μs, which is adequate for online applications.