Natural Resources Canada
Ressources naturelles Canada

CanmetENERGY
Leadership in ecoInnovation

REVIEW OF OPEN SOURCE CODE POWER GRID SIMULATION TOOLS FOR LONG-TERM PARAMETRIC SIMULATIONS

Canada

REVIEW OF OPEN SOURCE CODE POWER GRID SIMULATION TOOLS FOR LONG-TERM PARAMETRIC SIMULATIONS

Prepared by:

Marc-André Moffet
Frédéric Sirois, Ph.D.

École polytechnique de Montréal

and

David Beauvais

CanmetENERGY, Natural Resources Canada

Date

July 11th, 2011

CITATION

Marc-André Moffet, Frédéric Sirois and David Beauvais; Review of Open-Source Code Power Grid Simulation Tools for Long-Term Parametic Simulation, CanmetENERGY technical report 2011-137, July, 2011.

DISCLAMER

ACKNOWLEDGEMENT

Financial support for this project was provided by Natural Resources Canada through the Program on Energy Research and Development (PERD).

TABLE OF CONTENT

LIST OF TABLES

LIST OF FIGURES

1 INTRODUCTION

New grid simulation tools are required for the progressive development of a Smart Grid able to manage distributed generation, charging of electric cars or of many special protection systems [1] [2] [3]. Properly modelling and simulating the grid is among some of the challenges posed by the progressive deployment of a large number of distributed resources, including infrastructure related to flexible loads and storage. What is basically needed is greater granularity of information, as well as better performing analysis tools that are capable of processing a larger quantity of information over time.

The advent of smart grids has led to new grid functions that bring with them their own challenges for the planner. Integrating the decentralized generation of renewable energy may reverse loadflow direction and send power to the transmission network. A high penetration of this resource on a weak grid may result in voltage surges at the end of the distribution line, and thereby require a review of the voltage regulation and grid protection. With the electrification of transportation, charging electric cars may overload grid components upstream from electric vehicle charging stations. Finally, managing peak demand brings about changes to the load profile, thereby affecting grid planning criteria and practices. In the mid-term, the smart grid aims to take greater advantage of these distributed resources. In the event of a grid outage the goal is that, with distributed generation and additional support for storage, demand management and automated reconfiguration, it will be possible to island the load, in order to create an autonomous micro-grid and thereby maintain a community's power supply.

In order to address these numerous technical challenges, electrical distribution network engineers will need to add new techniques to their repertoire. They must be able to do "annual" type simulations (time series simulation), as well as long-term transitory simulations (in minutes) to evaluate, for example, the effect of a drop in solar generation due to a passing cloud on grid voltage, or to optimize management of demand and power storage resources. It is in this perspective that we will present three recently developed electrical distribution network simulation tools: GridLAB-D, OpenDSS and APREM. These software applications are first and foremost aimed at grid planners, not operators. They are open-source code and available free of charge: online, in the case of OpenDSS and GridLAB-D, or on request, in the case of APREM.

A description of these software applications and their functions is provided in this report. Two case studies are simulated with the software applications, and they serve as a basis for comparison concerning their respective performances.

2 DESCRIPTION OF SIMULATION TOOLS

No tool in the power grid simulation field can respond to all needs simultaneously. The different tools developed to date can be reasonably divided into three major groups. First, we have transient state simulation tools (time-based simulations, without limitation as to the duration of the simulations), such as EMTP-RV, EMTDC, SimPowerSystems, etc. These are also sometimes called “off-line simulations.” This type of tool may be used, among other things, to evaluate impacts on the power grid of all things that cause a transient state: lightening, turning a circuit breaker on/off, short-circuit, etc. Models are generally valid over large frequency ranges, from direct current to MHz.

Second, we have real-time software applications (time-based synchronized in real time with external inputs/outputs and, in general, external physical equipment/systems) such as RTDS, Opal-RT, Hypersim, etc. These simulators, also intended to study transient states, are used mainly to check how new equipment to be connected to the grid, such as protector relays, behaves. Given the demands of real time simulation, large calculators (multi-processors) are required. Nevertheless, certain concessions must be made with respect to the accuracy of the models and/or size of the simulations.

Third, certain grid analysis software applications in the frequency-based state allow for loadflow calculations, such as PSAF, ETAP or PSS/E. These software applications are used to find the steady state solution of a simple or complex electric circuit. The user enters the electrical parameters of the lines, transformers and nominal voltage of the nodes, generator production and load amount at a given time. The software then determines the voltages and angles at all buses and, as a result, the amplitude and direction of the power exchanges between each bus, thereby making it possible to check for overloaded lines or generators, and verify that the grid can duly sustain the load and generation conditions. We also find, related to frequency, other calculation functions or software applications for calculating short-circuit currents, harmonics, protection system adjustment, etc.

When planning electrical distribution networks, a common simulation practice is to consider only the worst case, i.e., peak load on the line. We therefore need to determine the active and reactive voltages and powers on each steady-state node in order to be able to determine whether the grid will eventually need to be upgraded. In order to conduct these studies, the following must be known: peak demand forecast (summer or winter, depending on the climate and clients), layout of the electrical distribution network and the different electrical parameters of the network equipment (conductor gauges, conductor length, electrical parameters of transformers, etc.). Software applications, such as Cymdist, that can take this type of information and calculate loadflow are commonly used [4].

Three tools were used in this report to do this type of simulation, namely: OpenDSS, GridLAB-D and APREM. The first two are not only free and open-source code, but they are also developed by research centres in the electrical power field. OpenDSS is developed by the ElectricPower Research Institute

(EPRI) and GridLAB-D by the Pacific Northwest National Laboratory (PNNL) under the US Department of Energy (DOE). The APREM software is developed by the École Polytechnique de Montréal.

2.1 OpenDSS

2.1.1 Description of the Software Application

OpenDSS [5-6] (Distribution System Simulator) is an open-source code software application originally developed by Electrotek Concepts in 1997. One of the reasons behind the development of this tool was the calculation of harmonics and interharmonics on the electrical distribution network. Different modules where later added, such as Monte Carlo analysis and annual simulations. In 2004, the software was purchased by EPRI in order to have a tool to simulate the advanced automation and modernization of the power grids. It was with this in mind that EPRI decided to make the source code available free of charge online on the SourceForge.net site. A wiki is also available online to assist in understanding and using the software.

OpenDSS works by command line with its own console. It is also designed to be used with Matlab or Excel. It can be downloaded for free online and does not require installation. It is programmed in Delphi (object-oriented Pascal) language. Based on how it was designed, it can be integrated into different software application through a COM interface. Companies can therefore use the calculation tool and easily integrate it into their software. The software does frequency-based rather than time-based analysis. It is therefore not a transient state analysis tool. OpenDSS is also not traditional loadflow software in the sense that it does not use any non-linear system resolution algorithm such as the Newton-Raphson algorithm. In fact, the software solves the circuit by using an impedance matrix, according to EMTP, and with a particular current injection method, which enables it to find the voltage and current at every circuit node with very few iterations. However, unlike traditional loadflow tools, with OpenDSS, it is the load that has a node, and not a node that has a load. A node can therefore have several types of loads connected to it. OpenDSS makes it possible to define the different elements of the electrical distribution network: lines, cable, capacitors, voltage regulators, transformers and loads. It can solve imbalanced three-phase circuits and do simulations taking into account the impedance of the distribution transformers. Even if the software works by command line, an interface was developed from the program for the graphs. Moreover, if geographical data are integrated, the software can draw the shape of the electrical distribution network.

2.1.2 Types of simulations

OpenDSS is used to do several types of simulations that are difficult to do with more traditional transient study or loadflow tools. The following are a few examples:

- Calculation of losses in a wind farm collection system: Losses in a wind farm collection system vary according to the power delivered by wind turbines, which fluctuates over time. This calculation can be done traditionally, taking into account average wind turbine power output. In exchange, with OpenDSS, and using the power delivered by each wind turbine each hour of the year, the calculation of total losses for all 8,760 hours is done in a few code lines only, unlike traditional tools that require heavier programming, often consisting of data exchange interfaces that perform poorly in terms of execution time.

- Calculation of the effect of a cloud passing over a solar power plant: When clouds pass in an area with photovoltaic panels, electricity generation will be down for a few moments (from a few seconds to a few minutes) and then resume according to a slope of just a few seconds. If these generation variation data are known, OpenDSS can easily help to determine voltage drop on the bus to which the solar generation is connected. We can therefore determine whether or not there will be ill-timed operation of the transformers with tap changers or capacitor banks to regulate this voltage variation.
- Impact of the arrival of electric cars on the electrical distribution network: The analysis capacities of OpenDSS make it possible to evaluate the impact of connecting electric vehicles to the power grids. EPRI used OpenDSS to analyze the impact on the Hydro-Québec grid [7].
- Short-circuit analysis using the Monte Carlo approach: OpenDSS makes it possible to carry out short-circuit analysis using the Monte Carlo approach. The software automatically generates short circuits randomly and finds the circuit solution under these conditions.
- Reactive power and voltagecontrol (Volt Var Control): In order to reduce consumed energy and losses, electric power companies use capacitors and voltage regulators at the station or on the line to keep voltage within the recommended limits. This new approach to voltage regulation requires measuring voltage at the end of the grid. Numerous scenarios of loadflow over the year can be studied with this dynamic tool.

2.2 GridLAB-D

2.2.1 Description of the Software

Gridlab-D [8] is anopen-source software application developed by PNNL (Pacific NorthWest National Laboratory), a laboratory with the US Department of Energy. GridLAB-D is also an electrical distribution network simulation application – which explains the letter D in the name. Unlike OpenDSS, GridLAB-D was first and foremost developed as a residential load simulator [9]. Other modules were then added for grid simulations. PNNL uses this tool to carry out analysis on the impact of the energy conservation, renewable energy integration and new technology program on the entire US power grid. The PNNL laboratory used one of the first versions of this software when showing how a smart water heater reacts to a market price signal in real time (OlympicPeninsula Project) [10].

GridLAB-D can be used to model a home's electric loads (dishwasher, refrigerator, freezer, dryer, etc.), as well as thermostat-controlled loads. The approach is different for household appliance and water heater simulation, or air conditioning and heating demand. The profiles of the household appliance loads were developed based on a study on the use of different residential loads by Americans (ELCAP – End-Use Load and Consumer Assessment Program 1983-1990). The software takes overall load curves, seen from the integrated network, for each of the devices and distributes them house by house, based on a top-down allocation. This approach enables different analyses on the overall impact of demand management, but does not reflect the specific dynamic of a house and its loads over time. It is rather an average load. As concerns the thermostat-controlled loads (heating, air conditioning, water heater), GridLAB-D includes a dynamic model that uses a discrete set of data provided by the user.

The software includes a water heater model and thermal parameters of a house to calculate heating, air conditioning and hot water needs. In fact, the number of square feet, wall and window insulation can be calculated. The type of heating/air conditioning (electrical, thermal) can then be defined, and the software calculates the need. In order to reflect temperature variations, it uses typical meteorological year (TMY) data. For the time being, only US data are available.

The method used to model heat exchanges is the ETP (Equivalent Thermal Parameter) method. This method makes it possible to convert thermal parameters to electric parameters. It represents the thermal dynamics of a house using resistances and capacitors. The temperature is represented by voltage. The calculation method for heating/air conditioning needs is not as developed as in the case of building mechanics software, such as Energy Plus or Trnsys, but it can still simulate heat exchanges for a time step of less than one minute. Also, as we will show below, simulation time is already quite important, so an overly complex model would slow the program down unnecessarily.

The GridLAB-D approach for determining heating, air conditioning and water heater needs is therefore to model the need of each house and go back up the network, from the bottom up. This is the reverse of the method used to analyze the impact of household appliances. In the first versions of the software, GridLAB-D used the Gauss-Seidel algorithm to solve the loadflow. However, this algorithm did not offer good large-scale network performance. That is why the current algorithm is the Newton-Raphson. At this time, GridLAB-D does not offer any graphic interface. However, like the OpenDSS, there is an integrated program that allows graphics to be traced in order to properly view the results.

The designers of this software would like other companies to use their source code and integrate it in their software applications. The programming language used is in general C and C++. Text files which describe the different network elements and loads and define the simulation parameters and type of results desired must be written for the software to operate. To run these text files,the Windows command window must be used, and then the name of the file to be run must be typed. An interface with Matlab was developed, but seems difficult to operate (see discussion group). This makes the program heavier to use than OpenDSS. Finally, PNNL offers group software training.

2.2.2 Types of Simulations

Several types of electrical simulations may be done with GridLAB-D, but what sets it apart is its ability to simulate demand management. The user may specify the simulation time parameters: the smallest time step value is one second and the biggest is one hour. Simulations over a time frame of several years can also be done using this software.

Several modules have been developed by the scientific community and are available on SourceForge. The main GridLAB-D modules are a load control module (demand side management – under development), a reliability module for evaluating SAIDI and SAIFI indexes following grid automation, a market module to simulate an electricity market, a module for real time calculation, i.e., state-changing loads in real time and a module including models for commercial, industrial and agricultural loads. A module to simulate a communications network was recently added. The software is being developed by the community, therefore, certain functions are not accessible to everyone, are lacking documentation or are not final.

The software is also used for loadflow calculations on the distribution networks. GridLAB-D was used to evaluate energy conservation gained from voltage control (CVR) on the American distribution network [11]. Twenty-four types of distribution circuits with loads were modelled with GridLAB-D, in order to model this conservation. These models are available with the software.

As previously mentioned, this software was developed for large-scale simulations. We can therefore simulate grids with tens of thousands of homes, each with its respective residential load, as well as large-size distribution networks. GridLAB-D also makes it possible to simulate these grids at the same time on different computers.

2.3 APREM

2.3.1 Software Description

APREM (Analyse Paramétrique des Réseaux Électriques [Parametric Analysis of Power Grids] with Matlab) was developed at Montreal's École Polytechnique in order to respond to certain specific Hydro-Québec needs [12]. APREM functions with Matlab and uses this software's object-oriented programming, combined with MEX files (files written in C++ and pre-compiled to accelerated calculations). APREM uses the increased node matrix method to solve the electrical circuit [13]. Like OpenDSS, it makes loop simulations possible by varying the load, the generation and even the topology of the grid. Of the three software applications proposed here, this one is the easiest to use. Furthermore, since it works with Matlab, an intermediate Matlab user with a basic understanding of loadflow can learn to use APREM in under an hour. The user must first of all define his grid (impedances, transformers, loads, sources, switches, etc.), and can then change any circuit parameter at will and recalculate the solution. Its ease of use makes it the ideal software for academic use. However, APREM offers fewer functions than the two previous software applications given more restricted budgets and development time. For example grid automations (capacitor bank, tap change) can be easily integrated in OpenDSS, which is not yet the case with APREM. However, APREM is able to associate an arbitrary number of functions (transformer tap change, capacitor bank control, grid reconfiguration, etc.) with each piece of grid equipment, which makes it possible to model any conventional or other control system found on the grid. For example, APREM enables the user to easily and dynamically change the grid topology, which is not the case with the other two software applications.

2.3.2 Types of simulations

APREM was initially developed for Monte Carlo-type reliability simulations. It can therefore simulate loops of thousands of iterations with topological (generator or working or non-working line, open or closed circuit breaker, etc.) or parametric (load or generation change) changes at each iteration. A topological validation tool is also included to prevent non-convergence of the electrical calculation in the event of an invalid topology resulting, for example from the islanding of a PQ load following the lost of the source, upstream. This means that APREM is specialized in simulations where parameters must be changed with each iteration. An optimized version for a fixed network topology is currently being developed.

Furthermore, APREM can be coupled, through Java, with an artificial intelligence module (Drools) for simulating smart automations on the grid, such as reconfiguration in the event of an outage.

2.4 Software Comparison

Table 2.1 sets out a qualitative summary of the features of the three software applications. This table provides an overview only. Since the three software applications are still under development/improvement, the features presented are valid at the time this report was drafted, but may be significantly different in 2 to 3 years, and perhaps even next year.

Table 2.1: Summary of Software Features

	OpenDSS	GridLAB-D	APREM
Available models of grid elements	Source, line, generator, capacitor, transformers with n phases and m windings, support for imbalanced loads	Line, transformer, regulator, capacitor, fuse, restart, generator (including solar and wind), support for imbalanced loads	Source (V, I), impedance (RLC), transformer, switch, generator, support for imbalanced loads
Available models of loads	7 models, including: constant PQ, constant Z, constant P-I	Constant Z, constant I, constant P Commercial, industrial and residential loads (Heating/air conditioning, several household appliances)	PQ, Z, possibility of programming P,Q = f(V,I)
Types of calculation	Loadflow, harmonics, time series simulations	Loadflow, time series simulations taking climate and utilisation into account	Loadflow, time series simulations
Types of study	Fluctuating generation, loss calculation, short-circuit currents, storage management, volt and var management	Volt and var management, fluctuating generation, load control, loss calculation	Reliability (Monte Carlo), loss calculation, fluctuating generation, reconfiguration (with Drools)
Integrated controls	Var-Volts, tap changer, storage	Var-Volts, tap changer, load controls depending on electricity cost, peak demand management, possible user programming	To be programmed by the user on a case-by-case basis
Calculation time	Very fast (through Matlab), even faster with direct use of the executable	Fast	Medium
Pre-processing	csv files, Matlab/Excel	csv files, Matlab link possible, but with limited functionality (see forum)	Matlab
Post-processing	csv files, Matlab/Excel, graphic tools	csv files, graphic tools	Matlab, Excel
Documentation	Source forge, Wiki, PDF Documentation [14]	Source forge, Wiki [13]	PDF Documentation (limited)
Code examples	Several examples	A few examples and training offered by PNNL	Limited
Internet forum	Active	Active	No
Initial software goal	Calculation of harmonics	Modelling of residential loads	Reliability (Monte Carlo)
Specific features	May be coupled with other software through a COM DLL interface	Use of climate files (TMY) for solar panels, wind turbines and heating/air conditioning, economic modules	Topological validations, dynamic addition/withdrawal of components
Developer	EPRI	PNNL	Polytechnique Montréal

3 FIRST CASE STUDY: CALCULATION OF DISTRIBUTION LINE LOSSES WITH ADDITION OF DISTRIBUTED GENERATION

Annual distribution line losses are usually calculated with the data from the peak load using a loss factor based on the grid use factor. With the addition of distributed generation, analyzing losses becomes more complex given that generation variation is added onto load variation. When distributed generation is producing power, it reduces the current carried from the sub-station to the load. Depending on the distributed generation capacity installed and the minimum line load, in certain instances, the direction of the current may be reversed and the line may provide power to the sub-station when there is a load dip and high generation. Since losses are proportional to the square of the current, the differential of the power produced and consumed on the line is not the same depending on whether a single annual value is used or a series of values are used over time. That is why using a software application for calculating time series loadflows is required in order to improve the accuracy of the result.

3.1 Methodology

In order to evaluate each tool's performance, a typical case was tested using the three software applications chosen for comparison. This case consists in evaluating the effect on a distribution line's losses of adding distributed generation which is variable over time. Although this case looks at distributed generation, it is also similar to calculating losses in a wind farm's collector network. The problem consists in calculating the annual losses of a distribution line, with or without distributed generation connected at the end of the distribution line. Since distributed generation and load data fluctuate over time, the software must be able to calculate loadflow at each unit of time for which we have data. In this case, there are data for each hour in a year. The software applications must therefore perform 8,760 loadflow calculations. The different parameters of this study are as follows: distributed generation, load and line model. They are described in detail below.

3.2 Parameter Description

3.2.1 Distributed generation Data

The type of distributed generation used is wind power. Solar could also have been used. Distributed generation consists of generation from three Vestas V66 2 MW wind turbines, for a total nominal capacity of 6 MW. The wind speed data come from the National Climate Archive [16] and where measured at Cap Chat in Gaspésie. They were measured at a height of 5 m, and since the average height of the V66 wind turbine is 78 m, we can use the following equation to bring the measured data to the right height [17]:

$$\frac{V_1}{V_2} = \left(\frac{h_1}{h_2}\right)^{0,14}$$

The wind data is therefore multiplied by: (78/5)0.14 = 1.47.

There are therefore 8,760 items of wind data (1 item for each hour from June 1, 2004, to May 31, 2005). Each item of wind data is associated with the Vestas V66 wind turbine power curve [18]:

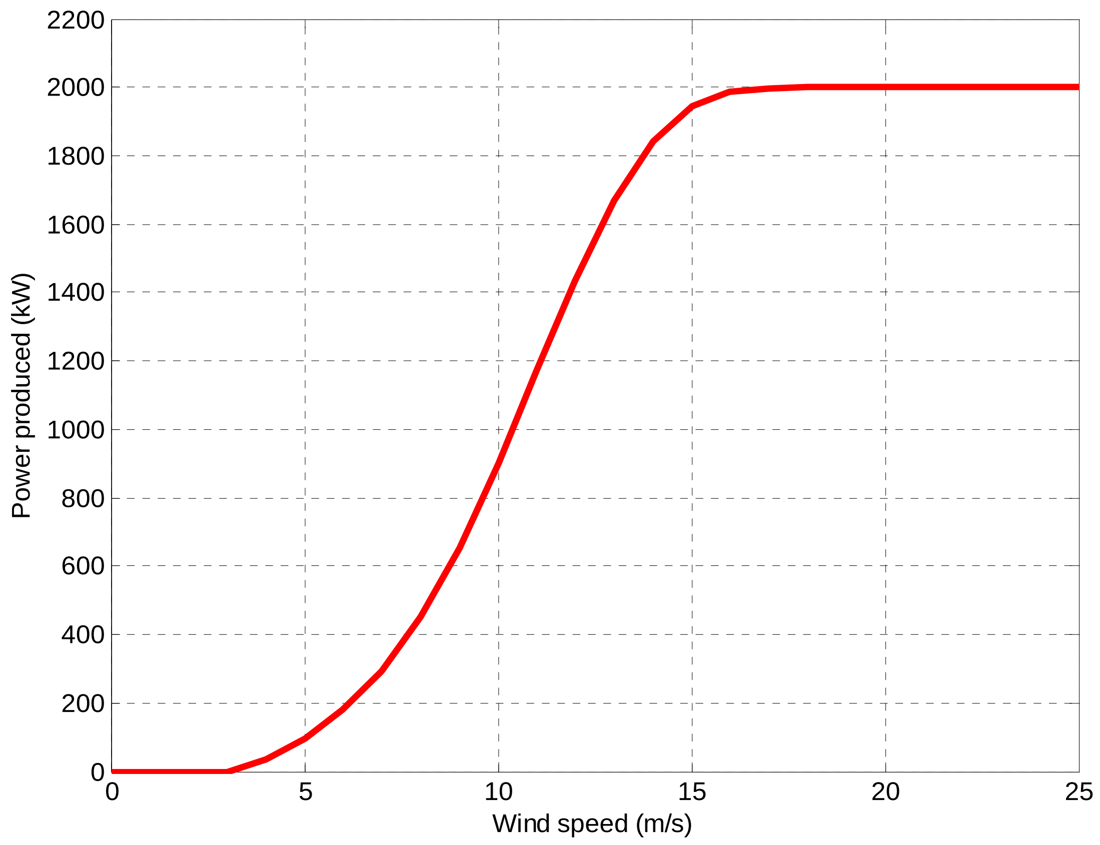

Figure 3.1: V66 wind turbine power curve

Using these parameters, and knowing that there are three 2-MW wind turbines, we are able to obtain a distributed generation value for each hour in a year.

3.2.2 Load Data

The distribution line load data are typical of a line supplying power to electrically heated homes (mostly resistive load), therefore with a power factor of nearly 99%. In order to facilitate the study, we used a unit power factor, although simulation with variable reactive power over time does not pose any problem to the three software applications. The factor for line use is:

$$\frac{P_{moy}}{P_{max[\square]}} = \frac{6,65}{13,37} = 49,8\ \%$$

The maximum load is 13.37 MW and the minimum load is 3.03 MW. Figure 3.2 presents the load profile for a typical winter week when load is at its maximum, and for a typical summer week, when load is at its minimum.

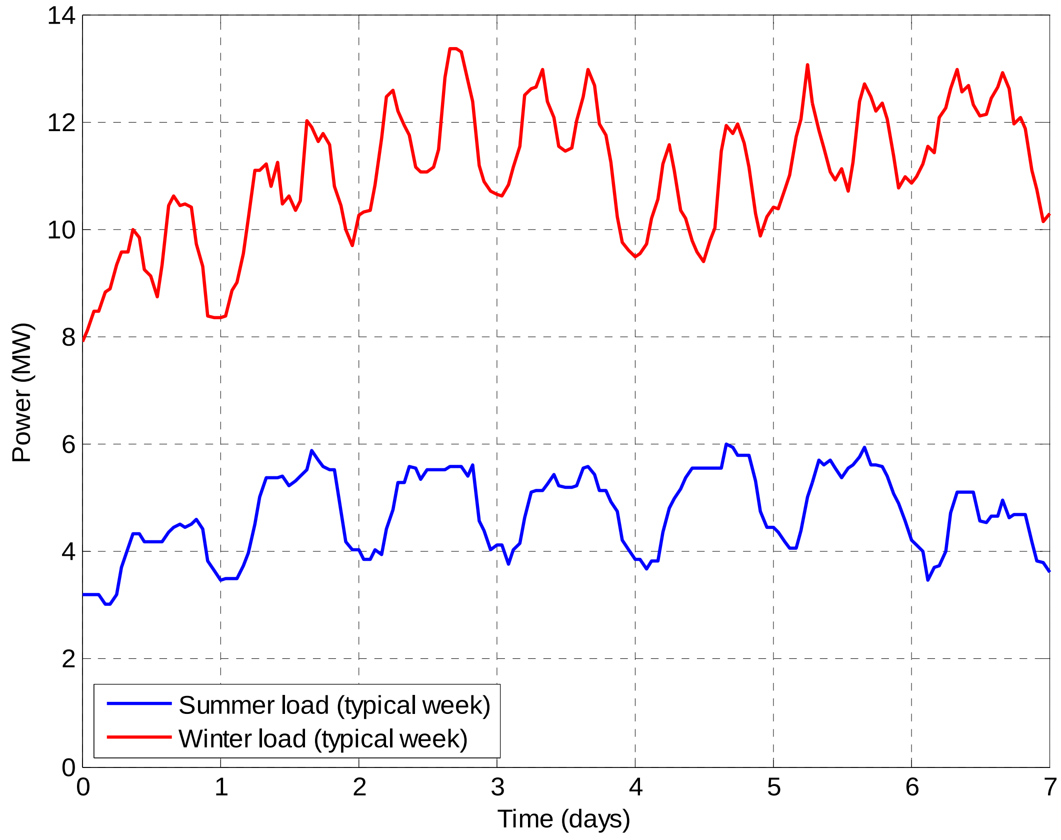

Figure 3.2: Load for a distribution line for a week in summer and winter

3.2.3 Electrical Distribution Line Model

In order to model a distribution line, the sub-station is first represented by a swing bus at one end of the line (25-kV line voltage). The line is 10 km long, with a 477 MCM aluminum conductor (direct sequence impedance 0.116 + j0.395 Ω/km). The three phases are modelled separately, but the coupling between the lines is neglected, because the load is evenly distributed over the three phases (therefore, the grid remains balanced). The coupled lines can, however, be modelled in all three software applications, which allows for easy modelling of imbalanced grids. Decentralized production placed at the end of the line is represented by a PQ bus, with Q = 0, although a reactive power control, defined by the user, can also be used.

For the load, four models were used. The first three models will consist of a load separated into 5, 10 or 20 equal parts along the 10-km line. Distributing a load this way along a line increases the calculation load. In the literature reviewed, we find a simplified model for evaluating losses and a voltage drop on this type of line [19]. This model, which will be our fourth, consists in placing two thirds of the load along a quarter of the line and the last third at the end of the line[1]. By having different models, we are able to observe the difference in software calculation time, based on the number of elements in play. The following are the four line models tested:

1. Load evenly distributed in five equal parts
2. Load evenly distributed in 10 equal parts
3. Load evenly distributed in 20 equal parts
4. Two thirds of the load along a quarter of the line at one third of the load at the end of the line

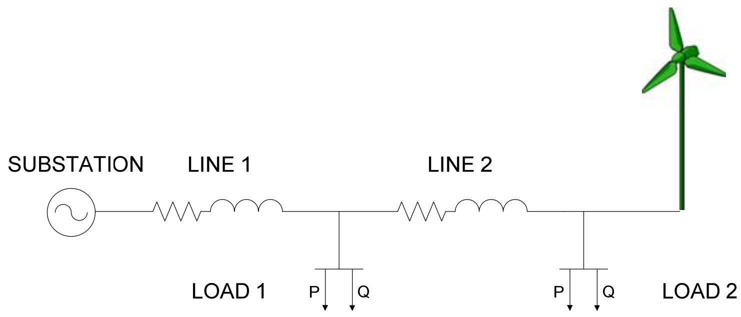

Figure 3.3: Simplified model of the distribution line (4) with decentralized wind-type generation at the end of the line

3.3 Simulation Implementation

Now that the model and its parameters have been defined, we can move to the simulation stage. The two calculations used are as follows:

- Calculation of losses along the distribution line at every hour of the year, without distributed generation

Calculation of losses along the distribution line at every hour of the year, without addition of wind generation at the end of the line

These two calculations are done for each of the four models, and with each of the three software applications. The reader is highly advised to read at this time the codes used with each of the software

[1]This model should be validated, in view of the recent introduction of distributed generation.

applications, presented in the appendices, in order to have an accurate idea on how to use them. This will make it easier to understand and assess the comparative analysis presented in the following section.

3.4 Simulation Case Analysis

The three software applications provide identical results for each of the scenarios reviewed. The results of each of the four line models are, however, different. The electrical results will first be presented and analyzed. The software results and performances will then be discussed. Figure 3.4 presents the results obtained for line model number 4 with APREM, OpenDSS and GridLAB-D. It is of note that the four line models provide similar graphs.

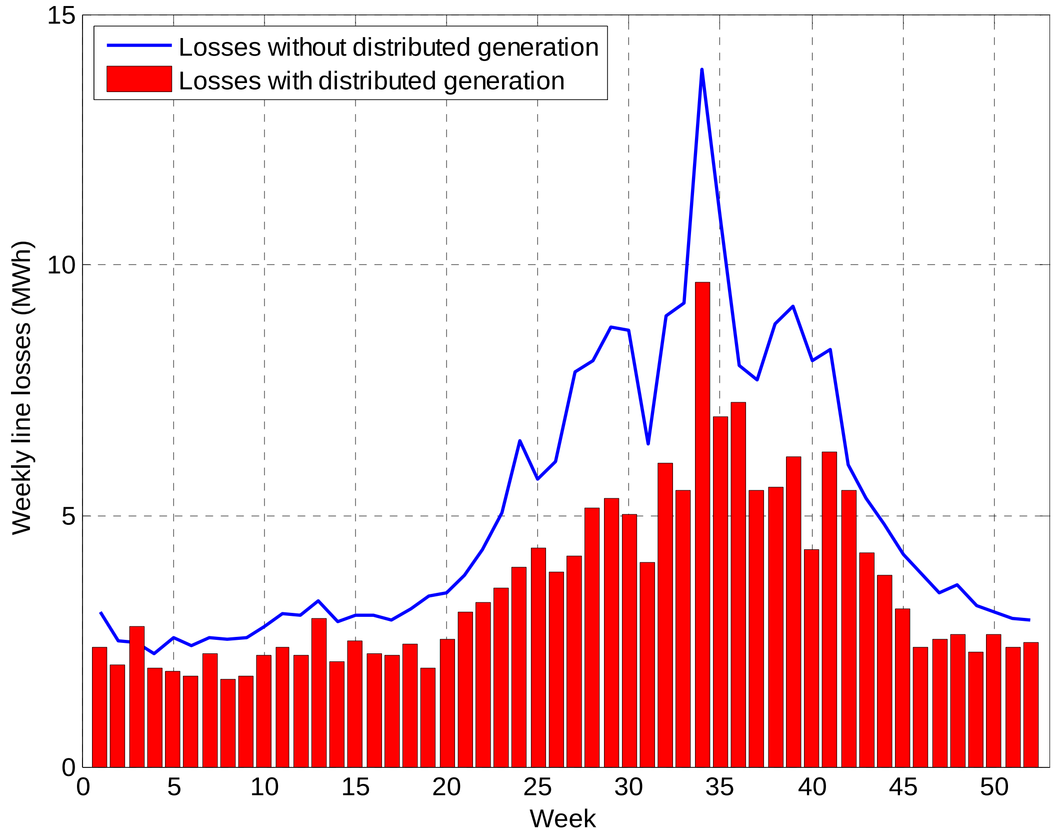

Figure 3.4: Weekly line losses with (red) and without (blue) distributed generation calculated with the line 4 model

Hourly losses were added and combined into weekly losses in order to better observe the effect of wind generation on the line losses. We notice in the graph that there is a week where losses with distributed generation are higher than without. This is due to the fact that, in that week, wind generation is higher than the line load. This therefore means that, at that moment, the distribution line is sending power to the distribution sub-station. Table 3.1 summarizes the results for the four load models.

Table 3.1: Comparison of results based on the different models

	Model 1	Model 2	Model 3	Model 4
Total annual power consumed by the load	58,280 MWh			
Peak line losses	151.4 kW	132.1 kW	122.9 kW	113.9 kW
Off-peak line losses	7.58 kW	6.6 kW	6.2 kW	5.7 kW
Annual distributed generation	14 731 MWh			
6-MW farm capacity factor	28.0 %			
Annual losses with/without distributed generation	351/235 MWh	307/210 MWh	286/199 MWh	265/188 MWh
Annual energy conservation	116 MWh	96.8 MWh	87.1 MWh	77 MWh
Annual savings (10¢/kWh)	$11,600	$9,680	$8,710	$7,700

Table 3.1 shows that the installation of decentralized wind-type generation at the end of a distribution line makes it possible to reduce line losses by about 80 MWh per year, which represents an approximate 30% decrease in annual losses. This reduction in losses includes only average distribution line voltage losses. In fact, loss reduction is still more considerable if we consider that, on an annual basis, most of the losses incurred on the transmission network are avoided. However, as can be seen in figure 3.4, medium voltage losses increased during periods of low load since, at that moment, distributed generation is higher than the load and the line sends power to the grid. In order to model this unusual situation and its impact upstream, the sub-station, regional transmission network or other distribution lines must be modelled. If needed, this type of calculation may also be done using the three software applications presented.

3.5 Comparative Software Analysis

3.5.1 Results

Table 3.2 sets out a comparison of the calculation time of the different software applications based on the line model used. Model 4 is the one with the fewest elements, and model 3 is the one with the most.

Table 3.2: Calculation time by software application

Line model	Number of elements	OpenDSS	GridLAB-D	APREM
1	15 loads, 15 lines	55 seconds	5 minutes	26 minutes
2	30 loads, 30 lines	1 min 50 seconds	8 minutes	53 minutes
3	60 loads, 60 lines	3 min 30 seconds (Coupled with Matlab)	13 minutes	115 minutes
4	6 loads, 6 lines	30 seconds (Coupled with Matlab)	2 minutes	10 minutes

The calculation time for this case study increases linearly with the number of elements. Thus, by doubling the number of loads, the calculation time was doubled. The previous table shows the impressive performance of the OpenDSS software in terms of calculation time. Moreover, these times are even shorter if the executable is used directly instead of passing through Matlab. GridLAB-D's performance is slightly slower than that of OpenDSS (by a factor of about 4). Moreover, the calculation time of OpenDSS includes the analysis of the results, which is not the case of GridLAB-D. As for APREM, the calculation time is longer by a factor of 5 to 20 in relation to the other software applications.

3.5.2 Discussion

The three software applications made it possible to simulate distributed generation, with calculation times that differ from one case to another. As concerns the distribution line modelling, the three software applications are comparable. However, the syntax and calculation time vary from one software application to another. For this type of simulation, OpenDSS is certainly the most appropriate because it was partially designed for this type of calculation. GridLAB-D performs reasonably well given that it was first and foremost designed for residential load and not strictly grid calculations. However, its use is more complex than that of APREM or OpenDSS. As for APREM, its relative slowness in doing this type of calculation is compensated by its ease of use, as program writing and results analysis is easier than with the two other software applications. Of course, this is only valid if the user is already familiar with Matlab.

It is important to note that the authors' prior in-depth understanding of Matlab influenced their assessment. Notably, the fact that OpenDSS and APREM were tested using Matlab as an interface gave them a comparative edge over GridLAB-D. Table 2.1 also shows that OpenDSS and GridLAB-D have good quality documentation and enough examples to be able to help the beginner user perform simulations easily.

4 Third Case Study: Calculation of Annual Energy Conservation Following Conservation Voltage Control

4.1 Introduction to the Case Study

The second case study relates to voltage control or CVR (Conservation Voltage Reduction). This application[2] aims at optimizing voltage and reactive power compensation to save energy and line losses.

This type of study requires specific capacities on the part of the power grid analysis software applications. First, the software must be able to model the different types of load. The ZIP model is the one most commonly used. This model makes it possible to define the different load behaviours when dealing with a voltage variation. The letter Z represents constant-impedance loads (when voltage drops, the current drops), the letter I represents constant-current loads (when voltage drops, the current remains constant), and the letter P represents constant-power loads (when voltage drops, the current increases).

This model has a few limits, in particular when the time comes to model thermostat-controlled loads (water heater, heating, air conditioning). In the case of these loads, when the voltage drops, the current drops. They therefore behave like constant-impedance loads. However, these devices must operate longer in order to be able to provide the total required amount of power, which risks reducing the diversity of the load and lead to peak load increase. Impact analysis therefore requires a tool for time series simulations. Since GridLAB-D is the only one of the three software applications to be able to implicitly model this phenomenon, it is of great importance for this type of study. PNNL used this software to evaluate the CVR potential in the United States [11].

In order to be able to compare the three software applications, only the ZIP model will be used.

4.2 Description of the Parameters

This case study consists of a simplified 30-buses distribution line with three-phase and single-phase branches, as well as single-phase loads. The loads – 21 in total – are represented by a ZIP model upstream from the transformers. There is therefore a Z load, I load and P load at each load bus. It is assumed that the voltage is to be kept at a minimum value of 114 V upstream from the transformer. This therefore represents a phase voltage of 13,680 V on the network side assuming a turns ratio of 14.4 kV / 120-240 V. The goal is therefore to get as close to this voltage as possible without being lower. We assume that the nominal voltage at the sub-station is 25 kV and that we can decrease this voltage in 200-V increments. The goal is therefore to slave the minimum voltage at the sub-station in order to have a voltage value close to 13,680 V on the bus where the voltage is lowest.

[2] See Hydro-Québec's CATVAR project [20]

The advantage of the three software applications is to be able to perform time series simulations. We are therefore going to vary the loads over the distribution network based on the values provided in the previous case study. Accordingly, the total power demanded by the loads is changed with each one-hour time step. Each of these power amounts is equally distributed among the Z, I and P loads. In order for the study to be more authentic, this distribution must also be changed according to the times and seasons over a year. These results will be the basis for estimating what amount of energy can be conserved annually with this type of control and what is the impact on peak load. It is important to specify that the goal of this study is merely to provide an overview of the software applications' possibilities. It does not on its own provide for a validation of the type of control reviewed, at least not without first accurately modelling the energy needs of the loads at all periods of the year.

Figure 4.1 presents the distribution network diagram used as well as the result provided by the OpenDSS circuit visualization tool. The wider the line, the bigger the current circulating in this part of the network. We note that the network consists of a main three-phase branch and secondary single-phase branches, to which the loads are connected.

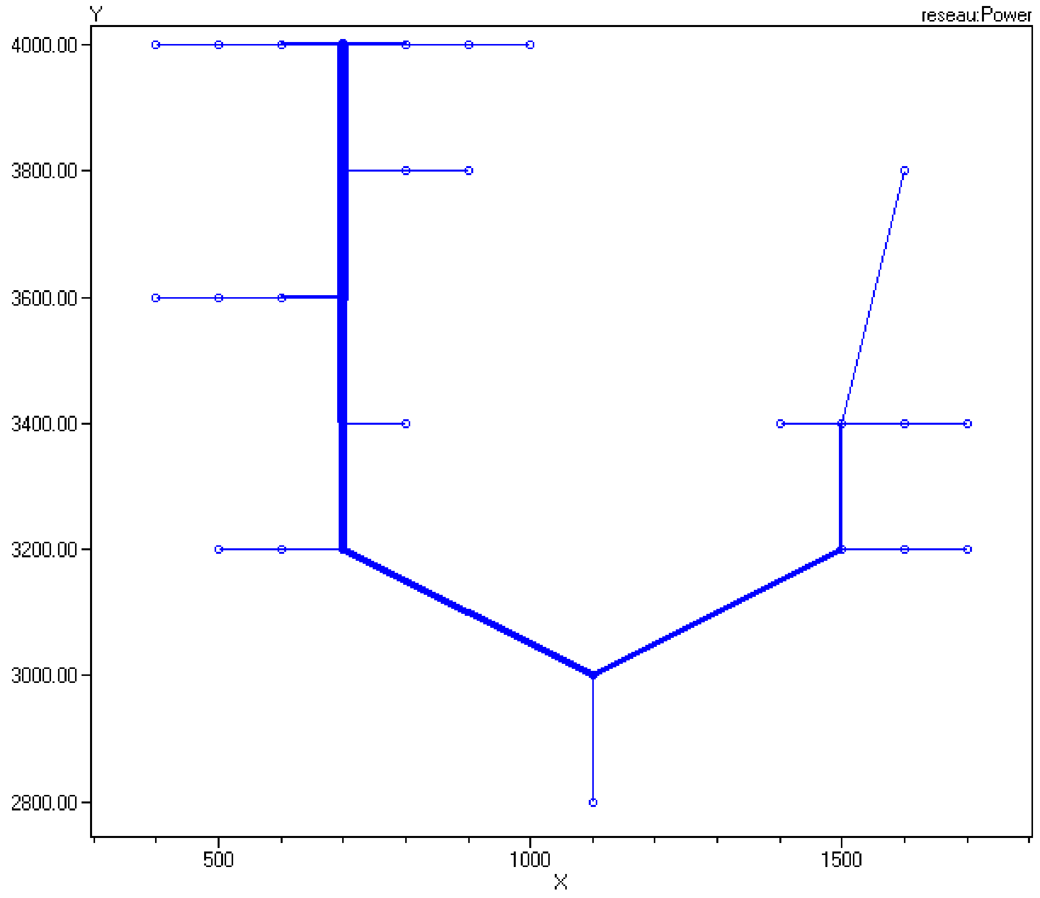

Figure 4.1: Diagram of the distribution network as represented by the OpenDSS visualization tool

4.3 Results

Table 4.1 provides an overview of the calculation time for each software application. Since the circuits are bigger and the calculation is more complicated (the right voltage needs to be found for each time step), the calculation times are longer than those found in the previous case study. The performance of each software application is fairly identical to those previously found. The table shows that it is the OpenDSS that performs best, even if used with Matlab and not simply with the executable.

Table 4.1: Calculation time by software application for an annual simulation on a 30-buses and 63-load circuit

OpenDSS	GridLAB-D	APREM
3 minutes (coupled with Matlab)	12 minutes	200 minutes

As for the calculation results, a 3.6% decrease in the total energy consumed by the loads is obtained. The total annual energy without voltage control is 55 GWh and the total energy with control is 53 GWh. As previously mentioned, these results do not lead to a conclusion on the validity of this type of voltage control, but rather a comparison of software application performance. To achieve this, more accurate data and load model would be needed.

4.4 Discussion

With these three software applications the impact of the conservation voltage control on the annual energy conservation of a distribution line can be simulated. Given the complexity and length of this calculation, we notice an even wider gap between APREM's calculation time performance and that of the other software. Voltage regulation can be easily programmed with the three software applications. With the GridLAB-D, there is even already a device that does this operation directly. Moreover, if we had wanted to integrate the thermostat-controlled circuits, only the GridLAB-D would have this function already built-in. This study has not integrated var control (capacitor control). However, since capacitor modelling is already built into the three software applications, the integration of this control could have been done by the three applications. The problem and control would have become more complex, and the calculation time longer.

CONCLUSION

This report presents three recent power grid simulation tools for conducting time series studies. The common feature among these software applications is that they are open source code and available for free on-line or on request. These tools make loop simulations relatively easy when compared to the existing tools, including, among other things, long-lasting time series simulations. Each of these three software applications offers different possibilities. GridLAB-D is more specialized for studies that include modelling residential loads, APREM is specialized in reliability studies, and OpenDSS is specialized in annual time series studies.

The conclusions of this report are based on two cases that can be simulated with the three software applications. Each application performed differently and their use has been characterized. Albeit the three applications were show to be capable of handling both studies, their calculation time and ease of use vary considerably. For the type of studies presented, OpenDSS had the best performance in terms of calculation time. However, for ease of use, APREM stands out from the rest. In the case of GridLAB-D, the type of study presented did not do justice to this software's possibilities. In fact, if residential load modelling had been mandatory, it is the only software that would have solved the problem in a reasonable time frame for the user, without having to program new functions oneself. Since these software applications are still being developed, it will be interesting in the future to follow their respective developments and evaluate their impact on the commercial software market.

REFERENCES

[1] F. Milano, L. Vanfretti, “State of the Art and Future of OSS for Power Systems,” 2009 IEEE PES General Meeting, Calgary, July 2009.

[2] R.C. Dugan, R. F. Arritt, T. E. McDermott, S. M. Brahma, K. Schneider, “Distribution System Analysis to Support the Smart Grid,” 2010 IEEE PES General Meeting, Minneapolis, July 2010.

[3] C. Abbey, “Evolution of network simulation tools to facilitate DER deployment,” PowerPoint presentation, 3rd International Conference on Integration of Renewable and Distributed Energy Resources, Nice, France, December 2008; www.3rdintegrationconference.com/pres/38_Abbey.pdf, accessed July 11, 2011.

[4] C. Kwok, F. De Léon, A. Morched, CYME International “Survey of Studies and Anlysis Tools Used for Assessment of Distributed Generation Integration in Canadian Distribution Systems,” CanmetENERGY report number 2006-070, May 2006

[5] SourceForge.net OpenDSS, http://sourceforge.net/projects/electricdss/ accessed July 11, 2011.

[6] R.C. Dugan, “Open Distribution Simulations System Workshop: Using Open DSS for smart distribution simulations,” EPRI PQ Smart Distribution 2010 Conference and Exhibition, Québec, June 14-17, 2010.

[7] A. Maitra, K.S. Kook, J. Taylor, A. Giumento, “Grid Impacts of Plug-in Electric Vehicles on Hydro Quebec’s Distribution System,” 2010 IEEE PES Transmission and Distribution Conference and Exposition, New Orleans, April 2010.

[8] SourceForge.net GridLAB-D, http://sourceforge.net/projects/gridlab-d/, accessed July 11, 2011.

[9] R.T. Guttromson, D.P. Chassin, S.E. Widergren, “Residential Energy Resource Models for Distribution Feeder Simulation,” Power Engineering Society General Meeting, IEEE, 2003.

[10] D.J. Hammerstrom, R. Ambrosio, T.A. Carlon, J.G. DeSteese, G.R. Horst, R. Kajfasz, L.L. Kiesling, P. Michie, R.G. Pratt, M. Yao, J. Brous, D.P. Chassin, R.T. Guttromson, O.M. Jarvegren, S. Katipamula, N.T. Le, T.V. Oliver, and S.E. Thompson. “Pacific Northwest GridWise™ Testbed Demonstration Projects; Part I. Olympic Peninsula Project”. PNNL-17167, Pacific Northwest National Laboratory, Richland, WA, October 2007.

[11] K.P. Schneider, JC Fuller, FK Tuffner, and R Singh. 2010. Evaluation of Conservation Voltage Reduction (CVR) on a National Level . PNNL-19596, Pacific Northwest National Laboratory, Richland, WA, July 2010. http://www.pnl.gov/main/publications/external/technical_reports/PNNL-19596.pdf accessed July 11, 2011

[12] F. Sirois, APREM – Analyse paramétrique des réseaux électriques avec Matlab, École polytechnique de Montréal, http://www.professeurs.polymtl.ca/f.sirois/These_F_Sirois.pdf accessed July 11, 2011

[13] J. J. Allemong R J. Bennon P. W. Selent, “Multiphase Power Flow Solutions Using EMTP and Newtons Method,” IEEE Transactions on Power Systems, Vol. 8, No. 4, November 1993, pp. 1455-1462.

[14] SourceForge.net OpenDSS, Main Page, http://sourceforge.net/apps/mediawiki/electricdss/index.php?title=Main_Page Accessed, July 2011.

[15] SourceForge.net GridLAB-D, Main Page, http://sourceforge.net/apps/mediawiki/gridlab-d/index.php?title=Main_Page accessed, July 2011.

[16] Environment Canada, National Climate Data and Information Archive http://climate.weatheroffice.gc.ca/Welcome_e.html accessed, July 2011

[17] Natural Resources Canada, Retscreen International, “Clean Energy Project Analysis,” 3rd Edition, January 2006, p. Eole 16.

[18] Natural Resources Canada, RETScreen International – Product Data, http://www.retscreen.net/ang/d_data_p.php accessed, July 2011

[19] W.H. Kersting, “Distribution System Modeling and analysis,” 2nd edition, CRC Press, p.57.

[20] Régie de l’énergie, “Projet de réduction de la consommation énergétique par une gestion optimisée de la tension du réseau de distribution – Projet CATVAR,” Demande R-3746-2010, Hydro-Québec Distribution, October 22, 2010, http://internet.regie-energie.qc.ca/Depot/Projets/84/Documents/R-3746-2010-B-0004-DEMANDE-PIECE-2010_10_27.pdf accessed, July 2011

APPENDIX 1 – EXAMPLE OF CODES WITH OPENDSS

A.1 – OpenDSS code using the executable for the first case study

As previously mentioned, OpenDSS can be used in its own environment or with software applications such as Matlab or Excel. In this appendix, we provide an example with its executable and an example with Matlab. We must first create our circuit with the executable (here called “reseau”):

```
new circuit.network
```

Then, the load data are entered. These data come from a CSV file, i.e., a series of numbers separated by commas. There is one item of data for each hour of the year, i.e., 8,760 in total. These data are loaded into a “loadshape” object, which is first given a name, in this case “essaiA.” Of note as well is that the values change every hour:

```
new loadshape.essaiA Npts=8760 Csvfile=Donnees0A.csv hour=1
new loadshape.essaiB Npts=8760 Csvfile=Donnees0B.csv hour=1
new loadshape.essaiC Npts=8760 Csvfile=Donnees0C.csv hour=1
new loadshape.essai2A Npts=8760 Csvfile=Donnees1A.csv hour=1
new loadshape.essai2B Npts=8760 Csvfile=Donnees1B.csv hour=1
new loadshape.essai2C Npts=8760 Csvfile=Donnees1C.csv hour=1
```

The same is done for the wind generation values:

```
new loadshape.wind Npts=8760 Csvfile=testwind.csv hour=1
```

Then, the different grid elements need to be created. First, the sources are created, using an object, “vsource”, which is named at random. In this case, a three-phase imbalanced circuit is assumed; therefore, three sources are used and the Phases=1 parameters are chosen. The “vsource” objects are connected to the ground and to a bus. The Bus name must therefore be specified. The nominal voltage must also be provided, which here is 14.43 kV (25 kV of line voltage):

```
new vsource.source Bus1=B2A Phases=1 basekv=14.43 pu=1
new vsource.sourceB Bus1=B2B Phases=1 basekv=14.43 pu=1
new vsource.sourceC Bus1=B2C Phases=1 basekv=14.43 pu=1
```

Second, the lines are created via the “Line” object. A Line object is connected between two Buses that the user must specify. Finally, the user enters the line’s R1 and X1 electrical parameters:

```
new Line.L1A Bus1=B2A Bus2=B3A Phases=1 R1=0.29 X1=0.9875
new Line.L1B Bus1=B2B Bus2=B3B Phases=1 R1=0.29 X1=0.9875
new Line.L1C Bus1=B2C Bus2=B3C Phases=1 R1=0.29 X1=0.9875

new Line.L2A Bus1=B3A Bus2=B4A Phases=1 R1=0.87 X1=2.9625
new Line.L2B Bus1=B3B Bus2=B4B Phases=1 R1=0.87 X1=2.9625
new Line.L2C Bus1=B3C Bus2=B4C Phases=1 R1=0.87 X1=2.9625
```

The last objects to be created are the loads via the "Load" object. Like the sources, this object is connected between the ground and a bus. The nominal voltage, nominal capacity, power factor and load type (constant power, constant current, constant resistance, etc.) must be specified. In our case, since we are looking for an annual simulation, we use the "Yearly" parameter to specify the annual value. In this case, the "Yearly" parameter is equal to essaiA, which is a previously specified "Loadshape" type object that contains the annual load values:

```
new Load.C1A Bus1=B3A Phases=1 Kv=14.43 Kw=10000 Pf=1 Model=6 Yearly=essaiA
new Load.C1B Bus1=B3B Phases=1 Kv=14.43 Kw=10000 Pf=1 Model=6 Yearly=essaiB
new Load.C1C Bus1=B3C Phases=1 Kv=14.43 Kw=10000 Pf=1 Model=6 Yearly=essaiC

new Load.C2A Bus1=B4A Phases=1 Kv=14.43 Kw=10000 Pf=1 Model=6 Yearly=essai2A
new Load.C2B Bus1=B4B Phases=1 Kv=14.43 Kw=10000 Pf=1 Model=6 Yearly=essai2B
new Load.C2C Bus1=B4C Phases=1 Kv=14.43 Kw=10000 Pf=1 Model=6 Yearly=essai2C
```

The annual wind generation values can be specified the same way with the "Generator" object:

```
new Generator.G1A Bus1=B4A Phases=1 Kv=14.43 Kw=10000 Pf=1 Model=1 Yearly=wind
new Generator.G1B Bus1=B4B Phases=1 Kv=14.43 Kw=10000 Pf=1 Model=1 Yearly=wind
new Generator.G1C Bus1=B4C Phases=1 Kv=14.43 Kw=10000 Pf=1 Model=1 Yearly=wind
```

Once the model is built, the parameters to be recorded need to be specified. The "energymeter" object is used, for which we specify the object whose parameters we wish to measure:

```
new energymeter.m3A Line.L1A LocalOnly=yes
new energymeter.m4A Line.L2A LocalOnly=yes
new energymeter.m3B Line.L1B LocalOnly=yes
new energymeter.m4B Line.L2B LocalOnly=yes
new energymeter.m3C Line.L1C LocalOnly=yes
new energymeter.m4C Line.L2C LocalOnly=yes
```

The simulation parameters must be specified:

```
set mode=yearly
set casename= essainumero1
set year=1
set demandinterval=true
Set overloadreport=true
```

To finish, the simulation needs to be launched and the results saved:

```
solve
edit energymeter.m3A action=save
edit energymeter.m4A action=save
edit energymeter.m3B action=save
edit energymeter.m4B action=save
edit energymeter.m3C action=save
edit energymeter.m4C action=save
```

The simulation results are recorded in an essainumero1.csv file, as specified in the simulation parameters. The case previously presented is easily solved with OpenDSS. However, the problem is with respect to the data preparation (pre-processing) and post-simulation results analysis (post-processing). The simulations were done with Matlab in order to facilitate these two steps.

A.2 – OpenDSS Code using Matlab

First, run OpenDSS through Matlab using this command:

```
[DSSStartOK, DSSObj, DSSText] = DSSStartup;
```

The DSSStartup function is as follows:

```
function [Start,Obj,Text] = DSSStartup
% Function for starting up the DSS
%instantiate the DSS Object
    Obj = actxserver('OpenDSSEngine.DSS');
%Start the DSS.   Only needs to be executed the first time w/in a
%Matlab session
    Start = Obj.Start(0);
% Define the text interface
    Text = Obj.Text;
```

The circuit already defined in OpenDSS is then loaded. This file (cir1.dss) contains the sources, lines, loads, but not the loadshape:

```
DSSText.command='Compile (C:\Users\MA\Desktop\OpenDSS_7_0_1\Calcul\cir1.dss)';
```

The interface variables must then be prepared:

```
% Set up the interface variables
    DSSCircuit=DSSObj.ActiveCircuit;
    DSSSolution=DSSCircuit.Solution;
```

Then, the losses at each hour of the year are simply calculated with a for. It is assumed here that the different Load vectors contain the load values for all times of the year:

```
for i = 1:8760
    DSSText.command=['Edit Load.C1A Kw=' num2str(ChargeA1(i))];
    DSSText.command=['Edit Load.C1B Kw=' num2str(ChargeB1(i))];
    DSSText.command=['Edit Load.C1C Kw=' num2str(ChargeC1(i))];
    DSSText.command=['Edit Load.C2A Kw=' num2str(ChargeA2(i))];
    DSSText.command=['Edit Load.C2B Kw=' num2str(ChargeB2(i))];
    DSSText.command=['Edit Load.C2C Kw=' num2str(ChargeC2(i))];
    DSSSolution.Solve;
    A = DSSCircuit.LineLosses;
    Pertes = [Pertes A(1)*1000];
end
```

The LineLosses function makes it possible to calculate the losses of all of the circuit's Line objects. The Losses vector contains the line loss values for all hours of the year. These data can then be easily analyzed and processed, and graphs easily created, using the Matlab environment.

APPENDIX 2 – EXAMPLE OF CODES WITH GRIDLAB-D

For simulations with GridLAB-D, text files with a .glm extension (GridLAB-D model) must first be created. This file must then be run using a control window with the following text, making sure to be in the GridLAB-D program directory:

```
gridlabd  nameoffile.glm
```

The base for using this software is the writing of text files. A wiki or examples available on-line can be used to ensure correct syntax. One code example is provided below. To start, we need to load the required modules. In this case, since we are simulating power flow, the “powerflow” module must be loaded. The “tape” module must be used to save or load data.

```
module powerflow;
module tape;
```

Then, the time parameters for the simulation need to be established. In this case, this being an annual simulation, the start and end times are a year apart:

```
clock {
      timezone EST+5EDT;
      starttime '2000-01-01 0:00:00';
      stoptime '2001-01-01 0:00:00';
}
```

The next step is to define the electrical parameters. First, the types of conductors are noted:

```
object overhead_line_conductor {
      name OH100;
      geometric_mean_radius 0.0244;
      resistance 0.306; //Ohm par mile
}
object overhead_line_conductor {
      name OH101;
      geometric_mean_radius 0.00814;
      resistance 0.306;
}
```

Then, the distance between conductors:

```
object line_spacing {
      name LS200;
      distance_AB 5.0;
      distance_BC 5.0;
      distance_AC 5.0;
      distance_AN 5.0;
      distance_BN 5.0;
      distance_CN 5.0;
}
```

The line configuration can be specified using the previous parameters:

```
object line_configuration {
      name LC300;
      conductor_A OH100;
      conductor_B OH100;
      conductor_C OH100;
      conductor_N OH101;
      spacing LS200;
}
```

The nodes are then defined. In our case, there are three nodes, including a balancing node:

```
object node {
      name Node1;
      bustype SWING;
      phases ABC;
      nominal_voltage 25000.0;
      }
object node {
      name Node2;
      phases ABC;
      nominal_voltage 25000.0;
      }
object node {
      name Node3;
      phases ABC;
      nominal_voltage 25000.0;
      }
```

The electrical lines connecting the nodes are then defined:

```
object overhead_line {

      name Link12;

      phases A|B|C;

      from Node1;
      to Node2;
      length 8200;
      configuration LC300;
      nominal_voltage 25000.0;
}
object overhead_line {
      name Link23;
      phases A|B|C;
      from Node2;
      to Node3;
      length 24600;
      configuration LC300;
      nominal_voltage 25000.0;
}
```

The next step is to specify the loads for each hour of the year. In order to do this, the “player” object is used and the data from a CSV file are loaded. These CSV files are presented in two columns, one for the temporary data, and the other for the load values at that time.

```
object load {
      name Load1;
      phases ABC;
      parent Node2;
      object player {
            property constant_power_A;
            file loadA1.csv;
      };
      object player {
            property constant_power_B;
            file loadB1.csv;
      };
      object player {
            property constant_power_C;
            file loadC1.csv;
      };
  nominal_voltage 25000.0;
}
object load {
      name Load2;
      parent Node3;
      phases ABC;
      object player {
            property constant_power_A;
            file loadA2.csv;
      };
      object player {
            property constant_power_B;
            file loadB2.csv;
      };
      object player {
            property constant_power_C;
            file loadC2.csv;
      };
  nominal_voltage 25000.0;
}
```

The wind generation data can be loaded the same way, as long as it is modelled by a negative PQ load.

```
object load {
      name Load3;
      parent Node3;
      phases ABC;
      object player {
            property constant_power_A;
            file WindA.csv;
      };
      object player {
            property constant_power_B;
            file WindA.csv;
      };
      object player {
            property constant_power_C;
            file WindA.csv;
      };
  nominal_voltage 25000.0;
}
```

The last step is to choose the data to be saved using a “recorder” object. In this case, we choose to save the line losses, specifying in which file the results will be saved.

```
object recorder{
name MeterCorderXA;
parent Link12;
property power_losses, power_in, power_out;
file pertes2A.csv;
}
object recorder{
name MeterCorderXB;
parent Link23;
property power_losses;
file pertes2B.csv;
}
```

The last step consists in running this file using the control window, as previously specified.

APPENDIX 3 – EXAMPLE OF CODES WITH APREM

The APREM software works in the Matlab environment and uses Matlab's object-oriented programming functions. A description of the code used to solve the first case study is provided below. This is the code to be used in an m file. First, a circuit needs to be created (here it is called "reseau"):

```
reseau = cCircuit();
```

Then, the circuit source is created. It is given then name "V1." It is connected between the "B1" node and ground ("0"). It has a voltage of 25 kV and an angle of 0. It will be the balancing bar.

```
reseau.addVsrc('V1', {'B1', '0'}, 25000, 0);
```

The next step is to create RL branches representing the lines. The branches are called L1 and L2. They are connected between nodes "B1" and "B2", and "B2" and "B3." The line impedances are 0.29 + j0.9875 Ω and 0.87 + j2.9625Ω, respectively (see section 2.2.3).

```
reseau.addImp('L1', {'B1', 'B2'}, 0.29 + 0.9875*1i);
reseau.addImp('L2', {'B2', 'B3'}, 0.87 + 2.9625*1i);
```

The PQ1 and PQ2 loads are then created. The load is a PQ type with Q = 0 and the power values contained in vectors Charge1 and Charge2, previously defined in Matlab. The nominal voltage is 25 kV.

```
reseau.addPQ('PQ1', {'B2', '0'}, Charge1(1), 0, 25000);
reseau.addPQ('PQ2', {'B3', '0'}, Charge2(1), 0, 25000);
```

Losses are calculated using the following loop:

```
Pertes = [];
for i = 1:8760
      reseau.set('PQ1', 'P', Charge1(i)); % Modification of the load
      reseau.set('PQ2', 'P', Charge2(i));
      reseau.solve(); % solve
      P1 = real(reseau.get('L1', 'S')); % Losses of L1
      P2 = real(reseau.get('L2', 'S')); % Losses of L2
      Pertes = [Pertes P1+P2]; % Recording of the losses
end
```

The same calculation must then be done, this time adding the distributed generation. This generation is connected between node "B3" and ground ("0"). The power at every hour of the year is pre-saved in the "Puissances" vector. A unit power factor is assumed for the source, and as electricity is sent to the grid, the power is negative. The voltage is 25 kV.

```
reseau.addPQ('Peol', {'B3', '0'}, -Puissance(1), 0, 25000);
```

The calculation must then be redone for each hour of the year. However, this time, in addition to varying the loads, the wind generation is also varied.

```
Pertesdistribuees = [];
for i = 1:8760
      reseau.set('PQ1', 'P', Charge1(i));
      reseau.set('PQ2', 'P', Charge2(i));
      reseau.set('Peol', 'P', -Puissance(i));
      reseau.solve();
      P1 = real(reseau.get('L1', 'S'));
      P2 = real(reseau.get('L2', 'S'));
      Pertesdistribuees = [Pertesdistribuees P1+P2];
end
```

The user can then analyze the results and draw graphs using the Matlab functions.