Simulation of Smart Fault Location, Isolation, and Service Restoration Scheme with Load Balancing in Distribution Grid

CHRISTIAN RUVALCABA, HA THU LE
Department of Electrical and Computer Engineering
Cal State Polytechnic University, Pomona
Pomona City, California 91768
UNITED STATES OF AMERICA

Abstract - Fault location, isolation, and service restoration (FLISR) technologies have been recognized as advanced and effective tools to increase resiliency and reliability of power distribution networks. As FLISR technologies are evolving, improvement is required for them to reach their full potential. This study enhances an existing FLISR scheme designed to automatically detect and isolate faults on a feeder by adding a new load balancing algorithm. The new algorithm enables automatic routing of power to faulty feeder load while protecting healthy feeders from overloading. By automatic load balancing, the new algorithm creates a FLISR scheme that is significantly more effective than the existing. The enhanced FLISR scheme is simulated using MATLAB Simulink where its control algorithms are implemented using Stateflow. Simulation results show that the enhanced FLISR scheme with the load balancing is effective in locating and isolating faults, as well as in re-routing power to loads of faulty feeders. The improved FLISR scheme is able to operate effectively on different feeders with different loading levels. The study outcomes are helpful for researchers and system planners where they can use the Simulink FLISR system to analyze diverse scenarios or create new FLISR versions. Furthermore, the simulation system can be used as an education tool to visualize FLISR technologies to technicians and the public. Overall, the study contributes some insightful understanding of the FLISR technology that promotes its implementation in Smart Grid to reduce service interruption and ensure better power supply to customers.

Keywords: Control algorithm, distribution automation, fault location isolation, load balancing, power distribution, service restoration, smart grid.

Received: May 30, 2023. Revised: June 19, 2023. Accepted: June 21, 2023. Published: June 21, 2023.

1. Introduction

Continuous electricity supply is a primary concern of power system operators following the restriction of the power industry, notably privatization and deregulation [1]. In addition, unexpected power outages in distribution systems have severe social and economic effects for utilities and consumers. Therefore, it is very important that the distribution system is reliable and efficient due to its linkage between end users and the upper levels of power delivery.

In order to mitigate the negative effects of power outages, a self-restoration scheme for distribution systems is created to detect and isolate faults and restore electricity around the faulted location. The self-restoration scheme, also known as fault location, isolation, and service restoration (FLISR) scheme, consists of a combination of heuristics, genetic algorithms, and advance communication technologies [2]. The generalized self-restoring follows the similar procedure of collecting electrical and operational system information to open and close switches dependent on the event that occurs [3]. FLISR schemes on distribution systems have the objective of performing fault location, isolation, and service restoration in an automated fashion, without or with limited distribution system operator and repair crew intervention [4]. The advantages of the scheme include system increased reliability due to outage duration reduction, efficient use of resources (e.g. operators, technicians), and increase productivity flexibility [5]. In addition, grid reliability is increased thanks to reduction in time required for locating and isolating faults on a feeder, as well as restoring customer service by distributing load to healthy areas. Self-restoring schemes are an inherent part of distribution automation and the

Smart Grid and expected to play a fundamental role in modern and future distribution systems [4]. It is worth noting that a drawback of this scheme is their dependence on switching technology such as protective and switching devices, adaptive protection, sensors, enterprise systems, and communications infrastructure. Therefore, the cost and effort to implement a self-restoration scheme likely are considerable, especially for existing distribution systems where existing protection systems are already built. However, the increase in reliability and efficiency for the consumer and operators outweigh the cost of implementing the scheme [6].

Different FLISR schemes have different algorithms to perform the fault isolation and load distribution. However, the basic concepts of FLISR schemes are similar, as follows. After a fault is occurred on a power line, a FLISR scheme detects it and isolates the faulty power line (or section of the line). Then, based on readings of relevant sensors, the control algorithm allows unfaulty sections of the affected line to be energized by neighboring feeders. The resulting benefits lead to a growing number of self-restoration projects being implemented by utilities as part of their system modernization plans.

The authors of [7] implemented their hardware-based FLISR scheme by using small-scaled DC power distribution system. The results of their project are successful in detecting and isolating a fault on a power line. However, their load balancing algorithm was not effective. Further, it is noted that multiple FLISR schemes focused on fault location and isolation without adequately addressing load balancing or overloading protection of neighboring healthy feeders when attempting to supply power to loads of faulty feeder(s) [1] [5] [19].

In this study, we aim to fill in the aforementioned gap by creating a new load balancing algorithm that works with fault location and isolation algorithm. It enables automatic distribution of load from the affected feeder to neighboring healthy feeders while protecting the healthy feeders from overloading. Adding the new load balancing algorithm results in an advanced FLISR scheme of higher effectiveness, as shown later by simulation. Our work aims to improve the FLISR scheme in [7] and help enhance other existing FLISR projects. In addition, we aim to provide a simulation tool to assist researchers and system planners to study FLISR schemes and create new versions. We adopt a distribution grid with real-world variables for simulating various fault scenarios to verify the improved FLISR scheme. The simulation is done using MATLAB Simulink with the Stateflow toolbox which has been used widely to design control algorithms. The following sections describe the existing FLISR scheme that we improve, the new load balancing algorithm that we create, along with the simulation system and results.

2. Existing FLISR scheme

The considered (existing) FLISR scheme in [7] consisted of two algorithms, fault isolation and load balancing, shown in Figures 1 and 2.

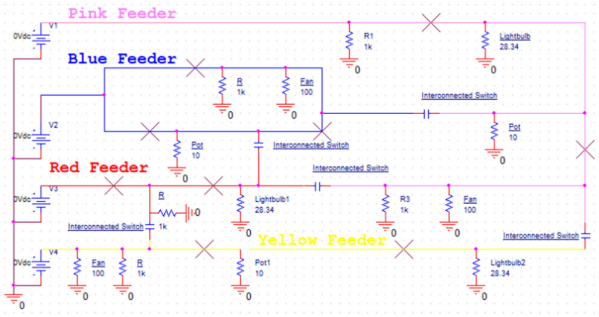

Fig. 1 Diagram of smart service restoration scheme based on PSPICE [7]

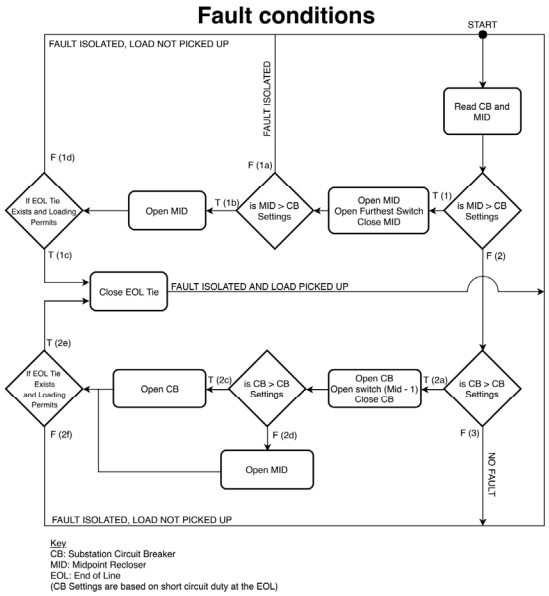

Fig. 2 Flowchart of fault detection and isolation algorithm [7]

The authors of [7] implemented their FLISR scheme on a small-scale DC distribution system and tested the scheme for different fault locations and load levels to show the accuracy of the scheme. The constructed distribution grid (Fig. 1), consists of 4 feeders with many switches and loads. The authors goals are to automatically detect and isolate faults and distribute affected loads to neighboring feeders. The results of the implemented scheme were successful in fault detection and isolation, but not successful in distributing power to affected loads, due to underdeveloped load balancing logic.

In efforts to resolve the previous study deficiency, we create an improved load balancing algorithm for use with the original fault detection and isolation algorithm. The newly-created load balancing algorithm improves the previous FLISR scheme, as explained in following sections.

3. Control algorithm

The control algorithm includes two logic algorithms: (a) fault detection and isolation, and (b) newly-created load balancing. The fault detection and isolation algorithm is the same algorithm developed by [7]. Fig. 3 shows the flowchart for the improved load balancing algorithm. Both the algorithms work together in order to isolate faults and balance load on the distribution grid.

3.1 Fault detection and isolation algorithm

Referring to Figures 2, 3 and 4, the fault isolation logic reads the current through the circuit breaker and midpoint sensors, and a value for the circuit breaker limit. The value of the circuit breaker limit is dependent on the rating of the circuit breaker. This allows system operators to have personal reset the breaker if it is tripped. Afterwards, a series of conditions are used to continuously compare the measured values of the circuit breaker and midpoint to the circuit breaker limit. The approach of the improved algorithm is that the fault is always assumed to be at a location ahead of the Far Switch (FS). Then a loop goes on to check the next section MID to FS and it repeats until it reaches the last section next to source Circuit Breaker (CB). If a fault is detected by the logic conditions, then a series of switches will open or close appropriately to isolate the fault. The switches are circuit breaker (CB), midpoint switch (MID), switch after the midpoint also called far switch (FS), and switch before the midpoint (SW1). Fig. 4 demonstrates a single feeder

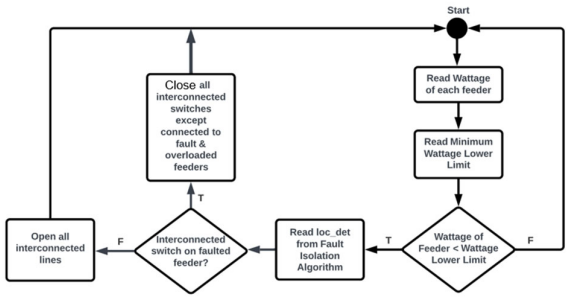

Fig. 3 Flowchart of improved load balancing logic.

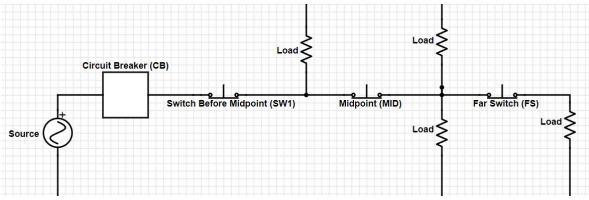

Fig. 4 Single feeder circuit with fault detection and isolation scheme.

circuit with the switches of the fault detection and isolation logic. Figures 5, 6, and 7 demonstrates implementation of the fault and detection algorithm of one of the feeders of Fig. 1 using Simulink Stateflow toolbox. The fault detection and isolation algorithm is implemented for all feeders.

In addition to the fault detection and isolation algorithm, we declare a new variable 'loc_det' in order to keep track of where the location of the fault is detected. The new variable is used in the load balancing algorithm which will be discussed in Section 3.2.

Pseudo code for fault detection and isolation logic:

Start State – Read the CB and MID sensor, and CB limit. Also, initiate all switches to be closed: CB, SW1, MID, and FS. Declare 'loc_det' = 0.

Continuous Loop – If the MID sensor is less than the CB limit and if the CB sensor is less than the CB limit go back to the Start State. This determines there is no fault on the feeder.

Phase I State – If the MID sensor is greater than the CB limit, go into the Phase I state and execute the following commands: open the MID switch, open the FS switch, then close the MID switch. These commands are to cut electricity flow beyond the FS switch. A fault is detected on the feeder and determined to be between the MID and FS switch or beyond the FS switch.

Phase Ia State – If the MID sensor is greater than the CB setting go to the Phase Ia state and execute the

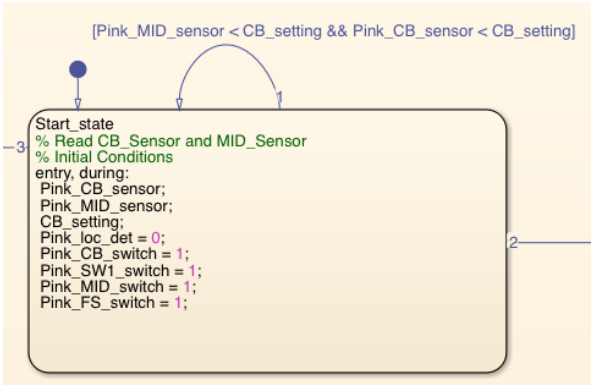

Fig. 5 Start state of fault detection and isolation algorithm for Pink feeder.

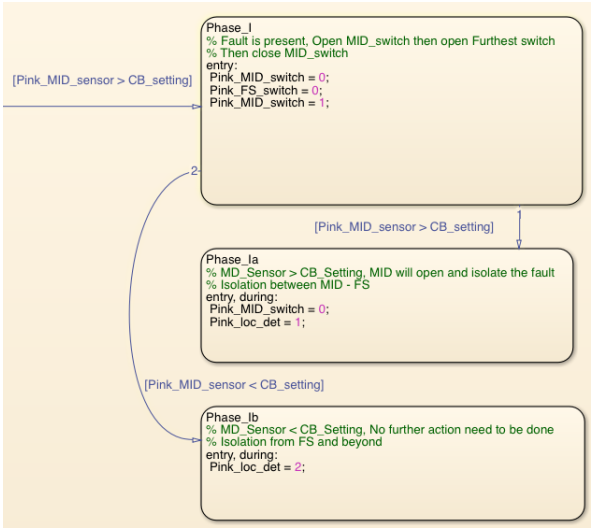

Fig. 6 Phase I states of fault detection and isolation algorithm for Pink feeder.

following commands: open the MID switch and update the 'loc_det' variable to 1. The fault is located and isolated between MID and FS.

Phase Ib State – If the MID sensor is less than the CB setting go to the Phase Ib state and execute the following commands: update the 'loc_det' variable to 2. The fault is located and isolated from FS and beyond.

Phase II State – If the MID sensor is less than the CB limit and the CB sensor is greater than the CB limit, go into the Phase II state and execute the following commands: open the CB switch, open the SW1 switch, then close the CB switch. These commands are to cut electricity flow beyond the SW1 switch. A fault is detected on the feeder and determined to be between the CB and MID switch or between SW1 and MID.

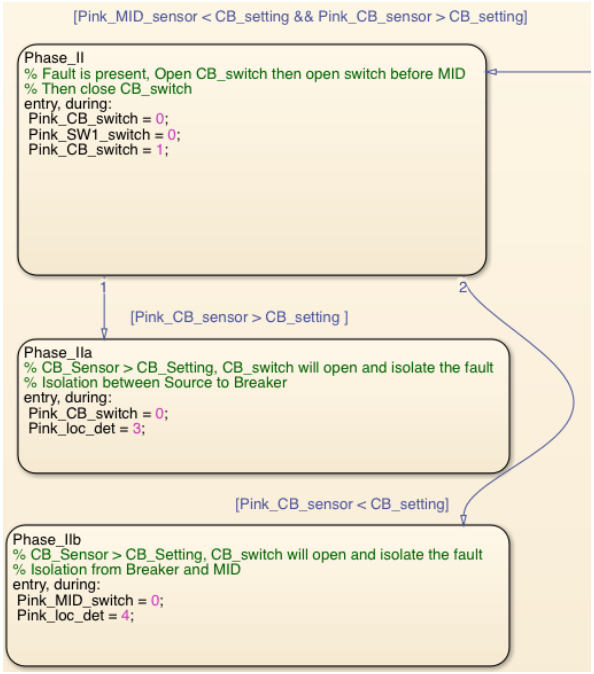

Fig. 7 Phase II states of fault detection and isolation algorithm for Pink feeder.

Phase IIa State – If the CB sensor is greater than the CB setting go to the Phase IIa state and execute the following commands: open the CB switch and update the 'loc_det' variable to 3. The fault is located and isolated between the source and SW1.

Phase IIb State – If the CB sensor is less than the CB setting go to the Phase IIb state and execute the following commands: open the MID switch and update the '*loc_det*' variable to 4. The fault is located and isolated between SW1 and MID.

3.2 Improved load balancing algorithm

The improved load balancing algorithm runs parallel with the fault detection and isolation algorithm. Running the algorithms in parallel allows the load balancing algorithm to execute commands at the same time as the fault detection and isolation algorithm is ongoing. This is important because the load balancing algorithm should execute commands once a fault is detected on the feeder.

The load balancing algorithm involves three key components, the wattage of the feeder, the interconnected switches, and the location of the fault. The wattage of the feeder measures its real power consumption.

It is necessary to analyze the power consumption of the feeder and the interconnected switches are needed to re-route power flow to the affected feeder. Lastly, the location of the fault is needed to know

which interconnected switches should be open or closed. The declared *"loc_det"* variable in Section 3.1 keeps track of the location of the fault.

The process of the load balancing involves two states. State 1 involves reading the wattage of the feeders and initializing all the interconnected switches as open. The logic stays in state 1 until the power of one feeder is less than a lower limit. We determine the lower limit to be 5% of the rated wattage. The reason for this is that the acceptable voltage drop according to the National Electric Code (NEC) is 5%. Since power is the product of current and voltage, we carried over the 5% acceptance rate for wattage.

Then, in state 2, the load balancing algorithm closes interconnected switches to allow power flow from neighboring feeder(s) into the affected (faulty) feeder. The closed interconnected switches are determined by overserving the grid topology and the location of the fault. All interconnected switches are set to close except the ones connected on the same line of the fault. The faulty power line is determined from the fault isolation algorithm. In addition, there is a relegation on the interconnected switches for situation where a healthy feeder cannot provide power to the affected feeder due to it being overloaded. The interconnected switches go through a logic that analyzes the current state of each neighboring feeder. If a neighboring feeder exceeds its power carrying limit, it is not used to supply power to the faulty feeder. This is to protect the heavy-loaded feeder from failure.

Figure 8 demonstrates a two-feeder circuit with the interconnected switches of the load balancing logic and Fig. 9 demonstrates the state 1 or start state of the load balancing algorithm in the distribution grid. Fig. 10 demonstrates the implementation of state 2 or start state of a feeder of the load balancing algorithm in the distribution grid. The load balancing algorithm is implemented for all feeders.

The adjustments in the construction and implementation of the newly-created load balancing algorithm improve the overall FLISR scheme, making it more reliable. Instead of having the load demand determined by just the circuit breaker, we use the location of the fault to determine the need for load balancing. In addition, we also set up the interconnected switches to route power flow around the faulty section of the affected feeder. This allows re-routing power from one feeder to another in an effective manner by spreading the load from the faulty feeder to neighboring feeders, thereby reducing number of customers being affected by the fault. Another improvement is the relegation of the interconnected switches. We used a condition to regulate the interconnected switches to close only if the healthy feeder is not overloaded. It is worth noting that this load balancing algorithm has a limit to the amount of load a neighboring feeder can provide or withhold. This ensures that neighboring feeders are not overloaded.

The advantages of the improved load balancing logic are abundant and reliable. The algorithm allows the feeders to work together to share load and provide continuous power around the faulty section of the faulty feeder. The idea is to maintain and analyze supply/demand through all feeders. In addition, the algorithms should decrease downtime time due to faults and grid reliability by minimizing service interruptions for customers around the fault.

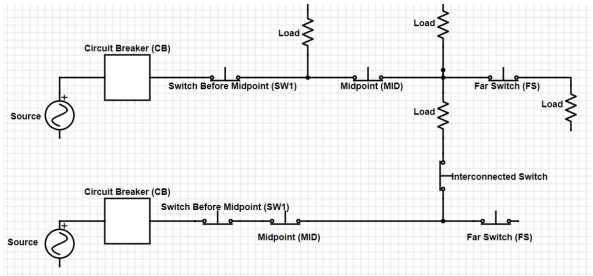

Fig. 8 Two-feeder circuit with load balancing implementation.

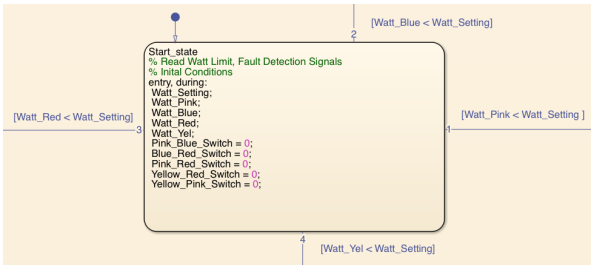

Fig. 9 Start state of improved load balancing algorithm.

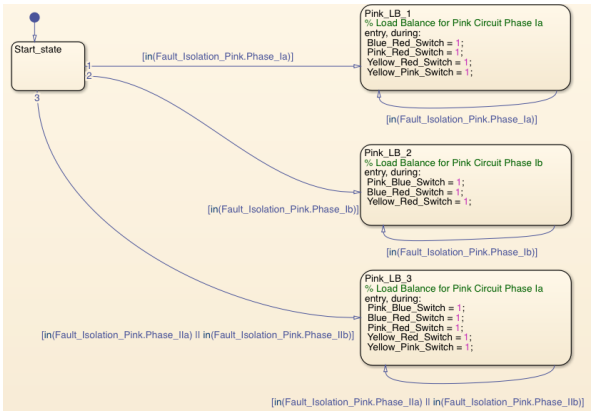

Fig. 10 Improved load balance state of Pink feeder.

Pseudo code for load balancing logic:

Start State – Read the wattage of all feeder, wattage lower limit, and initiate all interconnected switches to be open. The algorithm will stay in this state until the wattage of one of the feeders is less than the wattage lower limit, which indicates that the feeder is faulty.

Start State of Faulty Feeder – If the wattage of one of the feeders is less than the wattage lower limit, go into a new start state to activate load balancing logic for the faulty feeder.

Close Interconnected – Depending on where the fault is located by the fault isolation algorithm, the feeder-interconnecting switches not near the fault will close. This will allow power from other feeders to flow into the faulty feeder to help restore power to its loads as much as possible.

Overloading Protection – Using a condition to regulate the interconnected switches to close only if the healthy feeders are not overloaded. Also apply a limit to the amount of load that a neighboring feeder can provide or withhold.

4. Calculation of feeder wattage

This section demonstrates the calculations of the total wattage of each feeder in Fig. 1. The wattage is used in the load balancing algorithm presented in Section 3. Note that the resistant values used in [7] are based on data from an existing distribution network of Southern California Edison, a large power utility in California. Also in Fig. 1, DC voltage is used to power the LED strips representing feeders. Simulating with DC voltage on resistive loads gives a very close approximation of real-world values in terms of real power consumption.

As seen from Fig.1, the loads of the distribution grid are in parallel, and each feeder contains different amounts of loads. Using the rule of parallel resistor, we can calculate the equivalent total resistance (R_T) and get the total current with the equation:

$$I_T = V_s/R_T \quad (1)$$

where V_s is the source voltage and I_T is the total current. Next, we can use the following power equation:

$$W_T = V_s * I_T \quad (2)$$

The final solution of each feeder is presented in Table 1 through 4.

Table 1 Wattage of Pink feeder

Load	Resis-tor	Resis-tor	Bulb	Pot	Fan
Resistance (Ω)	1k	1k	28.34	10	100
R_T	6.78 Ω				
V_s	12.4 kV				
I_T	1828.90 A				
W_T	22.67 MW				

Table 2 Wattage of Blue feeder

Load	Resistor	Pot	Fan
Resistance (Ω)	1k Ω	10 Ω	100 Ω
R_T	9.00 Ω		
V_s	12.4 kV		
I_T	1377.77 A		
W_T	17.08 MW		

Table 3 Wattage of Red feeder

Load	Resistor	Bulb
Resistance (Ω)	1k Ω	28.34 Ω
R_T	27.55 Ω	
V_s	12.4 kV	
I_T	450 A	
W_T	5.5 MW	

Table 4 Wattage of Yellow feeder

Load	Resis-tor	Bulb	Bulb	Pot	Fan
Resistance (Ω)	1k	28.34	28.34	10	100
R_T	5.50 Ω				
V_s	12.4 kV				
I_T	2254 A				
W_T	27.95 MW				

5 Implementation and testing of improved FLISR scheme with load balancing

MATLAB Simulink is used to simulate the distribution grid of Fig. 1. The Simulink grid consists of four power sources, and many resistors, grounds, and manual and automated switches. There are current sensors for the CB and MID. The input of the switches use *"from"* tags to be the output of the algorithm and the current sensors use *"Goto"* tags to be the input of the control algorithms.

We simulate using DC voltage sources of 12.47kV, which is a standard distribution voltage level used on feeder lines. Overall, the simulation settings aim to ensure realistic conditions that, in turn, ensure the practicality of the FLISR scheme with respect to real-world distribution grids.

MATLAB Stateflow is used for the smart service restoration scheme, including (a) fault detection and isolation, and (b) load balancing algorithm.

The four distribution feeders of Fig. 1 are built in MATLAB Simulink and shown in Fig. 11 with same 12-47-kV DC sources. This is equivalent to a situation where the 4 feeders are supplied from the same substation. Figure 12 demonstrates the Stateflow implementation of the two logic algorithms with the input and output tags. This section demonstrates a step-by-step example of a fault between SW1 and MID on the red feeder.

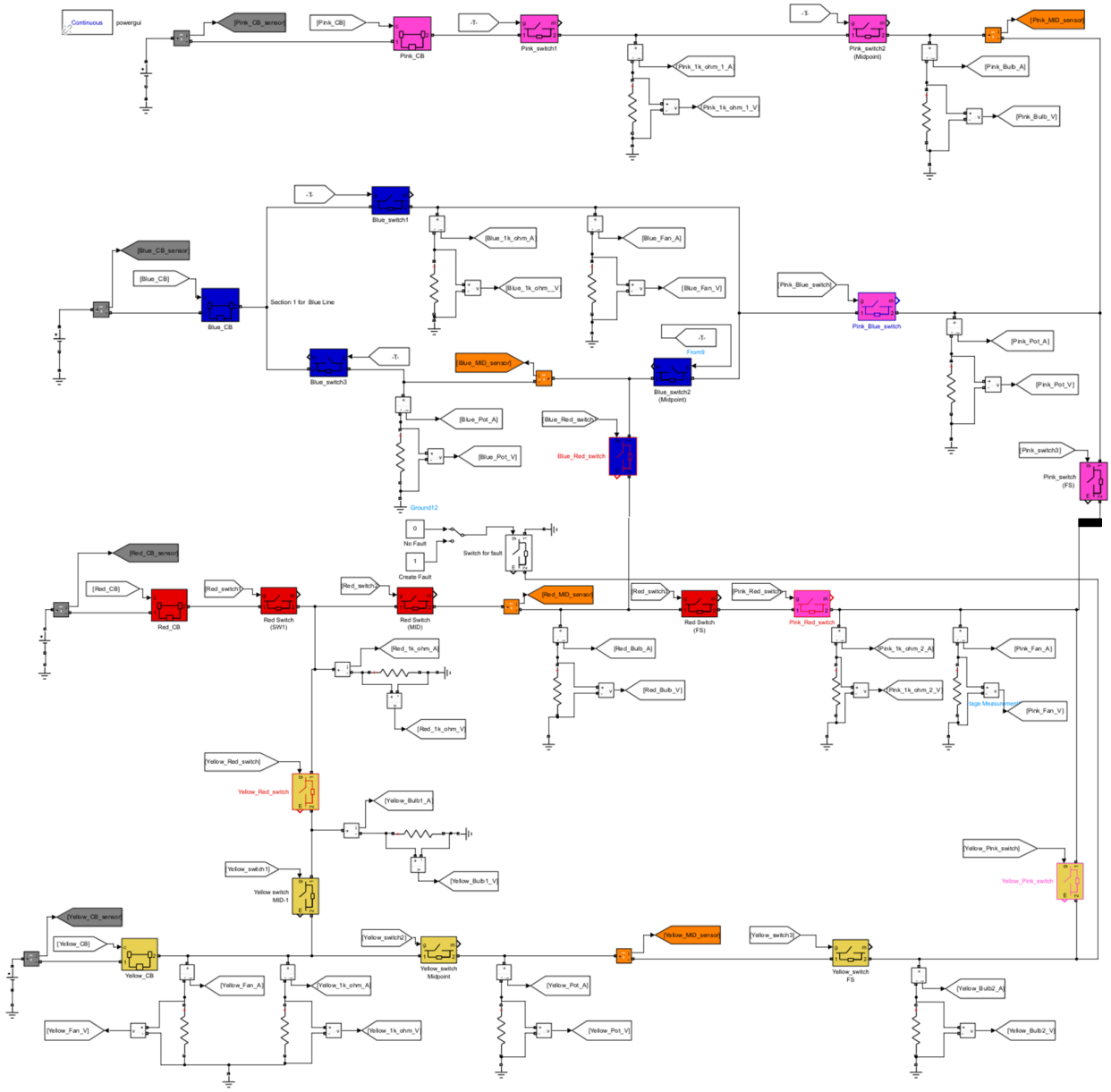

Fig. 11 MATLAB Simulink implementation of FLISR on distribution grid. The grid has 4 feeders (Pink, Blue, Red, and Yellow). The circuit breakers are highlighted and the fault creator is located in the middle of the diagram between Blue and Red feeders. A fault is created by connecting a feeder to ground.

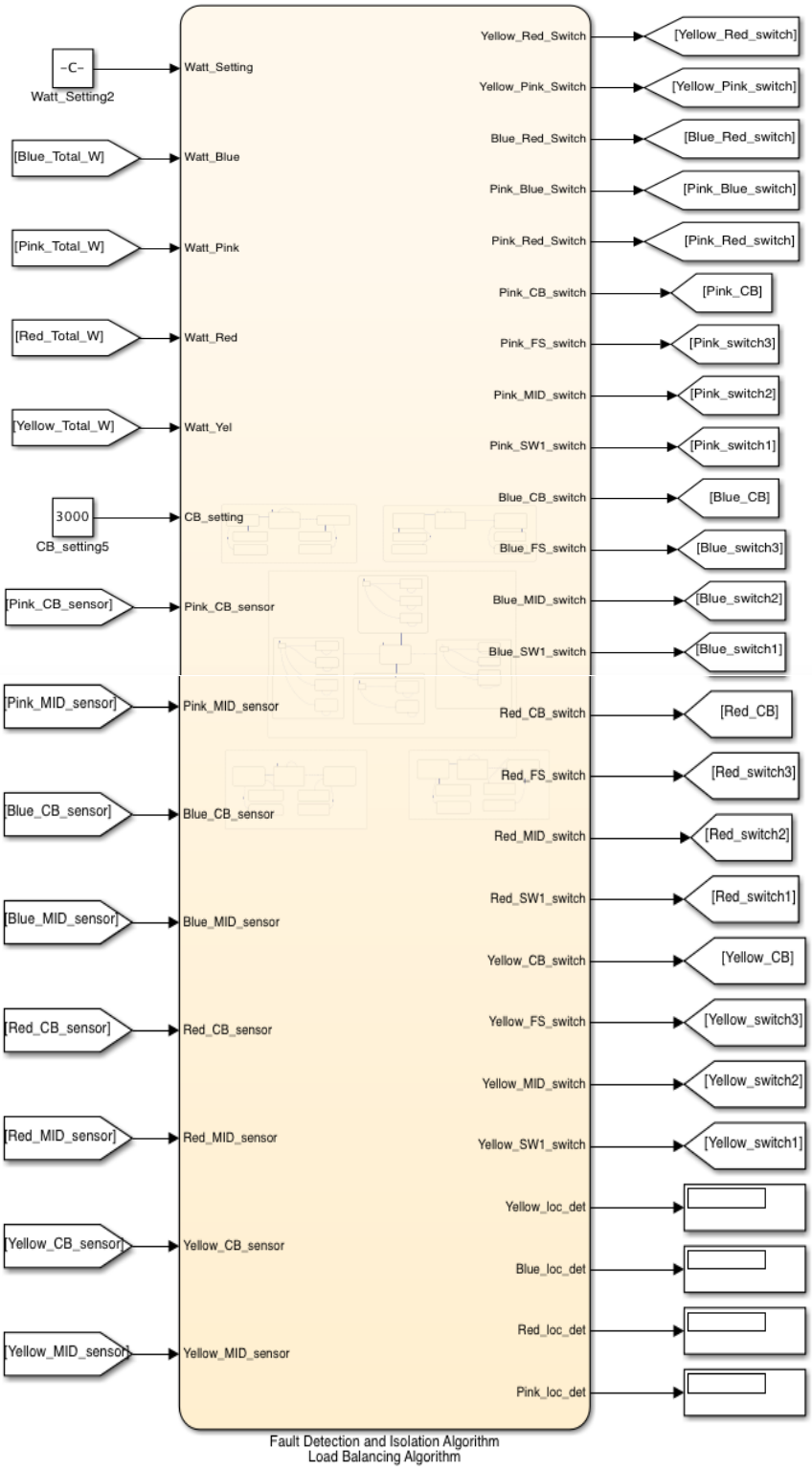

Fig. 12 Inputs and outputs of fault detection and isolation and load balancing algorithms. Complete Stateflow diagrams of fault location and isolation logics for Red and Pink feeders and load balancing logic for Red feeder are provided in Appendix.

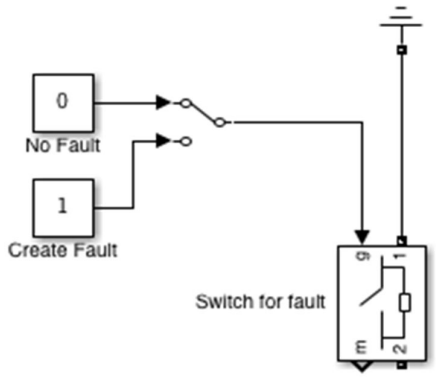

Fig. 13 Simulink manual fault creator

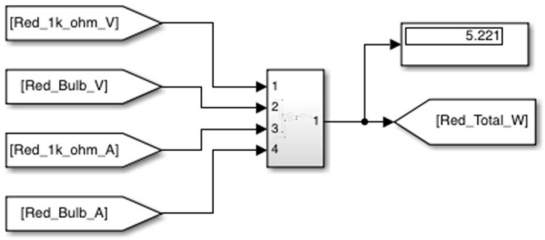

Fig. 14 Wattage system for Red feeder

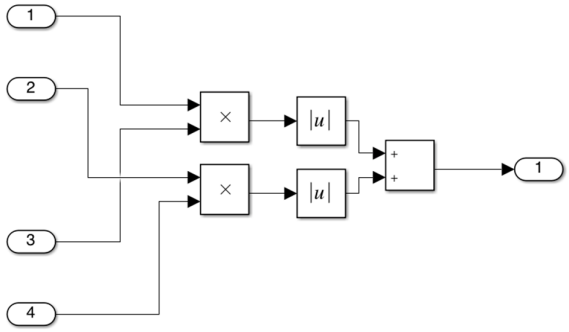

Fig. 15 Wattage calculation for Red feeder

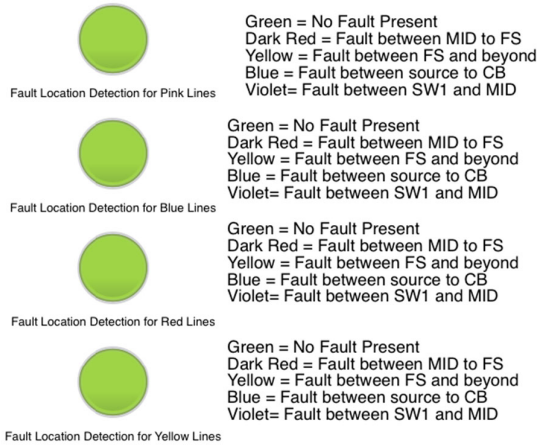

Fig. 16 Fault detection and location indicator.

Simulink switches are programmed to open when the switch input signal is '0' and close when the switch input signal is '1'. This is how both algorithms open and close switches inside the state commands. A fault is simulated with a manual switch, ground, and constants 1 and 0 to open and close the switch.

When the simulation starts the user can double click the manual switch to open or close the switch. The 0 constant sends an input signal to the switch to open and the 1 constant sends an input signal to the switch to close. For the fault creator switch shown in Fig. 13, it acts like a fault in the system when being connected to ground.

The wattage of each feeder is calculated using voltages and current sensors on all loads. The total wattage of the feeders is calculated by adding all the watts readings of all loads on the feeder. This calculation is performed for all feeders and the output values are tagged so they can be used as an input of the load balancing model. An example of the calculation using Simulink is shown in Figures 14 and 15 for the Red feeder.

The *"loc_det"* variable is connected to a Simulink button to keep track where the fault is located. The *'loc_det'* variable is initially 0 and changes to a number depending on the final state of the fault detection algorithm. The settings of the color code are changed for the visual representation of the location detection: 0 is Green, 1 is Red, 2 is Yellow, 3 is Blue, and 4 is Violet. Figure 16 demonstrates the implementation of the button for the fault locator.

5.1 Fault detection and isolation algorithm testing

The fault isolation logic is evaluated on Red feeder where a fault is induced before the midpoint with the circuit breaker limit of 3000A. The fault is induced by connecting the manual fault creator switch between the MID and SW1. Initially, the switch is open and, after a duration into the simulation, is closed to simulate the fault. Figure 17 demonstrates Red feeder circuit before the fault is implemented and Fig. 18 demonstrates part of the fault detection and isolation Stateflow chart of Red feeder before the fault is implemented. In addition, Table 5 presents the parameters of Red feeder before the fault creation.

The blue shading around the start state of Fig. 18 loops through the continuous loop path until a fault is implemented. After the fault is implemented, the loop goes through the phase II path and ends at the phase IIb state. Figure 19 demonstrates Red feeder

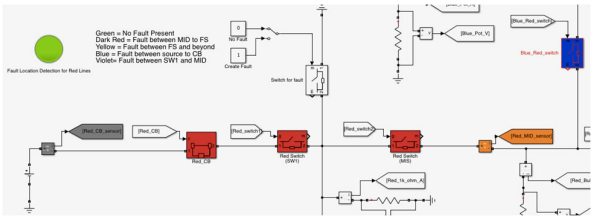

Fig. 17 Circuit view of Red feeder without fault implementation.

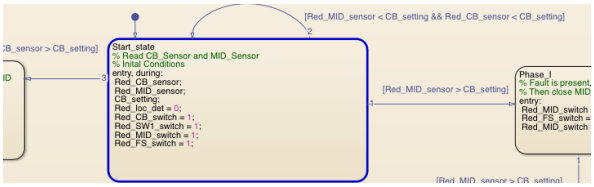

Fig. 18 Stateflow chart of fault detection and isolation algorithm of Red feeder without fault implementation.

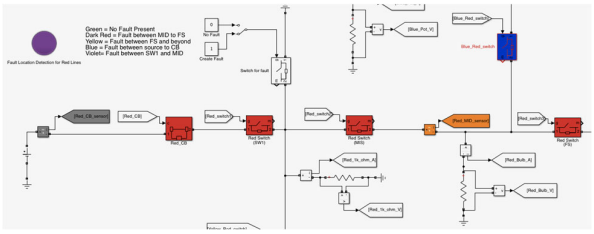

Fig. 19 Fault isolation simulation on Red feeder under fault condition

Table 5 Parameters of Red feeder for fault detection under no-fault condition

Parameter	Value
Circuit Breaker Sensor	449.87 A
Midpoint Sensor	437.35 A
Red CB Switch	1
Red SW1 Switch	1
Red MID Switch	1
Red FS Switch	1

schematic after the fault is implemented. Figure 20 demonstrates part of the fault detection and isolation Stateflow chart of Red feeder after the fault is implemented. In addition, Table 6 demonstrates the parameters of Red feeder after the fault is implemented.

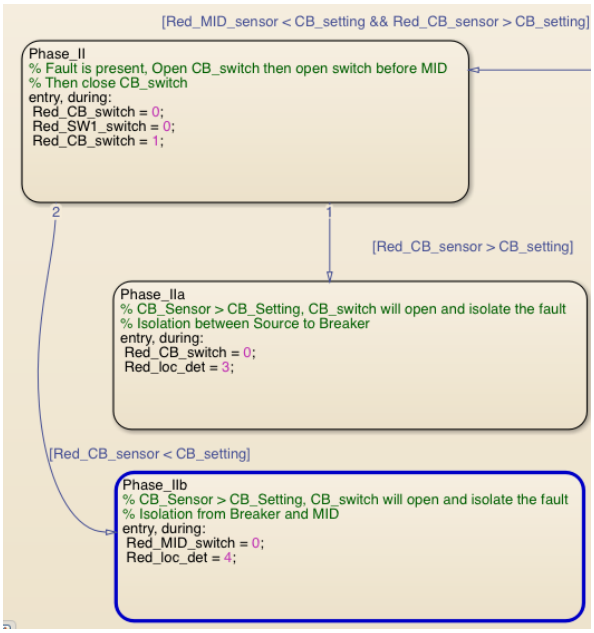

Fig. 20 Stateflow chart of fault detection and isolation algorithm of Red feeder with fault implementation.

Table 6 Parameters of Red feeder for fault detection under fault condition

Parameter	Value
Circuit Breaker Sensor	0.124 A
Midpoint Sensor	-0.124 A
Red CB Switch	1
Red SW1 Switch	0
Red MID Switch	0
Red FS Switch	1

5.2 New load balancing algorithm testing

This section describes the first evaluation of the new load balancing algorithm (presented in Section 3.2) via testing its effectiveness under a faulty condition of Red feeder.

Simulation setting and process: When the simulation takes place without the fault, the load balancing algorithm stays on the '*start_state*' block until the wattage of any of the feeders are less than the lower limit. Figure 21 demonstrates part of the load balancing algorithm Stateflow chart and Table 7 presents the total wattage of each feeder before the fault is induced. After the fault is induced, there is a moment where the logic must check for the final state of the fault detection and isolation algorithm. Figure 22 demonstrates part of the load balancing algorithm

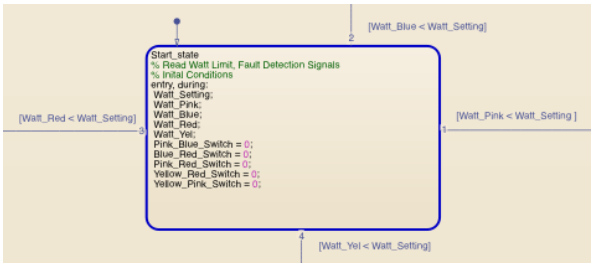

Fig. 21 Load balancing algorithm without fault implementation

Table 7 Total wattage of feeders without fault induction

Parameter	Value
Wattage of Blue Feeder	17.02 MW
Wattage of Pink Feeder	22.56 MW
Wattage of Red Feeder	5.57 MW
Wattage of Yellow Feeder	27.81 MW

Table 8 Total wattage of feeders measured immediately after the fault is implemented

Parameter	Value
Wattage of Blue Feeder	17.02 MW
Wattage of Pink Feeder	22.56 MW
Wattage of Red Feeder	38.74 kW
Wattage of Yellow Feeder	27.81 MW

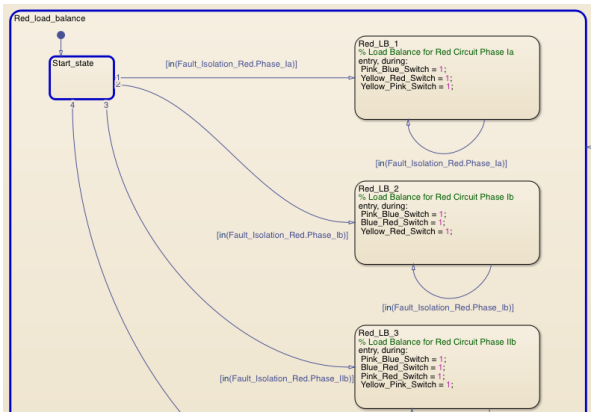

Fig. 22 Load balancing Stateflow for Red feeder immediately after the fault is implemented.

Stateflow chart for Red feeder directly after the fault is implemented and Table 8 presents the total wattage of each feeder directly after the fault is implemented. After some time, the fault is induced, the load balancing executes the commands and the circuit is in a steady state. Figure 23 demonstrates part of the load balancing algorithm Stateflow chart and Table 9 presents total wattage of each feeder after the fault is implemented.

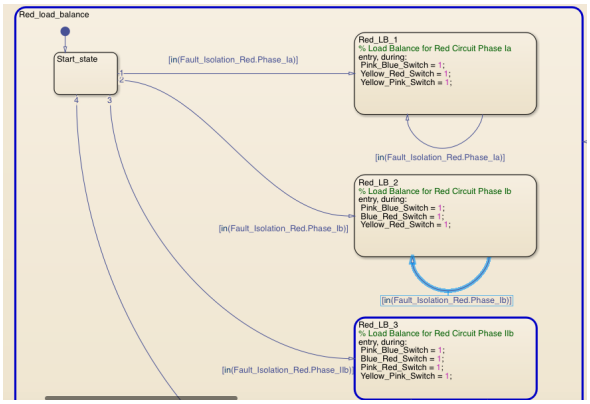

Fig. 23 Load balancing Stateflow for Red feeder after the fault is implemented.

Table 9 Total wattage of feeders measured after the fault is implemented

Parameter	Value
Wattage of Blue Feeder	17.00 MW
Wattage of Pink Feeder	22.56 MW
Wattage of Red Feeder	5.50 MW
Wattage of Yellow Feeder	27.81 MW

Explanation of results:

The simulation results in Figures 21, 22, and 23 show that the load balancing algorithm works properly where the control scheme correctly moves from one state to another in consistence with the fault location and isolation algorithm. Note that a bold blue line marks an active logic subset or an active state. Since a fault is induced on the Red feeder, after being detected and isolated by the fault location and isolation algorithm, the load balancing logic for the Red feeder is activated. The logic was initialized (bold blue line around the logic subset *"Red_load_balance"* and its *"Start_state"* in Fig. 22) and then *"Red_LB_3"* state was activated (bold blue line around this state in Fig. 23). This allows distributing load of the faulty Red feeder to neighboring feeders by closing certain inter-feeder switches (e.g. *"Pink_Blue_Switch=1"* means that this switch, shown in Fig. 11, is closed).

The load of the Red feeder before the fault was 5.57MW (Table 7), which dropped to 38.74kW during the fault (Table 8). Eventually, 5.5-MW load of the faulty Red feeder (out of the original 5.57MW) was distributed to neighboring feeders (Table 9). A load of 0.07MW of the faulty Red feeder was isolated due the fault and not supplied. The power consumed by the healthy feeders (Blue, Pink, Yellow) was essentially unchanged, suggesting that they could carry additional load from the faulty Red feeder while continuing to serve their own loads.

6. Simulation results of improved FLISR scheme with load balancing

The testing results in Section 5 show successful implementation of the improved FLISR scheme. They are summarized as follows:

a) The fault isolation algorithm works properly, as demonstrated by fault indicator in Fig. 16 and Stateflow chart in Fig. 18. The blue shading around a state represent the current state of the algorithm. When the fault is induced, the algorithm goes through the necessary steps in order to isolate the fault. This is highlighted how the blue shading concludes to the final state *'phase IIb'* and executes the necessary functions to isolate the fault.

b) The construction of the fault isolation algorithm works as expected. Comparing the switch values of Tables 5 and 6, the expected switches open in order to isolate the fault. The currents of the circuit breaker and midpoint also indicate this by how the values are near zero due to the isolation of the fault.

c) The load balancing algorithm works properly, as shown by Figures 21, 22 and 23. The blue shading around a state represent the current state of the algorithm. When the fault is induced, the algorithm goes through the correct steps to provide power around the faulty feeder. This is highlighted how the blue shading concludes to a final state that active interconnected switches to allow power flow around the isolated fault.

d) The construction of the load balancing algorithm functions properly. A comparison of Tables 7, 8, and 9 demonstrate how the load balancing algorithm successfully provides power around the isolated section of the fault. The load balancing logic detects the fault and operate the appropriate interconnected switches to allow power flow to Red feeder. This is shown by the change of the wattage of the Red feeder before, during, and after the fault is induced. Initially, the Red feeder power consumption was 5.57MW (Table 7). During the fault, it decreased to 38.74kW (Table 8). After the fault was isolated and the unfaulty sections of the Red feeder were supplied by neighboring feeders, it power consumption recovered to 5.5MW (Table 9). It follows that 0.07MW of original load is located in the faulty section and isolated.

e) In addition, the initial wattage shown in Table 7 is validated by the theorical calculations of Tables 1, 2, 3, and 4. This confirms that the simulated distribution system (Fig. 1) is constructed properly.

For further evaluation of the improved FLISR scheme, we simulate more fault scenarios which involve a fault on other feeders at different locations.

Scenario 1: A fault is induced between the source and SW1 on Pink feeder as shown in Fig. 24. This is phase IIa of the fault detection and isolation algorithm.

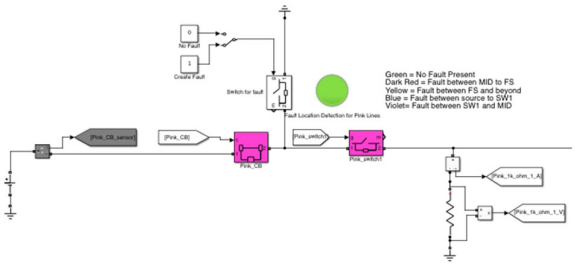

Fig. 24 Circuit of Pink feeder without fault induction.

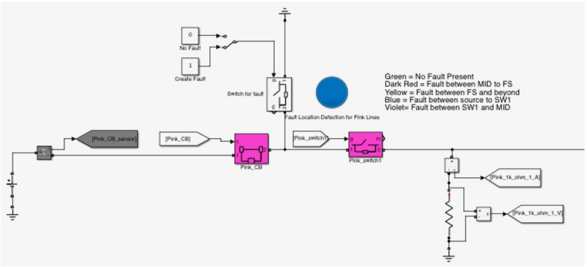

Fig. 25 Fault is induced between source and SW1 on Pink feeder.

Table 10 Wattage of all feeders before and after fault implemented on Pink feeder between the source and SW1

Feeders	Wattage before fault induction of improved FLISR scheme	Wattage after fault induction of improved FLISR scheme
Blue	17.02 MW	17.00 MW
Pink	22.56 MW	22.56 MW
Red	5.57 MW	5.56 MW
Yellow	27.81 MW	27.80 MW
Pink CB Sensor	1823.21 A	1.12 mA
Pink MID Sensor	1374.00 A	-449.22 A

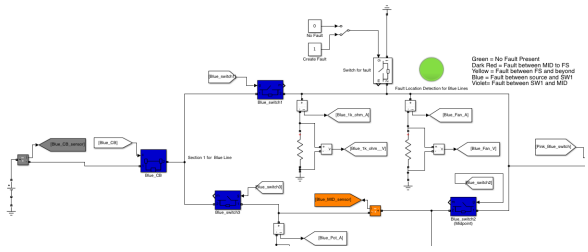

Fig. 26 Circuit of Blue feeder without fault implementation.

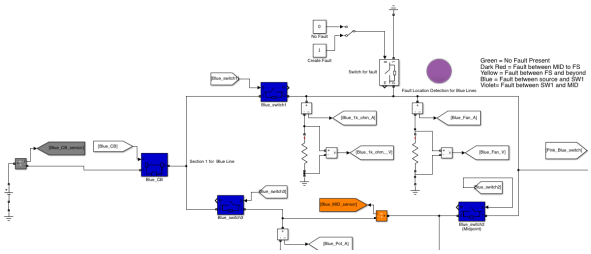

Fig. 27 Fault implemented between SW1 and MID on Blue feeder.

Table 11 Wattage of all feeders before and after fault implemented on Blue feeder between SW1 and MID

Feeders	Wattage before fault induction of improved FLISR scheme	Wattage after fault induction of improved FLISR scheme
Blue	17.02 MW	15.34 MW
Pink	22.56 MW	22.56 MW
Red	5.57 MW	5.56 MW
Yellow	27.81 MW	27.80 MW
Blue CB Sensor	1374.8 A	125.77 A
Blue MID Sensor	367.39 A	-18.92 A

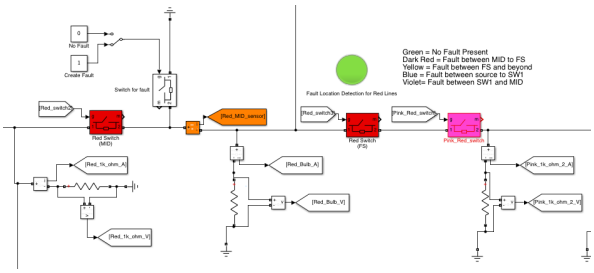

Fig. 28 Circuit of Red feeder without fault implementation.

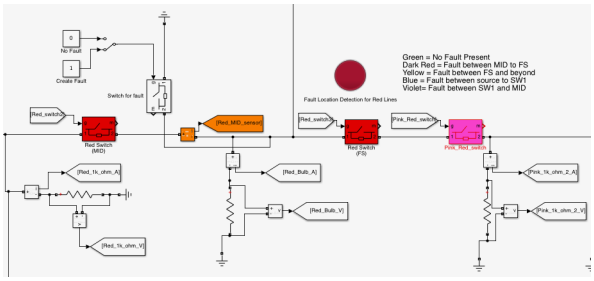

Fig. 29 Fault implemented between MID and FS on Red feeder.

Table 12 Wattage of all feeders before and after fault implemented on Red feeder between MID and FS

Feeders	Wattage before fault induction of improved FLISR scheme	Wattage after fault induction of improved FLISR scheme
Blue	17.02 MW	17.02 MW
Pink	22.56 MW	22.56 MW
Red	5.57 MW	0.15 MW
Yellow	27.81 MW	27.80 MW
Red CB Sensor	449.87 A	12.52 A
Red MID Sensor	437.35 A	0.124 A

Results: The fault detection and isolation algorithm correctly detects the fault and opens appropriate switches to isolate the fault as shown by the color of the fault indicator button in Fig. 25. The location of the fault disconnects the source from the circuit. Therefore, no power is provided to Pink feeder from the source (grid) side. However, the load balancing algorithm successfully provides power to Pink feeder by closing the interconnected switches of neighboring feeders. This is shown in Table 10 where the power of Pink feeder decreases but recovers thanks to the power supply from neighboring feeders. The current in the Pink MID sensor is negative because the current flows in the opposite direction, i.e. the Pink feeder is not fed from the grid source (positive current direction) but from the neighboring feeders (negative direction).

Scenario 2: A fault is simulated between SW1 and MID on Blue feeder as in Fig. 27. This is phase IIb of the fault detection and isolation algorithm.

Results: The fault detection and isolation algorithm correctly detects the fault and opens the appropriate switches to isolate the fault, as shown by the color of the fault indicator button in Fig. 27. The location of

the fault disconnects two loads. The load balancing algorithm successfully provides power back to Blue feeder by operating the interconnected switches of neighboring feeders, as shown in Table 11. The Blue feeder original power consumption was 17.02MW and, after the fault was isolated, recovered to 15.34MW. The decrease in the power consumption confirms that the two loads in the faulty section were isolated.

Scenario 3: A fault is simulated between MID and FS on Red feeder as in Fig. 29. This is phase Ia of the fault detection and isolation algorithm.

Results: The fault detection and isolation algorithm correctly detects the fault and opens appropriate switches to isolate the fault, as shown by the color of the fault indicator button in Fig. 29. The location of the fault disconnects one load and decreases the power drastically. However, the load balancing algorithm successfully provides power back to the Red feeder unfaulty sections by closing the interconnected switches of neighboring feeders, as shown by Table 12.

7. Issues to be considered for real-world implementation of improved FLISR scheme

The duration for the simulation of the FLISR scheme is 10 seconds and the fault is manually induced by double clicking a manual switch. In real-world situations timing depends on the specification and capability of communication and protection equipment installed along the feeder.

The realistic timing of the FLISR process would be shorter than a user double clicking on a screen. For example, it typically takes a circuit breaker around 5 - 7 cycles (83 - 117ms in a 60-Hz power system) to detect a fault current and trip to clear the fault. The same technology of the circuit breaker can be enhanced or replicated for the other switches involved in the FLISR scheme. The fault detection algorithm has at most four switches opening or closing for any given fault. This results in about 468 milliseconds to detect and isolate faults.

For the load balancing algorithm, the interconnected switches are independent from each other. Therefore, the time for switching should be the same as that for the circuit breakers, i.e., 5 – 7 cycles. In addition, certain delay time would be needed for each of the switches due to communication technologies. Nevertheless, this scheme is effective in speeding up the process to isolate faults on the power lines. Whereas it may take several hours to repair a faulty line, this improved FLISR scheme eliminates inspection time to search for the faulty line, as well as reduces power outage duration and the number of customers being affected by a fault, resulting in higher customer satisfaction.

As mentioned earlier, the FLISR technology is dependent on the communication and protection system of the feeder. For any self-restoring system, multiple switches, relays, reclosers, sectionalizers, faulted circuit indicator, and fuses are needed to operate the switching and protection schemes. Communication options may include fiber optic, cellular, or radio. Databases are also used to collect and analyse the data of all the sensors and implement them into specialized algorithm for the scheme.

The investment costs for implementation of FLISR schemes vary for several reasons, such as feeder sizes, number of neighbouring feeders, and number of load connections. In 2014, a report by the U.S. Department of Energy provides information of seven utility FLISR projects where the budget ranges from around $20 million to over $600 million per project, depending on project size [8].

8. Conclusion

This study creates a new fault location, isolation, and service restoration (FLISR) scheme that has an automatic load balancing algorithm developed by the study authors. Its performance is evaluated by simulation using realistic data. The simulation results have led to the following conclusion:

1) Testing with fault induction in multiple scenarios demonstrates that the improved FLISR scheme operates properly.
2) The measured current before and after a fault shows that the fault detection and isolation algorithm functions as expected.
3) Testing the change of power before and after a fault shows that the improved load balancing algorithm works properly.
4) The improved FLISR scheme is able to operate effectively on different feeders with different loading levels.

The major difference between the FLISR scheme developed by this study and some other FLISR schemes is that our FLISR scheme has an effective load balancing algorithm. It enables automatic re-routing of power to supply load of a faulty feeder while protecting healthy feeders from overloading during the service restoration process. As discussed in the Introduction section, lack of an adequate load balancing scheme has been a major deficiency of the study in [7]. Further, load balancing and overloading protection of healthy feeders have not been adequately addressed in some other studies [1] [5] [19]. It follows that our FLISR scheme helps fill in the existing gap and is superior with its capability to provide automatic power routing and feeder overloading protection in consistence with fault location and isolation.

In terms of broader impacts, the study outcomes are helpful for researchers and system planners. Apart from providing further understanding of the FLISR scheme and the new load balancing algorithm, the study provides a simulation tool that can be used to simulate and analyze various FLISR schemes. For example, researchers and system planners can use the Simulink FLISR system to simulate new scenarios or modify it to create new FLISR versions. In addition, the simulation system can be used as an education tool to visualize FLISR technologies to technicians and the public. These factors help expedite FLISR adoption as part of power distribution system automation. Overall, the study provides some insightful understanding of the FLISR scheme that, in turn, promotes its implementation in Smart Grid to reduce service interruption and ensure better power supply to customers.

Acknowledgement

The student author wants to acknowledge and thank Professor Ha Thu Le, the project advisor, for providing guidance and knowledge throughout his master and bachelor studies. Furthermore, the authors would like to thank the Master project committee members, Drs. Zekeriya Aliyazicioglu and Dennis Fitzgerald, for their time and feedback.

Appendix

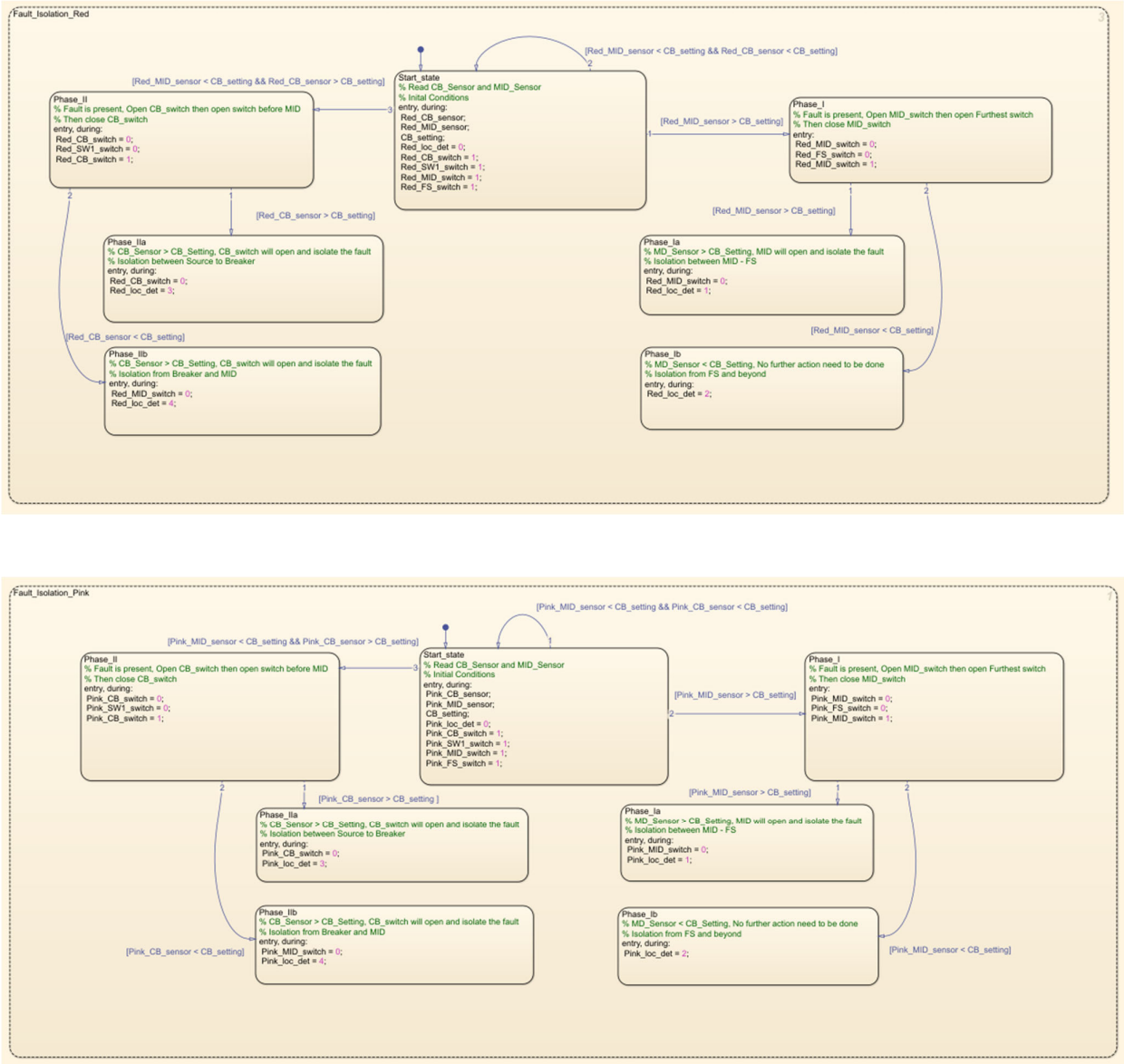

Fig. 30 Complete Stateflow diagrams of fault location and isolation logics for Red feeder (Top) and Pink feeder (Bottom)

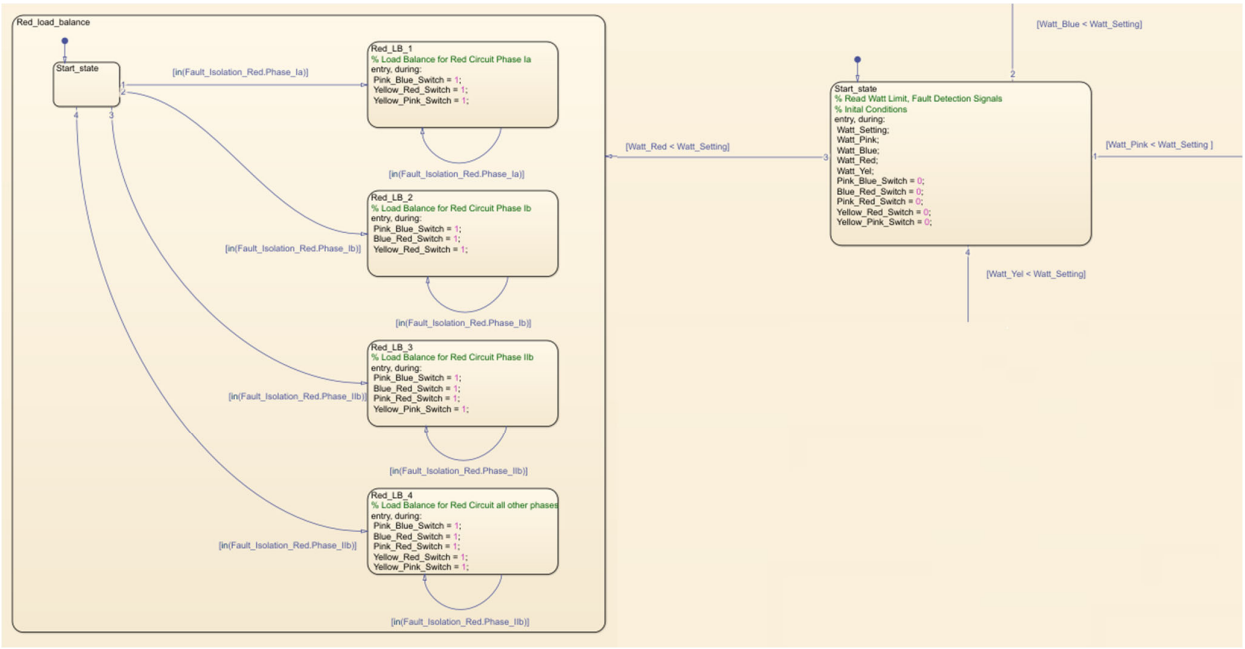

Fig. 31 Complete Stateflow diagram of load balancing logic for Red feeder. Stateflow load balancing logics for other feeders follow a similar principle.

References

[1] W. Al-Hasawi and M. Gilany, "Proposed Techniques for Identifying Open and Short Circuit Sections in Distribution Networks," *WSEAS Transactions on Power Systems,* vol. 4, no. 12, 2009.

[2] S. M. Amin and B. F. Wollenberg, "Toward a smart grid: power delivery for the 21st century," *IEEE Power and Energy Magazine,* vol. 3, no. 5, pp. 33-41, 2005.

[3] S. H. Asman, N. F. A. Aziz, M. Z. A. A. Kadir, U. A. U. Amirulddin and M. Izadi, "Determination of Different Fault Features in Power Distribution System Based on Wavelet Transform," *International Journal of Recent Tech-nology and Engineering (IJRTE),* vol. 8, no. 4, 2019.

[4] J. R. Aguero, "Applying self-healing schemes to modern power distribution systems," *IEEE Power and Energy Society General Meeting,* vol. 10, pp. 1-4, 2012.

[5] K. Cheena, " Self-Healing Besides Restoration Methods in Smart Grid Distribution Networks," *International Journal of Innovative Technology and Exploring Engineering (IJITEE),* vol. 9, no. 6, pp. 1123-1128, 2020.

[6] D. P. Le, D. M. Bui, C. C. Ngo and A. M. T. Le, "FLISR Approach for Smart Distribution Networks Using E-Terra Software—A Case Study," *Energies,* vol. 11, no. 12, 2018.

[7] Z. Cheben, J. Ilagan, J. Miller, L. Nguyen, K. Steidel, J. Ventura, S. Virnig and Ha Thu Le, "Simulation Board for Smart Service Restoration Schemes in Distribution Grid," *International Jornal of Power Systems,* vol. 5, 2020.

[8] Department of Energy, "Fault Location, Isolation, and Service Restoration Technologies Reduce Outage Impact and Duration," Report, 2014. https://www.smartgrid.gov/files/documents/B5_draft_report-12-18-2014.pdf

[9] C. Barreto, J. Giraldo, A. Cardenas, E. Mojica-Nava and N. Quijano, "Control Systems for the Power Grid and Their Resiliency to Attacks," IEEE Security and Privacy, vol. 12, no. 6, pp. 15-23, 2014.

[10] M. A. Elgenedy, A. M. Massoud and S. Ahmed, "Smart grid self-healing: Functions, applications, and developments," *First Workshop on Smart Grid and Renewable Energy (SGRE),* 2015.

[11] M. Eriksson, M. Armendariz, O. O. Vasilenko, A. Saleem and L. Nordström, "Multiagent-Based Distribution Automation Solution for Self-Healing Grids," *IEEE Transactions on Industrial Electronics,* vol. 62, no. 4, pp. 2620-2628, 2015.

[12] G. W. Arnold, "Challenges and Opportunities in Smart Grid: A Position Article," *Proceedings of the IEEE,* vol. 99, no. 6, pp. 922-927, 2011.

[13] M. J. Ghorbani, M. A. Choudhry and A. Feliachi, "A Multiagent Design for Power Distribution Systems Automation," *IEEE Trans. Smart Grid,* vol. 7, no. 1, pp. 329-339, 2016.

[14] D. Heirman, "What makes Smart Grid Smart," *IEEE Electromagnetic Compatibility Magazine,* vol. 1, no. 2, pp. 95-99, 2012.

[15] C. Jin-Lun, H. Chun, Z. Zhao-Xin, Q. Shuo, L. Jiao and Q. Qian, "Smart grid oriented smart substation characteristics analysis," *IEEE PES Innovative Smart Grid Technologies,* 2012.

[16] Y. J. Kim, J. Wang and X. Lu, "A Framework for Load Service Restoration Using Dynamic Change in Boundaries of Advanced Microgrids With Synchronous-Machine DGs," *IEEE Trans. Smart Grid,* vol. 9, no. 4, pp. 3676-3690, 2018.

[17] S. Mirsaeidi, D. M. Said, M. W. Mustafa, M. H. Habibuddin and K. Ghaffari, "Progress and problems in micro-grid protection schemes," *Renewable Sustainable Energy Rev,* vol. 37, no. 2, pp. 909-918, 2014.

[18] N. S. Miswan, M. I. Ridwan, A. Hayatudin and I. A. Musa, "Interoperability testing for Digital Substation in Smart Grid domain: A power utility perspective," *International Symposium on Technology Management and Emerging Technologies (ISTMET),* 2015.

[19] T. Nasser, "Electric Power System Fault Analysis," *WSEAS Transactions on Circuits and Systems,* vol. 19, pp. 19-27, 2020.

[20] P. Sharma, N. Batish, S. Khan and S. Arya, "Power Flow Analysis for IEEE 30 Bus Distribution System," *WSEAS Transactions on Power Systems*, vol. 13, no. 1, pp. 48-59, 2018.

[21] L. Xia, W. Qun, X. Hui and Z. Simeng, "Path Searching Based Fault Auto-mated Recovery Scheme for Distribution Grid with DG," *International Journal of Emerging Electric Power Systems*, vol. 17, p. 663, 2016.

[22] R. J. Yinger, "Self-healing circuits at Southern California Edison," *PES T&D*, 2012.

[23] X. Zhou, Y. M. H. Wang and Z. Gao, "Research on Power System Restoration Based on Multi-agent Systems," *Chinese Control and Decision Conference (CCDC),* 2018.

[24] R. Dantas, J. Liang, C. E. Ugalde-Loo, A. Adamczyk, C. Barker and R. Whitehouse, "Progressive Fault Isolation and Grid Restoration Strategy for MTDC Networks," *IEEE Trans. Power Delivery,* vol. 33, no. 2, pp. 909-918, 2018.

[25] X. Wang, J. Qi, Y. Hou, Y. Wang, W. Xu, D. Wang and Z. Jiao, "Studies on fault analysis and protection configuration schemes in an isolated micro-grid," *Conference & Exposition, National Harbor*, 2014.

[26] P. Jamborsalamati, A. Sadu, F. Ponci and A. Monti, "Design, implementation and real-time testing of an IEC 61850 based FLISR algorithm for smart distribution grids," *IEEE International Workshop on Applied Measurements for Power Systems (AMPS)*, 2015.

[27] Z. Zhang and G. Yang, "Interval Observer-Based Fault Isolation for Discrete-Time Fuzzy Interconnected Systems With Unknown Interconnections," *IEEE Transactions on Cybernetics*, vol. 47, no. 9, pp. 2413-2424, 2017.

[28] W. Liu, T. Kang, W. Cheng and F. Zhao, "The modeling of self-healing control system for distribution network based on UML," *Changsha*, 2015.

[29] B. Stott, J. Jardim and O. Alsac, "DC Power Flow Revisite,," *IEEE Transactions on Power Systems,* vol. 24, no. 3, pp. 1290-1300, 2009.

[30] J. Kim, F. Filali and Y. Ko, "Trends and Potentials of the Smart Grid Infra-structure: From ICT Sub-System to SDN-Enabled Smart Grid Architecture," *Applied Sciences-Basel*, vol. 5, no. 4, pp. 706-727, 2015.

[31] P. Sharma and N. Batish, "Power Flow Analysis for IEEE 30 Bus Distribution System," *WSEAS Transactions on Power Systems*, vol. 13, pp. 48-59, 2018.

[32] V. C. T., A. M. Parimi and C. Karri, "Interline Power Flow Controller with Control strategy to limit Fault Current in Electrical Distribution System," *WSEAS Transactions on Power Systems*, vol. 15, 2020.