CHAPTER 4
SENSOR FAULT DETECTION AND RESTORATION

4.1 Introduction

The closed loop control systems are very sensitive to component failure. The reason for this is; their control action may sometimes skin the fault from being identified, also sometimes they amplify the faults. Thus any fault in sensor, actuators or in the system itself can develop into breakdown of the loop. A breakdown in control system may lead to incorrect system output or even tripping of the whole system. The large economic loss due to these breakdowns increases the attention of researchers for making the system tolerable for these failures to acceptable level. A cost effective way to attain this goal is fault-tolerant control (FTC).

The present work is to develop a sensor fault tolerant system for DVR controller. The system can detect faulty sensor and even the missing sensor data can be restored. Thus control action of the close loop system is not affected and the quality of power is maintained across sensitive load.

In this chapter first the faulty or missing sensor is detected by a neural network based system and then the system for restoration of missing sensor data is developed using a search algorithm with neural network.

4.2 Overview of Sensor Fault Detection Systems

Sensors are very important component for any control system, and are very prone to failure. Some engineers consider them as weak or frail link of the system. Development of sensor fault tolerant system will increase the availability of the system, which is very important especially in safety critical applications.

Sufficient analysis or calculations should be performed on the sensor data before transmitting them to control system, to ensure their quality. If the faulty sensor reading (because of either low battery or calibration error or hardware issues) can be detected beforehand the breakdown of the close loop system or tripping of the plant can be avoided by using a proper restoration scheme for the faulty or missing sensor data. Some of the various possible reasons for the sensor fault are concluded in table 3.
The conventional method for the detection of sensor fault is to check and recalibrate a sensor periodically according to a set of scheduled processor. But because of increasing number of sensors in any industry this method becomes cost ineffective and even infeasible. This leads to the growth of more systematical approach for sensor fault detection, that are broadly classified as hardware and analytical redundancy approaches. In hardware redundancy approach one variable is measured by two or more sensors and by continuously checking the sensors data, the faulty sensor can be detected. The analytical redundancy method identifies the functional relationship between the variables through a mathematical model, thus no need of extra hardware.

Table 3 Taxonomy of sensor fault with definitions

<table>
<thead>
<tr>
<th>Fault</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Outlier</td>
<td>Isolated data point or sensor surprisingly distant from model</td>
</tr>
<tr>
<td>Spike</td>
<td>Multiple data point with much greater than estimated rate of change</td>
</tr>
<tr>
<td>stuck-at</td>
<td>Sensor value has zero variation for a long duration of time</td>
</tr>
<tr>
<td>High noise or variance</td>
<td>Sensor value experience high variations or noise</td>
</tr>
<tr>
<td>Connection or hardware</td>
<td>A failure in sensor hardware which causes inaccurate data recording.</td>
</tr>
<tr>
<td>Low Battery</td>
<td>Battery voltage drops to a point where a sensor no longer can report the data.</td>
</tr>
<tr>
<td>Calibration</td>
<td>Sensor reports the values that are offset from ground truth</td>
</tr>
</tbody>
</table>

Analytical redundancy method can be categorized according to the method of their getting the information about the system as; model based method, knowledge based expert systems, and data driven methods. Among these the model based method requires the exact mathematical model of the system. A knowledge based expert system incorporates the expert domain knowledge that is generally captured with a set of rules according to some knowledge representation formalism. The last that is data driven method does not require the deep understanding of the physical system. If adequate data history is available that represents the system performance, data driven methods can be
used. Artificial Neural Network is an important data driven analytical redundancy approach for sensor fault detection.

Testing has shown that the neural networks approach has proven to be an extremely powerful method for fault detection.

4.2.1 Neural Network for Sensor Fault Detection

The neural network, as the name suggests it is network of neurons connected together to form different layers of the network. The simple three layer neural network is shown in fig 4.1. The neurons transmit information to each other through synapses that will transform the information by a parameter called ‘weights’. Fig 4.1 shows weight matrix \([v]\) between input and hidden layer and weight matrix \([w]\) between hidden and output layer. The operation of artificial neural network depends on the three following parameters:

1. The structure of neural network i.e. number of neurons in each layer, number of layers, number of inputs and number of output.
2. The training rule that is the method through which the synaptic weights got updated. In the proposed work gradient descent back-propagation algorithm is used which is one of the supervised learning algorithm.
3. The activation function that translates a neuron's weighted input to its output.

Back-propagation algorithm necessitates the activation function used by the artificial neurons to be differentiable.

Auto-associative Neural Networks (AANN) is an alternative elucidation for Sensor fault Diagnostics. The AANN concept, which was developed by Kramer [67 68] can learn the co-relationship between plant variables. The synaptic weights of the neural network are trained to learn the co-relation between the system variables or sensor data. The structure of AANN is same as ANN having a input layer, an output layer and some hidden layers. Theoretically, it is sufficient for the AANN to contain three hidden layers [67]. However more hidden layers might be used for improved performance [69]. The architecture of a three hidden layer AANN is shown in Fig 4.2. The outputs of AANN are trained to resemble the input in case of healthy data. The output layer of AANN produces
a distorted version of the inputs but equal in dimension to the input in case of sensor failure.

For the detection of faulty sensor, all the sensor data are given as input to AANN, considering that there is some co-relation among these data. Synaptic weights are then adjusted to learn this co-relation through back propagation algorithm. Once trained the output of the auto-encoder matches the input closely for healthy sensor input and if some sensor is faulty, a considerable error is generated between input and output.

The non-matching of input and output of auto associative neural networks gives an indication that one or more sensor is faulty. But the determination of exact faulty sensor will still remain an issue. The researchers in [70, 71] proposed that the same system can even locate the exact faulty sensor. The error corresponding to each input or output can be used to detect the faulty sensors since the network has learned the relationship between the sensor data. The block diagram representation of faulty sensor detection system is shown in fig 4.3.

Fig 4.1 Neural network model with one hidden layer

For the detection of faulty sensor, all the sensor data are given as input to AANN, considering that there is some co-relation among these data. Synaptic weights are then adjusted to learn this co-relation through back propagation algorithm. Once trained the output of the auto-encoder matches the input closely for healthy sensor input and if some sensor is faulty, a considerable error is generated between input and output.

The non-matching of input and output of auto associative neural networks gives an indication that one or more sensor is faulty. But the determination of exact faulty sensor will still remain an issue. The researchers in [70, 71] proposed that the same system can even locate the exact faulty sensor. The error corresponding to each input or output can be used to detect the faulty sensors since the network has learned the relationship between the sensor data. The block diagram representation of faulty sensor detection system is shown in fig 4.3.
4.3 Implementation of missing sensor detection system

The detection of sensor failures in the present work is done using AANN. If the difference between AANN input and corresponding output exceeds a definite threshold (while other remaining still low) then that sensor is declared as faulty. If two sensors fail simultaneously, two corresponding errors of AANN will be more than defined threshold. Once a sensor fault is determined, that particular sensor is disconnected from the system to prevent any malfunctioning in the system. The block representation of the system for detecting the faulty sensor is shown in fig.4.4. It can be seen that left part of the figure is
same as for power quality improvement with DVR in chapter 3. The right part is auto-
encoder and a simple logic circuit based faulty sensor detection system.

The auto-encoder in this work is a three-layer feed-forward neural network with
bipolar hyperbolic tangent activation function in the hidden layer shown in fig. 4.5(a).
The relationship among sensor data is stored in hidden layer. Large dimension of the
output layer as compare to hidden layer is to make the output dimension same as that of
input.

![Block diagram of missing sensor detection system](image)

Although control of DVR requires only voltage at point of common coupling and
at the load point, four set of sensor data, voltage and current at point of common
coupling, and at the load point are taken as input to neural network.
A neural network is designed in MATLAB using “newff”, with bottle neck middle layer having eight neurons, nonlinear transfer function and two outer layers having thirty two neurons. The training set for the network is generated by generating a ‘mat’ file from Simulink model having data for PCC and load for different fault conditions and healthy condition. Then the network is trained using command ‘train’ to generate the output same as input for all the conditions i.e. the bottle neck layer will learn the co-relation between the inputs. The ‘gensim’ command gives a Simulink model for trained neural network. Its variation of mean square error of auto-encoder with epochs is shown in fig 4.5(b). It shows that mean square error during training reaches around 3 in 101 iterations.
Fig 4.6 Flow chart of Logic circuit for detection of faulty sensor
When the neural network is trained, its input and output no longer match if any sensor data is missing. But for detecting the faulty sensor the logic circuit is developed. The flow chart in fig 4.6 shows that when the error of auto-encoder is more than a pre-defined constant (em=80) and if there is no ground fault in that voltage, which is detected by increase in current in that branch. In this condition if voltage is zero continuously, that voltage sensor is a failed sensor, and similarly for current.

4.4 Results and discussion

The system is considered to have twelve sensors; three voltage sensors at PCC, three current sensors at PCC, three voltage sensors at load point and three current sensors at load point. An auto-encoder with twenty four inputs (twelve sensor values and their delayed values) is trained to detect failure of one or more sensor. To find out faulty sensor out of those twelve sensors a logic circuit is developed. The challenge for the system is to discriminate a ground fault with missing sensor condition, since in both the cases the values of voltage data is zero.
Fig 4.7 Faulty sensor detection during a two phase fault at PCC (a) PCC voltage (b) auto-encoder error

A 2 phase fault between phase a and c is forced on the system at 0.04 s, that causes a PCC line voltage (Vac) to be zero, at 0.07 a voltage sensor output at PCC (line voltage Vab) is switched to zero value (as sensor fault). The line voltage at PCC is shown in the fig 4.7(a). The error of auto-encoder, which is the difference between its input and output, is shown in the fig 4.7(b), auto-encoder has successfully detect that some sensor is missing sensor as error increases (fig.4.7 (b)) at 0.07s.

When auto-encoder error goes above 80 (a predefined constant), the logic circuit is initiated to detect the faulty sensor. Result of the logic circuit detecting the actual missing sensor is shown in fig 4.8. When at 0.07 a PCC voltage (vab) sensor output is forced to zero value it is detected as one at first output while all others remain zero.

When a load voltage sensor failed at 0.04, the error of auto-encoder is shown in the fig 4.9. The abrupt increase in the auto-encoder error at 0.07 gives an indication that, any one of the twelve sensors is faulty. The logic circuit detected the faulty sensor as a load voltage sensor shown in fig 4.10. Thus auto-encoder with a simple logic circuit is detecting missing PCC as well as load voltage sensors and can discriminate a ground fault from a missing sensor issue.
(a) Detection of fault in voltage sensor at PCC  
(b) Detection of fault in current sensor at PCC  
(c) Detection of fault in voltage sensor at load pt.  
(d) Detection of fault in current sensor at load pt.  

Fig 4.8 output of logic circuit detecting a faulty sensor at PCC
4.5 Overview of Faulty Sensor Data Restoration systems

An auto-encoder can learn the data correlations through scrutiny of historical data. Once trained, sensor data correlations embedded by the auto-encoder can be used by some search algorithms (e.g., PSO in this work) to restore missing data. The block representation of the scheme for failed sensor data restoration, used in the present work is shown in fig 4.11. For majority of systems the variables or the measured data are connected to each other through some relation. When this relation is sufficient to be learned by auto-encoder, the missing sensor data can be restored with the help of remaining healthy sensor data [72], [73].
Fig 4.10 Output of logic circuit detecting a faulty sensor at load
4.5.1 Particle swarm optimization

Particle Swarm Optimization (PSO) developed by Eberhart and Kennedy is a computational search and optimization method inspired by behavior of birds flocking in 1995. To improve the convergence speed and operation different extension to basic PSO algorithm were proposed by number of researchers. But in this work and even for many simple optimization problems the basic PSO algorithm has given sufficient results.

In PSO each particle has two important qualities [74], first is to explore different regions in the search space to reach an optimum value, and second is to confine the search in the most probable region so as to converge to optimum faster. Having these qualities the particles flutter in the search space by retaining its own best position till that time, which is called local best and also storing the information about best position attained till that time by any particle in the group, which is the global best.

Each particle has some position at any time represented by $x$, and a velocity that determines the next position of the particle [75].

$$x_i(t + 1) = x_i(t) + v_i(t + 1) \quad (24)$$

Where,
\[ v_i(t) = v_i(t - 1) + c_1 r_1 (localbest(t) - x_i(t - 1)) + c_2 r_2 (globalbest(t) - x_i(t - 1)) \]  

(25)

With \( c_1 \) and \( c_2 \) acceleration coefficient and \( r_1 \) and \( r_2 \) as random vector.

The PSO can be implemented by the following steps, to find the minimum of any given objective function PSO:

1. Set the number of particles in the search space to a fix value let it be \( N \). The value of \( N \) should not be too large as it will increase the convergence time and too small value of \( N \) will make it difficult to reach the optimum value for large search space.

2. Provide a position to each particle as \( x_1, x_2, ..., x_n \) between provided range of the positions (i.e. between some maximum and minimum possible position). These positions are usually uniformly distributed random numbers.

3. Now for each iteration ‘i‘, starting from 1 calculate the local best denoted by \( P_{best}(j) \), where \( j \) is the particle and global best denoted by \( G_{best} \). \( P_{best}(j) \) is the minimum value of objective function reached by particle \( j \) till that iteration. \( G_{best} \) is the minimum value of objective function among all particles.

4. Then compute the velocity for all particles using equation (25), assuming initial velocity to be zero. This velocity will then give a new position to each particle close to minimum of objective function. This new position is calculated by equation (24).

5. Calculate the value of activation function for each particle position. When the value of objective function for all particles points a similar value, the solution is said to be convergent. If this is not the case the complete procedure is repeated till particles converge to a same value.

The advantages and problem with PSO are discussed in [11]. PSO is an optimization and search algorithm with very simple calculations. The reason for its higher optimization ability is that only the particle closest to optimum can transmit the
information. The main issue with particle swarm optimization is many times it got stuck in local minimum, also it is not suitable for shaking optimization problems.

4.5.1.1 Basic Variants of PSO

Many forms of basic PSO have been established to improve speed of convergence and quality of solution. All these forms are characterized by some variation in parameters or their parameter acceleration. The four such variants are discussed below and summarized in table 4.

a. Velocity clamping

The velocity clamping bounds the velocity within some maximum and minimum allowable limit. If the velocity of any particle moves beyond this limit it is set to the boundary value (maximum or minimum). The velocity variation is now governed by equation (26) given below.

\[
v_{ij} = \begin{cases} 
  v'_{ij}(t + 1), & \text{if } v_{ij}(t + 1) < v_{\text{max}}(j) \\
  v_{\text{max}}(j), & \text{otherwise} 
\end{cases}
\]  

(26)

Sufficient care should be taken while choosing maximum and minimum velocity values. Larger value of maximum can sometime distract the particle from optimum value and too small value may result in small search space for the particles. Velocity clamping did not affect the position of the particle. It only reduces the size of the velocity change.

b. Inertia weight:

By using this parameter the change in particle position is controlled by changing the contribution of velocity in the current position. It also eliminates the need for velocity clamping.

c. Constriction Coefficient

Velocity update equation that includes constriction coefficient changes to equation (27) shown below

\[
v_{ij}(t + 1) = x[v_{ij}(t) + \phi_1(y_{ij}(t) - x_{ij}(t)) + \phi_2(y_j(t) - x_{ij}(t))]  
\]  

(27)

Where

\[
x = \frac{2k}{|2 - \phi - \sqrt{\phi(\phi - 4)}|}
\]

With
\[ \phi = \phi_1 + \phi_2 \]
\[ \phi_1 = c_1 r_1 \]
\[ \phi_2 = c_2 r_2 \]

Table 4 Basic Variants of PSO

<table>
<thead>
<tr>
<th>Basic Variant</th>
<th>Function</th>
<th>Advantage</th>
<th>Disadvantage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Velocity Clamping</td>
<td>Limits the velocity within limit.</td>
<td>Helps in fast convergence</td>
<td>May lead to non-convergence when all velocities become equal</td>
</tr>
<tr>
<td>Inertia Weight</td>
<td>The particle position variation is controlled by changing the contribution of velocity.</td>
<td>Increase the convergence ability</td>
<td>May lead to increased convergence time.</td>
</tr>
<tr>
<td>Constriction Coefficient</td>
<td>It ensures the stability of the algorithm during convergence.</td>
<td>Increase the convergence ability</td>
<td>May cause fluctuation in particle position.</td>
</tr>
<tr>
<td>Synchronous and Asynchronous Updates</td>
<td>Advantages of both synchronous and asynchronous updates</td>
<td>Ensures fine tuning and convergence rate.</td>
<td>Complicated algorithm as compare to PSO</td>
</tr>
</tbody>
</table>

d. Synchronous and Asynchronous Updates

In synchronous PSO which is the normal PSO algorithm the position and velocity of any particle is updated only after the performance evaluation of all the parameters is over. But in asynchronous update the position and velocity of any particle is updated as soon as its performance is evaluated. When both this update forms are merged together to improve the performance it is synchronous and asynchronous update form of PSO. The particles are divided in small groups whose updates are synchronous, while all the groups updates asynchronously.

4.6 Implementation of Faulty Sensor Restoration System

Once trained the input and output of Auto-encoder will match till all the sensor data are healthy. In case of failure of any sensor, there will be significant difference between auto-encoders input and output. This increase in auto-encoder error activates the PSO module of the system that form a loop with auto-encoder. The output of PSO block
is then given to auto-encoder in place of faulty sensor data and the error of auto-encoder is used as objective function for the PSO to be minimized so as to estimate the missing sensor data. The expression for the objective function for PSO is given by (28), which is the difference between the auto-encoder input (that includes healthy sensor data as well as the estimated missing sensor data be PSO) and output.

\[ f = \|E_s\| = \|S_H - \hat{S}_H(S_H, S_M)\| \] (28)

Where \(S_H\) represents the auto-encoder input, \(\hat{S}_H\) output of auto-encoder, and \(S_M\) represents the correct values of the missing sensor data. Once the error \(\|E_s\|\) is less than a preset threshold, the output of the PSO is considered to be the closest estimate of faulty sensor data.

In this work velocity clamping is used, that helps in fast convergence as summarized in Table 4. The velocities of particles are updated by equation (26).

The block representation of auto-encoder detecting the missing sensor and PSO module restoring the missing sensor value is shown in the fig 4.13. PSO block is enabled subsystem which is enabled only when the error of auto-encoder is above a predefined constant. The input to PSO is the auto-encoder error corresponding to faulty sensor. It will generate its output so as to minimize this error. The PSO has two outputs one given to auto-encoder in case of sensor failure so as to minimize its error, and another is the data corresponding to minimum error that is restored values of faulty sensor given to control system.

4.7 Results and Discussion

The input to the control system including a repetitive controller are three phase PCC line voltage and three phase load line voltage. In case of failure of a voltage sensor at PCC or load point, one input to the controller is zero that deteriorates the controller performance and affects the load voltage. But if missing sensor detection and restoration system is connected to the system missing sensor will be detected by auto-encoder and triggers the PSO block that will further restores the data. Thus controller performance remains unaffected by a sensor failure.
Initialize \( n (=50) \) particles position and velocity as 0, and \( n=1 \)

for \( n \)th particle set the position to a random value between -8 to 8

Set Pbest equal to position of \( n \)th particle, and transfer it to auto-encoder input and read the auto-encoder error for that input

\[ n = n + 1 \]

if \( n > 50 \)

For each particle compare the old error with present error, if old error is more than present error

set the current particle position as its best position, set old error as current error

Find the global best position among all particles with minimum error, give it as restored sensor value at the output

If the error corresponding to global best is less that predefined constant (0.3 here)

Generate it as restored output value

find the new particle position and velocity with PSO update equations, while limiting the position between -8 to 8 and velocity between -1 to 1

Fig 4.12 Flowchart for PSO
Auto encoder (Auto assosiativ-e neural network)

Substr-actor

Large inductive load

AC to DC Controller

Control Switch

Controlled Switch

PCC fault point

Step-Down transformer

Distribution System

Logic circuit

Embedded matlab function

PSO

AE error for failed sensor

If error> some constant

Missing sensor

Vp Ipc Vl Lo

Repetitive controller

Restored Sensor value

Fig 4.1
3 Block dia gram of mis sing sensor detection

Sensit ive load

Voltage phase for downstream

Downstream fault detector

DVR with passive filter

PWM controller

Controlled switch

Voltage magnitude

AE error for failed sensor

Restored Sensor value
4.7.1 Faulty sensor data restoration for PCC sensor failure

When a voltage sensor at PCC fails at t=0.02, the input to the repetitive controller is shown in fig 4.14(a), this causes the error of auto-encoder to rise and enables the PSO block. The PSO block along with auto-associative neural network restores the missing sensor value. The output of the PSO block is shown in fig 4.14(b), the red one is the output of PSO and blue is the actual PCC voltage of that phase. The restored three phase PCC voltage that will be the input to the repetitive controller is shown in fig 4.15(a).

It can be seen that although one PCC voltage sensor has failed the controller input is still very close to actual PCC voltage. With this PCC voltage input the controller performance will be maintained during sensor failure and load voltage remains almost unaffected as shown in fig 4.15(b).

![4.14(a) Three phase PCC voltage sensor output](image)
4.14 (b) PSO output and actual sensor value if not failed

Fig 4.14 Restoration of failed PCC sensor data
(a) three phase PCC voltage sensor output
(b) PSO output and actual sensor value if not failed

4.15(a) Three phase PCC voltage with restored missing sensor data
Fig 4.15 Restoration of failed PCC sensor data (a) three phase PCC voltage with restored missing sensor data, (b) load voltage

4.7.2 Data restoration for load sensor failure

When a load voltage sensor is failed at 0.02 the input to the controller without restoration system is shown in fig 4.16(a). But with a sensor fault detection and restoration system, this failure of sensor increases the error of auto-encoder as shown in fig. 4.16(b). This will trigger the PSO block that restores the missing sensor data within a cycle, as in fig 4.16(b) the error of auto-encoder decreases within a cycle, showing that the missing sensor data is restored by the PSO block. The output of PSO block is given to Repetitive controller through a sample and hold block so that sampling frequency of the signal can be maintained for proper operation of controller. To verify whether the PSO has restored the correct load voltage, the actual value of voltage and PSO block output are shown superimposing in fig 4.17(a). This restored voltage when given at the input of control system the load voltage remains least affected by sensor failure as in fig 4.17(b).
Fig 4.16 Restoration of failed load sensor data (a) output of three voltage sensors at load point (b) auto-encoder error
Fig 4.17 Restoration of failed load sensor data (a) Restore load voltage with actual load voltage, (b) load voltage
4.7.3 Faulty PCC voltage sensor data restoration during sag

In the previous case the failed sensor is restored in healthy system conditions but when the system is facing some power quality issues the PSO block must restore the faulty sensor data. Induction motor is switched in the system at t=0.03s causes 15% sag in PCC voltages; at 0.07s a voltage sensor at PCC is failed, the output of voltage sensors at PCC is shown in fig 4.18. This missing PCC voltage is restored by PSO with auto-encoder. The superimposed actual voltage and the voltage restored by PSO block are shown in the fig 4.19(a). Since there is a sag in the PCC voltage the restored PCC voltage is also a sagged voltage with a peak of around 514.6V. The restored PCC voltage is shown in the fig 4.19(b). When this restored PCC voltage is given to the control system, its performance remains unaffected by sensor failure and load voltage remains close to reference value even during a sag in voltage at PCC as shown in the fig 4.19(c).

Fig 4.18 PCC voltage showing sag at 0.03s and a sensor fail at 0.07s
4.19(a) Restored PCC voltage with actual voltage

4.19(b) PCC voltage with restored sag voltage
4.19(c) Load voltage

Fig 4.19 Restoration of failed PCC sensor voltage during sag in the system (a) restored PCC voltage with actual voltage (b) PCC voltage with restored sag voltage, (c) load voltage

4.7.4 Faulty load voltage sensor data restoration during sag

When a load voltage sensor failed during sag at PCC, the controller will get incorrect input so the DVR may insert incorrect voltage to the system. This will lead to a deviation of voltage across sensitive load from the reference value. With the sensor fault detection and restoration system, the auto-encoder will detect missing sensor and triggers the PSO block that restore the missing sensor data. The voltage sensor output at load in fig 4.20(a) shows sensor failure at 0.06s. The restored voltage by PSO with actual load voltage is shown in the fig 4.20 (b). It can be seen that PSO has restored the exact load voltage. Thus load voltage as shown in the fig 4.21, will be least affected by a load voltage sensor failure even during a sag in the system.
Fig 4.20 Restoration of failed load sensor data during sag at PCC (a) load voltage sensor output (b) Restored load voltage with actual voltage.
4.7.5 Co-simulation for PSO

Again by following the steps of co-simulation, the PSO block is replaced by Modelsim simulator loaded with the VHDL code for PSO block and results are generated. All the parameter of MODELSIM co-simulator is provided same as discussed for repetitive controller. The capacity and memory list for PSO is shown in fig. 4.22 and fig. 4.23. Results of co-simulation for a PCC voltage sensor failure with sag in the system are shown in fig. 4.24 (a) and (b) that is comparable with corresponding simulation results of Simulink. The output of PSO and the actual value of sensor output are shown together in fig. 4.24(a) (red is PSO output and blue is actual sensor value if not failed).

The voltage across considered sensitive load when a missing PCC sensor data is restored by the system is shown in fig 4.24(b).

When the load voltage sensor fails the results are shown in fig. 4.24. The input to the repetitive controller is difference between reference voltage and load voltage. When a load voltage sensor fails, the co-simulator block load with VHDL code for PSO restores the missing sensor data but at different frequency, that will deteriorate the performance of
the controller. To overcome the issue the restored data is given to controller through a sample and hold block. This block will change the restored voltage frequency and controller compensates any deviation in load voltage.

![Fig 4.22 Memory requirement for PSO implementation in FPGA](image1.png)

**Fig 4.22 Memory requirement for PSO implementation in FPGA**

![Fig 4.23 Memory list for PSO block](image2.png)

**Fig 4.23 Memory list for PSO block**
4.24(a) PSO co-simulator output with actual voltage

4.24(b) Load voltage

Fig 4.24 PCC sensor failure during co-simulation (a) PSO co-simulator output with actual voltage, (b) load voltage
4.8 Conclusion

The Simulink as well as MODELSIM simulations have verified the successful detection and restoration of missing sensor. In both healthy system condition, as well as when controller is compensating some power quality issues the auto-encoder and PSO have detected the missing sensor and missing data is restored by PSO for proper functioning of the control system.