KTH Electrical Engineering

RADPOW development and documentation

Lina Bertling
Patrik Hilber
Jolanta Jensen
Johan Setréus
Carl Johan Wallnerström

Royal Institute of Technology, KTH
School of Electrical Engineering
RCAM Research Group

Stockholm, Sweden
Mars 2008
TRITA-EE 2007:047

Abstract

This report summarizes the status of the computer program RADPOW. RADPOW is a program for Reliability Assessment of electrical Distribution POWer systems. It was developed at KTH School of Electrical Engineering, within the research program EKC and the research project on reliability of new electrical distribution systems. Further on, RADPOW has been used and further developed within the RCAM research group at KTH School of Electrical Engineering.

This status report contains a brief description of the RADPOW_2006 version, the Loadflow module from the RADPOW_1999_PH version and a description of the work done in the resulting RADPOW_2007 version. This version now includes a tested load flow module and the ability to calculate the latest component importance indices developed within the RCAM research group. The source code for the program has also been restructured and commented in a more detailed level than before.

Contents

1 Introduction

1.1 Background

RADPOW is a program for Reliability Assessment of electrical Distribution POWer systems. It was developed at KTH School of Electrical Engineering, within the research program EKC and the research project on reliability of new electrical distribution systems. Further on, RADPOW has been used and further developed within the RCAM research group at KTH School of Electrical Engineering [1].

1.2 Objective

This report aims to give an overview of the current status of the RADPOW_2007 version and a description of the new abilities that has been implemented in the program.

2 Versions of RADPOW

The first definitive version of the program is RADPOW_1999, developed by Lina Bertling and Ying He in their doctoral dissertations, [2][3]. In these theses and technical reports the algorithms and core modules of the program is explained and validated. In Table 1 the documents related to RADPOW has been summarized together with the different versions of the program.

The RADPOW_1999 version was followed by RADPOW_1999_PH that included the ability to run load flow calculations in the models. The additional module were referred to as Loadflow. This module did not worked properly for general systems and had memory leaks in its source code.

In the year of 2006 a Monte Carlo simulation module was implemented in the additional module Sim. The program was also converted from UNIX to a graphical user interface (GUI) in Windows. This version of the program is referred to as RADPOW_2006.

Based on the on the RADPOW_2006 version the current version of the program, RADPOW_2007, was developed and this work is described in this report. This version of RADPOW is compiled with Borland C++, Version 6.

Table 1: Summary of the different versions of RADPOW. The references in this table includes related theory and algorithms used in RADPOW and its modules. The latest version, RADPOW_2007, is described in this report.

Involved Authors	**Developed Modules**	**Developed classes and methods**	**Year**	**Version**	**Source**
Lina Bertling, Ying He	Mincut, Assbreak, Aafail, Lpind, Minpath, Netw, Branch, Comp, Sind	Main	1999	RADPOW_1999	[2], [3], [4], [5]
Philippe Rosett	Loadflow		2000	RADPOW_1999PH	[6]
Johan Setréus	Sim	Loadfile class, Graphical User Interface (GUI) classes	2006	RADPOW_2006	[7], [8]
Lina Bertling, Patrik Hilber, Jolanta Jensen, Johan Setréus, Carl Johan Wallnerström	Loadflow, Flowsolver,	Sensitivity method for Importance Indices	2007	RADPOW_2007	[9], [10], [11]

3 Updates in RADPOW_2007 version

The improvements in this version, compared with the previous, includes the following three stages:

1. Documentation, restructuring and annotating of the source code in C++.
2. Reconstruction and validation of the Loadflow module.
3. Implementation of new types of component importance indices for electrical distribution systems.

These three stages are described in Section 5 - 7 in this report.

4 Summary of methods and data flows in RADPOW_2007

In this section of the report the data structure and data flows are summarized for RADPOW_2007 version. These are all similar to the original figures for the RADPOW_1999 version in reference [2].

In RADPOW_2007 version there are three different types of system evaluations that can be initiated:

- Analytical calculation
- Iterative analytical calls for Importance indices calculation
- Monte Carlo simulation calculation

The evaluation method in RADPOW is load-point-driven, which means that all possible failures for each load point are deduced separately. The load point indices are then evaluated and these are then used for calculating the overall system indices. Figure 1 shows an overall general flow chart for this load-point-driven approach in RADPOW. The two main calculating methods in RADPOW are the analytical and simulation methods. The Importance indices calculation, described in Section 7, uses the analytical method consecutive times in order to determine the reliability importance of the components in the system.

The data flow between the modules in RADPOW_2007 is depending on which type of system evaluation that is performed. The general data flow for an analytical calculation in RADPOW_2007 is shown in Figure 2. The flow chart for the Importance indices calculation, shown in Figure 3, is very similar to the analytical chart, since the method makes consecutive calls to the analytical method. The data flow for the third system evaluation, the Monte Carlo simulation, is shown in Figure 4.

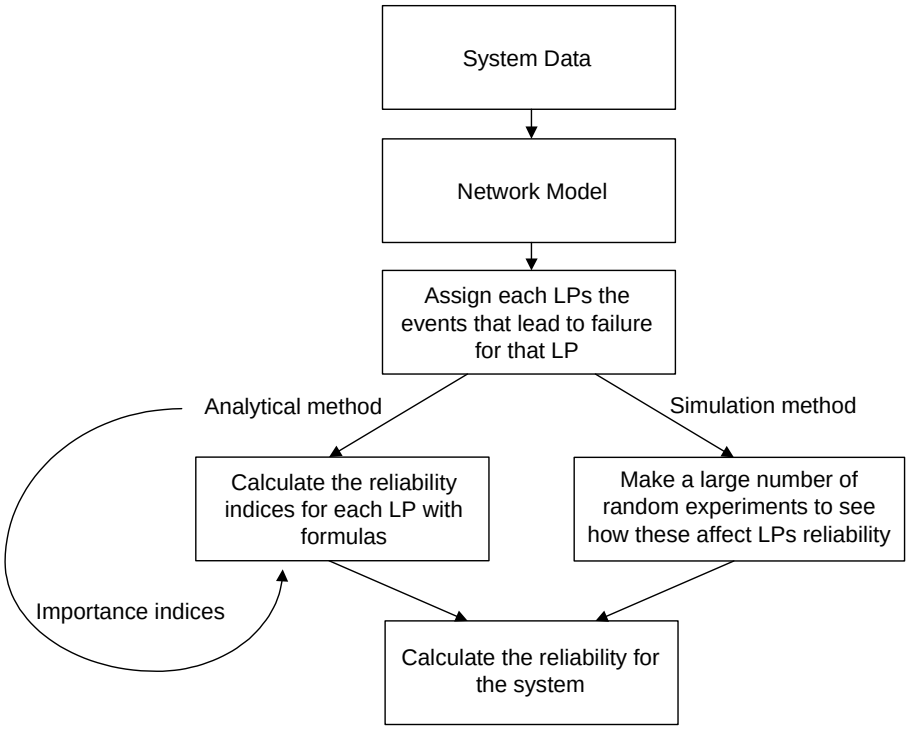

Figure 1: General flow chart and overview of the methods in RAD-POW_2007.

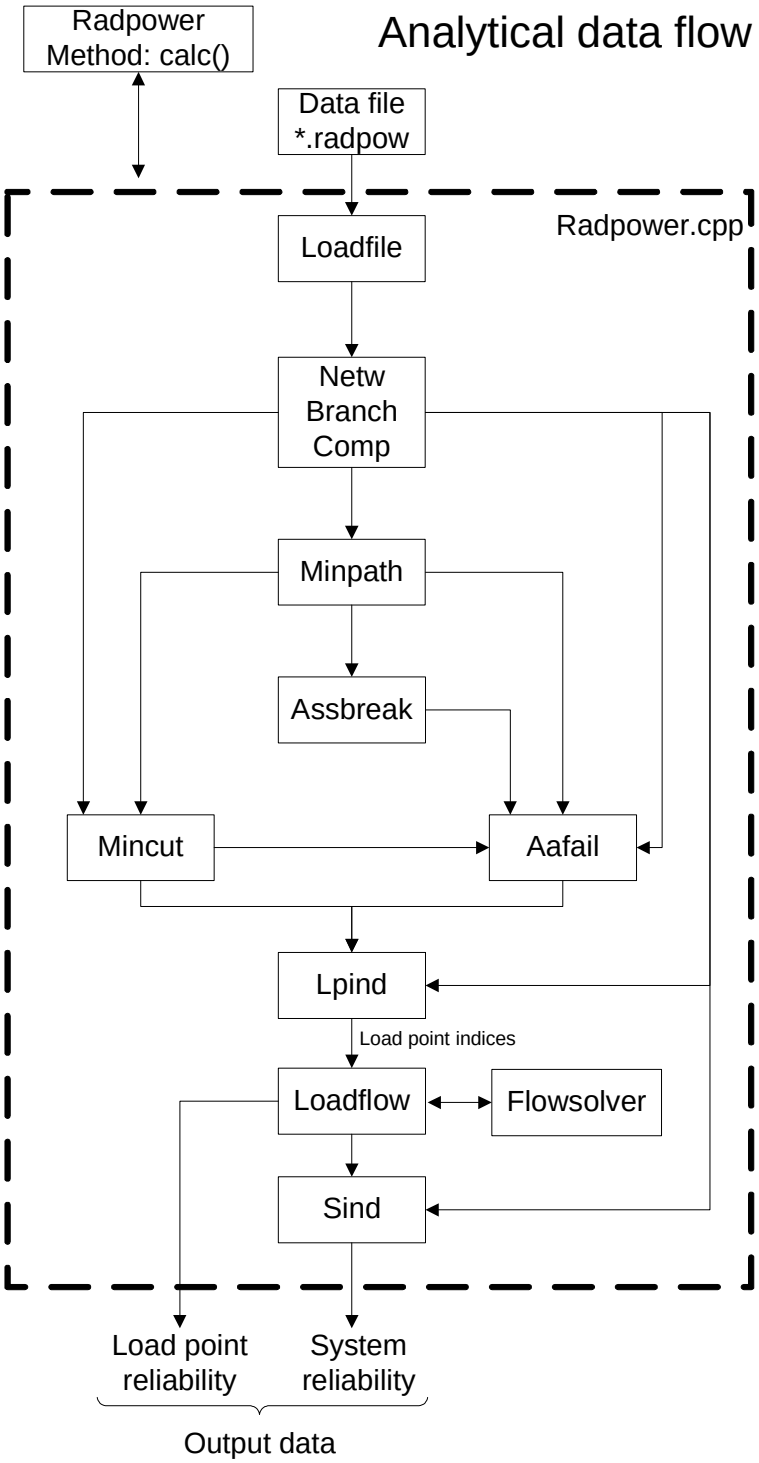

Figure 2: Data flow between the modules in RADPOW_2007 for an analytical calculation.

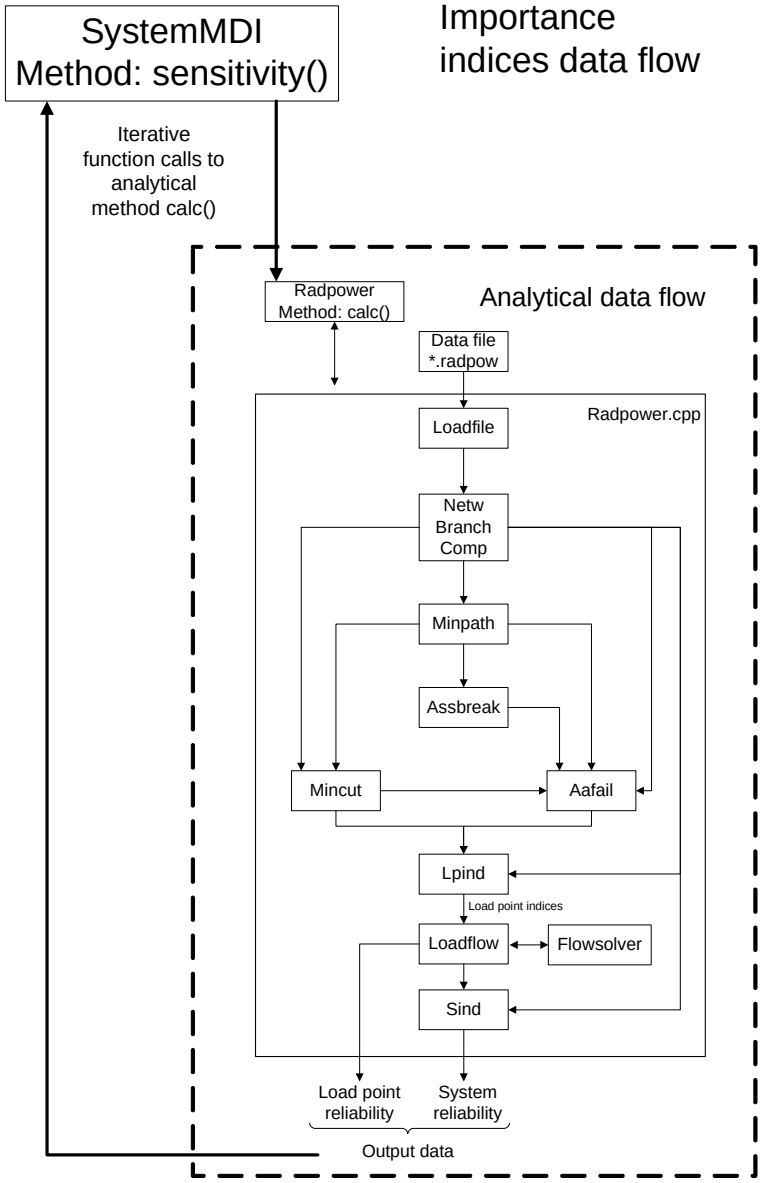

Figure 3: Data flow between the modules in RADPOW_2007 for an importance indices calculation. The method uses the analytical calculation method in RADPOW consecutive times, stores the result and evaluates the the importance indices for each component.

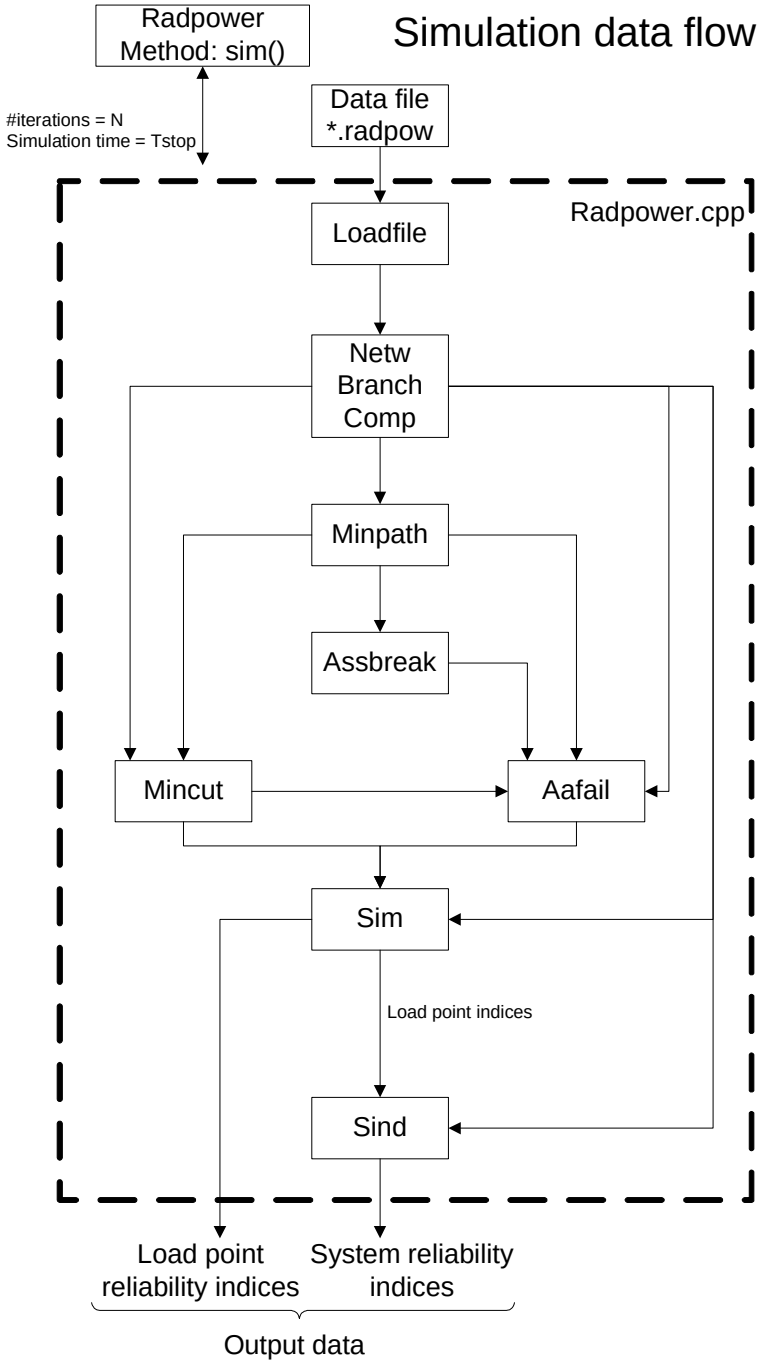

Figure 4: Data flow between the modules in RADPOW_2007 for a simulation calculation.

5 Stage 1 : Restructuring and documentation of the modules

The biggest change to the structure of the whole program is the introduction of a special headerfile "types.hh" which holds all the type declarations of the datastructures used by the program. In this way, every class which includes "types.hh" has access to all the data types used and does not need to include the other modules' header files. This approach shortens the compilation time and simplifies the dependencies between the modules.

In the Branch module, the function `getopenbrv` has been removed and moved to the Minpath module. The reason for the move was that the `getopenbrv` function calls functions declared in the Minpath module. The Branch, the Comp and the Netw modules declares extern variables which are never used. Those variables are now removed. The default constructors uses now a member initialization list instead of setting the values in the body of the constructor. The "initierad" flag keeps now updated, which was not always the case. In all modules, those instance variables which are declared but nor set and do not contain valid values are if possible set or annotated as invalid.

All the core modules (Mincut, Assbreak, Aafail, Lpind, Minpath, Netw, Branch, Comp, Sind) are thoroughly annotated. Every instance variable and every function and its arguments are annotated. The modules themselves are annotated too, i.e. they contain a description of the actions they perform, required input and produced output. The part of the class Radpower which calls the functions from the Loadfile module and the core modules has been annotated too. That means as previously annotation of the instance variables and functions but also annotation of the local variables and function calls.

6 Stage 2 : Reconstruction of the new Loadflow module

6.1 The Loadflow module in the RADPOW_1999_PH

The core modules in the RADPOW program computes system indices based on the Total Loss of Continuity (TLOC) criterion. This means that a load point is always supplied if there is at least one valid path to the supply point. No considerations is taken if a specific path's components (e.g. a overhead line or an cable) can handle the power flow without overload and tripping.

The aim of the Loadflow module is to include the Partial Loss of Continuity (PLOC) criterion to the result [6][12]. This part adds the types of events that leads to overloads in the lines in the system. One example of a situation when this is necessary is when a branch of three parallel lines requires at least two in order to function and handle the power flow. If only

the TLOC term is considered the branch fails (and the load point suffers an outage) when all three lines fails. In this case the PLOC term also adds the events of e.g. two overlapping line failures, since this leads to a line overload in the remaining line.

To calculate the PLOC term the module must for each failure in the system compute all buses voltage and angle, in order to calculate the active power flows in the lines. If there is an overload in one or more lines, there is an contingency and load shedding (total or partial) has to be made. The event that led to the interruption in the load point is included in PLOC, and this together with the TLOC gives the load point indices and also the system indices.

The fundamental theory of including the PLOC terms in RADPOW is quite well formulated in [6]. Unfortunately the module is not correctly implemented and had memory leaks. Therefore this module were rewritten and more structured than the previous one.

6.2 Implementation of a new Loadflow module

First an attempt was made to debug the Loadflow module implemented in the RADPOW_1999PH but it still executed correctly only on the smallest test system, Roset System demo [6]. The memory management is to erroneous (repeatedly writing and reading to and from not allocated memory) to make any attempts to fully correct the module impossible. A new Loadflow module was therefor developed. It consists of two classes: a Loadflow class and a Flowsolver class. Figure 2 shows the updated data flow diagram for RADPOW_2007 and the analytical method. The load flow calculation in RADPOW_2007 by default always performed. There is in this version of the program no option for not running the load flow, except a change of a parameter in the source code (which then requires a re-compilation).

The Loadflow class has the same structure as the other modules in RADPOW and contains a default constructor and an init method to set the data the module needs. The Flowsolver class supports the Loadflow class by performing the Newton-Raphson calculations. It builds the admittance matrix, solves for voltages and angles, computes the phase current on the transmission lines and checks if any line is overloaded. The Loadflow class holds in its instance variables the information about the whole system and decides which parts of the system are to be sent to the Flowsolver for computations. That means that the Loadflow class creates an instance of the Flowsolver class for each failure.

6.2.1 Flowsolver class

The Flowsolver class builds the admittance matrix, solves for voltages and angles using the Newton-Raphson method and computes the phase current

on the transmission lines. The correctness of the admittance matrix was verified for the following three test systems: Roset System demo [6], Test System 1b [4] and for Birka system [2]. The correctness of the Jacobian matrix and its inverse was verified for the Test System 1b and for Birka. The voltages, angles and phase current was verified for the Test System 1b. If any transmission line becomes overloaded the Flowsolver class computes and returns the failure's contribution to the load point indices. If no line is overloaded the contribution returned equals zero. The Flowsolver class calculates and logs the angles, the voltages and the phase current for the original system without failures too. It is recorded as failure 0, because no component has id number equal to zero.

The results from Flowsolver class is verified in [9]. In this report the results of the bus voltages and angles in RADPOW has been compared for two test systems with both NEPLAN [13] and MATLAB. The result from RADPOW equals the the results from these two programs.

6.2.2 Loadflow class

To be able to conclude which parts of the system ought to be sent to the Flowsolver class, the Loadflow implements an algorithm developed for this purpose. Given the load point's id number, the minimal cut sets and the additional active failures, the algorithm concludes which failures cause PLOC and for each such failure removes the branches which are without power supply and also those which do not influence the given load point and sends the remaining branches to the Flowsolver class for computations. In this way the Flowsolver class always works on the minimal and valid data i.e. there is no need to connect extra slack busses. The algorithm is implemented in the `calculate_flow` function in the loadflow.cpp file and described in detail in the function annotation and in Appendix A. The algorithm was tested for the Test System 1a, Test System 1b (specified in [4]) and the Birka system [2], all with correct results.

For each load point, the Loadflow class gathers the results sent by the Flowsolver and computes and returns the load point's indices, i.e. the failure rate lambda, the unavailability U and the energy not supplied LOE. The average duration rate r is set to zero - the right value is computed in the Radpower class. The computations may be done without or with load shedding and are based on constant power demand (computed as the average power demand). First step in load shedding (i.e. shedding the load(s) closest to an end of an overloaded line) was tested on Roset System demo [6], Test System 1a and Test System 1b and performed correctly. To test the second step in load shedding (i.e. shedding other connected loads) the input data files for Roset System demo, Test System 1a and Test System 1b must be modified, and this validation and test is left for future work.

Both the Loadflow and the Flowsolver class creates a log file of selected

data. The log file of the Loadflow class is simply called `loadflow.log`. The name of the log file of the Flowsolver class is built of the load point number, the failure number and the word flowsolver which gives the format `LP_FAILURENR_flowsolver.log` (e.g. 1_1_flowsolver.log). Those files becomes overwritten when another system is evaluated in RADPOW.

There are some demands the input data file must fulfil to ensure the correct execution of the Loadflow module. There must not be a component or a branch numbered 0. All the component and branch numbers start at 1 and the number of the next component or branch is a successor of the previous one.

For detailed description of the Loadflow class see Appendix A and Appendix B for the description of the Flowsolver class.

7 Stage 3 : Implementation of new component importance indices

7.1 Three major groups of indices

In RADPOW a number of additional reliability indices has been implemented within the PhD thesis in [10]. The indices, outlined below, utilize the concept of customer interruption cost (reliability worth) and traditional power system reliability measures (e.g. SAIDI and SAIFI) as measure of system reliability in order to establish the importance of the components.

Three major groups of indices are calculated in the tool:

1. Importance of the individual component's failure rate.
2. Importance of the individual component's repair time.
3. Component maintenance potential, i.e. how the system measures would be affected by an always available component (failure free).

For all of these three groups component importance is established with respect to SAIDI, SAIFI, CAIDI, ASAI, AENS and customer interruption cost. In total this results in 18 importance measures for every component. Which of the indices to use, in an optimization andor identification process of important components, depends on the objective. E.g. a company whose performance is measured in SAIFI and SAIDI will naturally study component reliability importance indices based on these.

One simplified interpretation of the indices of group 1 is that the values correspond to the expected "system reliability loss" in case of a failure of the studied component. The interpretation of group 2 is that the value corresponds to the expected value per year for a reduction of the repair time with one hour. These values can for example be compared with the cost of placing repair equipment closer to the studied component. The interpretation of

group 3 is the value (SAIDI, SAIFI, etc) that is possible to save on making the studied component perfect in reliability terms (i.e. failure free). Note that this is the potential savings from making one component perfect. Two components in parallel gets the same potential. If one of the components is made perfect there is no potential "left" for the other component. This gives an indication on how much system performance it is potentially possible to gain by making a component failure free.

7.2 Implementation of customer interruption cost

The customer interruption cost (reliability worth) for each load point is implemented in method `get_lp_outage_cost` in module Lpind. The total customer interruption cost for the system is implemented in module Sind in method `total_outage_cost`. The code implementation for each of these two methods are placed at the end of the file in both modules.

The input data for evaluating the customer interruption cost for each load point and then the system is

- cost per interruption (cpi) (SEK/int) and
- cost per hour (cph) (SEK/hours).

These two parameters are defined in the `ncuspow` section in the input data file (*.radpow) for each load point. For this reason two extra columns (cpi and cph) have been added under the `ncuspow` section. In order to use the RADPOW_2007 version these columns needs to be added in the input data file.

7.3 Algorithm and implementation of method Sensitivity for evaluation of importance indices

The indices are numerically calculated by alteration of the input data file to RADPOW. The calculations are performed according to the following process:

1. Basic run of RADPOW, with calculation of minimal paths, etc. Store system outputs (interruption costs, SAIDI, SAIFI, etc)
2. For component `i` increase the passive and active failure rate (or the repair time depending on which set of indices that are to be calculated) with a small value epsilon.
3. Call for a new reliability calculation (RADPOW without deducing minimal paths, etc, since this is already calculated)

4. The indices are now formed by subtracting the original values, step 1, from the new values, step 3. The differences are then divided by epsilon and stored. This results in a number of numerical partial derivatives of the systems reliability measures with respect to component failure rate/repair time (these measures are called component reliability importance indices).

5. Restore original values for component `i` (passive and active failure rate).

6. Perform step 2-5 for all components (increment `i`).

The approach for calculating the maintenance potential for the components is similar to the above. The difference is that instead of a small modification, epsilon, the failure rate is set to zero at step 2. And in step 4 only the differences are calculated (no division).

The implementation of the component importance indices is placed in the main graphical user interface `SystemMDI.cpp`, in the method `Sensitivity()`. The code is placed at the end of the file, where also an annotation is made. From the file, the analytical calculation engine in RADPOW is called consecutively times for each alternation, as described above. The result for each component and index is presented directly in the Log window in RADPOW. The result from RADPOW has been validated by earlier calculations performed in MATLAB.

8 Closure

8.1 Conclusions

The RADPOW program, including the restructured and annotated source code, the new Loadflow module and the ability to calculate three different groups of component importance indices, is refereed to as version RADPOW_2007. The implementation of these three activities and stages has been presented in this report.

The changes made to the modules of the RADPOW program do not influence the data flow in the program (as shown in Figure 2). The restructure of the program code shorten the compilation time and the general structure of the program becomes clearer and easier to follow.

The Loadflow module from the RADPOW_1999_PH has been replaced by a new one where the problems that occurred earlier has been fixed. The module Flowsolver has been implemented to support the Loadflow module by selecting the cases that is going to be evaluated and examined closer by load flow calculations. The electrical results from the Flowsolver module has been validated for three different test systems.

In the program a number of additional component reliability indices has been implemented by using RADPOW's built in analytical calculation engine consecutive times. The result for the proposed reliability indices are then presented directly to the user and this can be used as an priority list of the system's components.

8.2 Future work

For future work it would be preferable to test the module Flowsolver on several other systems. The load shedding policy that has been implemented in this module also needs more detailed tests and validation.

The option to turn off the Loadflow module in a calculation should also be implemented in the RADPOW graphical interface in future development. In the RADPOW_2007 version this is only possible in the source code, and it then requires a re-compilation. Other parameters for the Newton-Raphson algorithm iteration should also be possible to control by the user.

To make the development work more consistent and tracing of the changes easier, the use of a version control system should be considered in future work.

Appendix A - Stage 1 : Loadflow class

The Loadflow class uses its default constructor to open the log file named "loadflow", in which the run results are being recorded. The destructor closes the log file. The data needed by the Loadflow class is sent in two steps:

1. The arguments to the init method which is called only once.
2. The arguments to the calculate_flow method which is called for every loadpoint in the system.

The arguments to the init method are as follows:

- filenames of all files containing system data,
- the number of the loadpoints to be analysed,
- the total number of branches,
- the load points and their minimal paths,
- the components' id and type data,
- the components' reliability data,
- the total number of components,
- the branches in terms of components,
- the id numbers of the open points,
- the id numbers of the supply points,
- the number of supply points,
- the load points' customer and power data and
- the total number of the loadpoints.

The init method:

- reads and stores the data from the pflo and pfrx files (the read_pflo and read_pfrx methods),
- stores the received minimal paths in a more convenient data structure (the rearange_minpaths method),
- stores the received components' id and type data in more convenient form (the set_comps_idt method),
- stores the received components' reliability data,

- stores the received branches in terms of components in a more convenient data structure and for each component computes and stores which branch it belonges to (the set_comps_by_branches method),
- stores the received id numbers of the open points,
- stores the received supply points in a more convenient data structure (the set_supply_points method),
- stores the received load points' customer and power data in a more convenient data structure (set_lp_customers method), and
- prints chosen data to the log file.

The arguments to the calculate_flow method are as follows:

- the loadpoint's id number,
- the minimal cut sets for the load point and
- the additional active failures for the load point.

The calculate_flow function returns the indices for the given load point, i.e. the expected failure rate per year (lambda), the annual unavailability in hours per year (U) and the average loss of energy per year (LOE). The expected outage duration for a failure (r) is not computed by the Loadflow but by the radpower class. During its execution, the calculate_flow method:

- gets the components id numbers from the second order minimal cut sets for the given loadpoint,
- gets the first order additional active failures,
- gets the normally closed and normally open paths of the given load point,
- gets the normally closed paths of the other load points.
- For every component from the second order minimal cut sets, i.e. for every second order failure
  - the branches the component is member of are recorded as invalid,
  - from the n/c paths for the load point the paths which contain an invalid branch are removed (clean_path method),
  - a check is made if there are any n/c paths left, if no an error is logged and the remainder of the for statement is skipped,
  - from the n/c paths for other load points the paths which contain an invalid branch are removed (clean_paths method),

  - the branches included in the n/c paths of the load point checked are computed (compute_bvalid), and extended with branches from other load points' n/c paths connected to the n/c paths of the load point checked (extend_bvalid),
  - the components id numbers and types included in the branches are computed (pick_comps_ids, pick_components),
  - the number of busses among the components are computed,
  - the branches' .se and .re fields are mapped to the indices in the admittance matrix (pick_branches, set_y_se_y_re methods),
  - the relevant supply points are computed (pick_supply),
  - the type of the failed component is computed,
  - a check is made if the failed component is member of the additional active failures of the first order,
  - a check is made if there are an open point in the path of the checked load point
  - a Flowsolver object is created and the data is sent to its constructor (see Appendix B),
  - the calculate_flow method of the Flowsolver object is called to obtain the failed component's contribution to the indices of the checked load point (see Appendix B),
  - the lambda, the U and the LOE indices are updated with the values received from the Flowsolver's calculate_flow method,
  - the indices are printed to the log file.

- prints the load points indices (lambda, U, r = 0, LOE) to the log file and
- returns the indices.

The calculate_flow method creates an instance of the Flowsolver class for every failed component. To obtain power flow in a flawless system a dummy component which id number equals 0 is used. Because there is no real component with this id number it will not be found as a member of any branch and no paths will be removed from the system. The contribution of the dummy component to the load point indices equals zero values.

Appendix B - Stage 1 : Flowsolver class

The Flowsolver class is a helper to the Loadflow class. All data needed by the Flowsolver class is sent to its constructor. The arguments to the constructor are as follows:

- the load point's id number,
- all the load points' power flow data,
- all the load points' customer and power data,
- only valid branches and their power flow data,
- the mapping of pairs <key: index in the admittance matrix, value: the component id numbers>,
- id numbers of valid supply points,
- the number of valid busses,
- the id number of the failed component,
- the type of the failed component,
- the reliability data for all components,
- a flag indicating that the component is a member of additional active failures of first order
- a flag indicating that there is an open point in the system.

The constructor initiates instance variables with the values received, sets the size of the admittance matrix to the number of busses + 1, because of the artificial bus 0 which will be connected to the system, compiles the name of the log file of the load point's id number, the failure's id number and the "flowsolver.log" word and opens the file for writing. The destructor closes the log file.

The calculate_flow method of the Flowsolver class

- creates a mapping of pairs <key: the component id number, value: the index in the addmitance matrix> (set_comps method),
- sets the expected P and Q values (set_expected_pq method),
- removes superfluous loadpoints from the data received (set_expected_pq method)
- stores the expected P and Q values in original P and Q values in case the load shedding is run (the set_min_pq method),

- sets the minimi P and Q values (set_min_pq method, must be called after the set_expected_pq method),
- prints chosen data to the log file,
- compile statictics on paralell lines (build_Y_bus_helper method),
- sets the admittance matrix (the build_Y_bus method),
- computes and sets the voltages and angles (the newton_rapson method),
- if the newton_rapson converges the phase current on the branches is computed (the phase_current method) and the branch data is printed to the log file. Further if there are overloaded branches, the contribution of the failure to the load point indices is computed (the indices method). The overloaded branches are printed to the log file. If there are no overloaded branches the contribution of the failure equals zero.
- if the newton_rapson method does not converge the contribution of the failure equals zero,
- the contribution of the failure consist of the components failure rate per year (lambda), its annual unavailability in hours per year (U) and the loss of energy (LOE) caused by the failure.

If the LOAD_SHEDDING flag is set to 1, the calculate_flow method calls the shed_the_load method to perform the load shedding before the indices are computed. This method returns true if load shedding was succesfully performed. In this case the LOE is based on the difference between the P required and the P supplied. Notice that if the load of an other load point that the load point which indices are being computed, has been shed, the P supplied is considered to be equal the P required and LOE equals zero. If the shed_the_load method returns false, the P supplied is considered to be zero and the LOE is based on P required.

References

[1] Lina Bertling. The RCAM research group - history and status. Technical report, Royal Institute of Technology (KTH), February 2008.

[2] L. Bertling. *Reliability Centred Maintenance for Electric Power Distribution Systems.* Doctoral dissertation, Department of Electrical Engineering, KTH, Stockholm, Sweden, August 2002. ISBN 91-7283-345-9.

[3] Y. He. *Modelling and Evaluating Effect of Automation, Protection, and Control on Reliability of Power Distribution Systems.* Doctoral dissertation, Department of Electrical Engineering, KTH, Stockholm, Sweden, September 2002. ISBN 91-7283-351-3.

[4] Y. He, L. Bertling. The verification of the reliability assesment computer program radpow. Technical report, Department of Electrical Power Engineering, KTH, 1998. A-EES-9808.

[5] L. Bertling. Status report for the computer program radpow. Technical report, Department of Electric Power Engineering, KTH, 2000.

[6] P. Rosett. Power Flow Assessment for Reliability Evalution in RADPOW. Master's thesis, Department of Electrical Engineering, KTH, 2000. B-EES-0005.

[7] J. Setréus. Development of a simulation module for the reliability computer program radpow. Master's thesis, School of Electrical Engineering, KTH, June 2006. XR-E-ETK 2006:010.

[8] J. Setréus, L. Bertling, S. Mousavi Gargari. Simulation method for reliability assessment of electrical distribution systems. *To be published at the Nordic conference on Nordic Distribution and Asset Management (NORDAC)*, August 2006.

[9] Johan Setréus. Verification of results from the loadflow module in RADPOW_2007. Technical report, Royal Institute of Technology (KTH), Mars 2008. TRITA-EE 2008:013.

[10] P. Hilber. *Maintenance Optimization for Power Distribution Systems.* PhD thesis, Royal Institute of Technology (KTH) School of Electrical Engineering, Stockholm, Sweden, April 2008. TRITA-EE 2008:012, ISSN-1653-5146.

[11] P. Hilber, L. Bertling. Component reliability importance indices for electrical networks. In *Proceedings 8th International Power Engineering Conference, IPEC 2007*, Singapore, 2007.

[12] R. Billinton and R.N. Allan. *Reliability Evaluation of Power Systems*. Plenum publishing corporation, New York, 2 edition, 1994. ISBN 0-306-45259-6.

[13] BCP Busarello + Cott + Partner AG. Neplan, power system analysis. www.neplan.ch, 2008.