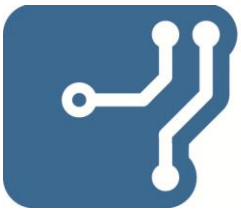


INESCPORTO®
INSTITUTO DE ENGENHARIA DE SISTEMAS
E COMPUTADORES DO PORTO
LABORATÓRIO ASSOCIADO

EFACEC DMS PROJECT

ESP – FLF – 092

Fault Location Finder Specification

Version 2.1

André Moreira

Ivan Franchin

Jorge Pereira

Manuel Matos

28/09/2012

Authors: **André Moreira, Ivan Franchin, Jorge Pereira, Manuel Matos**

File Reference: ESP_FLF_092v1_1.doc

REVISION HISTORY:

Internal reference: ESP-DLD-092 **Revision:** A03

Revision A00 (11-12-1998)

1. First version.

Revision A01 (31-07-2000)

1. Configuration file removed.
2. FAULTDATA structure changed (dFaultCurrent field was added).

Revision A02 (10-02-2012)

1. APIs algorithms changed: CandidatesDetection; TraceForDistance; TraceAndSimplify.
2. FAULTRESULT structure changed (new fields were added)

Revision A03 (28-09-2012)

1. APIs algorithms updated: CandidatesDetection, TraceForDistance and TraceAndSimplify.
2. FAULTRESULT structure changed.
3. FUZZYDATA structure was removed because it was not being used, not even existed in the source code
4. FAULTDATA was change to include the new attribute cStartNode.
5. Added a new API, GetFarthestTerminalFromBus.

Table of Contents

1. INTRODUCTION 38
1.1 ORGANIZATION 41
1.2 LEXICAL AND GLOSSARY 46
1.2.1 Lexical 47
1.2.2 Glossary 53
2. FUNCTIONAL MODEL 59
2.1 INTRODUCTION 61
2.2 SYSTEM DEFINITION 83
2.2.1 System inputs 93
2.2.2 System outputs 104
2.2.3 Error Codes 111
2.2.4 APIs 122
2.3 SYSTEM MODEL 155
2.3.1 Candidate locations detection process 162
2.3.2 Fuzzy inference process 292
3. STRUCTURES 332
3.1 GENERIC ASSUMPTIONS 333
3.2 FAULTDATA STRUCTURE 338
3.3 FAULTRESULT STRUCTURE 366
BIBLIOGRAPHY 390

List of Tables

TABLE 1 - LEXICAL 52
TABLE 2 - DENOMINATIONS 56
TABLE 3 - EXPRESSIONS 58
TABLE 4 - RETURN ERROR CODES 117
TABLE 5 – RUNFAULTLOCATION PARAMETERS 130
TABLE 6 - RUNFAULTLOCATION RETURN VALUES 135
TABLE 7 – REMOVEFAULTRESULT PARAMETERS 141
TABLE 8 – GETFARTHESTTERMINALFROMBUS PARAMETERS 149
TABLE 9 - GETFARTHESTTERMINALFROMBUS RETURN VALUES 154
TABLE 10 - CANDIDATESDETECTION PARAMETERS 180
TABLE 11 - CANDIDATESDETECTION RETURN VALUES 186
TABLE 12 - TRACEFORDISTANCE PARAMETERS 231
TABLE 13 – TRACEFORDISTANCE RETURN VALUES 237
TABLE 14 – TRACEANDSIMPLIFY SIMPLIFY PARAMETERS 272
TABLE 15 – TRACEANDSIMPLIFY RETURN VALUES 277
TABLE 16 - FUZZYINFERENCE PARAMETERS 306

TABLE 17 - FUZZYINFERENCE RETURN VALUES 311
TABLE 18 – FAULTDATA STRUCTURE. 353
TABLE 19 - MODULE PARAMETERS 364
TABLE 20 – FAULTRESULT STRUCTURE. 385

List of Figures

FIGURE 1 - PROCESS DIAGRAM FOR THE FLF APPLICATION 161
FIGURE 2 - FUNCTIONAL MODEL FOR CANDIDATESDETECTION FUNCTION 175
FIGURE 3 - FUNCTIONAL MODEL FOR TRACEFORDISTANCE FUNCTION 216
FIGURE 4 - FUNCTIONAL MODEL FOR TRACEANDSIMPLIFY FUNCTION 265
FIGURE 5 - FUNCTIONAL MODEL FOR FUZZYINFERENCE FUNCTION 302

1. Introduction

This document describes the specification of the Fault Location Finder (FLF) module. The document includes the interface structure and functions related with it.

1.1 Organization

The main information in this document is presented in the following two chapters:

- Functional Model -Describes the data transformation and presents the input and output data for this module, and the functionality of each Application Program Interface (API);
- Structures - Describes the structures used to maintain the data associated with a network study;

1.2 Lexical and Glossary

1.2.1 Lexical

Example	Format	Meaning
Instituto	italic	terms in a foreign language
Bold	bold	name of chapters or sub-chapters also highlights relevant parts or words in the text
Lexical Glossary	underlined	highlights relevant parts or words in the text, or concepts
Important	bold and underlined	very important concepts or remark
(URD)	parenthesis	default values, acronyms
[...]	angle brackets	optional values, bibliographic references
ESA	Upper case	software tools, product names, companies, acronyms

Table 1 - Lexical

1.2.2 Glossary

API	Application Program Interface
DMS	Distribution Management System
ET/DE	EFACEC – Industrial Electronic Division Development Department
GUI	Graphical User Interface
INESC	*Instituto de Engenharia de Sistemas e Computadores*
FLF	Fault Location Finder

Table 2 - Denominations

Applications	Software that implements specific functionality(e.g., Timetag, Watchdog).
Library	A set of reusable software.

Table 3 - Expressions

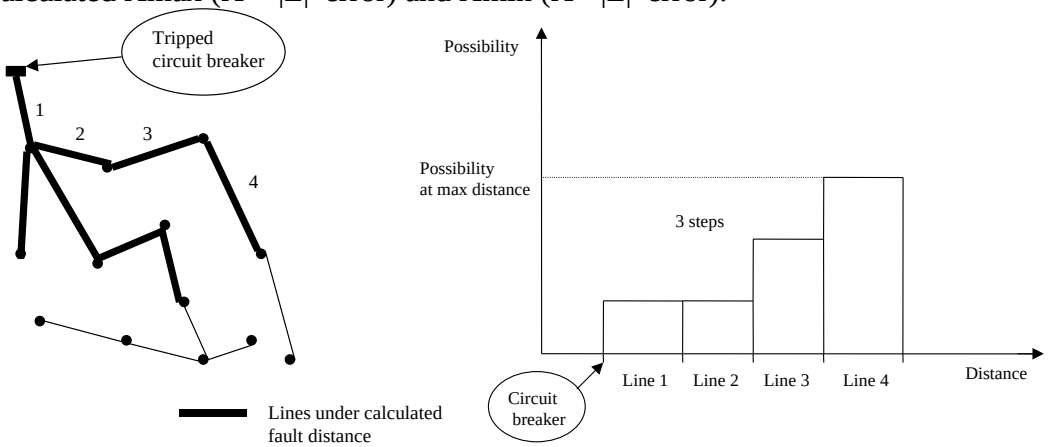
2. Functional Model

2.1 Introduction

The Fault Location Finder module implements a solution, oriented **towards distribution medium voltage networks**, that provides a list of possible fault locations, together with their degree of possibility. It is intended to be used **in radial or meshed sub-networks of the whole network**.

The module is meant to incorporate three types of information:

Electrical information, the short circuit current value, distance value or impedance value provided by the circuit breaker that tripped due to the fault. This information, if available, is used to make an estimate of the minimum and maximum fault distance, ~~considering the fault to be a three phase symmetrical, zero fault resistance short circuit (so giving the longest distance)~~. If this information is not available, the module considers all the line segments of the networks to be equally likely locations for the fault. If the information is available, and so the maximum distance calculated, the possibility the fault has happened in a line segment between the breaker and that distance is considered to vary in a stepwise ascending way starting at the breaker. The minimum distance is given taking in consideration an error given by the user in equivalent impedance considering the value provided by the circuit breaker that tripped due to the fault. The error must be interpreted as a variation on the value of the impedance Z. Given R and X it should be calculated the value of |Z|. With this value it is calculated Xmax (X + |Z|*error) and Xmin (X - |Z|*error).



- Logical information, in what concerns the state (opened/closed or active/not active) of the switching gear or detectors in the sub-network below the circuit breaker. Special care is to be given to devices such as fault detectors and reclosers, since the information they provide performs a great role in narrowing the list of possible candidates to be the fault location. For example the information that a certain recloser changed the state (tripped close) after the fault, clearly indicates

the fact that the line segments above it and in parallel branches departing on the node the recloser is connected to, should not be considered as likely candidates to be the fault location. In case a telephone service that customers can call to in case they experience a power failure exists, the information that a given bus is out of power supply can be used to make an estimate of the current state of reclosers who are listed as with current state "unknown", because the information on their state is not available at real time.

All this information is used by the module to produce an educated guess of the line segment the fault is more possibly located at, or a list of line segments that are possible locations of the fault.

A fuzzy inference method is addressed in section 2.3.2. where some adjustments considering weather information and lines proneness to a fault is considered.

- Empirical information, that models the knowledge and the information owned by the user. The user is supposed to have previously classified each of the line segments of the sub-network according to how sensible to the weather they are. In that way, with the information on the current weather condition, provided by the user when the module is executed, each line segment will have a value of possibility to be the fault location according to the previously entered data. The user is also supposed to classify the zone each of the line segments belongs to according to its hazard proneness, meaning the line segment is lightly or heavily exposed to such phenomena as nearby repairings, crossing a forest area, being in a site of former incidents etc., or even not prone at all. In this way the experience of the operator may have a direct influence on the performance of the module calculations.

2.2 System definition

The structures to be used with the FLF module, for maintaining input data, output data and internal results will be shared by the whole set of function. The set of information used by the FLF module is:

- internal DATASET structure, produced by the Application Program Interfaces;

The events are generated by the **user** through the graphical user interface (GUI), which does not make part of this module.

All the parameters are provided by the caller by it own initiative and not requested by the FLF module. The caller is also responsible for the proper activation of the module.

The **user** is also responsible to indicate the circuit breaker tripped due to the fault. If a case happens in which more than one circuit breaker trips, the **user** is responsible in giving as an input to the module **only the reference to the most hierarchically important breaker.**

The circuit breaker at a substation is, for example, hierarchically more important than the circuit breaker on a capacitor bank located in the sub-network below it.

If the sub-network in study is equipped with reclosers, care must be taken to the fact that, even though a fault exists**, the final state of the breaker may be "closed"**, due to the action of the recloser. The information on the circuit breaker to be used as input for the module should then be gathered in the following way: either the information on the fact that the circuit breaker experienced a reclosure cycle is available, and therefore its identification, or the information on the fact that one of the sub-network reclosers has changed state must be used by the **user** to determine the circuit breaker in question.

2.2.1 System inputs

The inputs of this module are:

- reference to the tripped circuit breaker
- switching devices status after fault
- network topology
- start node
- type of data entry
- value of data entry (fault current, impedance or distance)
- error to calculate minimum and maximum distance
- penalty percentage for unreliable fault current data
- possibility of fault occurrence at minimum estimated distance calculated from fault current
- possibility of fault occurrence at maximum estimated distance calculated from fault current
- number of possibility steps from the circuit breaker between the minimum and the maximum estimated distance

There are several other inputs that can be used by the module in the fuzzy inference system, such as:

- current weather condition
- weather sensibility classification of line segments
- zone hazard proneness classification of line segments
- minimum possibility for alpha level data display
- possibility values for weather states normal, variable and storm
- possibility values for zone proneness not prone, prone and highly prone
- list of node references where power shortage was reported by phone.

The several inputs are used by the FLF module to calculate the output results.

2.2.2 System outputs

The outputs for this module are:

- overall result:
  - success
  - error code

- the fault location results:
  - list of line segments with fault possibility between a minimum and maximum value of equivalent minimum and maximum impedance
  - list of equipments between circuit breaker until line segments with fault possibility
  - list of equipments in the fault zone

These outputs, which guide the user in the restoration process, are stored in the FAULTRESULT structure, using arrays, which are described in chapter 3.

2.2.3 Error Codes

For a better understanding and management of the entire application, the error codes returned by the FLF API functions are now represented by the following constants:

Errors constants:

Name	Value	Meaning
SUCCESS	0	success, no errors
FLF_E_FAULTDATA	600	invalid fault data handle
FLF_E_DATASET	601	invalid data set handle
FLF_E_FAULTRESULT	602	invalid fault result handle
FLF_E_PARAM	603	invalid parameter

Table 4 - Return error codes

These error codes are returned by the whole set of API functions of this module. Each API function uses only some of these error codes as it will be specified further on. The caller must verify the return value and upon it abort or continue the interactive process.

There are also some return codes from internal INESC API functions that have to be considered (see return codes of internal INESC API functions used by this module).

The error values begin at 600 to avoid confusion with the set of errors returned by the INESC API functions used in this module. This allows retrieving easily the error that occurred.

2.2.4 APIs

RunFaultLocation

This API runs the Fault Location Finder Application existing in the Power Functions Library. As results an integer is returned stating the overall result (0 = success, >0 = errors have occurred).

Parameters:

Type	Unit	Name	Description
DATASET*		pDataSet	pointer to the data set handle
UID		idSwitch	tripped circuit breaker id
FAULTDATA*		FaultData	store information about the Fault Location Finder parameters.
FAULTRESULT**		FaultResult	address of pointer to a structure that will hold the Fault Location Finder overall results.

Table 5 – RunFaultLocation parameters

This function returns zero (SUCCESS) if there were no problems during the process, otherwise returns a value different from zero stating the type of error that was found.

return codes:

Constant	Meaning
SUCCESS	no error, function succeeded (SUCCESS)
ERROR_MEMORY	not enough memory
FLF_E_FAULTDATA	invalid FAULTDATA handle (is null)
FLF_E_FAULTRESULT	invalid FAULTRESULT handle (is null)
FLF_E_DATASET	invalid DATASET handle (is null)
ERROR_UID	invalid element identifier (INESC API)

Table 6 - RunFaultLocation return values

RemoveFaultResult

This API removes the contents of FAULTRESULT structure previously created by RunFaultLocation API.

Parameters:

Type	Name	Description
FAULTRESULT **	FaultResult	address of pointer to a structure that will hold the Fault Location Finder overall results.

Table 7 – RemoveFaultResult parameters

GetFarthestTerminalFromBus

This API finds which terminal of a switching device is farthest from a busbar. The "farthest" criterion is obtained by the number of equipments present between the switching device terminal until the busbar. As results an integer is returned stating the overall result (0 = success, >0 = errors have occurred).

Parameters:

Type	Unit	Name	Description
DATASET*		pDataSet	pointer to the data set handle
UID		idSwitch	switching device id
char*		cTerminal	return parameter where is informed the farthest switching device terminal from a busbar: 0 - (UNDEFINED) it was not possible to determine the farthest terminal; 1 – (FROM) is the farthest terminal; 2 - (TO) is the farthest terminal.

Table 8 – GetFarthestTerminalFromBus parameters

This function returns zero (SUCCESS) if there were no problems during the process, otherwise returns a value different from zero stating the type of error that was found.

return codes:

Constant	Meaning
SUCCESS	no error, function succeeded (SUCCESS)
FLF_E_DATASET	invalid DATASET handle (is null)
ERROR_EQUIPMENT_UID	invalid switching device identifier
ERROR_EQUIP_TYPE	invalid equipment type

Table 9 - GetFarthestTerminalFromBus return values

2.3 System Model

The FLF module has two functional blocks:

- **Possible candidates for fault location detection processes** - for a given tripped circuit breaker, this process builds a list of line segments that are likely candidates to be fault locations.

- **Fuzzy inference process** – for a given list of candidates and its classification according to the weather sensibility and hazard proneness zone they belong to, this process is responsible in building the FAULTRESULT structure data (previously initialised) to be returned by the module.

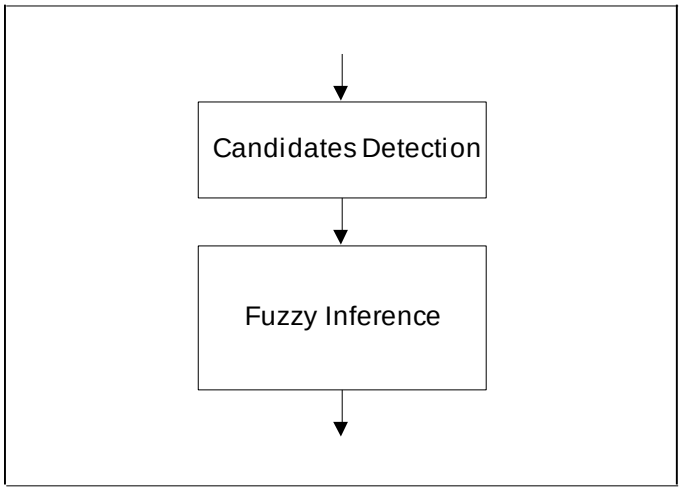


Figure 1 - Process diagram for the FLF application

2.3.1 Candidate locations detection process

The candidate location detection process finds the line segments that are likely candidates to be the location of the fault that caused the tripping of the circuit breaker to isolate the fault in the network.

In case the module uses the fault current the circuit breaker opened (if available) to make an estimate of the interval between a minimum and a maximum fault distance (considering the fault to have happened with zero fault resistance). The fault current data might be considered to be unreliable. If so, it is increased by a given percentage that increases the minimum and maximum fault distance and so possibly the amount of line segment candidates.

In case the module uses the distance itself it will be used the line segment distance to encounter the fault location. It can be considered that can exist an error on the distance, if so the module takes that error in consideration to calculate the minimum and maximum distance of the fault location.

The module can also use the short circuit impedance (if available) to calculate the minimum and maximum fault distance. On the impedance data it can be considered that exists an error. If so, the error will be considered to calculate the minimum and maximum fault distance.

Information on the existence of network devices like fault locators and reclosers is taken into account to narrow down the list of possible candidates.

If the fault current, the distance or the impedance values are not available, the function considers the entire network tree that starts at the circuit breaker to be an equally possible (possibility 0) location for the fault. If it is available, the possibility is considered to vary stepwise in distance from the breaker, from 0, at the breaker, to a given value of possibility at the maximum fault distance.

This function operates on valid FAULTDATA and FAULTRESULT structures.

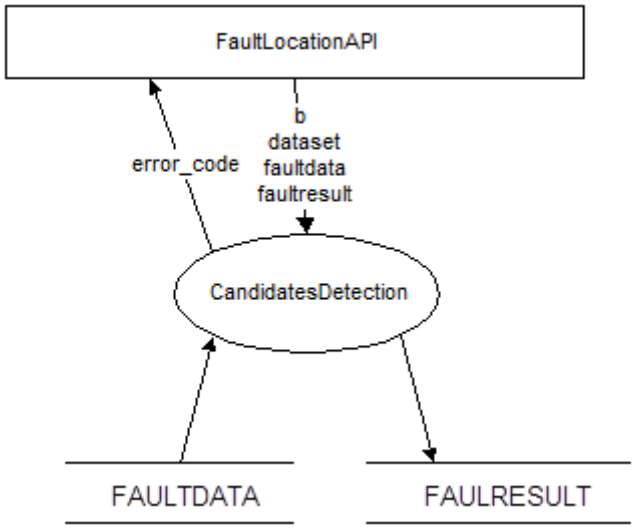


Figure 2 - Functional model for CandidatesDetection function

CandidatesDetection

After a tripped circuit breaker has been specified, this function builds an array of the line segments that are likely candidates to be the fault location.

Parameters:

Type	Unit	Name	Description
UID		b	Unique identifier for the tripped circuit breaker
DATASET *		dataset	Pointer to a data set handle
FAULTDATA *		faultdata	Pointer to a fault data handle
FAULTRESULT *		faultresult	Pointer to a fault result handle

Table 10 - CandidatesDetection parameters

This function returns zero (SUCCESS) if there were no problems during the process, otherwise returns a value different from zero stating the type of error that was found.

return codes:

Constant	Meaning
SUCCESS	no error, function succeeded (SUCCESS)
ERROR_MEMORY	not enough memory
FLF_E_FAULTDATA	invalid FAULTDATA handle (is null)
FLF_E_FAULTRESULT	invalid FAULTRESULT handle (is null)
FLF_E_DATASET	invalid DATASET handle (is null)
ERROR_UID	invalid element identifier (INESC API)

Table 11 - CandidatesDetection return values

Algorithm:

```
if faultdata is null then
        return FLF_E_FAULTDATA
endif

if faultresult is null then
        return FLF_E_FAULTRESULT
endif

if dataset is null then
        return FLF_E_DATASET
endif

//**For maximum distance**//
if ( FaultData->cDataType == 0x01) {
        FaultData->dFaultCurrent = initFaultCurrent;
        FaultData->dFaultCurrent *= (1-FaultData->dFaultDataUP/100.0);
        FaultData->dFaultCurrent *= (1-FaultData->dDistanceError/100.0);

}
else if ( FaultData->cDataType == 0x02)
        FaultData->dFaultDistance =
                initFaultDistance * (1+FaultData->dDistanceError/100.0);
else if ( FaultData->cDataType == 0x03)
        FaultData->FaultImpedance.dReal =
                initFaultImpedance.dImag + sqrt(quad(initFaultImpedance.dReal) +
                quad(initFaultImpedance.dImag)) * (FaultData->dDistanceError/100.0);

// trace tree under circuit breaker for candidates given the estimated fault current
do tmp = TraceForDistance(faultresult)
if tmp is not SUCCESS then
        return tmp
endif
// detect reclosers and fault detectors and simplify the list of candidates
```

```
do tmp = TraceAndSimplify(faultresult)
if tmp is not SUCCESS then
        return tmp
endif

//**For minimum distance**//
if ( FaultData->cDataType == 0x01) {
        FaultData->dFaultCurrent = initFaultCurrent;
        FaultData->dFaultCurrent *= (1+FaultData->dFaultDataUP/100.0);
        FaultData->dFaultCurrent *= (1+FaultData->dDistanceError/100.0);
}
else if ( FaultData->cDataType == 0x02)
        FaultData->dFaultDistance =
                initFaultDistance * (1-FaultData->dDistanceError/100.0);
else if ( FaultData->cDataType == 0x03) {
        FaultData->FaultImpedance.dReal =
                initFaultImpedance.dImag - sqrt(quad(initFaultImpedance.dReal) +
                quad(initFaultImpedance.dImag)) * (FaultData->dDistanceError/100.0);
        FaultData->FaultImpedance.dReal =
                max(0.0,FaultData->FaultImpedance.dReal);
}

// trace tree under circuit breaker for candidates given the estimated fault current
do tmp = TraceForDistance(faultresultaux)
if tmp is not SUCCESS then
        return tmp
endif
// detect reclosers and fault detectors and simplify the list of candidates
do tmp = TraceAndSimplify(faultresultaux)
if tmp is not SUCCESS then
        return tmp
endif

//the results are stored in the faultresult structure
do JoinResults(faultresult, faultresultaux)

//faultresultaux is removed
do tmp = RemoveFaultResult(faultresultaux)

do GetCandidatesEquipsPath();

do GetEquipsInTheFaultZone();

return SUCCESS
```

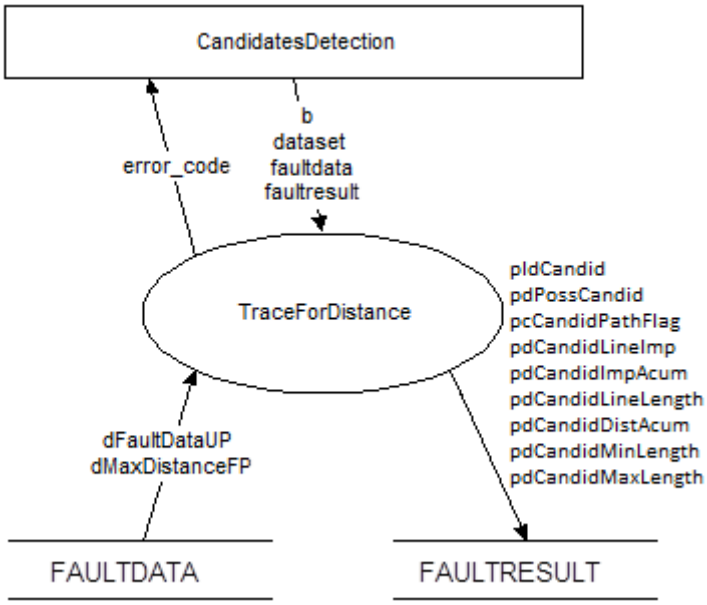


Figure 3 - Functional model for TraceForDistance function

TraceForDistance

After a tripped circuit breaker has been specified, if the fault current opened by the breaker is available, this function calculates an estimate of the fault distance as seen by the breaker and a list of the candidate line segments. If the fault current data is considered to be unreliable, the current is considered to be larger than the registered amount by a percentage set by the user when the FLF module is called. Then, considering the fault was a three phase symmetrical, zero fault resistance short-circuit (worst case), the total worst-case fault impedance as seen by the circuit breaker is calculated and determined the maximum fault distance. It is then calculated an impedance taking into account a user specified distance error in order to determine the minimum fault distance.

If the distance is available, after a tripped circuit breaker has been specified this function calculates the candidates line segments based on the distance itself of each one of the candidates. It can be considered that can exist an error on the distance, if so this function takes that error in consideration to calculate the minimum and maximum distance of the fault location.

Finally, if the impedance is available, after a tripped circuit breaker has been specified this function calculates the candidates line segments based on the impedance itself of each one of the candidates. It is calculated the module of the impedance based on the resistance and reactance parameters. It can be considered that can exist an error on the impedance, if so this function takes that error in consideration to calculate the minimum and maximum distance of the fault location.

After, a trace of the tree (sub-network) is made ~~simulating a three-phase symmetrical zero fault resistance short-circuit~~ in each of the end nodes of the line segments, starting at the root (circuit breaker node). For that, the sub-network below the end node of the line segment in question is removed, and the equivalent impedance at the circuit breaker node is calculated.

If the value is lower than the calculated equivalent fault impedance, the line segment in analysis is removed. If the line segment is between the minimum and maximum fault impedance, it is considered to be a candidate, and its reference stored. The process is repeated in a top down approach until all branches are checked or, in each branch, until a non-candidate line segment is found.

Parameters:

Type	Unit	Name	Description
UID		b	Unique identifier for the tripped circuit breaker
DATASET *		dataset	Pointer to a data set handle
FAULTDATA *		faultdata	Pointer to a fault data handle
FAULTRESULT *		faultresult	Pointer to a fault result handle

Table 12 - TraceForDistance parameters

This function returns zero (SUCCESS) if there were no problems during the process, otherwise returns a value different from zero stating the type of error that was found.

return codes:

Constant	Meaning
SUCCESS	no error, function succeeded (SUCCESS)
FLF_E_FAULTDATA	invalid FAULTDATA handle (is null)
FLF_E_FAULTRESULT	invalid FAULTRESULT handle (is null)
FLF_E_DATASET	invalid DATASET handle (is null)
FLF_E_PARAM	invalid parameter
ERROR_UID	invalid element identifier (INESC API)

Table 13 – TraceForDistance return values

Algorithm:

```
if faultdata is null then
        return FLF_E_FAULTDATA
endif

if faultresult is null then
        return FLF_E_FAULTRESULT
endif

if dataset is null then
        return FLF_E_DATASET
endif

if faultdata.dFaultDataUP is less than zero then
        return FLF_E_PARAM
endif
```

```
if faultdata.dMaxDistanceFP is less than zero then
        return FLF_E_PARAM
endif

if fault current, fault distance or fault impedance is available
        if (FaultData->cDataType == 0x01 ) then //fault current at breaker is available
                if FaultData->dFaultDataUP is different from zero then
                        faultcurrent = faultcurrent * FaultData->dFaultDataUP
                endif
        endif
        repeat
                end_of_trace = false
                if branches available then
                        if there are available nodes at current branch then
                                do remove sub-tree bellow ending node
                                if (FaultData->cDataType == 0x01 ) then
                                        do calculate fault impedance as seen from the breaker
                                endif
                                if (FaultData->cDataType == 0x01 ) then
                                        do add line segment to candidates list
                                        if calculated impedance is bigger than the fault
impedance then
                                                canditate state = 1 //it is a candidate for distance
                                        else
                                                canditate state = 0
                                        endif
                                else if (FaultData->cDataType == 0x02 ) then //fault distance at
breaker is available
                                        do add line segment to candidates list
                                        if  line length is between minimum and maximum
FaultData->dFaultDistance then
                                                canditate state = 1 //it is a candidate for distance
                                        else
                                                canditate state = 0
                                        endif
                                else if (FaultData->cDataType == 0x03 ) then //fault impedance
at breaker is available
                                        do add line segment to candidates list
                                        if  line impedance is between minimum and maximum
FaultData->FaultImpedance then
                                                canditate state = 1 //it is a candidate for distance
                                        else
                                                canditate state = 0
                                        endif
                                endif
                                do increment ncandidates
```

do move to next node
else mark node as terminal
endif
else end_of_trace = true
endif
until end_of_trace
else all line segments on the sub-network are candidates with possibility 1
endif

return SUCCESS

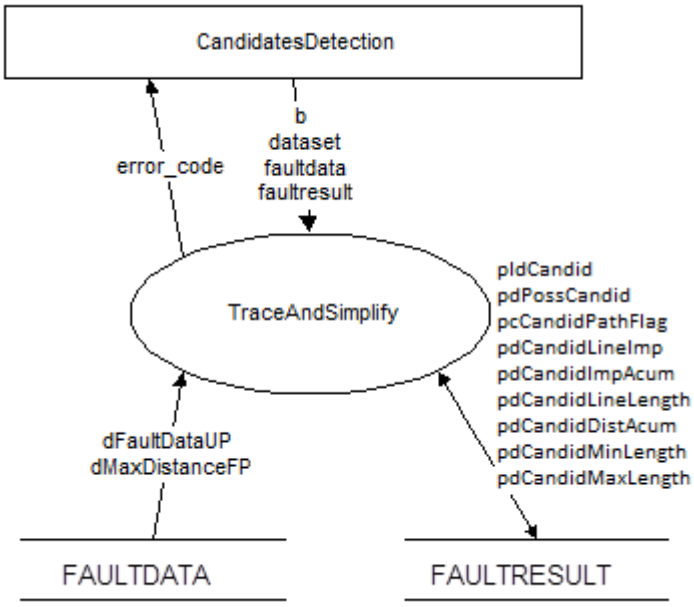


Figure 4 - Functional model for TraceAndSimplify function

For the calculation of the equivalent fault impedance as seen by the circuit breaker, the following formula is used:

$Z_{eq} = \frac{V_i}{I_{cc}}$, where Z_{eq} is the equivalent impedance as seen by the circuit breaker in p.u., V_i is the voltage in the node after the circuit breaker (towards the sub-network) before the fault, in p.u., and I_{cc} is the measured fault current at the circuit breaker in p.u.

TraceAndSimplify

Parameters:

Type	Unit	Name	Description
UID		b	Unique identifier for the tripped circuit breaker

DATASET *	dataset	Pointer to a data set handle
FAULTDATA *	faultdata	Pointer to a fault data handle
FAULTRESULT *	faultresult	Pointer to a fault result handle

Table 14 – TraceAndSimplify simplify parameters

This function returns zero (SUCCESS) if there were no problems during the process, otherwise returns a value different from zero stating the type of error that was found.

return codes:

Constant	Meaning
SUCCESS	no error, function succeeded (SUCCESS)
FLF_E_FAULTDATA	invalid FAULTDATA handle (is null)
FLF_E_FAULTRESULT	invalid FAULTRESULT handle (is null)
FLF_E_DATASET	invalid DATASET handle (is null)
FLF_E_PARAM	invalid parameter
ERROR_UID	invalid element identifier (INESC API)

Table 15 – TraceAndSimplify return values

Algorithm:

```
if faultdata is null then
        return FLF_E_FAULTDATA
endif

if faultresult is null then
        return FLF_E_FAULTRESULT
endif

if dataset is null then
        return FLF_E_DATASET
endif

if faultdata.dMaxDistanceFP is less than zero then
        return FLF_E_PARAM
endif

if faultdata.iNumPSteps is less than one then
        return FLF_E_PARAM
endif

do build a list of possible paths between the root and the ending nodes
set path = first_path
repeat
        set current node = root node
        repeat
```

if there is an active fault locator or a tripped recloser in this node then
do mark all line segments above as not-candidates
do mark all line segments below that have state equals to 1 as candidates (in this case the state is changed to 2. It indicates that this candidate was detected by a fault locator or recloser)
endif
set current node = next node
until end_of_path is reached
set path = next_path
until all the paths are checked

return SUCCESS

2.3.2 Fuzzy inference process

The fuzzy inference process uses the available data about the weather sensibility (not sensible, moderately sensible or highly sensible) and the hazard proneness zone (not prone, prone or highly prone) the network line segments are in, together with the list of the candidate line segments and the possibility the fault has occurred in each of them obtained by the candidates detection process, to calculate the final possibility that each is the fault location. Information about the current weather condition is also used in the following way:

- Normal weather is considered not to influence any lines. Then all line segments possibility they are they fault location due to the weather is set to zero.
- Variable weather is considered to influence only “highly sensible” line segments. Then, all line segments classified as “highly sensible” get a value of possibility to be the fault location equal to dPWHighlySens.
- Stormy weather is considered to influence both “sensible” and “highly sensible” line segments. All line segments classified as being “sensible” get a value possibility to be the fault location due to weather equal to dPWSens, while line segments classified as being “highly sensible” are set to a possibility value equal to dPWHighlySensStorm.

The fuzzy inference process also builds a list of line segments with fault possibility above a given value, together with the identification of the line segment the fault is most possible to have occurred in.

This function operates on valid FAULTDATA and FAULTRESULT structures.

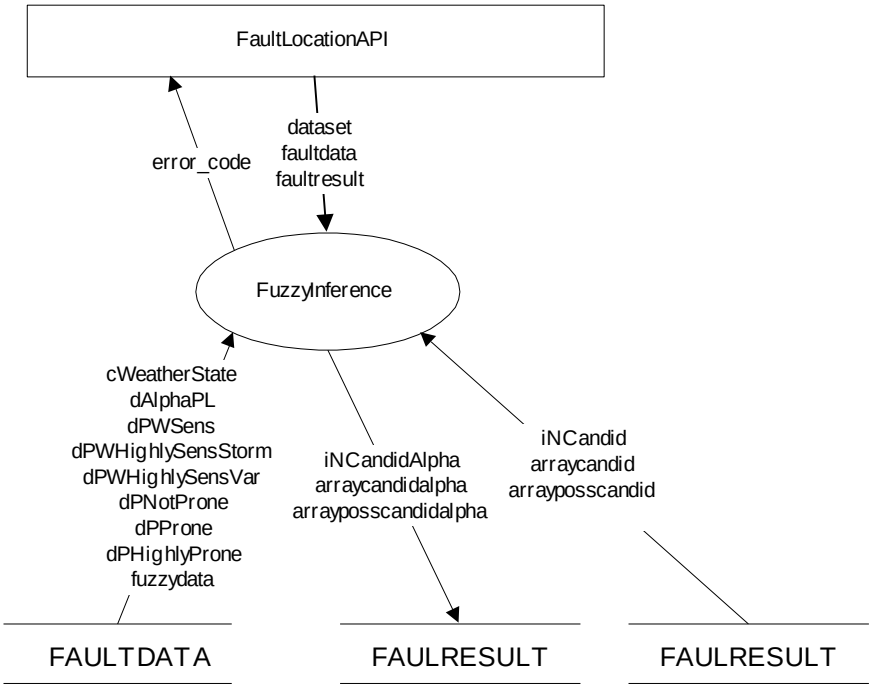


Figure 5 - Functional model for FuzzyInference function

FuzzyInference

Parameters:

Type	Unit	Name	Description
DATASET *		dataset	Pointer to a data set handle
FAULTDATA *		faultdata	Pointer to a fault data handle
FAULTRESULT *		faultresult	Pointer to a fault result handle

Table 16 - FuzzyInference parameters

This function returns zero (SUCCESS) if there were no problems during the process, otherwise returns a value different from zero stating the type of error that was found.

return codes:

Constant	Meaning
SUCCESS	no error, function succeeded (SUCCESS)
ERROR_MEMORY	not enough memory
FLF_E_FAULTDATA	invalid FAULTDATA handle (is null)
FLF_E_FAULTRESULT	invalid FAULTRESULT handle (is null)
FLF_E_DATASET	invalid DATASET handle (is null)
ERROR_UID	invalid element identifier (INESC API)

Table 17 - FuzzyInference return values

Algorithm:

if faultdata is null **then**

return E_FAULTDATA
endif
if faultresult is null **then**
return E_FAULTRESULT
endif
if dataset is null **then**
return E_DATASET
endif
set candidate = first_candidate
repeat
if weather and hazard proneness information for current candidate is available **then**
do calculate new possibility including weather and current weather state and hazard proneness information
endif
set candidate = next_candidate
until all the candidates are tested
do build list of candidates with possibility above faultdata.dAlphaPL
do search list of candidates for the one with maximum possibility
return SUCCESS

Method to combine the several values of possibilities:

A fuzzy set F in universe X is defined as being a set of ordered pairs of the form:

$F = \{(x, \mu_F(x)) \mid x \in X\}$, where μ_F is called the grade of membership of **x.**

In our case the universe **X** is the set of line segments of the sub-network. It is then possible to define the following three fuzzy sets:

- One that includes the candidate line segments and the possibility they are the location of the fault (grade of membership) that were found using the fault current with the CandidatesDetection API.
- One that includes all the line segments in the sub-network and who's possibility of being the location of the fault (grade of membership) is built from the line segment information on weather sensibility and the current state of weather.
- One that includes all the line segments in the sub-network and who's possibility of being the location of the fault (grade of membership) is built from the line segment information on the hazard proneness zone it is located in.

Therefore the fuzzy set that includes all the three rules (information on fault current, weather and hazard proneness) is the union of the previously discussed fuzzy sets. The degree of membership of an element of the fuzzy set after the union (FaultCur∪Weather∪Hazard) of the other three (FaultCur, Weather and Hazard) is calculated as following:

$$\mu_{FaultCur \cup Weather \cup Hazard}(x) = \mu_{FaultCur}(x) + \mu_{Weather}(x) + \mu_{Hazard}(x) - (\mu_{FaultCur}\mu_{Weather} + \\ + \mu_{FaultCur}\mu_{Hazard} + \mu_{Weather}\mu_{Hazard}) + \mu_{FaultCur}\mu_{Weather}\mu_{Hazard}$$

If a given member does not have grade of membership information, it is considered to be 0.

3. Structures

3.1 Generic assumptions

The data structures described will be used to maintain input data, output data and internal results. All structures shall be shared by the whole set of functions of the FLF module.

The FLF module needs the interface structure DATASET in order to maintain and calculate several results.

The FAULTDATA is a structure that contains the several parameters necessary to run the FLF module.

The FAULTRESULT is a structure that contains the output results during and after running FLF.

3.2 FAULTDATA structure

Type	Unit	Name	Description
char		cWeatherState	Current weather state: 0 – Normal 1 – Variable 2 – Storm
double		dAlphaPL	Alpha possibility level
double		dPWSens	Possibility of fault location if line segment sensible and weather storm
double		dPWHighlySensVar	Possibility of fault location if weather variable and line segment highly sensible
double		dPWHighlySensStorm	Possibility of fault location if weather storm and line segment highly sensible
double		dPNotProne	Possibility of fault location if in a not prone zone
double		dPProne	Possibility of fault location if in a prone zone
double		dPHighlyProne	Possibility of fault location if in a highly prone zone
char		cDataType	Flag that indicates the type of the data entry: 0 – no entry 1 – fault current 2 – fault distance 3 – fault impedance
char		cStartNode	Flag that indicates the start node: 0 - node that is opposite to the production side (as implemented before); 1 - FROM node; 2 - TO node;
double	kA	dFaultCurrent	Value of short circuit current provided by the circuit breaker
double	km	dFaultDistance	Value of short circuit distance provided by the circuit breaker
COMPLEX	(Ω,Ω)	FaultImpedance	Value of short circuit impedance provided by the circuit breaker
double	%	dDistanceError	Percentage of error to determine minimum and

			maximum distance
double	%	dFaultDataUP	Penalty (percentage) if fault data current is not reliable
double		dMaxDistanceFP	Possibility of fault location at the maximum calculated distance
int		iNumPSteps	Number of possibility steps from the circuit breaker until the maximum calculated distance
UID*		telephone info	Array of identifiers of nodes reported by phone as being out of power supply

Table 18 – FAULTDATA structure.

If any the values of cWeatherState, dAlphaPL, dPWSens, dPWHighlySensVar, dPWHighlySensStorm, dPNotProne, dPProne, dPHighlyProne, dFaultCurrent, dFaultDataUP, dMaxDistanceFP or iNumPSteps is set to –1, the value in question is set to its default value, as in Table 19.

Parameter	default value	unit
cWeatherState	0	
dAlphaPL	0.0	
dPWSens	0.4	
DPWHighlySensVar	0.6	
DPWHighlySensStorm	0.8	
dPNotProne	0.1	
dPProne	0.6	
dPHighlyProne	0.8	
cDataType	unknown	
dFaultCurrent	unknown	kA
dFaultDistance	unknown	km
FaultImpedance	unknown	(Ω,Ω)
dFaultDataUP	0.0	%
dDistanceError	unknown	%
dMaxDistanceFP	1.0	
INumPSteps	1	

Table 19 - Module parameters

3.3 FAULTRESULT structure

Type	Unit	Name	Description
int		iNCandid	Number of line segments candidate to be the fault location
UID*		pIdCandid	Array with the unique identifiers to the candidate line segments
double*		pdPossCandid	Array with the indexed possibilities of each candidate line segment
double*	Ω	pdCandidLineImp	Array with the impedances of each candidate line

			segment
double*	Ω	pdCandidImpAcum	Array with the accumulated impedances of each candidate line segment
double*	km	pdCandidLineLength	Array with the length of each candidate line segment
double*	km	pdCandidDistAcum	Array with the accumulated distance of each candidate line segment
double*	%	pdCandidMinLength	Array with the minimum distance of each candidate line segment in percentage.
double*	%	pdCandidMaxLength	Array with the maximum distance of each candidate line segment in percentage.
char*		pcCandidPathFlag	Array with the path flag of each candidate line segment: 0 – it does not belong to a path between the min and max distance 1 – it belongs to a path between initial point (breaker) and the min distance 2 – it is at min distance 3 – it is at max distance 4 – it is between min and max distance 5 – it has min and max distance 6 – distance not defined
char*		pcCandidSimpFlag	Array with the flag of each candidate line segment that indicates 0 – it is not candidate; 1 – it is candidate for distance; 2 – it is candidate detected by a fault locator or recloser
char*		pcCandidAlpha	Array with the flag of each candidate line segment that indicates 0 – candidate with possibility below level dAlphaPL; 1 – candidate with possibility equal or above level dAlphaPL;
int*		iNEquipsPath	Array with the number of equipments in the path between the circuit break until each candidate line segment
UID**		pIdEquipsPath	Array with the unique identifiers to the equipments in the path between the circuit break until each candidate line segment
int		iNEquipsZone	Array with the number of equipments in the fault zone between the minimum and maximum.
UID*		pIdEquipsZone	Array with the unique identifiers to the equipments in the zone between the minimum and maximum.

Table 20 – FAULTRESULT structure.

DATASET structure: contains all internal information related to a specific system or sub-system.
For more information about this structure and other non described function or data, refer to INESC document:

EFACEC_DMS PROJECT
Topology Processor Specification

Bibliography

- P. Järventaustra, P.Vehro, and J. Partanen, "Using fuzzy sets to model the uncertainty in the fault location process of distribution networks", IEEE Transactions on Power Delivery, pp 954-960, April 1994.

- Hans J. Zimmermann, "FuzzySets, Decision Making, and Expert Systems", Kluwer Academic Publishers.