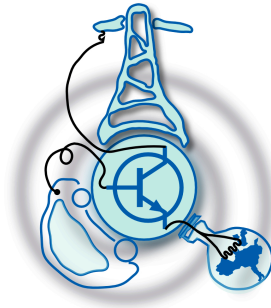


Web Based Tool to Analyze Unbalanced Distribution Systems

by
Aitor Fuillerat García



Submitted to the Department of Electrical Engineering, Electronics,
Computers and Systems
in partial fulfillment of the requirements for the degree of
Master of Science in Electrical Energy Conversion and Power Systems
at the
UNIVERSIDAD DE OVIEDO

November 2017

© Universidad de Oviedo 2017. All rights reserved.

Author

Certified by

Pablo Arboleya
Associate Professor
Thesis Supervisor

Certified by

Islam El-Sayed
Post-Doc Researcher
Thesis Supervisor

Web Based Tool to Analyze Unbalanced Distribution Systems

by

Aitor Fuillerat García

Submitted to the Department of Electrical Engineering, Electronics, Computers and
Systems

on November 15, 2017, in partial fulfillment of the
requirements for the degree of

Master of Science in Electrical Energy Conversion and Power Systems

Abstract

This project seeks to be a interactive web tool that applies advanced technologies as visual analytics and data driven documents (D3), that allows the user to evaluate the impact of the distributed generation in distribution grids. It consider unbalanced power flows and it has the possibility to include residential photovoltaic generation and energy storage elements with different control strategies.

The features of this tool goes from, the possibility of make a better approach in the grid infrastructure plannification having into account the different levels of distributed renewable generation installed, to the economic analysis impact of this distributed generation due to the fact that, it is able to determine power peaks and energy at the transformer level. In addition to, this tool will be useful for dimensioning photovoltaic installations and storage elements.

Thesis Supervisor: Pablo Arboleya

Title: Associate Professor

Thesis Supervisor: Islam El-Sayed

Title: Post-Doc Researcher

Acknowledgments

Special thanks to Pablo Arboleya and Islam El-Sayed whom made, that this master thesis, could be made in the best conditions possible and for helping me every time I needed. Also, I would like to thank the whole LEMUR team. Professors, researchers and students for having a perfect working environment. Specially, I would like to thank Alejandro Gancedo, Tania Cuesta and Andres Fernandez that were part of this project too, with complementary works, and to the master course coordinators, Jorge Garcia and Cristina Gonzalez-moran, who were always available for helping me with any issue.

Finally, I would like to thank my EECPS mates, my parents for the unconditional support (also economical), my girlfriend Ana that has been always supporting me during these months and my friends that have missed me lately.

Contents

1	Introduction	13
2	Residential Energy Demand, Generation and Storage	15
2.1	Distribution System	15
2.2	Residential Energy Demand	16
2.2.1	Nearly-Zero Energy Buildings.	18
2.3	Residential Distributed Generation	18
2.4	Residential Energy Storage	21
3	Power Flow	23
3.1	Grid Modelling	23
3.2	Formulation	24
3.3	Bus types	26
3.4	Methods To Solve Power Flow	26
3.4.1	Gauss-Seidel Method	26
3.4.2	Newton-Raphson Method	27
3.5	EPRI OpenDSS	27
3.5.1	OpenDSS	28
4	Web tool: FloWI	35
4.1	Front-end Development	37
4.1.1	Design and programming	38
4.1.2	Upload files	38

4.1.3	Visualization of results through on click event	39
4.1.4	Visualization of results through slider event	41
4.2	Functionality development	43
4.2.1	Read Input Data	43
4.2.2	Draw the map	44
4.2.3	Draw the graphics	45
4.2.4	Perform the visualization simulation in the map	45
5	IEEE European Low Voltage Test Feeder	47
5.1	Introduction	47
5.2	Description	48
6	Study Cases and Results	51
6.1	Node	52
6.2	Line	57
6.3	Global variables	64
7	Conclusions	75
8	Bibliography	77
A	HTML Source Code	79
B	CSS Source Code	91
C	JavaScript Source Code	99

List of Figures

2-1	Power system components.	16
2-2	Unbalanced and radial distribution.	17
2-3	Generation power profiles.	20
2-4	Generation and storage power profiles.	22
3-1	Power system components.	25
3-2	Storage default mode.	32
3-3	Storage default mode.	34
4-1	FloWI Initial Page.	36
4-2	FloWI before files were uploaded.	36
4-3	Bus Graphics Layout.	39
4-4	Illustration of how power demanded is represented in the map.	42
6-1	Node 36 load power profile.	53
6-2	Node 36 consumed and generated power profile.	53
6-3	Node 36 net power that flows through the feeder.	54
6-4	Node 36 stored energy profile and I/O power.	54
6-5	Node 36 power flowing through the feeder.	55
6-6	Node 36 power monotone curve.	55
6-7	Line graphics in base case, active power (kW), Currents (A), Losses (kW) and monotone curve.	58
6-8	Line curves in the case of 25% of PV panels in the nodes.	59
6-9	Active power flow	60

6-10	Line curves in the case of 75% of PV panels in the nodes.	61
6-11	Line curves in the case of 25% of PV panels and storage in the nodes. . . .	62
6-12	Line curves in the case of 75% of PV panels and storage in the nodes. . . .	63
6-13	Global graphics for the base case.	65
6-14	Global graphics for the 75% of nodes with PV panels case.	66
6-15	Global graphics for the 75% of nodes with PV panels and storage case. . . .	67
6-16	Real-time simulation of consumed power.	68
6-17	Real-time simulation of voltage level.	69
6-18	Real-time simulation of voltage level with PV panels installed.	70
6-19	Real-time simulation of generated power with PV panels installed.	71
6-20	Real-time simulation of net power with PV panels installed.	72
6-21	Real-time simulation of demanded power with PV panels installed and stor- age devices.	73

List of Tables

4.1	FloWI Elements.	37
6.1	Conclusions from the node analysis.	56
6.2	Conclusions from the global variables analysis.	74

Chapter 1

Introduction

During the recent years has increased the global awareness and understanding that there is a necessity of changing the traditional fossil fuel based energy system, towards a new era of sustainability, low CO₂ emissions and high efficiency. Researchers, companies and governments must join forces to reach this goal. More policies as the 20/20/20 agreement must be approved, not only in the European Union but in the whole world. The 20/20/20 agreement, that basically stands: “20% percent of CO₂ emissions, 20% more of renewables capacity installed by the year 2020”, suppose a path to move to this new energy system.

Among the main factors that composed this post-industrial revolution, it has to be considered the changes in the electrical energy generation, where the distributed generation, located at the costumers site, will grow in the electrical grids. Also, the installation of all the new generation devices and its maintenance will have a positive impact on the local economy focused on the electrical installations sector. And it is not only the generation, but the storage too. There is a need to store the energy produced during the middle hours of the day, so it can be consumed later when the people arrive home and switch on the TV, the computer, charge his smartphone and cook. First batteries operating models have been launched into the market last year by the company Tesla, along with solar roof tiles and electric cars. There are more companies researching and selling this kind of goods and electric cars too, but most important thing is that people are buying, which basically means that the movement towards this new energy system is happening, right now.

Other factor that must be considered is that the energy demand will grow. Even if the appliances in the homes are more efficient, or people substitute classic light bulbs by LEDs, the energy demand is higher. Then, the efficiency is a main issue. Nearly zero energy buildings seem to be a good answer to this claim. These buildings, have really good isolation materials in their walls, so there is a slight thermal flow between inside and outside the building. Then it reduces the necessity of heating and cooling during winter and summer seasons. Also they must have their own generation units, solar or wind turbines and batteries to store it, and supply it when needed.

With this new concept, the role of the utilities and the system operator will change and have more weight on the grid stability and electricity quality. This is why this project has been developed. FloWI tries to be an analysis tool for measuring the impact of the distributed generation and storage devices in the distribution grid. Providing good and intuitive results, with a good visualization based on data driven documents and interactive results and simulations.

This document describes the residential generation and demand in chapter 2. Then the power flow concept, the resolution algorithms and the description of the softwares used to obtain the power flow results is described in chapter 3. The web tool FloWI is completely described in chapter 4. Then in the next chapter it is explained a study case presented by the Institute of Electrical and Electronic Engineering (IEEE), to perform some simulations with the given data using FloWI. Finally in chapter 6, the results of some study cases are compared showing the main features of FloWI. The codes used to develop this tool are included in the appendixes.

Chapter 2

Residential Energy Demand, Generation and Storage

This chapter is a brief introduction to the distribution system and the energy demand in distribution grids and also the residential distributed generation and energy storage.

2.1 Distribution System

The distribution system begins under the distribution substation. This substation is fed by one or more sub transmission lines or by a high voltage line. Figure 2-1 shows a scheme of the power system components.

The substations are composed by three-phase transformers or three single-phase transformers interconnected that reduce the voltage to the distribution voltage level. There are few standard levels, 34.5 kV, 23.9 kV, 14.4 kV, 13.2 kV, 12.47 kV, and, in older systems, 4.16 kV [1]. Then this substation serve one or more primary feeders, usually radial. This means that power flow only has one path from the substation to the consumer. The distribution system will have more than one substation and there will be also in-line transformers and distribution transformer. This ones reduce the voltage level to the low voltage (LV) consumer level, which is 230/400 V the standard in Europe. Then, even if the main feeder

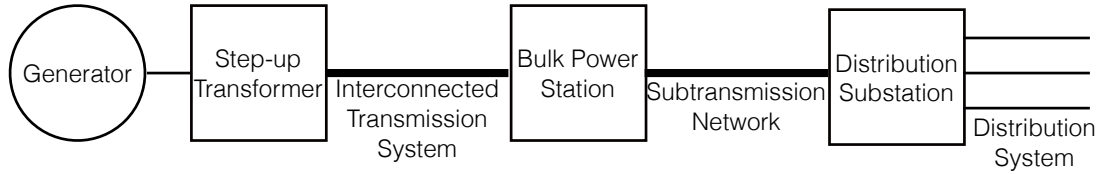


Figure 2-1: Power system components.

has 3 phases, each consumer is connected just to one of them.

The distribution feeders have unbalanced power flows because of the unequal number of single-phase loads connected to each phase. So, as the loads vary along the feeders, the voltage drop will vary too between the transformer and the consumers. This is a fact that voltage drop is not equal on the 3 phases, having a lower voltage level at those phases with higher load values and distances. This requires the fact of introducing voltage regulation, which typically is accomplished by using load tap changing (LTC) transformers to adapt the LV value to the requirements of the system at some instants, having a higher LV level when voltage drop happens. Figure 2-2 shows how the radial distribution, and how the loads are single-phase and not each phase has the same amount of loads connected to it, so it is unbalanced.

2.2 Residential Energy Demand

Along the power system the consumption can be at three levels. First, at high voltage, which is called industrial consumption, where large factories with many loads and an almost 24 hour consumption. Those factories have their own substation and are connected to

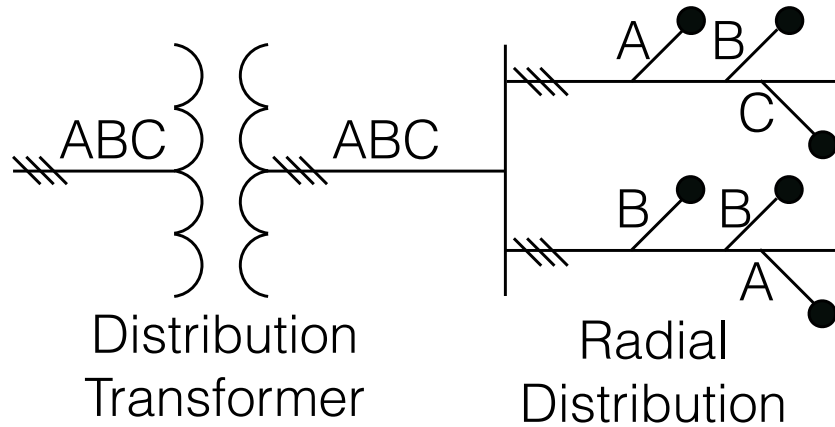


Figure 2-2: Unbalanced and radial distribution.

the high voltage grid in order to obtain a lower price in the electricity bill. Then, there are other consumers at medium-voltage. Large buildings like hospitals, medium factories and malls. These also have their own transformer from medium voltage to low voltage. And finally, there are all the small consumers, that include flats, apartments and houses, which is the residential demand and also small companies offices and services. All those consumers are connected to the low voltage grid and usually in a radial distribution.

Is quite complicated to analyse the residential energy demand because there is no steady-state load. In fact, the loads are constantly changing due to any appliance or to the lighting when a light-bulb turns on or off. But it is very important for the future trends on the energy consumption to study the energy demand of the consumers in the residential market which will be useful to move towards the nearly zero energy buildings (nZEB). This is because usually, the maximum demand hours are in the morning and in the night when people are in their homes and the appliances are working. However, the solar photovoltaic (PV) generation, depends on the sunlight and its better performance is during the middle hours of the day when probably there is no consumption or just the base load of the house (freezer, standby appliances, Wi-Fi). So the generated energy would be only used during these hours to fit the base consumption. But, if it is possible to know the energy consumed during the whole day, it can be possible to install storage in order to save that energy for the peak of demand. This will be explained in the following sections.

The annual report from the Spanish system operator "Red Electrica de España (REE)" of year 2016 says that for the first time in history the maximum demand was in a summer day instead of winter and also the energy demand during the whole summer was higher than in winter. Also the residential consumption has represented the 23.5% of the electrical energy demand in Spain. It is almost one quarter of the total and therefore, any reductions that can be performed in this sector, will reduce the dependence on fossil fuels resources to generate electricity. [4].

2.2.1 Nearly-Zero Energy Buildings.

nZEB are buildings environmental friendly as the energy consumed should come from renewable resources in most of the cases installed in the buildings themselves or nearby. The European Union has already set policies to transform the residential demand making it more efficient, reliable and with less CO₂ emissions. In the Energy Performance of Buildings Directive 2002/91/EC and its recast (Directive 2010/31/EU). This recast demands that "Member States shall ensure that by 31 December 2020 all new buildings are nearly zero-energy buildings; and after 31 December 2018, new buildings occupied and owned by public authorities are nearly zero-energy buildings". But also it is possible to transform existing buildings in nZEB. So the study of the energy demand is very important in order to implement auto-consumption solutions. In [2], it has been developed a software to assess professionals and companies in the selection of the best renewable-distributed generation-mix suitable to be installed in residential buildings considering technical, economic and reliability aspects. [3]

2.3 Residential Distributed Generation

In the traditional power system, the energy comes from the power plants, and goes through the transmission system and distribution system to the consumers. The concept of residen-

tial distributed generation or auto-consumption means that, the generated energy does not have to be transmitted somewhere else, because it is almost consumed at the same generation point. However, due to the unpredictable nature of the renewable resources, consumers with installed generation that want to guarantee their energy supply must be connected to the grid. In that case, the control systems and performance of the distribution utility will be crucial for the system. The main features of the residential distributed generation in words of [5] and [6] are:

- Demand peaks will be reduced.
- Quality of electricity will increase.
- Transmission and distribution losses will be reduced.
- Generation flexibility.
- Business opportunity and economical growth.
- Number of interruptions will be reduced.

The massive penetration of this new concept into the system will suppose a significant change in the transmission system and distribution system. As the lines that have been only unidirectional should be bidirectional to permit power flows from some distributed generation points. Figure 2-3 illustrates the behaviour of a residential auto-consumption power profile.

- Region 1: There is only power demanded, basically the base consumption of the home. And around 07:30 hours the power demand increase due to kitchen appliances when the inhabitants are preparing breakfast, or water heating during the shower.
- Region 2: The demand decrease when people leave their home to going to work. Also the PV panels starts to generate cause it is a sunny day.
- Region 3: This is the most critical moment for the system stability and a challenging endeavour for the utilities. When the people is working is when the PV panels are

generating more energy. Then it could be an excess of power. This power must flow to the grid instead that to the house. Then is many neighbours have installed PV panels, it could be a problem to the transformer located in the substation.

The good part is of course, that the emissions of CO₂ during this period could be reduce quite a lot and the consumers are paying nothing for the energy cause they are self-sufficient during this time.

- Region 4: People are coming back home and the sun is going down. People after work like to watch TV, cook, have a shower, charge its electronic devices and many other electrical energy consumption behaviours that we have available nowadays. Then the consumption increase and it is covered by the PV panels but not for long.
- Region 5: The consumption is at is maximum and the generation is practically zero at this time. So the home is consuming all the energy from the grid. Also those are the hours when the energy is more expensive due to the necessity of including expensive generation technologies based on burning fossil fuels and with high CO₂ emissions.
- Region 6: During this time the demand used to decrease when people are going to sleep and falls to the base consumption again.

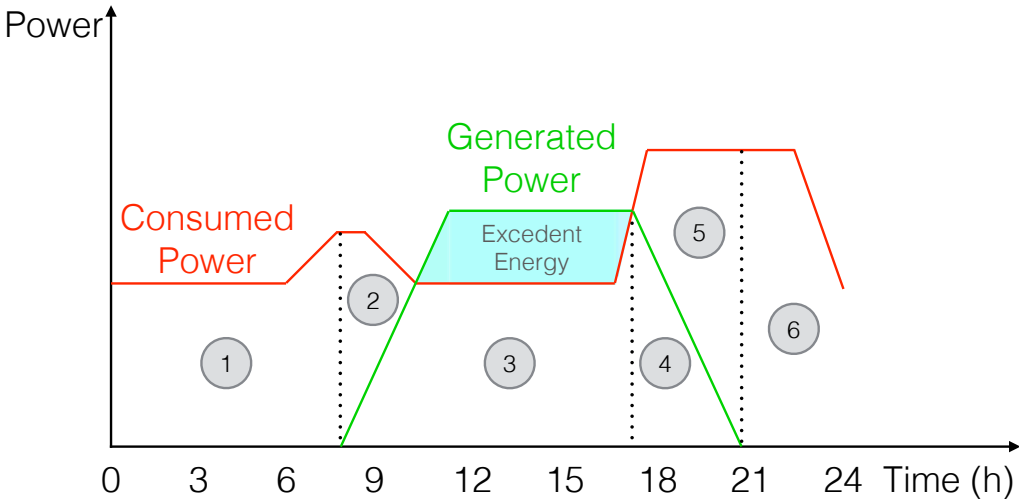


Figure 2-3: Generation power profiles.

2.4 Residential Energy Storage

As mentioned before, PV solar generation and wind generation used to occur when the residential demand is low, basically the base demand. If the installed generation capacity is higher than the base demand, then it can occur that at some point the generation is higher than the demand. If this happens, that energy must be injected to the grid. And on the other hand, when the demand is higher than the generation, it will be necessary to supply it from the grid.

It is very difficult to maintain the system stability when many PV panels and wind turbines are injecting power. One of the main problems could be that, as the system is unbalanced, loads at each phase are different. It could be possible that, at the substation point, one of the phases has so much power injected that the power is flowing to the high-voltage side while the other two are flowing to the low-voltage side. This is called reverse power flow and could cause many problems. To avoid that energy storage is becoming an essential element in the power system.

In the case of residential energy storage, the purpose is to charge the batteries with the excess of power during generation hours, so it can be consumed during the demand hours until they are discharge and starts to demand from the grid.

For the residential energy storage, as is shown in OpenDSS chapter 3 of this document, there are various control strategies to charge and discharge the batteries. Among them we can find the case where the battery does not have current control, so it will always charge and discharge at its rated power. Other where it has current control, so the power will be adapted to the required demand. Other case could be the economical optimization in case that some users only have the batteries installed but no generation. In that case, it will be possible to charge the battery during the low price hours, typically during the night, and then use this stored energy during the high price hours.

Figure 2-4 illustrates the behaviour of a residential consumption having both generation and storage. The power demand does not vary from the previous case with only generation. But there is an important difference in this case. In region 3, when there is an excess of power, it is not a problem any more. All this power can be stored to use it later. In regions 5 and 6, the net power will be the difference between the demanded from the grid and the given by the battery. Then the maximum power demand is reduced, the bill will be smaller and the utilities maybe will not have to generate as much from fossil fuel plants as if there were no storage elements installed.

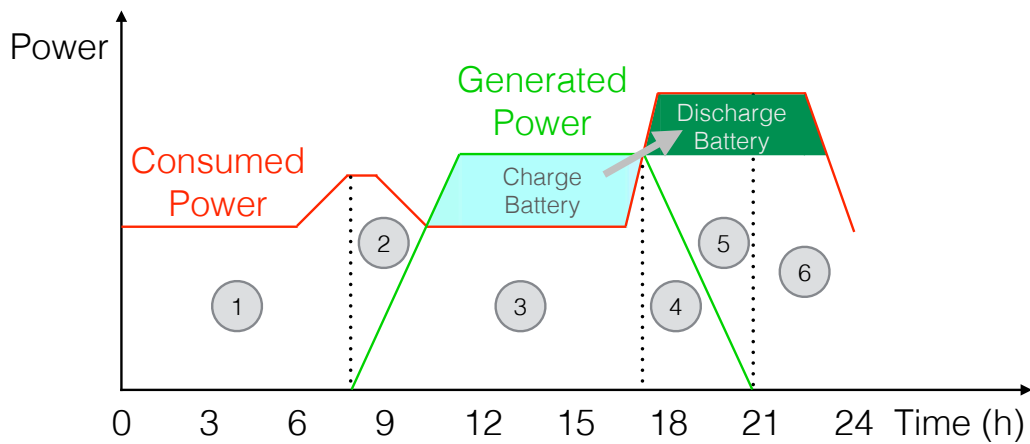


Figure 2-4: Generation and storage power profiles.

Chapter 3

Power Flow

This chapter includes a brief description of the grid modelling and the power flow problem. In the early years, the grid was very small, with few interconnections and with radial distribution. During the 20th century, electrical network has become more and more complex and there are a lot of interconnections inside the grid. Power flow consists in finding the steady-state operating point of an electric power system. It means that, knowing the power demanded in the buses and the power supply by generators or by a transformer in a micro grid, we can obtain bus voltages and power flowing through all grid elements. [7]

In the system operation it is used for analyse the voltage level and identify their possible deviations out of the operation boundaries and also power flowing through the lines and ensure that they are not overload. This is why it is useful to study future and hypothetical scenarios with new elements included.

In this chapter is explained the grid modelling, the power flow formulation, the different methods to solve the power flow problem and finally it is included a description of the software EPRI OpenDSS. This software developed by EPRI has been used in this work to solve the power flow of the grid used and its results are visualized with the web tool FLOWI.

3.1 Grid Modelling

As this analysis refers to the whole network the representation is more complex. It is necessary to introduce the admittance matrix concept. Consider a generic bus, connected

by series admittances to another buses, as we can see in figure 3-1. Also it has to be considered the small shunt admittance connected to the neutral or ground. The net current injected to the bus is shown in equation 3.1.

$$I_i = \sum_{j \in i} y_{ij}(V_i V_j) + y_{si} V_i \quad (3.1)$$

V_j stands for the complex voltage at bus j . Then we can rearrange the terms in what is called "from" bus and "to" bus. It is expressed in equation 3.2.

$$I_i = \left[\sum_{j \in i} y_{ij} + y_{si} \right] V_i - \sum_{j \in i} y_{ij} V_j \quad (3.2)$$

And doing that for the whole set of buses we obtain equation 3.3, where Y is the admittance matrix and the column vectors V and I represent the node voltages and net injected currents [7].

$$I_i = YV \quad (3.3)$$

This is how a three node equation system looks like:

$$\begin{bmatrix} I_1 \\ I_2 \\ I_3 \end{bmatrix} = \begin{bmatrix} Y_{11} & Y_{12} & Y_{13} \\ Y_{21} & Y_{22} & Y_{23} \\ Y_{31} & Y_{32} & Y_{33} \end{bmatrix} \cdot \begin{bmatrix} V_1 \\ V_2 \\ V_3 \end{bmatrix}$$

3.2 Formulation

Starting the formulation from the net complex power injected to bus i as is expressed in equation 3.4, and applying this equation to all the buses and combining with 3.3, it constitute a system of $2n$ complex equations in terms of the complex unknowns S , V and I . Then the complex power can be expressed as in equation 3.5

$$S_i = V_i I_i^* \quad (3.4)$$

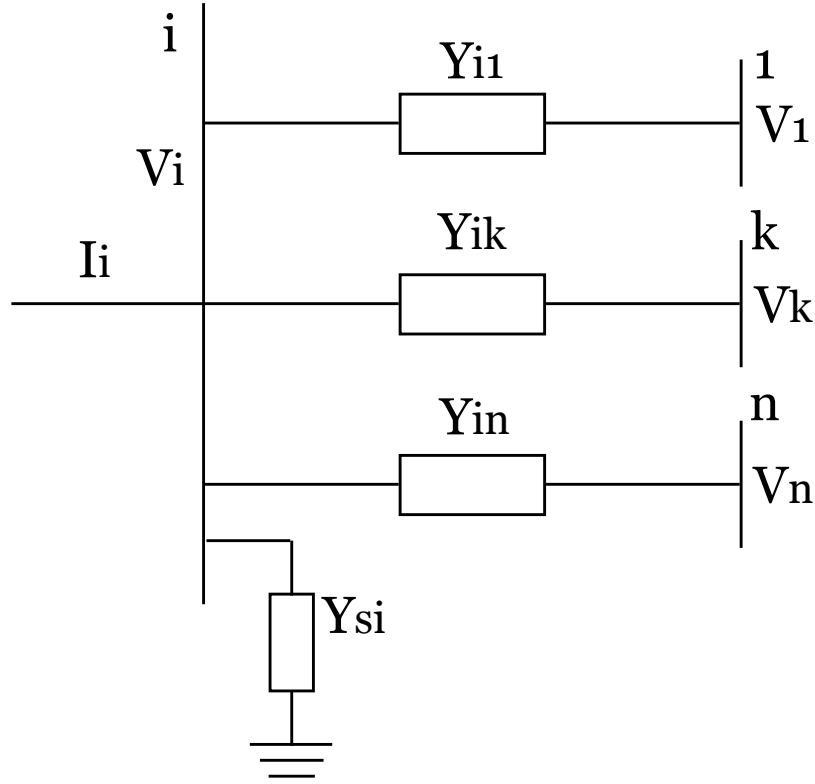


Figure 3-1: Power system components.

$$S = \text{diag}(V)[YV]^* \quad (3.5)$$

Then, expressing the complex power in terms of active and reactive power, as in equation 3.6:

$$S = P + jQ \quad (3.6)$$

And expressing the elements of the admittance matrix using rectangular coordinates as in 3.7:

$$Y = G + jB \quad (3.7)$$

We obtain the expressions 3.8 and 3.9 in polar form.

$$P_i = V_i \sum_{j=1}^n V_j (G_{ij} \cos \Theta_{ij} + B_{ij} \sin \Theta_{ij}) \quad (3.8)$$

$$Q_i = V_i \sum_{j=1}^n V_j (G_{ij} \sin \Theta_{ij} + B_{ij} \cos \Theta_{ij}) \quad (3.9)$$

3.3 Bus types

Each bus provides two nodes and four unknowns depending on the kind of bus.

- Slack bus. It is a bus with generation so it has a voltage regulation that keeps the voltage magnitude to the specified value. It has the particularity that its active power is left as an unknown and the phase angle is set to zero. It is a mathematical artefact so it can be performed a power flow solution to the system.
- Generation or PV buses: At least one generator and a regulator that keeps the voltage magnitude to an specified value. Also the power injected at these kind of buses is specified.
- Load or PQ buses: In this case the active power and reactive power values are specified.

3.4 Methods To Solve Power Flow

We are going to focus on the methods to solve non-linear equation systems. These methods are based on an iterative process. The two main methods are Gauss-Seidel and Newton-Raphson.

3.4.1 Gauss-Seidel Method

For a vector with initial conditions x_i^0 , the result of the new vector by applying the Jacobian method would be $x_i^{k+1} = [x_1^k, x_2^k, \dots, x_n^k]$. This method will finish when the difference between two consecutive parameters is smaller than the convergence parameter determined.

3.4.2 Newton-Raphson Method

This method is based on first order approximations of the non-linear equations. The general form of procedure with this method is to find an x value that fits equation 3.10:

$$f(x) = 0 \quad (3.10)$$

By using the Taylor series expansion of this equation around x^k and keeping the first two terms we obtain equation 3.11:

$$f(x) \simeq f(x^k) + F(x^k)(x^{k+1} - x^k) \quad (3.11)$$

Where F is the Jacobian matrix of $f(x)$. The iterative process stop when the method arrives to two similar consecutive values, as is expressed in equation 3.12.

$$\max |f_i(x^k)| \leq \epsilon \quad (3.12)$$

3.5 EPRI OpenDSS

Developed by the Electric Power Research Institute (EPRI), OpenDSS is a power system simulation tool for electric utility power distribution systems. Besides of performing the common analysis on the utility power distribution systems it also supports new types of analyses designed to meet future needs. Among them it can be highlighted the smart grid, grid modernization and renewable energy research. [8].

As this tool offers all of that possibilities, it fits perfectly the requirements for this project. So is going to be our tool to solve the power flow problem of the system and generate the result files.

OpenDSS has another important features. One is that is has an important advantage respect to other similar software to solve power systems and is that its license and its code are free. Other important feature is that OpenDSS is based in a quasi static model in sequential mode. This means that the simulation can be customized to users needs. It is possible to choose if the solution is made for a day, a year... And also it is possible to choose the data

interval, each minute or hour... This is a huge advantage because it allows us to introduce real data obtained from smart meters.

Main features of OpenDSS:

- Evaluation of losses due to unbalanced load.
- Develop of distribution generation models for the IEEE feeder test.
- THD analysis.
- Study of wind farms and PV plants.
- Open conductor failure study, for one or three phase transformer connections.

3.5.1 OpenDSS

To proceed with a case in the Opendss we need to execute the main.dss file. It has the main instructions to do and some declarations of the electrical grid to simulate. For a simple case, for instance, a system with three buses, the declarations for the whole grid can be included in this main.dss file, but for a big case, it would be better to make the declarations in .txt files and make a call from the main.dss to these text files. Then we are going to explain all the data files needed.

- **Line Code:** In this file are defined the different types of lines existing in the system, considering in they have one or more phases, its material and other physical parameters. For example, the declaration of a LineCode would be the following: `New LineCode.2c007 nphases=3 R1=3.97 X1=0.099 R0=3.97 X0=0.099 C1=0 C0=0 Units=km`
- **Lines:** Here we defined the name of the line; generally, it would be named as LINEX, where X is a number depending on the number of lines that the system has. It is also specified the starting and end point of the line, Bus1 or from bus and Bus2 or to bus. Furthermore every line needs a line code to specified its physical parameters.

So it has to make reference to one of the line codes defined in the previous file. The declaration of a line would be the following: New Line.LINE1 Bus1=1 Bus2=2 phases=3 LineCode=4c70 Length=1.098 Units=m

- Storage: This file will be only included if we want to perform a case where there are storage elements installed in the system. The storage elements have several working modes. In this document we have focused on the default mode and follow mode. There is more information in [9].

In this mode we need to specified the trigger values for charge and discharge. Figure 3-2 illustrates the working mode. Inside the trigger values the storage element remains in standby. Once the net power (demanded power - generated power) curve overcomes the discharge trigger it starts to discharge. The same happens if the net power curve goes down the charge trigger.

- For declaring a storage element it is necessary to specified the maximum charge and discharge power with the parameter `kwrated`.
- Also the maximum storage energy is specified in `kwrated` parameter.
- The minimum storage level is set in the parameter `% stored`
- By default, the storage element is programmed to start charging at 2 AM because of the price of the energy. This is interesting for cases when there are storage elements in the houses without having generation. This kind of strategy could save some money buying energy during cheaper hours to use it during the afternoon peak demand. However, in our case, we suppose that our consumers will have both generation and storage. Then it is not necessary any more to start charging our batteries from the grid any more. To clear this parameter we have to write `TimeChargeTrig=-1`. However, during the testing period of `OpenDSS` trying to find the better solution by using the storage elements, the default mode was not a good choice. It is because once the trigger is overcome,

and it starts to charge or discharge, it always do that at the maximum power, which in this case is 5 kW. This behaviour causes huge voltage drop and power demand when the PV panels start to generate energy during the morning. Then follow mode was a better option. It consists in calculate the difference between generation and demand, and the constraints that the maximum energy stored level is 100% and the minimum 20%. Then we have created net power curves, and added it to a shape as in the loads and generators, so the storage element can follow that shape and simulate the real behaviour of a current controller for charging the battery.

The declaration of a storage element would be: `New Storage.Bat1 phases=1 Bus1=908.1 daily=ShapeBat1 kv=0.24 kwrated=5 pf=1.0 kwhrated=8 state=IDLING DischargeTrigger=1 ChargeTrigger=-0.8 TimeChargeTrig=-1 %stored=20`

- **Load Shapes:** The load shapes are used to associate a load profile to other elements like loads, generators or storage. It has 1440 values, each one for each minute of the day. Through this file we can specified the behaviour of an element during 1440 instants in the day. The specification of a load shape could be: `New Loadshape.Shape1 npts=1440 mininterval=1 csvfile=Loadprofile1.txt`
- **Generators:** This file specifies the generators inside the grid. It is only necessary when we are going to perform a case with distributed generation. If we do not have any generator it is not necessary to include it as happens with the storage file. For declaring a new generator it could be: `New Generator.Gen1 phases=1 bus1=908.1 kv=0.24 kw=1 pf=1.00 conn=weye status=variable model=1 daily=ShapeGen1`. In the declaration we can find that bus1 is the bus is connected in this case 908, and the phase to which this bus is connected to the grid. Also, the daily parameter refers to the load shape attached to the generator to simulate its generation profile during a day.
- **Transformers:** This file define the transformers existing in the system. It can be specified as: `New Transformer.TR1 windings=2 Buses=[SourceBus 1] Conns=[Delta`

Wye] kvs=[11 0.416] kvas=[800 800] XHL=4 sub=y

In the case that we have studied in this document there is only one transformer in the grid, so it has all the needed information to perform a simulation.

- **Loads:** This file defines the loads existing in the system. It has to be defined its name, the number of phases (three phase or single phase), the bus where it is connected, kV reference value, kW reference value, power factor, connection (wye or delta), and we have to specified that the performance will be for a day so we must write `daily = Shape`. Where shape will be one of the load shapes defined in the previous file. The declaration of the loads could be: `New Load.LOAD1 Phases=1 Bus1=907.1 kv=0.24 kw=1 PF=0.95 model=1 status=variable conn=wye daily=Shape`.

Note that as happens in the declaration of the generator, when the bus in which we connect the load is specified it is necessary to include the phase. So we have to write `bus1=907.X` where X could be 1, 2 or 3 standing for phases a, b or c.

- **Monitors:** This file specifies the measures that we want to do in the elements of the system. These monitors shows different parameters depending on the configuration we choose. After simulation, the monitors will create a results folder, with results files in .csv format. Here it is included the specification of a monitor attached to the element LOAD1 and measuring its active and reactive power: `New monitor.LOAD1PQ element=Load.LOAD1 terminal=1 mode=1 ppolar=no`.

The parameter mode is used to set which kind of variables we want to measure with the monitor. Mode 0 measures voltage and current, mode 1 measures active and reactive power and mode 3 is used to monitored the storage elements. It shows the stored energy, the state of the battery (charging, discharging or standby), input power, output power, losses, and charging energy.

Then after we know all the data that we need to perform a power flow case, we need to know how to configure the main.dss. First of all, if we are going to declare the elements in different .txt files we have to tell Opendss where to find them. So, in order to do that we have to write: `“set datapath/.../.../data”` with the address of our data documents. Then we have to tell Opendss to open them. We can do that by writing `“redirect”` follow by the name

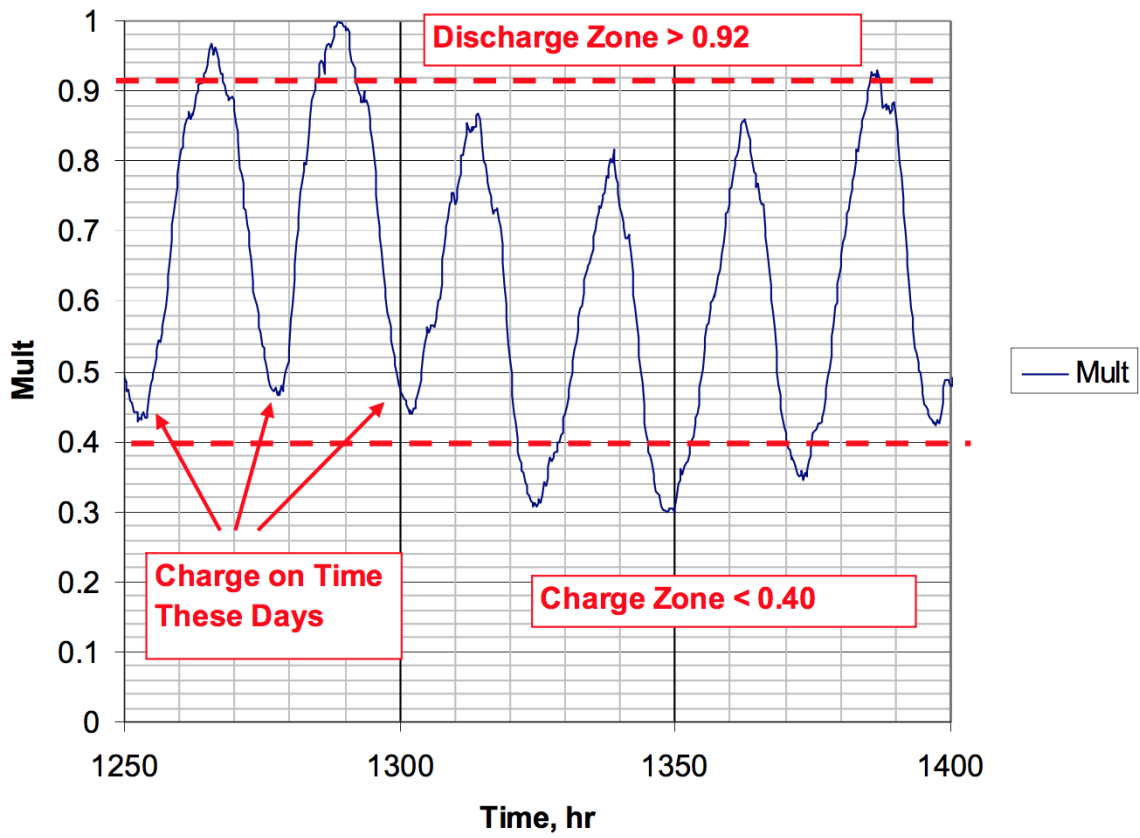


Figure 3-2: Storage default mode.

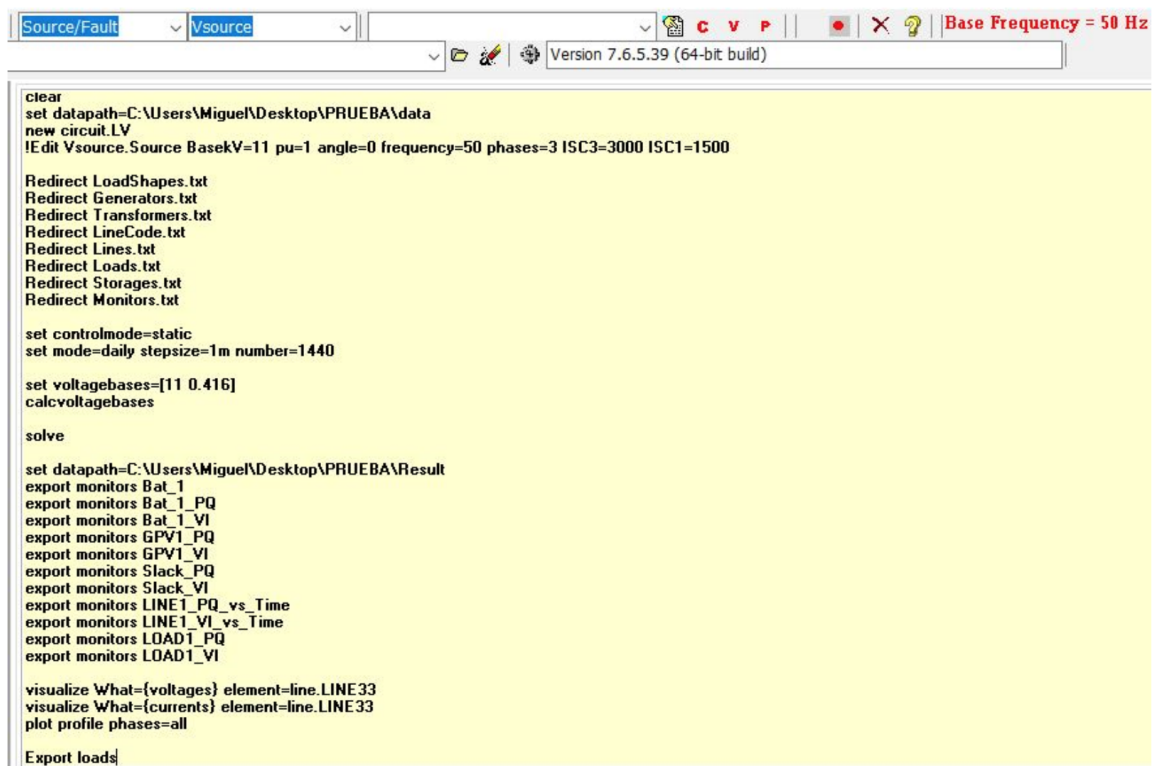
of the file, for instance for the line code file, we must type “Redirect LineCode.txt”. We have to do that for all the .txt files. It is mandatory to call these files in the order they have been explained previously. This is because some of them have parameters that depend on other files, so Opendss needs to read all the data in the correct order.

After finish will the calls, it is necessary to specify a few more things. Is necessary to specified the control mode as static, the simulation mode as daily, to perform a single day solution, the step of one minute and 1440 instants. And finally the bus voltages, that in the case performed in this document are 11 kV and 416 V. These declarations must be like that:

“set controlmode = static”, control mode is specified.
"set mode = daily stepsize = 1m number = 1440", time mode is specified. Number refers to number of steps.
“set voltagebases = [11 0.416]”, high voltage base and low voltage base are specified.
“calc voltagebases”
“solve”, with this command the power flow calculation is initialize.

At this point Opendss has all it needs to perform a single day power flow solution. However, we want to know the results so we can work with them. For doing that, we need to specified a path to a folder where we want to store the simulation results, “set datapath/.../.../results”. Then, we have to tell Opendss which monitors we want to export. So we have to write “export monitors NAME OF MONITOR”. In a large system we have to write many lines for export monitors, more or less two for each line, two for each bus, one for each generator and one for each storage element.

These are all the files and specifications needed to perform a case in Opendss. Once this is ready, we have to select all the main.dss lines and click enter (CTRL + A then ENTER). For make it more understandable, figure 3-3 shows a main.dss file of a simple base case.



```
Source/Fault  vsource  Base Frequency = 50 Hz
Version 7.6.5.39 (64-bit build)

clear
set datapath=C:\Users\Miguel\Desktop\PRUEBA\data
new circuit.LV
!Edit Vsource.Source BaseV=11 pu=1 angle=0 frequency=50 phases=3 ISC3=3000 ISC1=1500

Redirect LoadShapes.txt
Redirect Generators.txt
Redirect Transformers.txt
Redirect LineCode.txt
Redirect Lines.txt
Redirect Loads.txt
Redirect Storages.txt
Redirect Monitors.txt

set controlmode=static
set mode=daily stepsize=1m number=1440

set voltagebases=[11 0.416]
calc voltagebases

solve

set datapath=C:\Users\Miguel\Desktop\PRUEBA\Result
export monitors Bat_1
export monitors Bat_1_PQ
export monitors Bat_1_VI
export monitors GPV1_PQ
export monitors GPV1_VI
export monitors Slack_PQ
export monitors Slack_VI
export monitors LINE1_PQ_vs_Time
export monitors LINE1_VI_vs_Time
export monitors LOAD1_PQ
export monitors LOAD1_VI

visualize What={voltages} element=line.LINE33
visualize What={currents} element=line.LINE33
plot profile phases=all

Export loads
```

Figure 3-3: Storage default mode.

Chapter 4

Web tool: FloWI

FloWI stands for Flow Web Interface. This means that it is a web tool designed for the visualization and study of the power flow results obtained from Opendss as explained in chapter 3. This tool is based on data driven documents (from now onwards D3) for the data representation, advanced visualization systems, and real time events. The tool has been designed specifically to study the impact of distributed generation and nearly-zero energy buildings inside the distribution grid. The idea of creating this tool as a web page was that it is possible to use it from every operation system. As it works in a browser, every device that can navigate with a browser will be able to use FloWI. In fact, D3 supports every modern browser except Internet Explorer 9 and below. FloWi has been tested in Chrome and Safari with successful performance. Its design is modern and minimalist creating an easy and fast-learning user experience. In figure 4-1 we can see the FloWI main page before uploading any files for their visualization in Safari.

So, the needed files to make it work are, all the system data which defines the buses, lines, transformers and loads. Those files are to draw the system network in the map. Then those files must be complemented by the power flow results in format .csv so the web code can process it. In this way, FloWI is able to draw the network which has all the power flow simulation results during one day, every minute, related to each bus or each line so it is possible to start the visualization of every parameter. In figure 4-2 we can see how FloWI has draw the low voltage test European feeder by the IEEE, with an installation of PV panels and batteries on the 25% of the consumption buses.

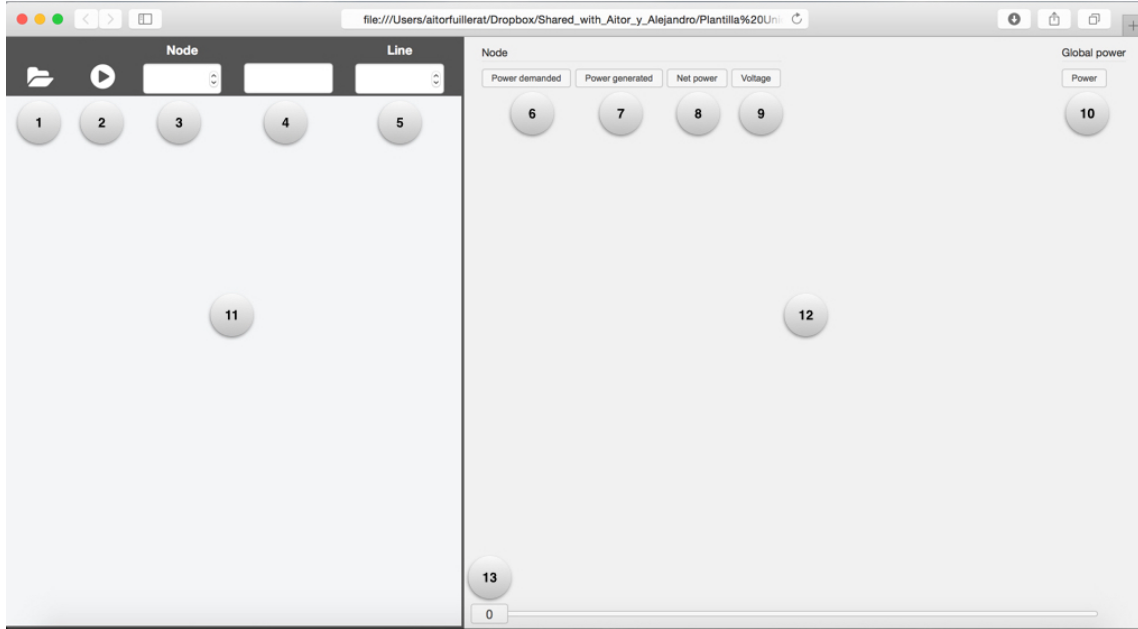


Figure 4-1: FloWI Initial Page.

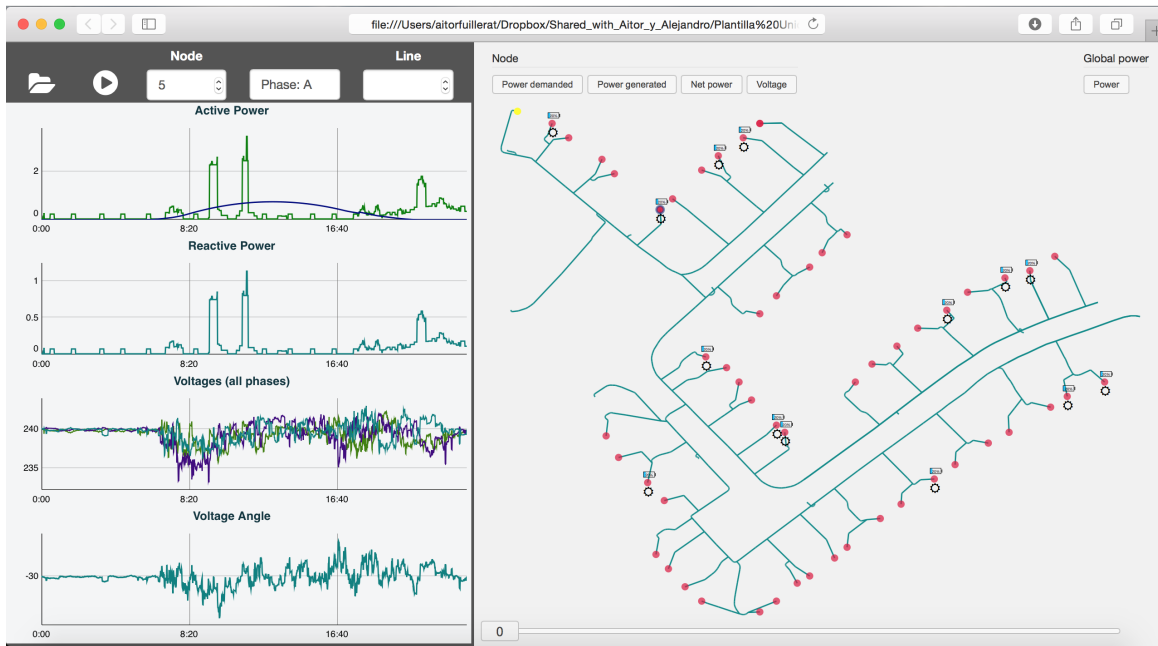


Figure 4-2: FloWI before files were uploaded.

Table 4.1: FloWI Elements.

Number	Description
1	Upload zip files
2	Play and pause simulation
3	Node selector
4	Phase indicator
5	Line selector
6	Show power demanded on each bus during simulation
7	Show power generated on each bus during simulation
8	Show net power on each bus during simulation
9	Show voltage level on each bus during simulation
10	Draw global variables in the graphics area
11	Graphics area
12	Map area
13	Slider: time representation

As can be seen, the screen is divided in two parts. The right part is to represent the system, and the left part to draw graphics related to the buses or lines that we select. There are also other elements as the menu on the top-left and a slider representing the time at the button. All the elements and its description are explained in table 4.1.

4.1 Front-end Development

This section talks about the front-end development or the user experience interface. It means all the elements and the interactivity that the web offers. So basically it is the visual part.

FloWI has been designed with the intention of creating a great user experience. The interface is simple, minimalist and intuitive. Once the electrical system data are ready, they can be uploaded into the web, all included in a zip file and FloWI will process it automatically. The format of the data and results is explained in chapter 3, among all the other specifications about Opendss.

4.1.1 Design and programming

The web has been designed using three programming languages: HTML, CSS and JavaScript. HTML stands for Hyper Text Mark-up Language. It is the standard language for creating web pages. It gives the structure to the web. In fact, any web, is composed of many HTML block elements as titles, paragraphs, lists or boxes.

CSS stands for Cascading Style Sheets. This language describes how HTML elements are going to be displayed on screen. So basically, CSS defines styles on the web page, including the design, layout and variations in display for different devices and screen sizes. With CSS is possible to modify the position of every element, their colors, sizes and text fonts.

JavaScript was invented by Brendan Eich in 1995, it is used to program the behaviour of web pages. JavaScript was used from buttons on click events, upload files events, to drawing the data and the graphics related to the system. For this web, three main libraries have been used in order to achieve a better performance. D3 (Data Driven Documents) is designed for data representation. It has been used to draw all the network (buses and lines), and include extra symbols for those buses which have PV generation installed or storage installed. For drawing the graphics it has been used dygraph. This library read the input data in the format data magnitude over time, so in this case, it represents voltage, current, power and phase over time. This library also makes possible to make zoom over the graphics on those areas of interest both in axis x and y. The third library used is JQuery. It is a JavaScript library and its purpose is to make it much easier to use JavaScript. Actually, it is very helpful to save time and lines of code just by calling a function, where if not, it would be necessary to write many lines of JavaScript.

In appendix A it is the full HTML code of the web. In appendix B is the CSS code of the web. In appendix C is the JavaScript code of the web.

4.1.2 Upload files

On the top-left menu we can see a file icon. By clicking on it, an event will generate a new window to choose the zip file with all the data. Then, if we click on "Accept" FloWI will automatically plot on the right side of the screen the map of the electrical system.

4.1.3 Visualization of results through on click event

In this paragraph is explained the results that is possible to see from the different elements of the system. This are called on lick events because to see the results you have to click on the element of the network you are interested in. So it is possible to evaluate the buses, lines, storage elements and transformer buses. We are going to analyse this more deeper. See figure 4-3 to know how graphics are draw. Note that in the third graph related to the voltage, a zoom has been made over 9:00 to 10:00 hours to demonstrate how it works.

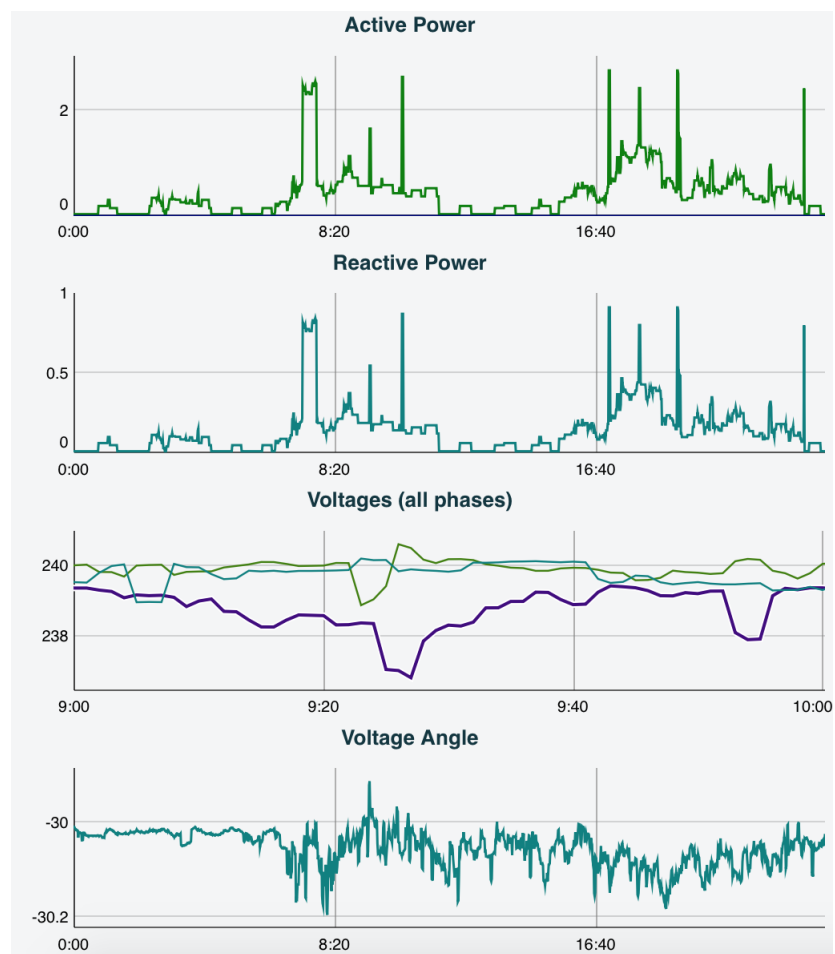


Figure 4-3: Bus Graphics Layout.

- Buses: Each load in the system is represented as a red circle. To select a bus, we can click on it or we can select it from the bus selector that is in the menu, as can be

seen in figure 4-2. Once the bus is selected, FloWI will draw a blue stroke around it. This way it will be easy to see where this bus is located. At the same time, four graphs will appear on the left side of the screen. Those graphs are the active power both demanded and generated in kW. The reactive power in kW. The voltage level in the three phases in V. The voltage angle in degrees. And next to the bus selector in the menu, it will appear to which phase is the load connected.

If the bus has PV panels installed, FloWI will draw a solar symbol under the bus in the map. If it has storage elements, FloWI will draw a battery symbol over the bus.

- **Lines:** When selecting a line, the property width will duplicate its value so it will be highlighted and easy to visualize. To select a line we can click on it, or we can select the line we desired to analyse with the line selector on the menu. Then, it will appear the four graphics on the left side of the screen. For the lines those graphics will show the active power through the three phases of the line in kW. The losses in the three phases in kW. The currents in the three phases in Amps. The total active power through the line, summarizing the three phases and reordering its values from the maximum to the minimum. So, this way, will be possible to know how much time there is a power value flowing through the line.
- **Batteries:** In case of being installed, its symbol will appear in the map. By clicking on it, the battery will be highlighted and on the graphics will appear the active power consumed and generated in that bus in kW. The net active power in kW. The battery charge profile in kWh. The active power in and out in kW.
- **Global variables:** At the top-right side of the screen, there is a bottom that says global variables. If we click on it, then it will appear four graphics with global variables. Those graphics are the global demanded and generated active power in kW. The net active power in kW. The net active power reorganized from the maximum to the minimum in kW. The global losses in the three phases in kW.

4.1.4 Visualization of results through slider event

Slider events are those ones that can be visualized as a time variation simulation during one day. The slider can be played as a video simulation of the whole day, while we can choose what variables we want to analyse in real time or we can handle the slider and move it as we want, just as the slider of a video in a smart phone.

So, in order to select what variables we want to analyse, there are four buttons on the top of the screen, over the map. Those buttons are for showing, demanded power, generated power, net power and voltage value in per unit. This is better explained in figure 4-1 and table 4.1. So for power variables, a color circle will appear around the bus. The radius of this circle is proportional to the kW value. So the higher the power the bigger the circle. In the case of the net power, there will be two colors for the circle. This is to differentiate between positive and negative net power. See figure 4-4 where it is shown how power demanded simulation looks like. It can be seen that the simulation is paused at that instant of the afternoon. The green circles represent the buses where there is power demand and the bigger the circle the higher the demand. So it looks easy to see where is power demand and how much is it.

In the case of voltage, there are five ranges of voltage, each of them related to a coloured circle. So in case that the voltage is not at its nominal value boundaries that are $\pm 2\%$, FLOWI will draw an orange circle if the voltage is over that boundary or a blue circle if the voltage is below that boundary. Both colors will become more intense if the voltage overcome the boundary of $\pm 10\%$.

Then while simulation is running, it is possible to have a global vision of system's variables and where and when power is either demanded or generated and voltage is between its boundaries or not. All of this makes so simple and easy to analyse the results from a power flow simulation and compare the same grid among different scenarios. As this tool has been designed to represent distributed generation and storage, it will be easy to analyse the effect that will have on the system the fact of including those elements.

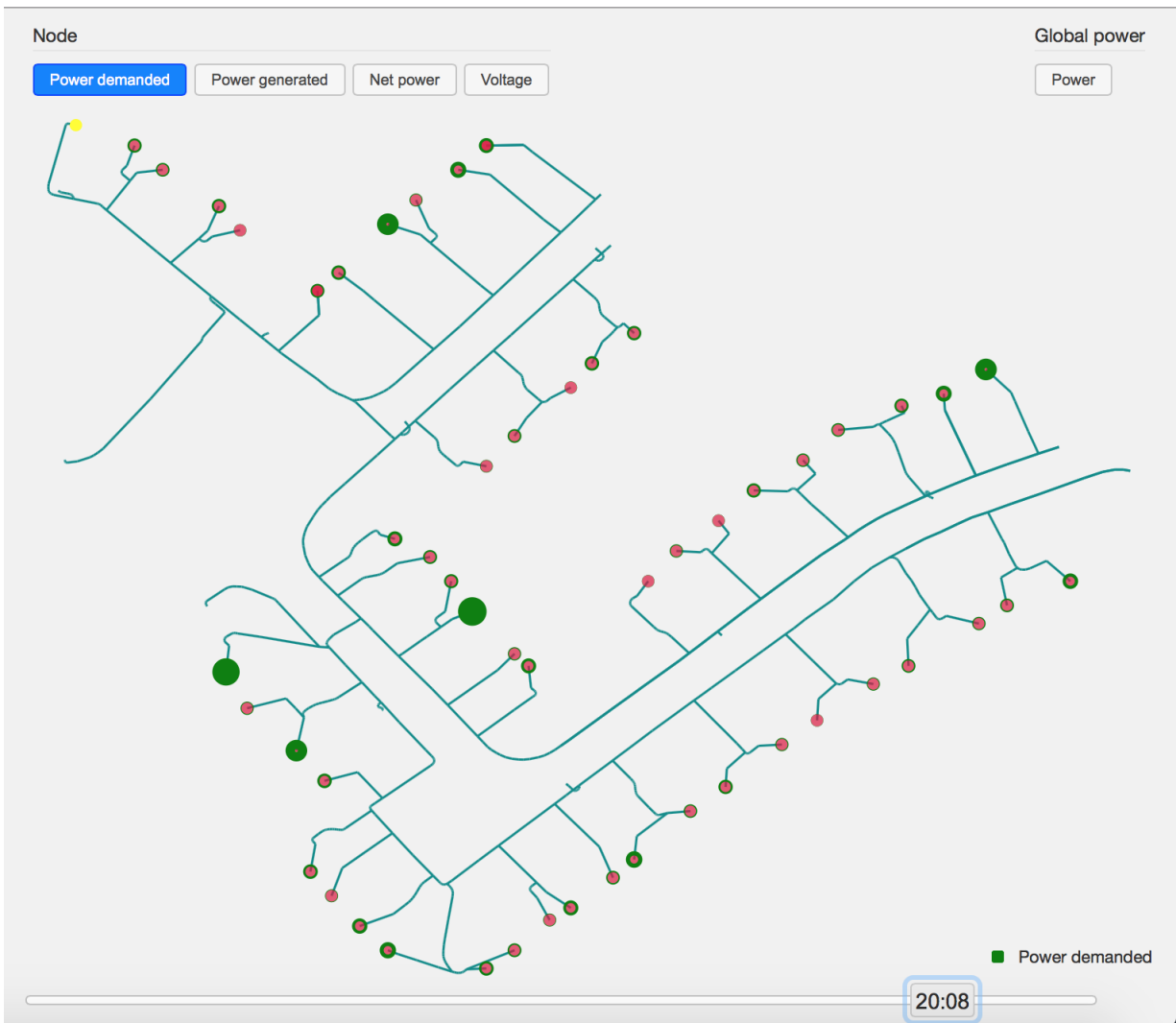


Figure 4-4: Illustration of how power demanded is represented in the map.

4.2 Functionality development

The functionality means the internal work that the web does but the user does not see. This is all the algorithms, functions or calculations. Every event or process have a code behind. So in this section is explained how the web works, not as a simulation tool but as a programming code.

4.2.1 Read Input Data

To start the simulation, first thing to do is click on the file button on the top-left side of the window. It will be better explained in the user guide. But once the file is selected and we click on "Accept", this button will call to the function UploadFile(event). To see this function look appendix C, lines 455-464. This function will start the reading process by calling all the reading functions and the PapaParse configuration.

Once the zip file has been uploaded by the user, FloWI is going to read all the files and show the results. To read a ".zip" file it has been used the JavaScript library "PapaParse". This library obtain all the data included in the file and read them.

The way that it reads the data must be configured. In appendix C, lines 22-40, it is the configuration of PapaParse to specify to the function how to read the data. It specifies what is the delimiter between two data, what is the end of the line and if the file has a header or not, so it must skip the first row.

Then it starts to read each ".csv" file included in the ".zip" file. It is important that the input files are called as the IEEE European LV Test Feeder have called them. Because this function is going to seek for that specific file name and read it. This specifications will be explained in the user guide.

Once the program has finished reading the input data it starts to read the result data in the same way as the input ones. It is also important to use the same names that OpenDSS gave to the result data. For the case of the active power generated due to PV panels or the storage results, it works exactly the same way as the other result data. But there is an exception, thus it is possible to include or not PV panels and storage, and if included, it can be in some of the load buses but they does not have to be in all of them, is necessary to first initialize

the reading array with zeros. It is because if there is no PV panel or no storage at any bus, it will consider their curves as zeros when simulating or drawing. And if there is some of this elements, the function rewrite this array with the data obtained from OpenDSS. To see the reading functions, see appendix C, lines 62-268.

After that, there are some functions to obtain the data of interest inside the files that have been read and generate the arrays used for the simulation and for drawing at the graphics area. To see these functions see appendix C, lines 307-452.

4.2.2 Draw the map

For drawing the map it has been used the JavaScript library D3, that stands for Data Driven Documents, [10]. This library allow us to represent data variables on the screen and make it interactive. It is possible to click on it, make zoom on it and draw different elements in function of different events.

In that case, we want to draw the buses and the lines in the map. As FloWI has already read the files with all the data, it now process it and works in the following way.

First, it creates an svg element over the map section that has been specified in the HTML code and its size that has been specified in the CSS code. See appendix A, lines 164-234 and appendix B, lines 14-17.

Then, it creates an x-axis and y-axis scales to make the data fit the available space. Then it uses these scales to adapt the bus coordinates to the map size and plot the buses with loads as red dots in the map. After that, it draws the lines connecting the buses. At this point the map is ready and its result can be seen in figure 4-2. To see the code to draw the map, see appendix C, lines 1587-1664. After that, it is also included the possibility of drawing the PV panel symbol and the battery symbol related to the buses where they are connected. The tool will only draw those elements when they are included in those buses. So in the case of the PV panel, if exists, it will draw a solar icon. For the case of the battery, it will draw two rectangles, one exterior rectangle with black borders to use it as a frame, and an

interior one, which width is variable in function of the battery percentage of charge. And also the text inside of the rectangle that shows the percentage of charge of the battery, that actualizes its value in function of the data.

See appendix C, lines 1587-1809 to know the necessary code to draw all the elements in the map.

4.2.3 Draw the graphics

The graphics that appear on the left side of the screen as can be seen in figure 4-2, are draw for one of the following reasons: That a bus is selected, a line is selected, a battery is selected or the global variables button is clicked. In any of this cases, the graphics related to each elements will be draw. It works in the way that every element in the map mentioned before works as a button so it can be clicked. In the case of the buses and the lines it can be also done by selecting from the bus or line selector button, see figure 4-1. Then if an element is selected, there is a function that process which kind of element we are selecting and also what is its number. For example, if we click on load number one, the program will know that we are selecting a load, so it will call the data results related to that load and then plot it. To draw the graphics it has been used the JavaScript library Dygraph. It is a fast, flexible open source JavaScript charting library. The graph that is created is interactive: you can mouse over to highlight individual values. You can click and drag to zoom. Double-clicking will zoom you back out. Shift-drag will pan. The code to draw the graphics for each element is in appendix C, lines 473-1314.

4.2.4 Perform the visualization simulation in the map

The visualization of the results in the map is related to the slider, which defines the variable time and the buses power and voltage results. It works in the way that, the slider represents 1440 minutes, which is equal to 24 hours. Then, the result arrays have 1440 elements, because of the OpenDSS and the IEEE test feeder that defines the case with a minute res-

olution. So for each slider position, we can point to that same position in the result arrays and represent it. Basically that means that if we move the slider to the hour 10:00, which is equal to minute 600, we access to position 600 of power demanded results and all the other arrays. Then once we read those values, we create a function to decide which one we are going to represent. This decision is made by clicking on the buttons placed on the map. See table 4.1. Then FloWI will represent the variables related to the clicked button in the instant showed by the slider.

To make the slider move by its own and represent a fast-motion simulation of the whole day, the code includes a function that increments its value from 0 to 1439 each 200 ms. So it is going to take 4 minutes and 54 seconds to do a whole day simulation. For making the play button initialize the simulation, it has been used the function "onclick()" which calls this function. There is also another function to hide this button and show a pause button instead. See appendix C, lines 1341-1361.

Chapter 5

IEEE European Low Voltage Test Feeder

IEEE stands for Institute of Electrical and Electronic Engineering. As part of it, there is also the IEEE PES Distribution System Analysis Subcommittee's Distribution Test Feeder Working Group. The working group began as a task force with four radial test feeders presented in 1991. Since then various test feeders have been added. [11].

This case is the one used to prove both the power flow tool OpenDSS and obtain the power flow results for this grid. And also to represent this system and its results in the web tool FloWI.

5.1 Introduction

In 2015, it was published the European Low Voltage Test Feeder. Until now, the test feeders were based on North American systems. However, it is common to see low-voltage distribution systems both radial and meshed in Europe. The purpose of this test feeder is to provide a benchmark for researchers who want to study low voltage feeders, which are common in Europe, and their mid- to long-term dynamic behaviours.

In order to evaluate the system properly and obtain good conclusions about concepts such as energy storage and photovoltaic generation, it requires an element of time to truly understand the behaviour.

So the European low voltage test feeder features are:

- The test feeder is at the voltage level of 416 V (phase-to-phase). It is the typical voltage value in European low voltage distribution systems.
- Load shapes are given with a one-minute resolution over a 24 hours period. This allow us to perform a time-series simulation in OpenDSS.
- Time-series simulation results over a one-day period and static power flow calculation results at some key moments are provided.

5.2 Description

It is a radial distribution feeder with base frequency of 50 Hz. It has one connection point to medium-voltage through a step-down transformer at substation. It steps the voltage down from 11 kV to 416 V. Then all the buses below the connection to medium-voltage point are at 416 V. The coordinates of this buses are specified at "Buscoords.csv" file.

Medium-voltage system is modelled as a voltage source with impedance. All data of this substation bus is included in the file "Source.csv". The three-phase has a rated MVA of 0.8 and delta/grounded-wye connection.

The distribution lines are defined by line codes and their length. All needed data about the lines are included in the files "Lines.csv" and "lineCodes.csv".

For the loads, they have been modelled as a constant PQ loads. Each load is defined using kW and power factor. Loads shapes are defined for time-series simulation. Each load shape is linked to a load profile that defines for each minute the multiplier value of the load. It means that the value of a load at a specific time is determined by its base kW value times the multiplier of the load profile. All needed data are included in the files "Loads.csv", "LoadShapes.csv".

As a result of all of that, the test provides the load shape, or the consumption of active and reactive power of each load for every minute during 24 hours. In this test feeder it is considered to be 905 buses, 906 lines and 55 loads.

Chapter 6

Study Cases and Results

Once the tool have been finished, some tests and simulations have been carried on to prove that it is working in the way it was expected. After the verification that it is well implemented, we can move on to different study cases, from the user point of view, based on the LV test European feeder.

For the simulation of the generation using PV panels, the data profiles have been obtained from the software GenMix, created by Felix Manuel Lorenzo Bernardo y Edwin Xavier Dominguez Gavilanes in the research group: LEMUR.

In this chapter, it is exposed the study of the variables in a load bus, in one of the main feeder lines and the system global variables. All this variables have been studied for different cases, which are:

- Base case: There are just loads simulating the consumers behaviour.
- 25% of PV generation: Here, one half of the consumers have PV panels installed.
- 75% of PV generation: Here, almost all the consumers have PV panels installed.
- 25% of PV generation and storage: Here, one half of the consumers have PV panels and batteries installed.
- 75% of PV generation and storage: Here, almost all the consumers have PV panels and batteries installed.

From the user side, the tool shows information of the nodes, the lines, distributed generation devices, and storage. This information can be studied in an individual scale and global scale. As individual, it is possible to analyse just one node or line, in order to know its active power profile, the generated power, voltage level or storage device variables. Or in a line, can be analyse the current through the line, the power, and how much time the power through the line is bigger or smaller than a determinate value. It is also possible to know in which direction the power is flowing, in case that some nodes were injecting in the grid. From a global point of view, this tool offers plenty of information for studying the impact of installing distributed generation a storage in a micro-grid. it is possible to have a global vision of how much power is consumed or generated at any time during the whole day. It is also possible to see where are the main consumptions, or the voltage level with just looking into the map were a real-time simulation is showing the behaviour of the system.

6.1 Node

The selected node for the simulation is number 36. It is because this node is located far from the slack bus, so the study of the voltage drop would be interesting. Furthermore, the decision of what buses will have PV panels and storage installed, has been made randomly and this node is one of the 25% buses that will have PV panels and storage in the following cases. This node is connected to phase B.

So, here in that node, could be interesting to see its load active power profile, which has a typical residential profile. More or less constant low consumption during the first hours of the day, then it has some increasing during the morning and after that it goes down again until the afternoon. During the late afternoon it has some peaks that emulates, the typical behaviour in an average European home. The maximum active power peak as is shown in figure 6-1 is 2.53 kW.

Then, at this node (and any), would be interesting to know its power consumption, the

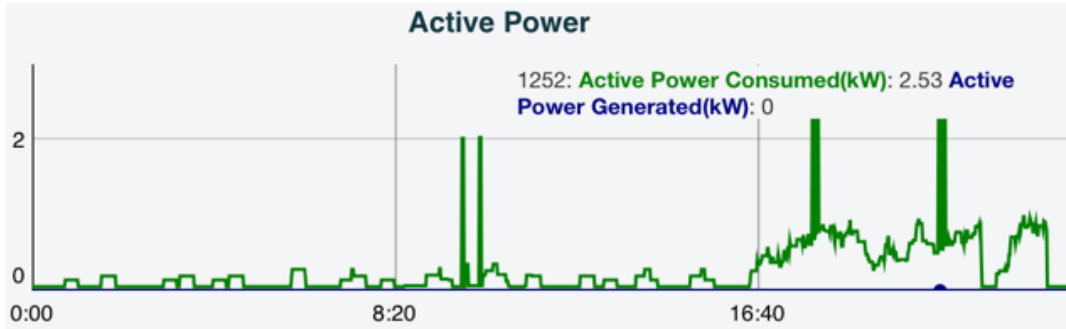


Figure 6-1: Node 36 load power profile.

generated power that in the base case, see that in figure 6-1 is zero, and its voltage level.

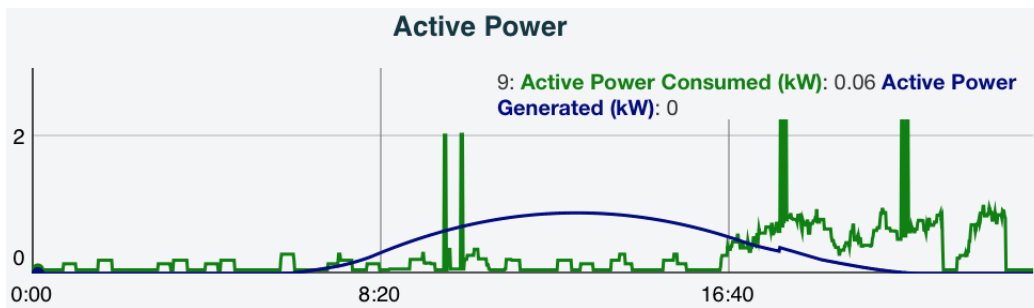


Figure 6-2: Node 36 consumed and generated power profile.

Figure 6-2 illustrates the generated power profile for the case with PV panels installed. It can be seen that the PV panels generate energy from 08:00 until 20:00 more or less. So we have 12 hours of generation. And we can appreciate in figure 6-3, which shows the active power through the feeder, the net power in the node. We can see that during the middle hours of the day, the generation overcomes the consumption and therefore, it injects power into the grid. This is one of the challenging problems for the stability of the grid that has been introduced in the chapter 2 of the present document.

However, it is obvious that, from the consumers side, this scenario is quite more interesting in economical terms, due to the reduction of a significant amount of power demanded from the grid. But, as explained in chapter 2, having only generation can not be that good as having generation plus storage. The benefits will be both for the utilities (in terms of

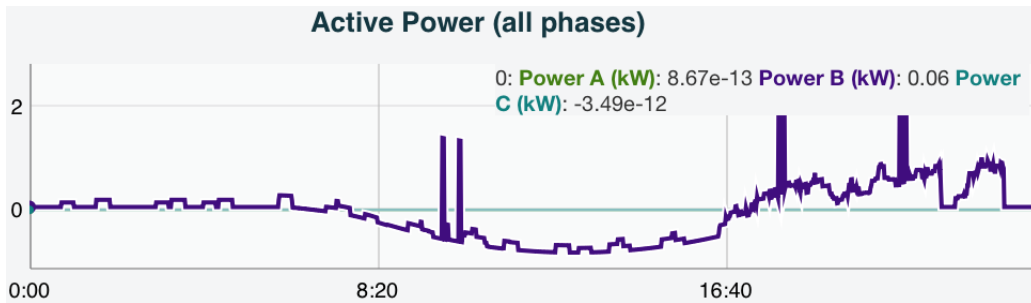


Figure 6-3: Node 36 net power that flows through the feeder.

system stability) and for the consumers (considerable bill reduction).

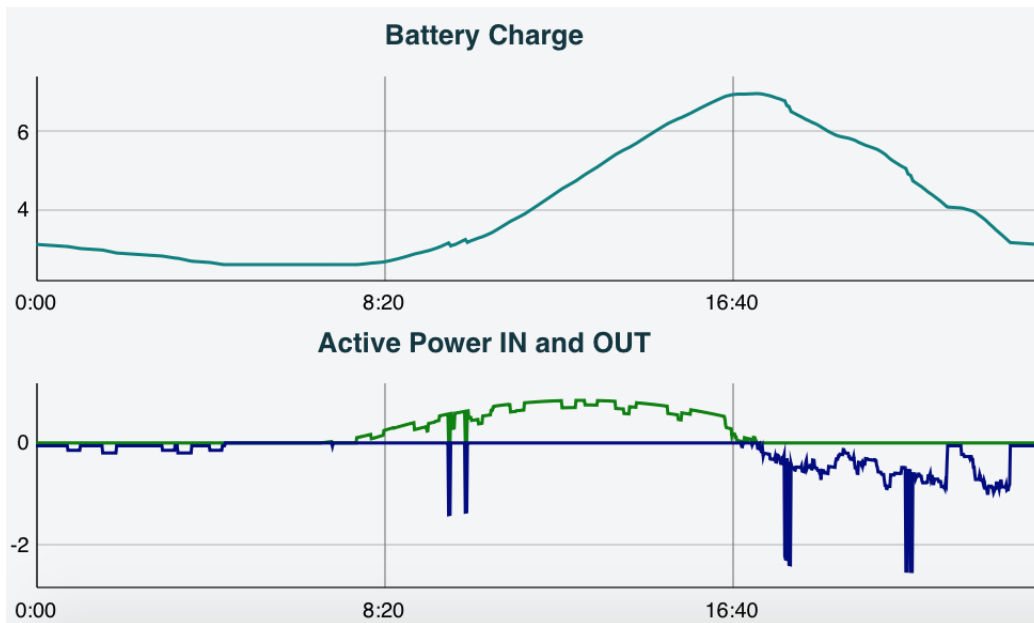


Figure 6-4: Node 36 stored energy profile and I/O power.

Figure 6-4 shows in the first graphic, the stored energy profile. The rated energy of the storage elements that have been introduced in OpenDSS is 7kWh. Which is the maximum they can store. The graphic shows how the battery increase its stored value when the generated power is bigger than the demanded. The second graphic shows the power that flows in (green) or out (blue) the storage device. It simulates that the battery has a current controller to adapt the charging power ratio. The battery discharge when the demand is bigger than the generation, and the stored value is over 20%.

For the simulation of the storage elements behaviour it has been simulated this case for various days, in such form that, the initial storage values are not the minimum, but a value obtained from many iterations. This was made with the intention of obtain a better approach to a real scenario, where the values of the batteries at the beginning of the day will be different, in function of previous days generation and consumption profiles. Then the battery of this node has a storage value of 24% at the beginning of the day.

Even if this graphic do not illustrate quite well the difference in the power demand among having or not storage devices installed, figure 6-5 illustrates it quite well. We can see that this node will only need to obtain energy from the grid during few instants in the morning, and then it is self sufficient for the rest of the day.

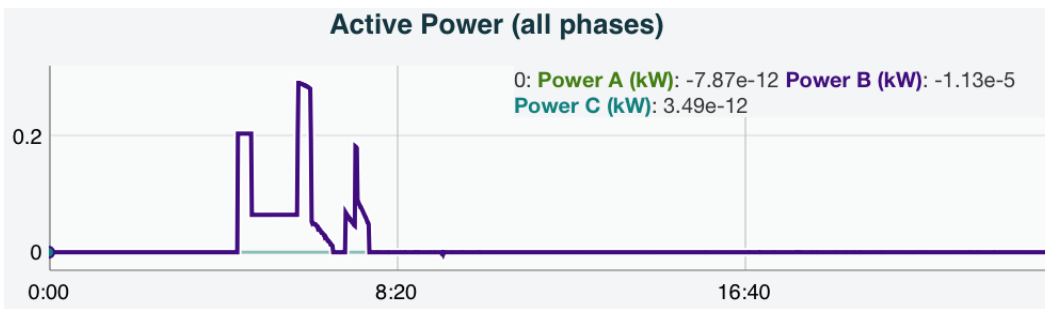


Figure 6-5: Node 36 power flowing through the feeder.

Figure 6-6 illustrates the monotone curve of power consumption in node 36 with the use of storage devices. That shows that it has been consuming power from the grid for around 200 minutes.

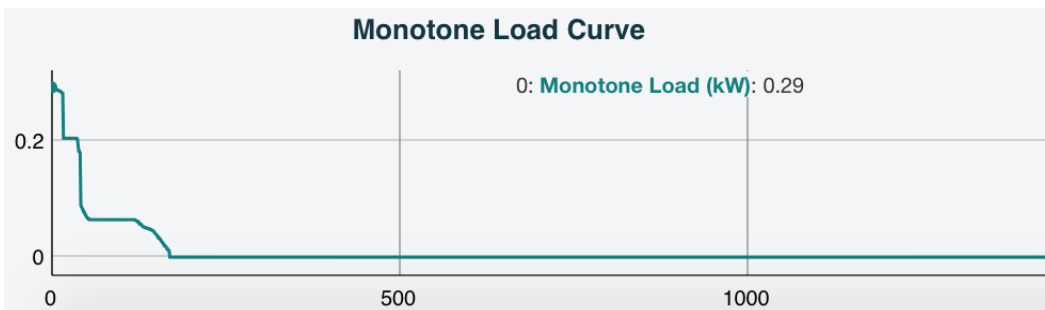


Figure 6-6: Node 36 power monotone curve.

Table 6.1: Conclusions from the node analysis.

	Base case	Generation case	Generation and storage
Self-sufficient time (min)	0	400	1200
Max power in feeder (kW)	2.4	2.4	0.3
Power injected in the grid	No	Yes	No

In table 6.1, are compared some variables. How much time the node is self-sufficient, the maximum power through the feeder that connects the node to the rest of the grid, and if at any time, some excess energy is injected in the grid.

As conclusions for this case, we have studied one node, that has a single-phase load in three different scenarios. Just with consumption, with PV panels generation and with storage devices included. It can be seen that, once there are PV panels installed, and the generation overcomes the demand, the node will become a generation node. As can be seen in next study cases, one single node injecting into the grid will not make the difference, but when the number of nodes that have PV panels installed without storage devices grows, the problem is huge. It will not be consumption enough inside this micro-grid to consumed all this generation, therefore it will be injected to the high-voltage side of the transformer, and that could cause huge damage to this device when the power is not flowing through the same direction in the three phases. We can not forget that, there will be no problems only for the utilities but for consumers as well. If the substation transformer is threatened, it will be more maintenance interventions and bills will grow. Also, all that energy, from the consumers point of view is money wasted. Unless there is a specific legislation that make sure that some utilities is going to pay for that energy, it will not come back and later, that energy will be needed after the sunset. Then, the installation of storage devices, could be an answer both for utilities and consumers. As can be seen in this node, with this load profile, and generation, it could be practically self-sufficient during all day in a sunny day like this scenario. It has positive impacts both in system stability, in reducing maintenance costs of the grid, in reducing the consumers bill, and in reducing CO₂ emissions.

6.2 Line

Analysing a line with FloWI will report information about the currents, power and losses in each of the three phases. It will be useful for the capacity studies and design. Here in this case it will be analysed one of the main feeders of the system, that will show the power exchange between the high-voltage side and the low-voltage side, and the effect of having or not generation and storage devices.

Figure 6-7, illustrates the four graphics that FloWI calculates and shows when the line is selected. The first one is the active power flowing through the three phases, the second one are the currents in the three phases, the third are the losses and finally the monotone curve, summarizing the power through the three phases of the line. With this last curve we can know how much time the power is bigger than some value, which is useful to understand the dimension of power flowing through it and for how much time. This figure, is from the base case, without generation or storage.

Then, based on those curves, we can easily understand the magnitude of the power, current and losses through the line, and in this case, through the grid, as it is part of the main feeder. Power and current, are always positive, as expected, thus there is no generation distributed among the grid and consequently it flows only in one direction.

Figure 6-8 illustrates the curves from the case where the 25% of nodes has PV panels installed. It can be rapidly seen, that, during the hours around 14:00, the power curve goes down zero. It means that during that instants, the nodes are injecting in the grid, and globally, they are overcoming the grid power demand, so the only choice is to inject it to the high-voltage side. This could be specially dangerous for the system stability and for the transformer, if one of the phases has a positive power flow (from high-voltage to low-voltage) and if in any other phase is all the way around.

It actually happens in this case, as can be seen in figure 6-9, that is a zoom-in image

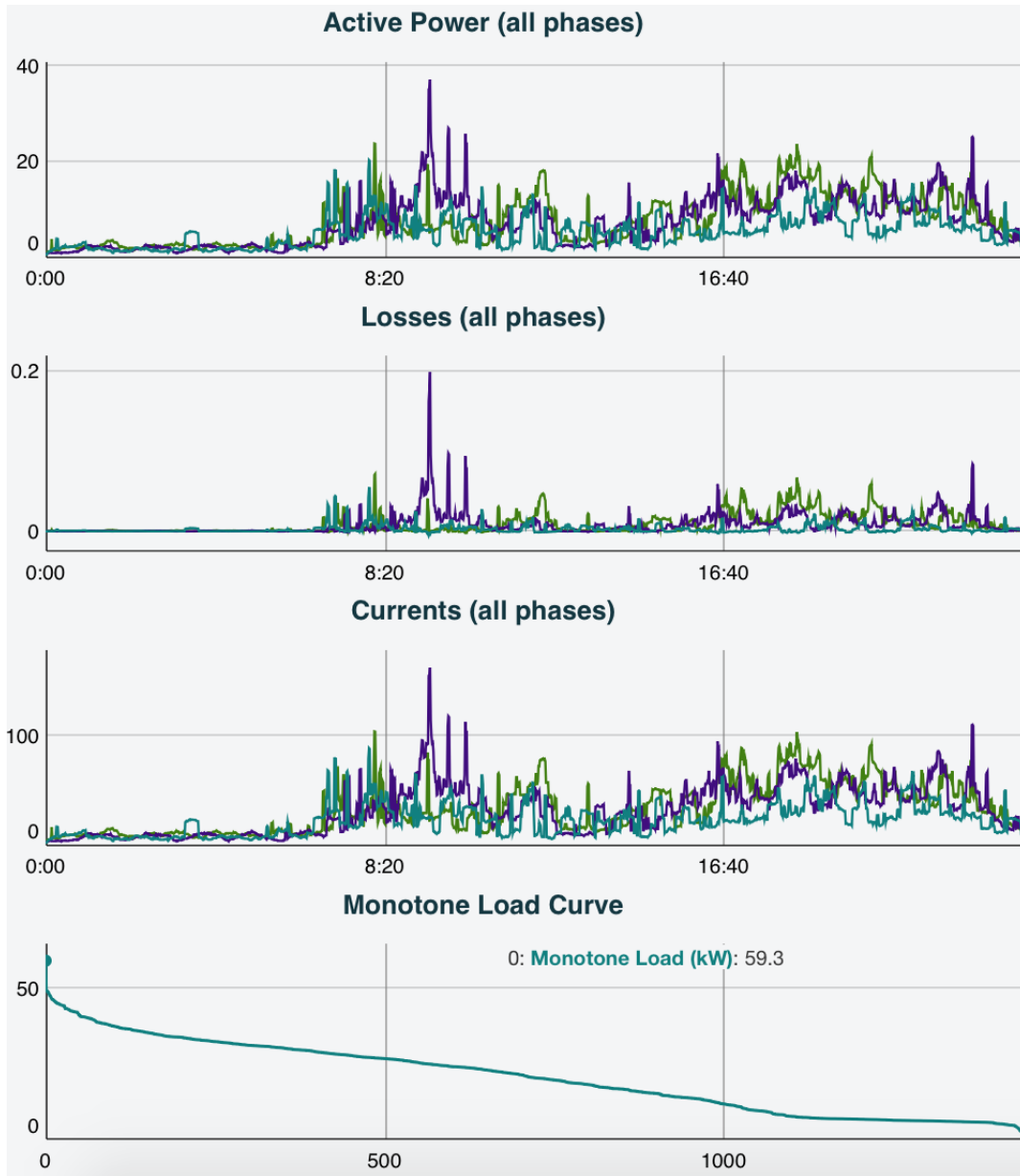


Figure 6-7: Line graphics in base case, active power (kW), Currents (A), Losses (kW) and monotone curve.

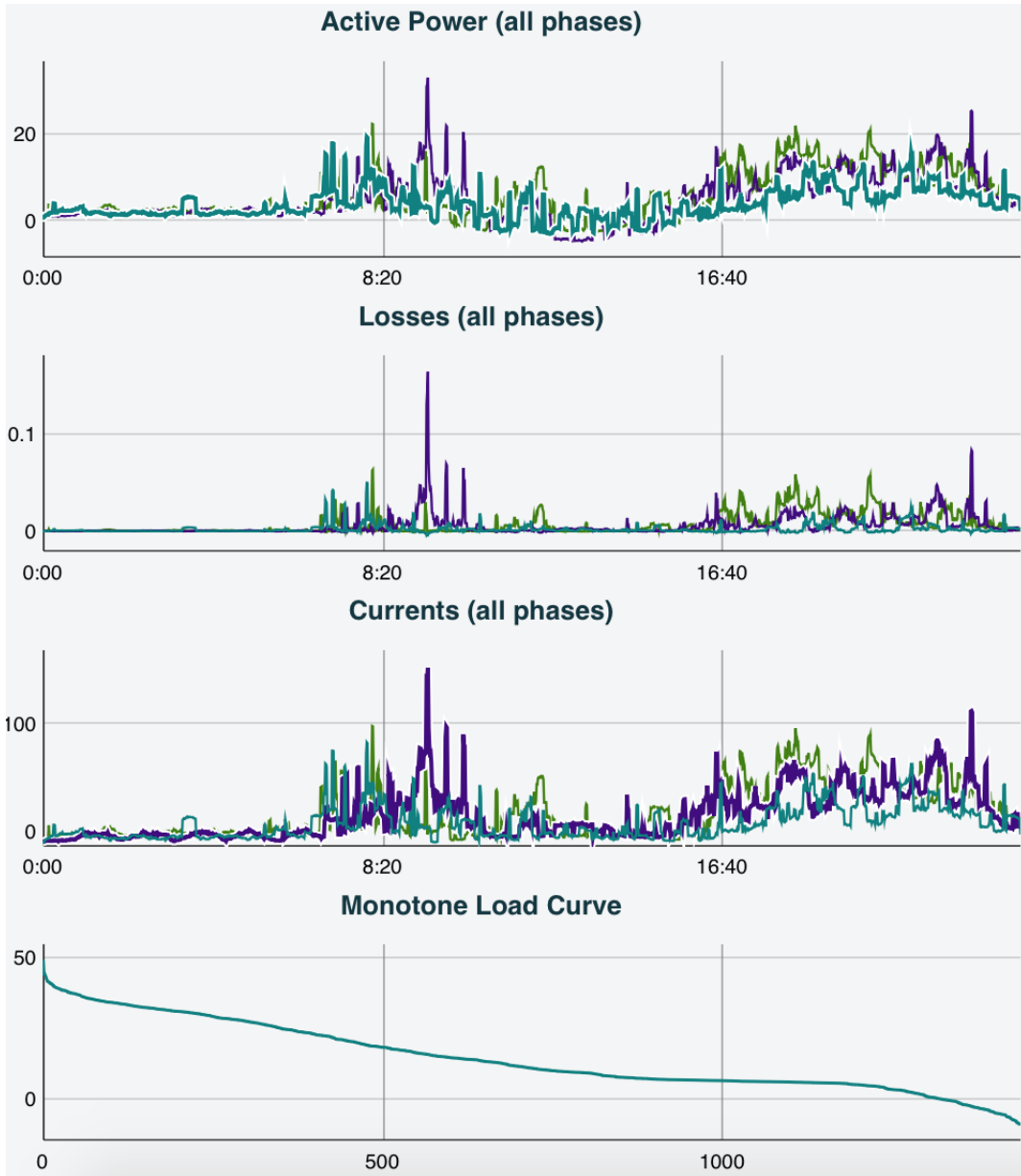


Figure 6-8: Line curves in the case of 25% of PV panels in the nodes.

of the active power in the case of 25% of PV panels installed. Here one of the phases has negative values, and the other two positive values and from time to time it happens just the opposite.

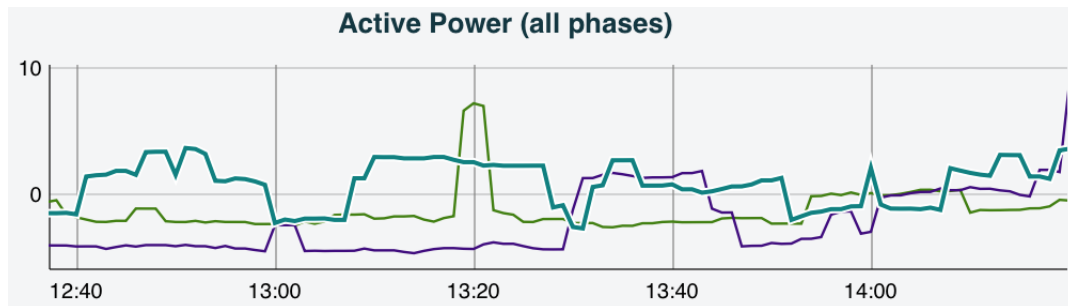


Figure 6-9: Active power flow .

Figure 6-10 illustrates the variables through the line for the case when the 75% of the nodes have PV panels installed. Comparing it to the previous case of 25% of panels installed and the base case, allows us to see that the power flowing towards the high-voltage side, has a quite important value, bigger than before. It is because there is no chance to store the excess power. But it takes only a look to understand what is happening in the line.

Figure 6-11 illustrates the behaviour for the case when there are installed PV panels and storage devices in 25% of the nodes. Here it can be immediately appreciated the effect of the storage devices, thus there is no more negative power values flowing to the high-voltage side. In fact the power flowing through the line, during that hours is smaller because excess power is now storage where it is produced.

Figure 6-12 illustrates the behaviour for the case when there are installed PV panels and storage devices in 75% of the nodes. Here, as the scenario is supposed to be a sunny day, the storage devices could get fully charged during the peak of generation and lower consumption. As can be seen, in one of the phases (phase C) there is an important power injection to the high voltage side. This phase is the one with less nodes connected to it and lower consumption. This information, suggest that maybe, some nodes connected in phases

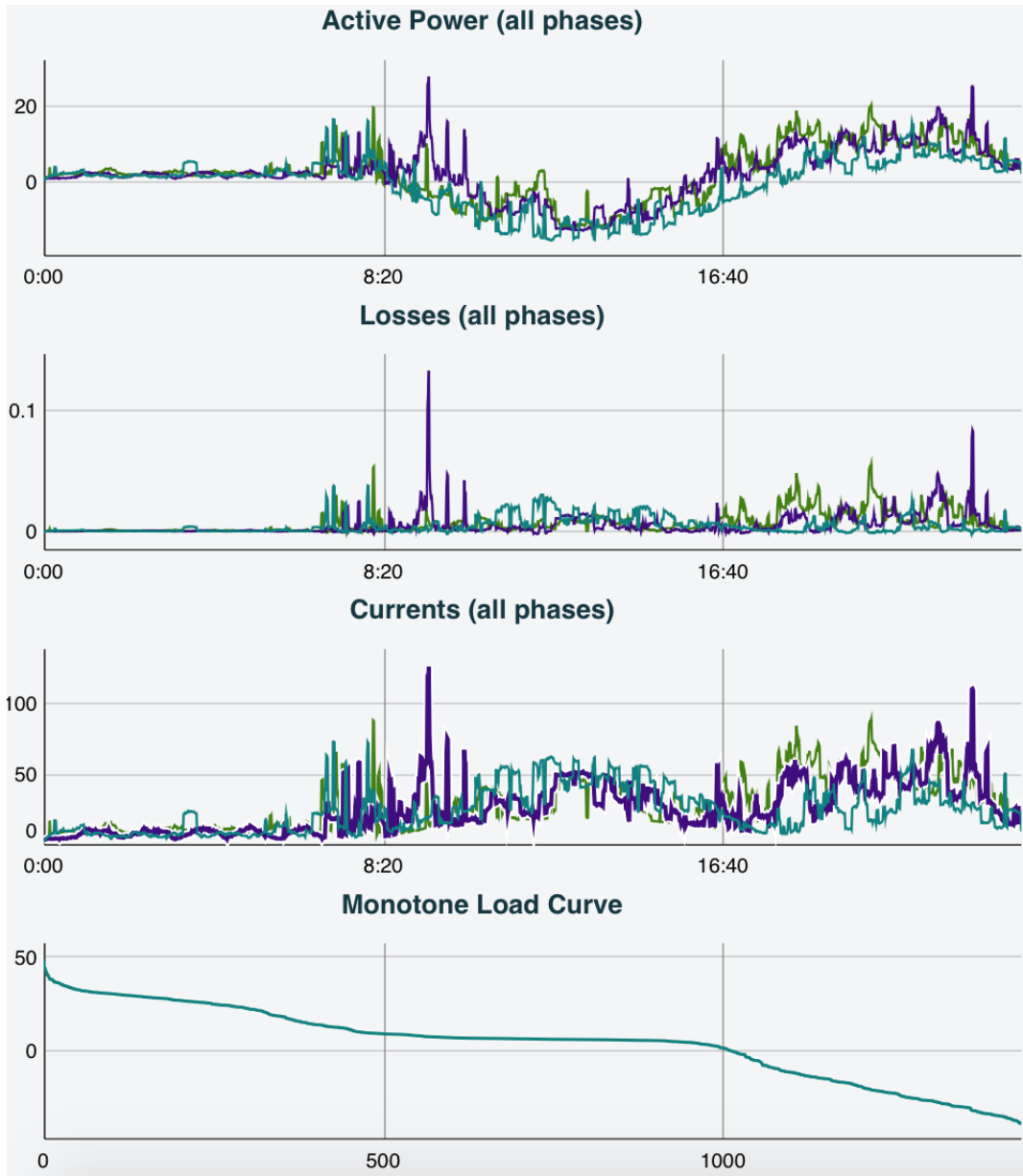


Figure 6-10: Line curves in the case of 75% of PV panels in the nodes.

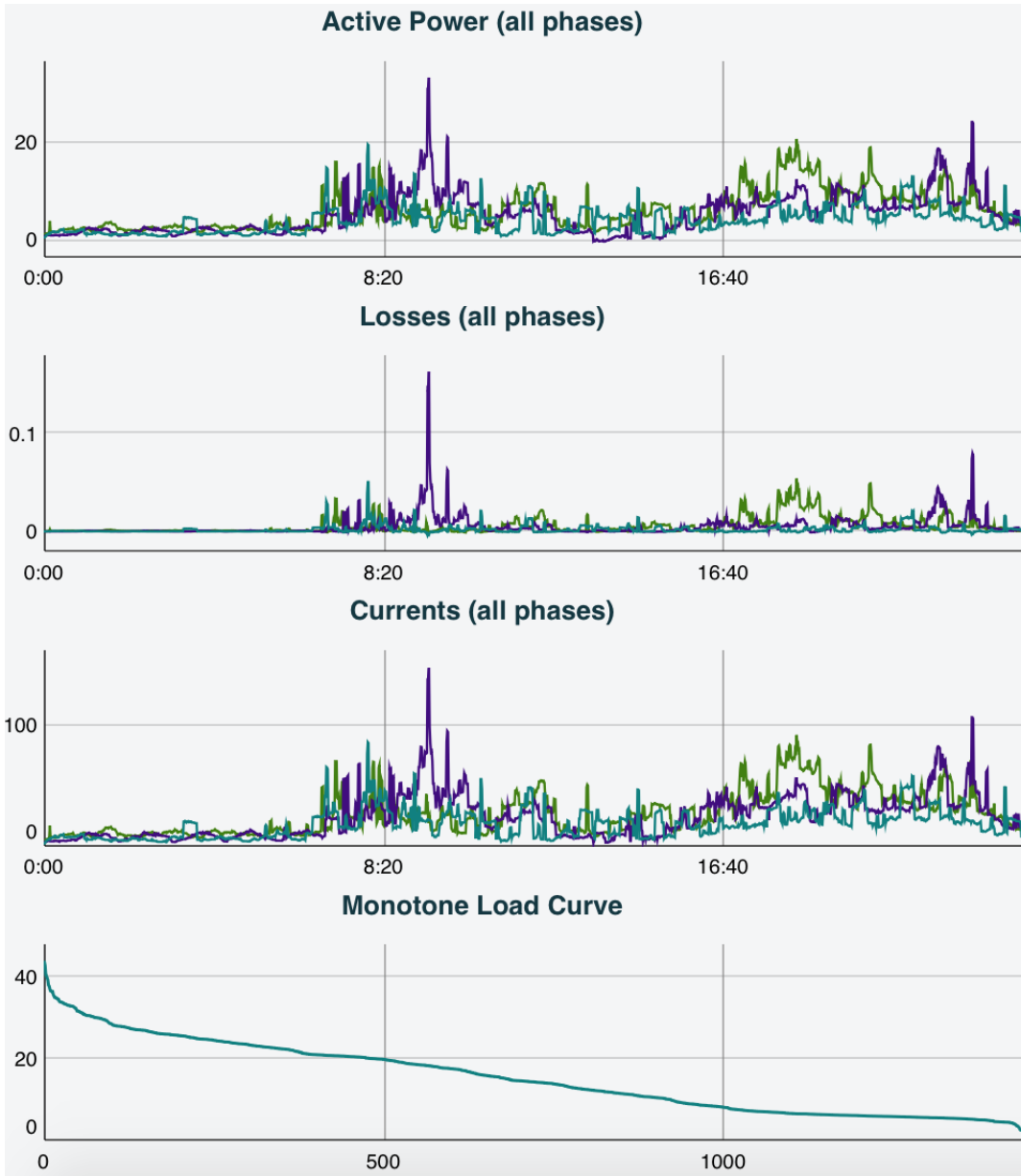


Figure 6-11: Line curves in the case of 25% of PV panels and storage in the nodes.

A and B should be instead connected to phase C, in order to balance as much as possible the grid. Also, in this case, the currents are smaller, thus the power flowing from and to the high-voltage side has been reduced. Therefore, currents through the line get smaller. Due to that fact, line dimensions could be reconsidered thus there is no need to have a line with the same characteristics as in the base case.

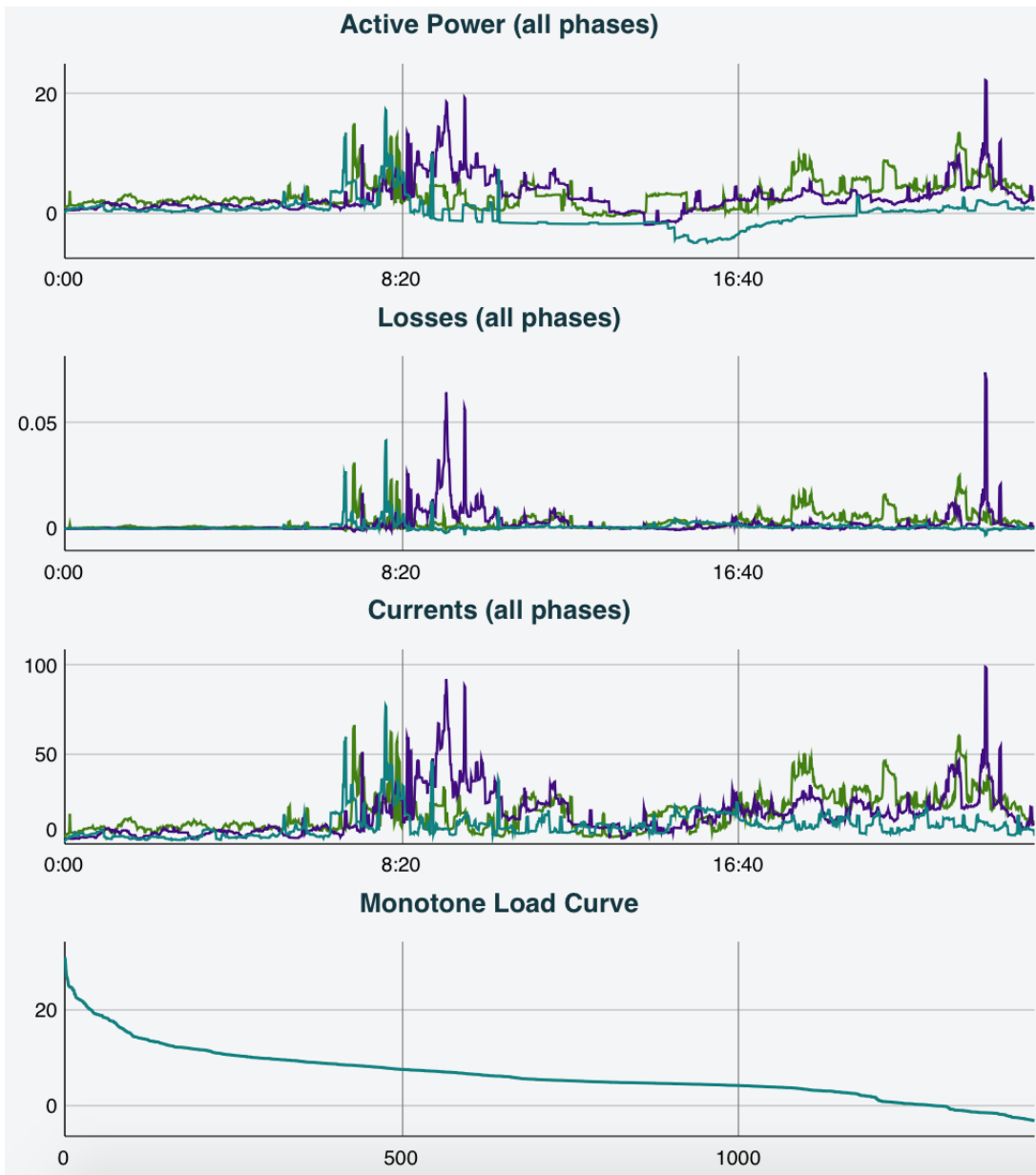


Figure 6-12: Line curves in the case of 75% of PV panels and storage in the nodes.

As conclusions from the study of the line, it has been analysed the variations in power and current through the line for the five cases. It has been clearly seen, that including much distributed generation without storage devices, could become a challenge for the utilities to keep the system stability. Furthermore, in a radial distribution like this one, a bad distribution of the consumers connected to each phase, could make that in some phases consumers have to consumed power from the utilities and however, in other phase, consumers have their storage devices 100% charged and they must inject power into the grid. Then, it could be really interesting to redistribute consumers to make the grid as balanced as possible.

6.3 Global variables

This case illustrates the global vision of the grid that FloWI can offer. Here there are analysed global variables that appear in the graphics display and also how the real-time simulation could show and help in monitoring what is happening in the grid, in terms of power consumption, generation, the voltage level, the amount of energy stored in the batteries and its status.

First, we are going to compare the graphics of the global variables for the base case, the 75% of PV panels case and the 75% of PV panels and storage devices case.

Figure 6-13 illustrates the graphics with the global variables of the base case. As can be seen, there is only power consumed, and the monotone curve is always positive so there is no power injection into the high-voltage side.

Figure 6-14 illustrates the graphics of the global variables for the case where 75% of the nodes have PV panels installed. It can be seen in the first graphic, how the generation and consumption curves are overpassed. Then, the net power illustrates where the grid is consuming power from the substation (positive values) and when it is injecting power to the high-voltage side (negative values). To understand better the magnitudes of time and

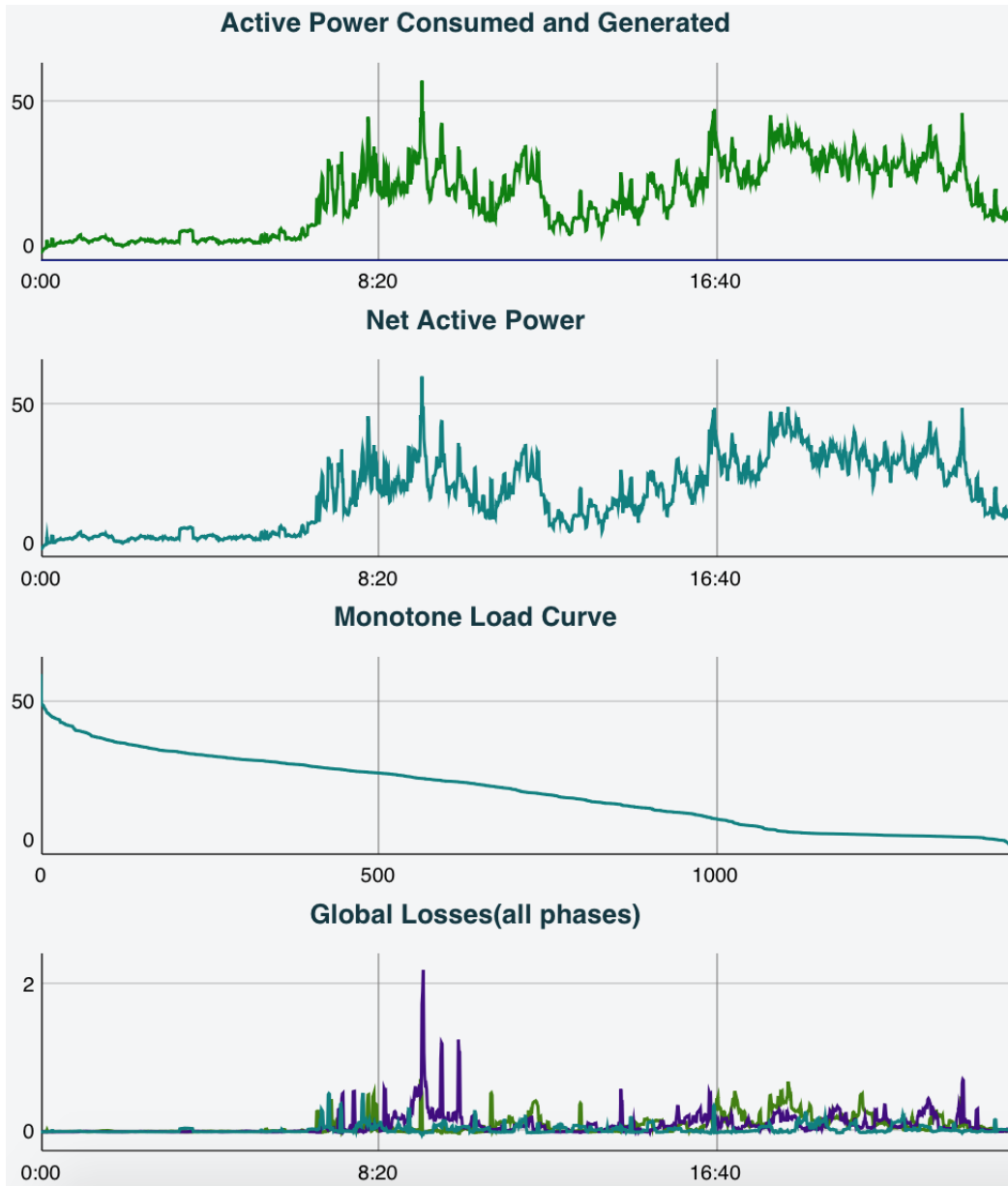


Figure 6-13: Global graphics for the base case.

power, is better looking to the monotone curve, where the power is ordered from higher value to lower value for each minute of the day. Therefore we can easily identified, that the grid is consuming power from the substation for 1000 minutes a day, and it is injecting power during the rest of the day.

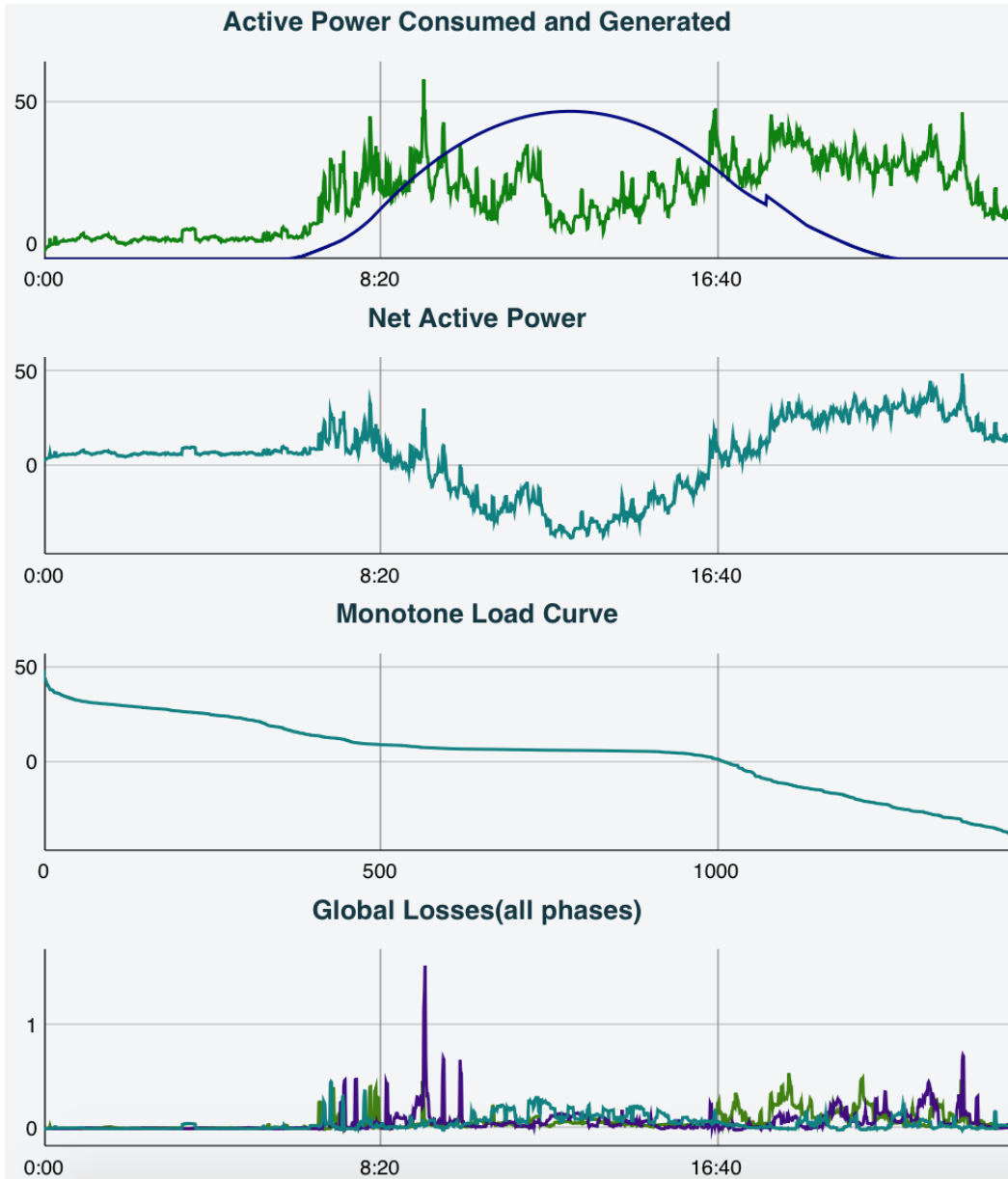


Figure 6-14: Global graphics for the 75% of nodes with PV panels case.

Figure 6-15 illustrates the graphics of the global variables for the case where 75% of the nodes have PV panels and storage devices installed. Comparing it to figure 6-14, the

net power curve is more flat, and the monotone curve shows that there is no as much power injection as in the previous case, due to the majority of the excess power is stored and used along the day.

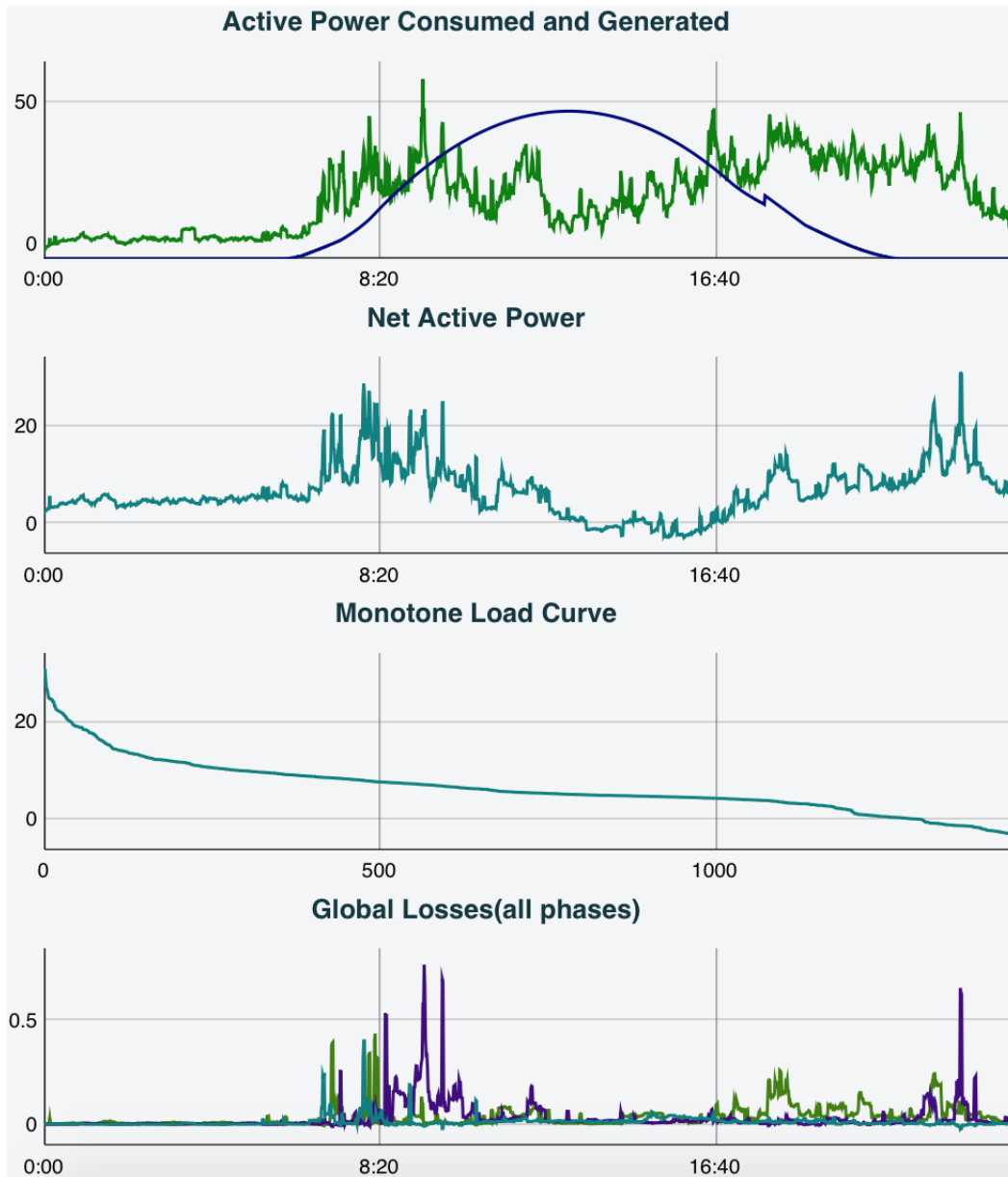


Figure 6-15: Global graphics for the 75% of nodes with PV panels and storage case.

But if looking this graphics could give a good approach of the power exchange during one day in this grid, the real-time simulation could add a lot of valuable information and data. By clicking of the play button, the minute simulation begin. And it can be configured,

as explained in chapter 4, how to visualize the different variables.

For example, the maximum peak of consumption occurs at 09:25 hours. In figure 6-16, can be seen what nodes are consuming (green circle), and the magnitude of its consumption with the circle size. This helps quite a lot to analyse these nodes separately if needed, and locate where are the biggest consumptions in the grid, and in which phase they are connected.

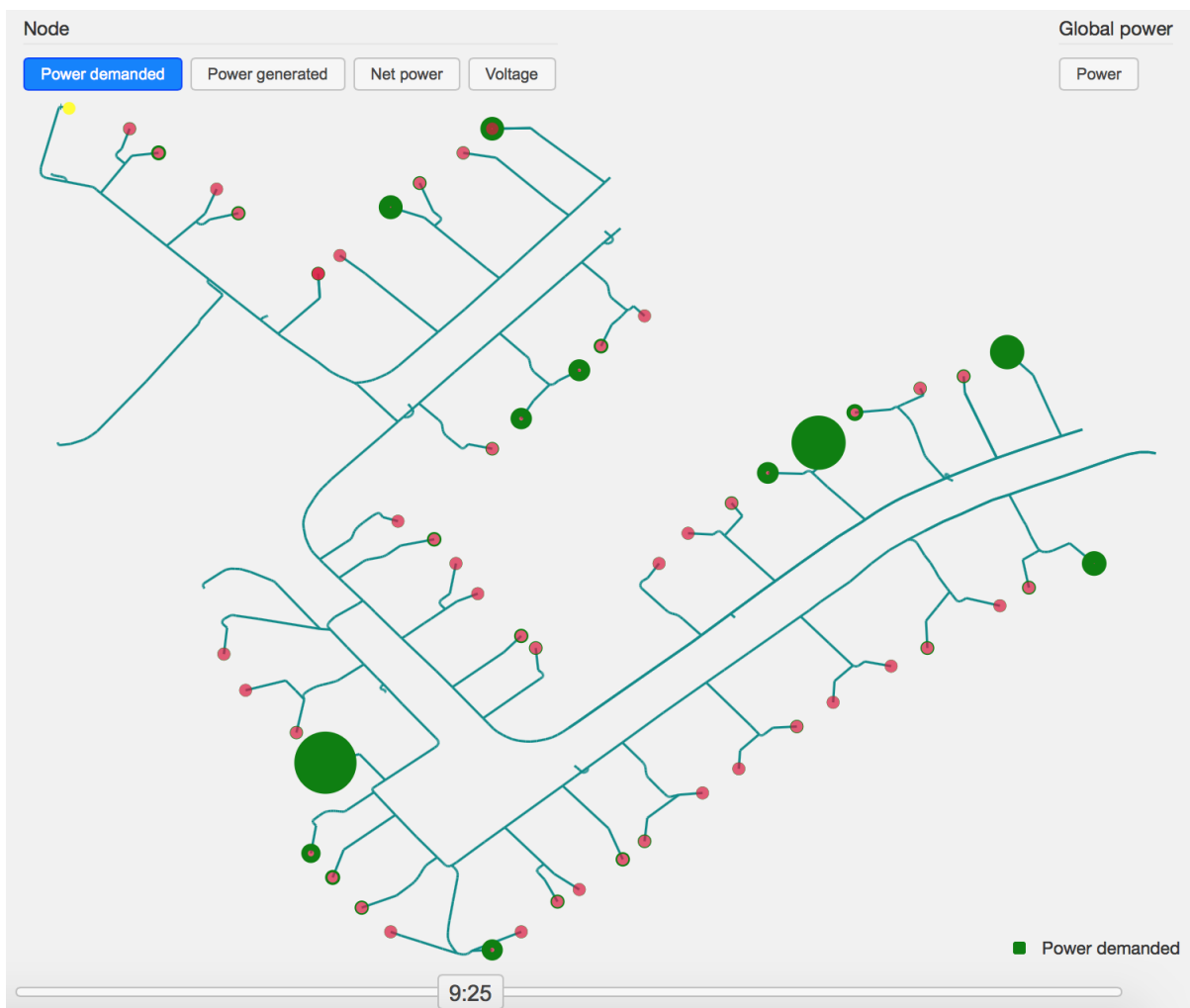


Figure 6-16: Real-time simulation of consumed power.

Then, it could be interesting to analyse the voltage level during this instant, to locate if there are nodes, with voltage values under its nominal value. Figure 6-17 illustrates, the

real time simulation of voltage level in the nodes. In the right-low corner, there is a legend, specifying the voltage ranges and its associated colors. As can be seen, there are many nodes with a voltage level below its nominal but inside the boundaries. Noticed that, the more distance from the substation (yellow node), the more nodes have low voltage levels. This will help to analyse in what phase that nodes are connected and if there is other phase with a higher voltage level, to compensate the system. In this particular case, all the marked nodes but four, are connected to phase B, which can be known by clicking on them. Therefore, it could be interesting to redistribute some of phase B nodes to other phase.



Figure 6-17: Real-time simulation of voltage level.

For the case with 75% of nodes with PV panels installed, there are still low voltages at

that time of maximum consumption, but is quite attenuated by the generated power which elevates the voltage. However, at some point during the middle hours of the day, what happens is the opposite and some nodes have voltage levels above its nominal value. Figure 6-18 illustrates this scenario. Noticed that it is possible to know in which nodes are PV panels installed looking to the sun symbol under them. In this figure it is painted in green because they are generating. When they are not generating they will be painted in black.



Figure 6-18: Real-time simulation of voltage level with PV panels installed.

It can be also possible to see which nodes are generating, with the coloured sun symbols and by clicking in the button generated power. It will show, like in the case of the consumed power, the nodes that are generating with a green circle around them, and changing its size

to represent the power magnitude. And so happens with the net power if we want to analyse if the node is consuming or injecting power in the grid.

Figure 6-19 illustrates the real-time generated power analysis.

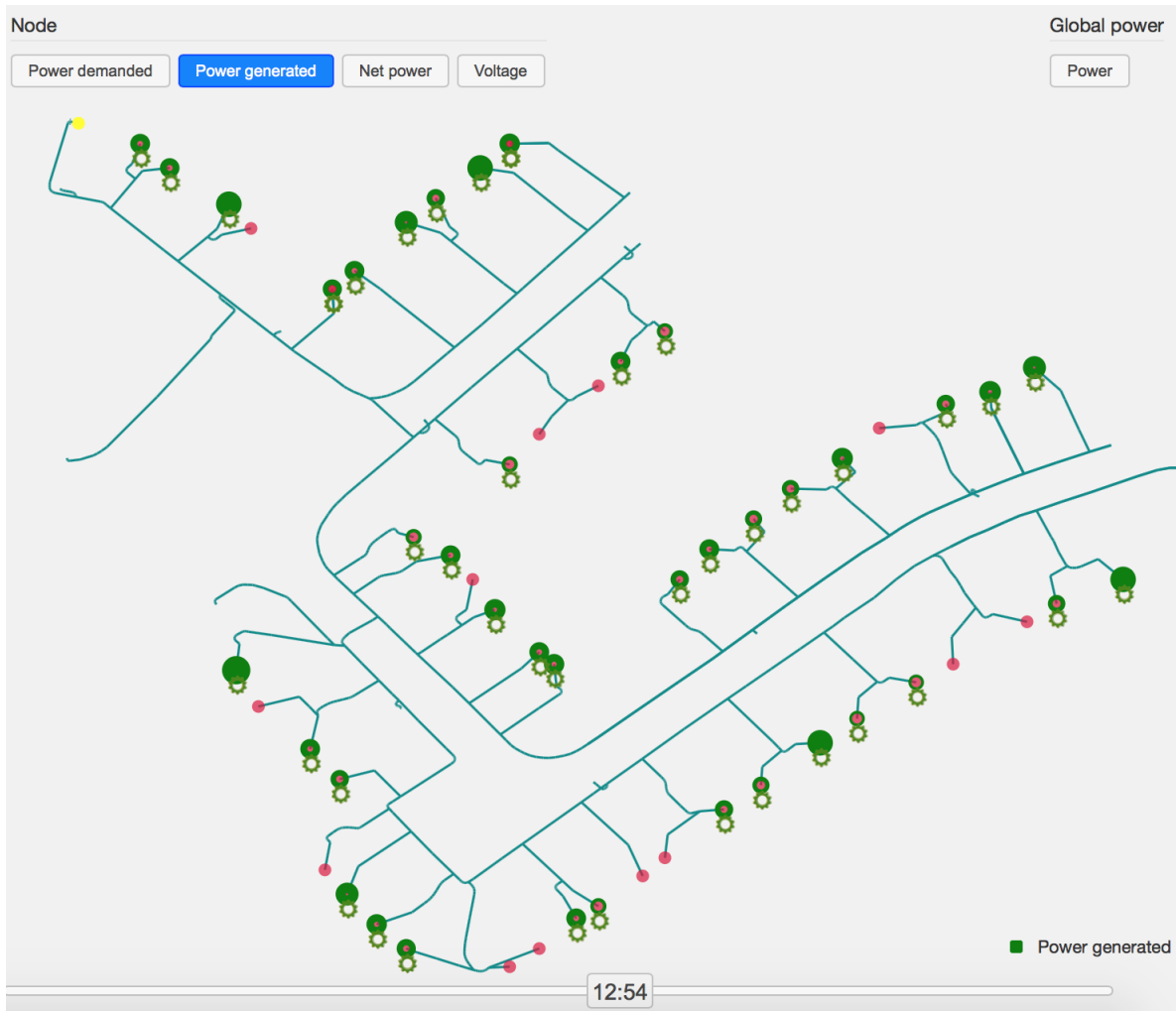


Figure 6-19: Real-time simulation of generated power with PV panels installed.

Figure 6-20 illustrates the real-time net power analysis.

Finally, in the cases where there are storage devices, it will appear a battery symbol over the node, indicating its percentage of energy stored, and its status (idling, charging and discharging). Figure 6-21 illustrates one area of the map, that it was zoomed-in. It is

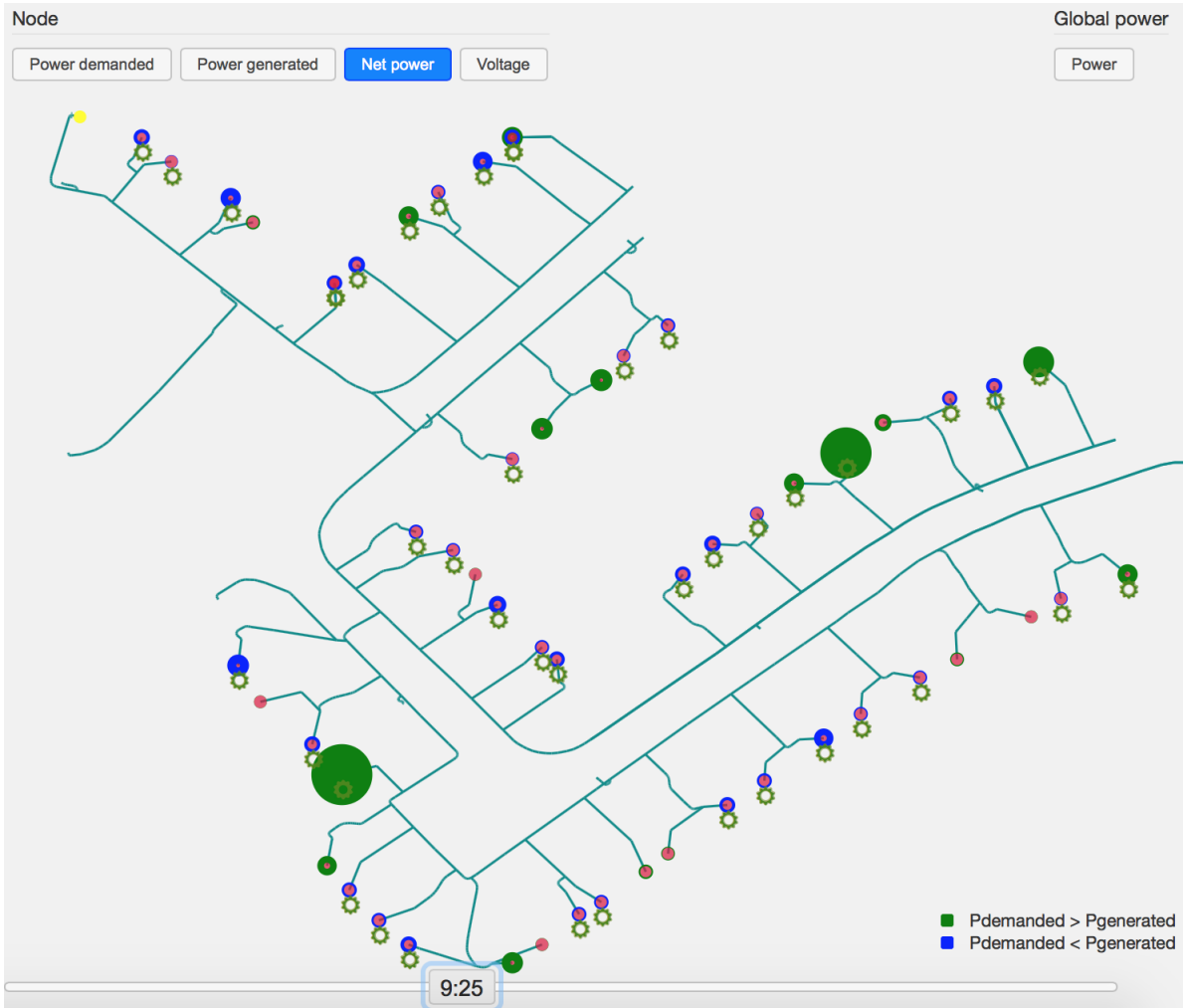


Figure 6-20: Real-time simulation of net power with PV panels installed.

possible to see the nodes, with sun symbols, representing that there are PV panels installed and with battery symbols. The batteries are either green (charging), red (discharging) and blue (idling). And as the power demanded button is clicked, the green circles around the nodes, represent that there is consumption in this precisely instant. This shows the potential of this tool. As can be monitored the data of every single element in the grid, and its behaviour.

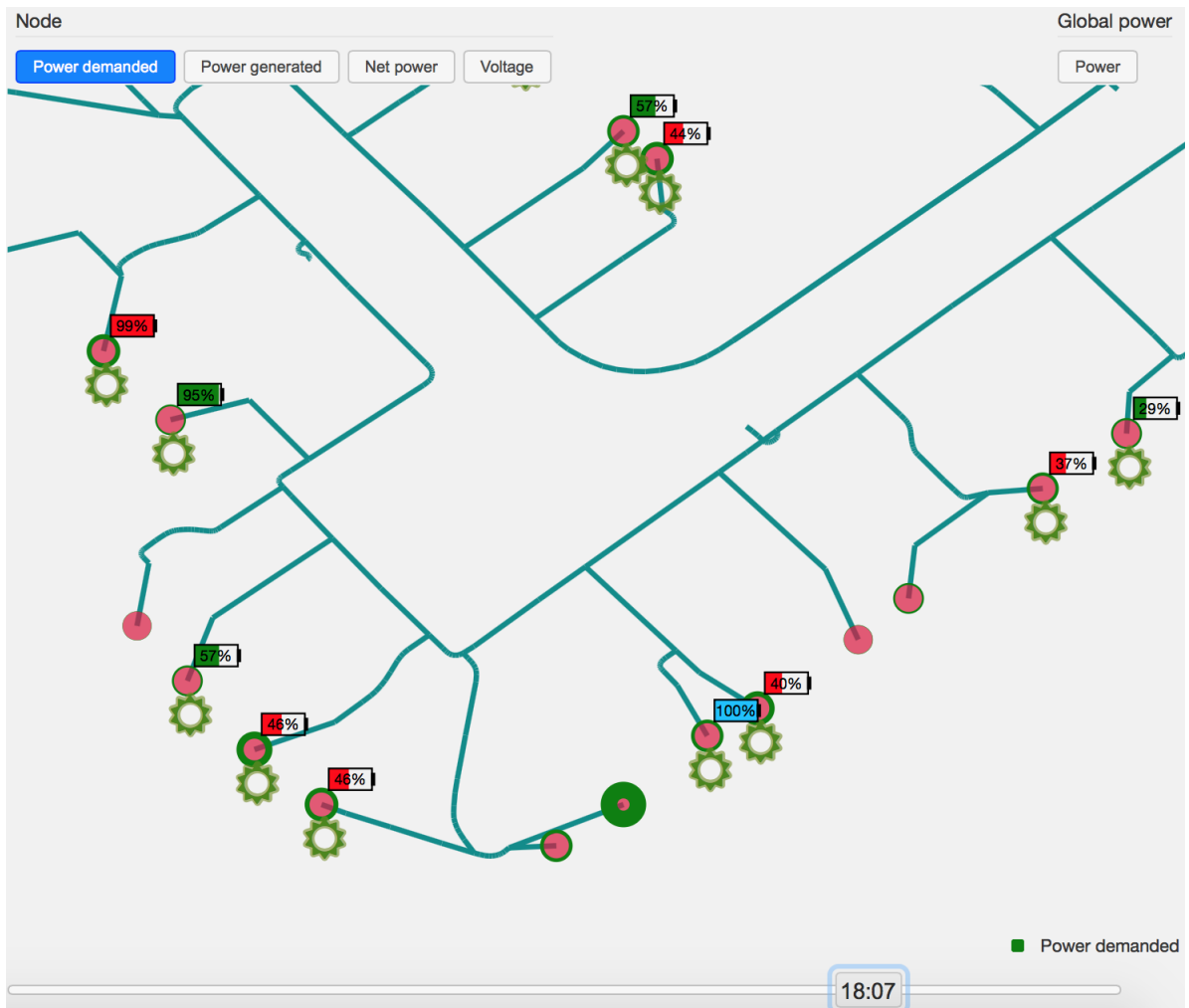


Figure 6-21: Real-time simulation of demanded power with PV panels installed and storage devices.

Table 6.2 shows a recompilation of the main concepts studied in this case and the results obtained with FloWI.

Table 6.2: Conclusions from the global variables analysis.

	Base case	Generation case	Generation and storage
Power injected into the grid	No	Yes	Yes
Time injecting to the HV-side(min)	0	430	140
Transformer threatened	No	Yes	Yes
Voltage over nominal value	No	Yes	No
Voltage below nominal value	Yes	Yes	Yes
Voltage out of boundaries	No	No	No
Global losses reduction(%)	-	15%	50%
Efficiency increase	-	Yes	Yes

To conclude this study, we have demonstrated the potential of the tool in measuring the impact of including distributed generation and storage in the distribution system. FloWI provides data about the power consumption, power generation, voltage level, power injection into the grid and storage percentage of the batteries. These features, make possible to reconsider the connections of the nodes to the three phases, to balance the system as much as possible, and to re-dimension the lines through the grid in the scenario of storage devices.

Chapter 7

Conclusions

In this thesis we have exposed a brief introduction into the residential power consumption and generation, and its importance in the electrical system and for the environment. The trend on create buildings that are almost self-sufficient (nearly-zero energy buildings), is growing fast. As the consumers that are interested in installing PV panels in their homes and storage devices. Some companies have already launch their products covering that demand. Therefore, governments and utilities, should create policies to regulate its use and move the electrical system into an industry as free of CO₂ emissions as possible.

In terms of technical aspects we have defined the load flow problem formulation and two different iterative methods to solve it, Gauss-Seidel and Newton-Raphson. Then we have presented the software used to solve the power flow for a micro grid using Eprri OpenDSS, which with using the input data we are given, it makes all the mathematical resolution and generates .csv files with the results. But, as the main purpose of this thesis is to represent real-time simulations of power flow solutions, a new web tool has been created to accomplished the desire outcome. It is called FloWI, and it read all the grid parameters, input data and results data, and creates an interactive visualization of the system, that enables the final user to measure the impact that distributed generation and storage could have in the system.

FloWI was created to be as simple and interactive as possible, creating a modern environment. his features, along with the possibility of using it in any operative system where crucial to determine that FloWI must be created as a web tool, so it can be used in any

browser. That allow us to work with this tool, as we do with modern web sites, clicking on any system element, and visualize its main parameters. Also, it can carried out real-time simulations, where the user can see the behaviour of the system in terms of power consumption, generation, voltage level or energy stored.

Finally we have study different scenarios, proving the many possibilities that FloWI has. As can be seen this kind of data representation enhance quite a lot the analysis of electrical distribution systems, and make it easier to have a modern, global vision of the grid, to control it in a better and more effective way. It allows to simulate possible scenarios, by modifying the input data, or study existent cases in order to improve them.

Chapter 8

Bibliography

- [1]: William H. Kersting. "Distribution system, modelling and analysis".
- [2]: Felix Manuel Lorenzo Bernardo, Edwin Xavier Dominguez Gavilanes. "Computer Tool for assessing the power generation-mix in residential Nearly Zero-Energy Buildings".
- [3]: Pablo Arboleya, Miguel Huerta, Bassam Mohamed, Cristina Gonzalez-Moran, Dept. of Electrical Engineering, University of Oviedo, Gijon, Spain, Xavier Dominguez, Dept. of Automation and Control, national poly-technical school of Quito, Ecuador "Assessing the Effect of nearly-Zero Energy Buildings on Distribution Systems by Means of Quasi-Static Time Series Power Flow Simulations".
- [4]: "Red Electrica de España annual report".
- [5]: C.S. Antonio y col. GENERACION DISTRIBUIDA, AUTOCONSUMO Y REDES INTELIGENTES: UNED - Universidad Nacional de Educacion a Distancia, 201.
- [6]: Rasmus Luthander y col. "Photovoltaic self-consumption in buildings: A review".
- [7]: Antonio Gomez-Exposito, Antonio J. Conejo, Claudio Cañizares. "Electric Energy Systems Analysis and Operation".
- [8]: R. C. Dugan, "Reference guide: The open distribution system simulator (openss)," Electric Power Research Institute, Inc, vol. 7, 2012.
- [9]: R. Dugan, "Openss storage element and storage controller element," 2010.
- [10]: Scott Murray. "Interactive Data Visualization for the web".
- [11]: IEEE, "The IEEE European Low Voltage Test Feeder".

Appendix A

HTML Source Code

```
1 <!DOCTYPE html>
2 <html>
3
4 <head>
5 <meta charset="utf-8">
6
7 <meta name="viewport" content="width=device-width, initial-
  scale=1">
8
9 <!-- Latest compiled and minified CSS -->
10 <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com
  /bootstrap/3.3.7/css/bootstrap.min.css">
11
12 <!-- jQuery library -->
13 <script src="https://ajax.googleapis.com/ajax/libs/jquery
  /3.1.1/jquery.min.js"></script>
14
15 <!-- Latest compiled JavaScript -->
```

```

16 <script src="https://maxcdn.bootstrapcdn.com/bootstrap
    /3.3.7/js/bootstrap.min.js"></script>
17
18 <script src="https://use.fontawesome.com/b4f3cf1867.js"></
    script>
19
20 <script src="./js/papaparse.min.js"></script>
21 <script src="./js/jszip.min.js"></script>
22 <script src="./js/async.min.js"></script>
23 <script src="./js/d3.v3.js"></script>
24 <script src="./js/underscore-min.js"></script>
25 <script src="./js/jquery-3.2.0.js"></script>
26 <script src="./js/jquery-ui.js"></script>
27 <script src="./js/dygraph.min.js"></script>
28 <link rel="stylesheet" href="./js/jquery-ui.css">
29 <link rel="stylesheet" href="./js/dygraph.css">
30 <!-- <link rel="stylesheet" href="./js/jquery-ui.
    structure.css"></link>
31 <link rel="stylesheet" href="./js/jquery-ui.theme.css"></
    link> -->
32 <!-- Libreria para cambiar a formato minutil -->
33 <script src="./js/numeral.min.js"></script>
34 <!-- Libreria de estilos para los botones del mapeado -->
35
36 <script src="./js/code.js"></script>
37 <link rel="stylesheet" href="estilos.css">
38
39 <!-- Prueba para botones del mapa -->
40 <script>
41 $(function () {

```



```

42 $("input[type='radio']").checkboxradio({
43   icon: false
44 });
45 });
46
47 </script>
48
49 <title>FloWI</title>
50
51 </head>
52
53 <!-- programa de prueba para la web con bootstrap -->
54
55 <body>
56
57
58
59
60
61
62
63
64 <div class="row full-height">
65 <div class="col-sm-5 full-height">
66
67 <div class="cabecera">
68 <!-- Boton para zip -->
69 <div class="menu" id="upload">
70 <input type="file" name="" id="file" onchange="uploadZip(
      event)">

```

```

71 <label for="file"><i class="fa fa-folder-open fa-2x" aria-
    hidden="true"></i></label>
72
73 </div>
74 <!-- boton para empezar simulacion -->
75 <div class="menu" id="simulate">
76 <button class="button button3" onclick="startSimulation()">
77 <i class="fa fa-play-circle fa-2x" aria-hidden="true"></i>
78 </button>
79 <div id="pause">
80 <button class="button button4" onclick="stopSimulation()">
81 <i class="fa fa-pause-circle fa-2x" aria-hidden="true"></i>
82 </button>
83 </div>
84
85 </div>
86
87 <div class="menu" id="node">
88 <form>
89 <div class="form-group">
90 <label for="usr"><span class="nodecolor">Node</span></label>
91 <input type="number" class="form-control" id="load_index"
    min="1" max="56" onchange="SelectNodeFromButton()">
92 </div>
93 </form>
94 </div>
95
96 <div class="menu" id="phase">
97 <form>
98 <div class="form-group">

```

```

99 <!-- Selector de fase -->
100 <label for="sel1"></label>
101 <input class="form-control" id="sel1" onchange="SelectLoad()
    ">
102 <!--<option>Phase:</option>
103 <option>Phase:A</option>
104 <option>Phase:B</option>
105 <option>Phase:C</option>-->
106 </div>
107 </form>
108 </div>
109
110 <div class="menu" id="line">
111 <form>
112 <div class="form-group">
113 <label for="usr"><span class="nodecolor">Line</span></label>
114 <input type="number" class="form-control" id="line_index"
    onchange="SelectLineFromButton()">
115 </div>
116 </form>
117 </div>
118
119 </div>
120
121 <div class="graficas">
122 <div class="graph" id="P">
123 </div>
124
125 <div class="graph" id="Q">
126 </div>

```

```

127
128 <div class="graph" id="V">
129 </div>
130
131 <div class="graph" id="delta">
132 </div>
133 </div>
134
135
136
137 </div>
138
139 <div class="col-sm-7 full-height" id="columnaizda">
140 <div class="botonesmapa">
141 <div class="widget1">
142 <fieldset>
143 <legend>Node</legend>
144 <label for="radio-1">Power demanded</label>
145 <input type="radio" name="radio-1" id="radio-1" onclick="
      DibujaP () ">
146 <label for="radio-2">Power generated</label>
147 <input type="radio" name="radio-1" id="radio-2" onclick="
      DibujaPgen () ">
148 <label for="radio-3">Net power</label>
149 <input type="radio" name="radio-1" id="radio-3" onclick="
      DibujaPneta () ">
150 <label for="radio-4">Voltage</label>
151 <input type="radio" name="radio-1" id="radio-4" onclick="
      DibujaV () ">
152 </fieldset>

```

```

153 </div>
154 <div class="widget2">
155 <fieldset>
156 <legend>Global power</legend>
157 <label for="radio-5">Power</label>
158 <input type="radio" name="radio-1" id="radio-5" onclick="
      DibujaPotenciasGlobales()">
159 <!--<label for="radio-6">Current</label>
160 <input type="radio" name="radio-1" id="radio-6">
161 <label for="radio-7">Losses</label>
162 <input type="radio" name="radio-1" id="radio-7">-->
163 </fieldset>
164 </div>
165 </div>
166
167 <div id="map">
168
169 <div id="legend-voltage">
170 <div class="legend-color">
171 <svg class="circulo" height="8" width="8">
172 <circle cx="4" cy="4" r="5" stroke="" stroke-width="" fill="
      hsla(26, 100%, 54%, 1)" />
173 </svg>
174 <svg class="circulo" height="8" width="8">
175 <circle cx="4" cy="4" r="5" stroke="" stroke-width="" fill="
      hsla(28, 50%, 70%, 1)" />
176 </svg>
177 <svg class="circulo" height="8" width="8">
178 <circle cx="4" cy="4" r="5" stroke="" stroke-width="" fill="
      hsla(200, 50%, 70%, 1)" />

```

```

179 </svg>
180 <svg class="circulo" height="8" width="8">
181 <circle cx="4" cy="4" r="5" stroke="" stroke-width="" fill="
      hsla(249, 100%, 19%, 1)" />
182 </svg>
183 </div>
184 <div class="legend-text">
185 <li>
186 <ul>V &ge; 1.1 p.u.</ul>
187 <ul>V &ge; 1.02 p.u.</ul>
188 <ul>V &le; 0.98 p.u.</ul>
189 <ul>V &le; 0.9 p.u.</ul>
190 </li>
191 </div>
192 </div>
193
194 <div id="legend-netpower">
195 <div class="legend-color">
196 <svg class="circulo" height="8" width="8">
197 <circle cx="4" cy="4" r="5" stroke="" stroke-width="" fill="
      green" />
198 </svg>
199 <svg class="circulo" height="8" width="8">
200 <circle cx="4" cy="4" r="5" stroke="" stroke-width="" fill="
      blue" />
201 </svg>
202 </div>
203 <div class="legend-text">
204 <li>
205 <ul>Pd demanded &gt; Pgenerated</ul>

```

```

206 <ul>Pdemaned &lt; Pgenerated </ul>
207 </li >
208 </div >
209 </div >
210
211 <div id="legend-genpower">
212 <div class="legend-color">
213 <svg class="circulo" height="8" width="8">
214 <circle cx="4" cy="4" r="5" stroke="" stroke-width="" fill="
      green" />
215 </svg>
216 </div >
217 <div class="legend-text">
218 <li >
219 <ul>Power generated </ul>
220 </li >
221 </div >
222 </div >
223
224 <div id="legend-conpower">
225 <div class="legend-color">
226 <svg class="circulo" height="8" width="8">
227 <circle cx="4" cy="4" r="5" stroke="" stroke-width="" fill="
      green" />
228 </svg>
229 </div >
230 <div class="legend-text">
231 <li >
232 <ul>Power demanded </ul>
233 </li >

```

```

234 </div>
235 </div>
236
237 </div>
238
239 <!-- Introduccion del boton de slider -->
240 <div id="slider">
241 <div id="custom-handle" class="ui-slider-handle"></div>
242 </div>
243
244 </div>
245
246 </div>
247 <script>
248 var w = $("body").width();
249 $("#pause").hide();
250 $(document).ready(function () { // Script
    para el boton pause del slider
251 $(".button3").click(function () {
252 $(".button3").hide();
253 $("#pause").show();
254 });
255 });
256 $(document).ready(function () {
257 $("#pause").click(function () {
258 $("#pause").hide();
259 $(".button3").show();
260 });
261 });
262 </script>

```


263 </body>

264

265 </html>

Appendix B

CSS Source Code

```
1  html ,body{
2  height: 100%;
3  margin: 0px 0px 0px 0px;
4  padding: 0px 0px 0px 0px;
5  }
6  .full-height{
7  height: 100%;
8  background-color: #F1F1F1;
9  }
10 #columnaizda{
11 margin-left: -15px;
12 background-color: #F1F1F1;
13 }
14 #map{
15 text-align: center;
16 height:88%;
17 }
18 #legend-voltage ,
19 #legend-netpower ,
```

```
20 #legend-conpower ,
21 #legend-genpower{
22 display: none;
23 position: absolute;
24 right: 2%;
25 bottom: 5%;
26 font-size: 11px;
27 line-height: 4px;
28 }
29
30 .legend-color{
31 position: relative;
32 right: -30px;
33 bottom: 2px;
34 }
35
36 .circulo{
37 display: block;
38 margin-bottom: 6px;
39 }
40
41 .legend-color ,
42 .legend-text{
43 display: inline;
44 text-align: left;
45 float: left;
46 }
47
48 li{
49 list-style: none;
```

```
50 }
51
52 .menu{
53 text-align: center;
54 margin:2%;
55 margin-left: 15px;
56 padding-top: 5px;
57 height:30px;
58 float:left;
59 max-width: 100px;
60 }
61 #simulate{
62 margin-top:22px;
63 }
64 #node ,
65 #line{
66 margin-top: 0px;
67 color: black;
68 }
69 #phase{
70 margin-top: 4.5px;
71 color: black;
72 }
73 .nodecolor{
74 color: white;
75 font-size: 14px;
76 }
77
78 .graficas{
79 border-bottom-width: 3px;
```

```
80 border-bottom-style: solid;
81 border-bottom-color: #545454;
82 border-left-width: 3px;
83 border-left-style: solid;
84 border-left-color: #545454;
85 }
86 .graph{
87 text-align: center;
88 border-right-width: 3px;
89 border-right-style: solid;
90 border-right-color: #545454;
91 height: 25%;
92 display: block;
93 position: relative;
94 top: 0px;
95 margin-left: -20px;
96 }
97 #file{
98 opacity: 0;
99 overflow: hidden;
100 }
101 #upload{
102 width: 50px;
103 margin-top: 7px;
104 color: white;
105 }
106 #upload:hover{
107 color: deepskyblue;
108 }
109 .button {
```

```
110 border: none;
111 color: white;
112 text-align: center;
113 text-decoration: none;
114 display: inline-block;
115 font-size: 16px;
116 -webkit-transition-duration: 0.4s; /* Safari */
117 transition-duration: 0.4s;
118 cursor: pointer;
119 }
120 .button3,
121 .button4 {
122 color: white;
123 background-color: #545454;
124 }
125 .button3:hover {
126 color: limegreen;
127 }
128 .button4:hover {
129 color: yellow;
130 }
131 .cabecera{
132 height: 10%;
133 background-color: #545454;
134 margin-bottom: 0px;
135 }
136 .graficas {
137 height: 90%;
138
139 }
```

```

140 /* Slider */
141 #custom-handle {
142 width: 3em;
143 height: 1.6em;
144 top: 50%;
145 margin-top: -.8em;
146 text-align: center;
147 line-height: 1.6em;
148 }
149 #slider{
150 width: 95%;
151 position: relative;
152 top: 0px;
153 height: 1%;
154 }
155 /* Botones del mapa*/
156 .botonesmapa{
157 height: 9%;
158 position: relative;
159 top: 10px;
160 }
161 .botonesmapa legend{
162 font-size: 12px;
163 }
164 .botonesmapa label{
165 font-size: 10px;
166 margin-top: -20px;
167 }
168 .widget1 {
169 margin-left: 5px;

```



```

170 float: left;
171 height: 100%;
172 }
173 .widget2{
174 margin-right: 5px;
175 float: right;
176 height: 100%;
177 }
178 /* Estilos graficas */
179 #P,
180 #V,
181 #Q,
182 #delta{
183 background-color: #F4F5F6
184 }
185
186
187 /* Estilos para las graficas de Potencia Activa */
188 .dygraph-legend{
189 display: in-line;
190 }
191 .dygraph-title {
192 font-family: sans-serif;
193 font-size: 13px; /*text-shadow: gray 2px 2px 2px; */
194 margin-left: 7px;
195 color: #113641;
196 }
197 .dygraph-axis-label-y{
198 font-family: sans-serif;
199 font-size: 10px ;

```

```
200 }
201 .dygraph-axis-y{
202 margin-left: 2px;
203 }
204 .dygraph-axis-label-x{
205 font-family: sans-serif;
206 font-size: 10px ;
207 }
```

Appendix C

JavaScript Source Code

```
1 // Variables iniciales
2 var reader = new FileReader();           // reader class (no
    la usamos)
3 var zip = new JSZip();                   // zip class
4 var selected_load = null;
5 var position = null;
6 var sliderP = [];
7 var sliderPgen = [];
8 var sliderPneta = [];
9 var sliderV = [];
10 var sliderBat = [];
11 var sliderBatState = [];
12 var botonP;
13 var botonPgen;
14 var botonPneta;
15 var botonV;
16 var valorslider = 0;
17 var stop;
18 var selected_line = null;
```

```
19 var selected_bat = null;
20
21 // Parametros de configuracion de lectura de datos
22 var config = {
23   delimiter: ",", // auto-detect
24   newline: "", // auto-detect
25   quoteChar: '"',
26   header: true,
27   dynamicTyping: true,
28   preview: 0,
29   encoding: "",
30   worker: false,
31   comments: "#",
32   step: undefined,
33   complete: undefined,
34   error: undefined,
35   download: false,
36   skipEmptyLines: true,
37   chunk: undefined,
38   fastMode: undefined,
39   beforeFirstChunk: undefined,
40   withCredentials: undefined
41 };
42 var config_2 = {
43   delimiter: ",", // auto-detect
44   newline: "\n",
45   quoteChar: '"',
46   header: true,
47   dynamicTyping: true,
48   preview: 0,
```

```

49 encoding: "",
50 worker: false ,
51 comments: "#",
52 step: undefined ,
53 complete: undefined ,
54 error: undefined ,
55 download: false ,
56 skipEmptyLines: true ,
57 chunk: undefined ,
58 fastMode: undefined ,
59 beforeFirstChunk: undefined ,
60 withCredentials: undefined
61 };
62 // LECTURA DE DATOS
63 var zip_files , data_obj;
64 // Coordenadas
65 function read_Buscoords(cb) {
66 var data;
67 var filename = "Buscoords.csv";
68 zip_files [ filename ]. async (" string "). then ( function ( csv_data )
        {
69 data = Papa.parse ( csv_data , config ). data ;
70 cb ( null , data );
71 });
72 }
73 // Codigos de linea
74 function read_LineCodes ( cb ) {
75 var data ;
76 var filename = "LinesCodes.csv";
77 zip_files [ filename ]. async (" string "). then ( function ( csv_data )

```

```

    {
78 data = Papa.parse(csv_data , config).data;
79 cb(null , data);
80 });
81 }
82 //Lineas
83 function read_Lines(cb) {
84 var data;
85 var filename = "Lines.csv";
86 zip_files [filename] . async (" string "). then (function ( csv_data )
    {
87 data = Papa.parse(csv_data , config).data;
88 cb(null , data);
89 });
90
91 }
92 //Cargas
93 function read_Loads(cb) {
94 var data;
95 var filename = "Loads.csv";
96 zip_files [filename] . async (" string "). then (function ( csv_data )
    {
97 data = Papa.parse(csv_data , config).data;
98 cb(null , data);
99 });
100
101 }
102 //Shapes de las cargas (minutales)
103 function read_LoadShapes(cb) {
104 var data;

```

```

105 var filename = "LoadShapes.csv";
106 zip_files[filename].async("string").then(function (csv_data)
    {
107 data = Papa.parse(csv_data, config).data;
108 cb(null, data);
109 });
110
111 }
112 //Fuente de tension
113 function read_Source(cb) {
114 var data;
115 var filename = "Source.csv";
116 zip_files[filename].async("string").then(function (csv_data)
    {
117 data = Papa.parse(csv_data, config).data;
118 cb(null, data);
119 });
120
121 }
122 //Transformador
123 function read_Transformer(cb) {
124 var data;
125 var filename = "Transformer.csv";
126 zip_files[filename].async("string").then(function (csv_data)
    {
127 data = Papa.parse(csv_data, config).data[0];
128 cb(null, data);
129 });
130
131 }

```

```

132 // Perfiles de las cargas
133 function read_LoadProfiles(cb) {
134   var data;
135   var filename = "LoadProfiles.csv";
136   zip_files[filename].async("string").then(function (csv_data)
      {
137     data = Papa.parse(csv_data, config).data;
138     cb(null, data);
139   });
140
141 }
142 //NUDO SLACK
143 // Potencia
144 function read_slack_pq(cb) { // Esta
      funcion lee los datos de cada uno de los archivos con P y
      Q para el nudo slack
145   var data;
146   var filename = "LV_Mon_slack_pq.csv";
147   zip_files[filename].async("string").then(function (csv_data)
      {
148     var new_csv_data = "hour,sec,P1,Q1,P2,Q2,P3,Q3," + csv_data.
      slice(csv_data.indexOf("\n")) // Creamos una nueva
      variable para cambiar las cabeceras del archivo
149     data = Papa.parse(new_csv_data, config_2).data;
150     cb(null, data);
151   });
152 }
153 //Tension(tres fases)
154 function read_slack_vi(cb) {
155   var data;

```



```

156 var filename = "LV_Mon_slack_vi.csv";
157 zip_files[filename].async("string").then(function (csv_data)
    {
158 var new_csv_data = "hour,sec,V1,VAngle1,V2,VAngle2,V3,
    VAngle3,I1,IAngle1,I2,IAngle2,I3,IAngle3," + csv_data.
    slice(csv_data.indexOf("\n")) // Creamos una nueva
    variable para cambiar las cabeceras del archivo
159 data = Papa.parse(new_csv_data, config_2).data;
160 cb(null, data);
161 });
162 }
163 //CARGAS
164 // Potencia
165 function read_LV_Mon_load_pq(index, cb) {
    // Esta funcion lee los datos de cada
    uno de los archivos con P y Q en funcion del tiempo de
    las 55 cargas
166 var data;
167 var filename = "LV_Mon_load" + index + "_pq.csv";
168 zip_files[filename].async("string").then(function (csv_data)
    {
169 var new_csv_data = "hour,sec,P1,Q1,P2,Q2," + csv_data.slice(
    csv_data.indexOf("\n")) // Creamos una nueva variable
    para cambiar las cabeceras del archivo
170 data = Papa.parse(new_csv_data, config_2).data;
171 cb(data);
172 });
173 }
174 // Tension
175 function read_LV_Mon_load_vi_1(index, cb) {

```

```

176 var data;
177 var filename = "LV_Mon_load" + index + "_vi_1.csv";
178 zip_files[filename].async("string").then(function (csv_data)
    {
179 var new_csv_data = "hour,sec,V1_1,VAngle1_1,V2_1,VAngle2_1,
    I1_1,IAngle1_1,I2_1,IAngle2_1," + csv_data.slice(csv_data
    .indexOf("\n")) // Creamos una nueva variable para
    cambiar las cabeceras del archivo
180 data = Papa.parse(new_csv_data, config_2).data;
181 cb(data);
182 });
183 }
184 function read_LV_Mon_load_vi_2(index, cb) {
185 var data;
186 var filename = "LV_Mon_load" + index + "_vi_2.csv";
187 zip_files[filename].async("string").then(function (csv_data)
    {
188 var new_csv_data = "hour,sec,V1_2,VAngle1_2,V2_2,VAngle2_2,
    I1_2,IAngle1_2,I2_2,IAngle2_2," + csv_data.slice(csv_data
    .indexOf("\n")) // Creamos una nueva variable para
    cambiar las cabeceras del archivo
189 data = Papa.parse(new_csv_data, config_2).data;
190 cb(data);
191 });
192 }
193 function read_LV_Mon_load_vi_3(index, cb) {
194 var data;
195 var filename = "LV_Mon_load" + index + "_vi_3.csv";
196 zip_files[filename].async("string").then(function (csv_data)
    {

```

```

197 var new_csv_data = "hour , sec , V1_3 , VAngle1_3 , V2_3 , VAngle2_3 ,
    I1_3 , IAngle1_3 , I2_3 , IAngle2_3 ," + csv_data.slice(csv_data
    .indexOf("\n")) // Creamos una nueva variable para
    cambiar las cabeceras del archivo
198 data = Papa.parse(new_csv_data , config_2).data;
199 cb(data);
200 });
201 }
202 //LINEAS
203 // Potencias
204 function read_LV_Mon_lines_pq_12(index , cb) {
205 var data;
206 var filename = "LV_Mon_line" + index + "_pq_vs_time_12.csv";
207 zip_files[filename].async("string").then(function (csv_data)
    {
208 var new_csv_data = "hour , sec , P1 , Q1 , P2 , Q2 , P3 , Q3 ," + csv_data .
    slice(csv_data.indexOf("\n"))
209 data = Papa.parse(new_csv_data , config_2).data;
210 cb(data);
211 });
212 }
213 function read_LV_Mon_lines_pq_21(index , cb) {
214 var data;
215 var filename = "LV_Mon_line" + index + "_pq_vs_time_21.csv";
216 zip_files[filename].async("string").then(function (csv_data)
    {
217 var new_csv_data = "hour , sec , P1 , Q1 , P2 , Q2 , P3 , Q3 ," + csv_data .
    slice(csv_data.indexOf("\n"))
218 data = Papa.parse(new_csv_data , config_2).data;
219 cb(data);

```

```

220 });
221 }
222 // Tensiones
223 function read_LV_Mon_lines_vi(index , cb) {
224 var data;
225 var filename = "LV_Mon_line" + index + "_vi_vs_time.csv";
226 zip_files[filename].async("string").then(function (csv_data)
    {
227 var new_csv_data = "hour,sec,V1,VAngle1,V2,VAngle2,V3,
    VAngle3,I1,IAngle1,I2,IAngle2,I3,IAngle3," + csv_data.
    slice(csv_data.indexOf("\n"))
228 data = Papa.parse(new_csv_data , config_2).data;
229 cb(data);
230 });
231 }
232 // Generacion PV
233 function read_LV_Mon_gpv_pq(index , cb) {
234 var data = new Array(1440);
235
236 for (var i = 0; i < 1440; i++) {
237 data[i] = { P1: 0, P2: 1 };
238 };
239 var filename = "LV_Mon_gpv" + index + "_pq.csv";
240 if (zip_files[filename]) {
241 zip_files[filename].async("string").then(function (csv_data)
    {
242 var new_csv_data = "hour,sec,P1,Q1,P2,Q2," + csv_data.slice(
    csv_data.indexOf("\n"))
243 data = Papa.parse(new_csv_data , config_2).data;
244 cb(data);

```

```

245 });
246 } else {
247   cb(data);
248 }
249 }
250 //BATERIAS
251 // Perfiles de baterias
252 function read_LV_Mon_bat_p(index , cb) {
253   var data = new Array(1440);
254
255   for (var i = 0; i < 1440; i++) {
256     data[i] = { kWh: 0, kWOut: 0, kWIn: 0, State: 2 };
257   };
258   var filename = "LV_Mon_bat_" + index + ".csv";
259   if (zip_files[filename]) {
260     zip_files[filename].async("string").then(function (csv_data)
261       {
262         var new_csv_data = "hour ,sec ,kWh, State ,kWOut,kWIn, Losses ,
263           Idling ,kWhChg," + csv_data.slice(csv_data.indexOf("\n"))
264         data = Papa.parse(new_csv_data , config_2).data;
265         cb(data);
266       });
267   } else {
268     cb(data);
269   }
270 }
271 // En principio no la vamos a utilizar
272 // function read_LV_Mon_bat_vi(index , cb) {
273 //     var data = new Array(1440).fill(0);
274 //     var filename = "LV_Mon_bat_" + index + "_vi.csv";

```

```

273 //      if ( zip_files [ filename ] ) {
274 //          zip_files [ filename ]. async ( " string " ) . then ( function
                ( csv_data ) {
275 //              var new_csv_data = " hour , sec , V1 , VAngle1 , V2 ,
                VAngle2 , I1 , IAngle1 , I2 , IAngle2 , " + csv_data . slice ( csv_data
                . indexOf ( "\n " ) )
276 //              data = Papa . parse ( new_csv_data , config_2 ) .
                data ;
277 //              cb ( data ) ;
278 //          } ) ;
279 //      } else {
280 //          cb ( data ) ;
281 //      }
282 // }
283
284 var read_fcns = {
285 " Buscoords " : read_Buscoords ,
286 // " LineCodes " : read_LineCodes ,
287 " Lines " : read_Lines ,
288 " Loads " : read_Loads ,
289 // " LoadShapes " : read_LoadShapes ,
290 // " Source " : read_Source ,
291 " Transformer " : read_Transformer ,
292 // " LoadProfiles " : read_LoadProfiles
293 " pq " : read_pq ,
294 " vi_1 " : read_vi_1 ,
295 " vi_2 " : read_vi_2 ,
296 " vi_3 " : read_vi_3 ,
297 " slack_pq " : read_slack_pq ,
298 " slack_vi " : read_slack_vi ,

```

```

299 "lines_pq_12": read_lines_pq_12 ,
300 "lines_pq_21": read_lines_pq_21 ,
301 "lines_vi": read_lines_vi ,
302 "gpv_pq": read_gpv_pq ,
303 "bat_p": read_bat_p ,
304 //" bat_vi": read_bat_vi ,
305 };
306
307 //Funcion de lectura de los datos con indice , es decir ,
      casos repetitivos (55 cargas,905 lineas)
308 function read_pq(cb) {
309 var loads_data = {};
310 async.each(
311   _.range(55),
312   function (index , cb_in) {
313     read_LV_Mon_load_pq(index + 1, function (pq_data) {
314       loads_data[index + 1] = pq_data;
315       cb_in();
316     })
317   },
318   function finish(err) {
319     cb(null , loads_data);
320   })
321 }
322 //Lectura de perfiles de generacion
323 function read_gpv_pq(cb) {
324 var loads_data = {};
325 async.each(
326   _.range(55),
327   function (index , cb_in) {

```

```

328 read_LV_Mon_gpv_pq(index + 1, function (pq_data) {
329   loads_data[index + 1] = pq_data;
330   cb_in();
331 })
332 },
333 function finish(err) {
334   cb(null, loads_data);
335 })
336 }
337 // Lectura perfiles de las baterias
338 function read_bat_p(cb) {
339   var loads_data = {};
340   async.each(
341     _.range(55),
342     function (index, cb_in) {
343       read_LV_Mon_bat_p(index + 1, function (pq_data) {
344         loads_data[index + 1] = pq_data;
345         cb_in();
346       })
347     },
348     function finish(err) {
349       cb(null, loads_data);
350     })
351 }
352 // function read_bat_vi(cb) {
353 //     var loads_data = {};
354 //     async.each(
355 //         _.range(55),
356 //         function (index, cb_in) {
357 //             read_LV_Mon_bat_vi(index + 1, function (

```



```

        vi_data) {
358 //             loads_data[index + 1] = vi_data;
359 //             cb_in();
360 //         })
361 //     },
362 //     function finish(err) {
363 //         cb(null, loads_data);
364 //     })
365 // }
366 // Lectura de las tensiones de cada fase
367 function read_vi_1(cb) {
368 var loads_data = {};
369 async.each(
370   _.range(55),
371   function (index, cb_in) {
372     read_LV_Mon_load_vi_1(index + 1, function (vi_data) {
373       loads_data[index + 1] = vi_data;
374       cb_in();
375     })
376   },
377   function finish(err) {
378     cb(null, loads_data);
379   })
380 }
381 function read_vi_2(cb) {
382 var loads_data = {};
383 async.each(
384   _.range(55),
385   function (index, cb_in) {
386     read_LV_Mon_load_vi_2(index + 1, function (vi_data) {

```

```

387 loads_data[index + 1] = vi_data;
388 cb_in();
389 })
390 },
391 function finish(err) {
392 cb(null, loads_data);
393 })
394 }
395 function read_vi_3(cb) {
396 var loads_data = {};
397 async.each(
398   _.range(55),
399   function (index, cb_in) {
400     read_LV_Mon_load_vi_3(index + 1, function (vi_data) {
401       loads_data[index + 1] = vi_data;
402       cb_in();
403     })
404   },
405   function finish(err) {
406     cb(null, loads_data);
407   })
408 }
409 //Lectura de las potencias de las lineas
410 function read_lines_pq_12(cb) {
411 var lines_data = {};
412 async.each(
413   _.range(905),
414   function (index, cb_in) {
415     read_LV_Mon_lines_pq_12(index + 1, function (pq_data) {
416       lines_data[index + 1] = pq_data;

```

```

417  cb_in();
418  })
419  },
420  function finish(err) {
421  cb(null, lines_data);
422  })
423  }
424  function read_lines_pq_21(cb) {
425  var lines_data = {};
426  async.each(
427  _.range(905),
428  function (index, cb_in) {
429  read_LV_Mon_lines_pq_21(index + 1, function (pq_data) {
430  lines_data[index + 1] = pq_data;
431  cb_in();
432  })
433  },
434  function finish(err) {
435  cb(null, lines_data);
436  })
437  }
438  //Lectura de las tensiones y corrientes de las lineas
439  function read_lines_vi(cb) {
440  var lines_data = {};
441  async.each(
442  _.range(905),
443  function (index, cb_in) {
444  read_LV_Mon_lines_vi(index + 1, function (vi_data) {
445  lines_data[index + 1] = vi_data;
446  cb_in();

```

```

447 })
448 },
449 function finish(err) {
450 cb(null, lines_data);
451 })
452 }
453
454 // function of upload text file
455 function uploadFile(event) {
456 var file = event.target.files[0];           // file uploaded
457 reader.readAsText(file);                   // start reading the
      file
458 reader.onload = function (event) {           // run this
      function after reading file data
459 var data_txt = event.target.result;         // obtain data as
      text in data_txt
460 // Parse CSV string
461 var data = Papa.parse(data_txt, config);
462 // console.log(data.data);
463 };
464 }
465
466 function readFile(entry, filename) {
467 entry.async("string").then(function (csv_data) {
468 data[filename] = Papa.parse(csv_data, config).data;
469 // console.log(data);
470 // console.log(1);
471 });
472 }
473 //SELECCION DE LA LINEA

```

```

474 //Desde el boton
475 function SelectLineFromButton() {
476 line_index = +$("#line_index").val()
477 // console.log(load_index+1)
478 selectLine(line_index);
479 }
480
481 // Funcion select line
482 function selectLine(line_index) {
483 if (selected_line != null) {
484                                     // Permite quitar
485                                     el stroke de una carga ya seleccionada
486 d3.select("#line_" + selected_line).attr("stroke-width",
487     1.5).attr("stroke", "darkcyan") // haciendo click
488     en una nueva
489 }
490 selected_line = line_index;
491 d3.select("#line_" + selected_line).attr("stroke-width", 4).
492     attr("stroke", "darkcyan") // en una nueva carga
493 $("#line_index").val(line_index);
494 $("#load_index").val(null); // resetear el boton de nodos y
495     la fase
496 $("#sel1").val(null); // Resetear boton de la fase
497
498 var dataP1_line_12 = [];
499 var dataP2_line_12 = [];
500 var dataP3_line_12 = [];
501 var dataP1_line_21 = [];
502 var dataP2_line_21 = [];
503 var dataP3_line_21 = [];

```

```

498 var dataLosses_1 = [];
499 var dataLosses_2 = [];
500 var dataLosses_3 = [];
501 var dataI1_line = [];
502 var dataI2_line = [];
503 var dataI3_line = [];
504 var dataP_acumulada = [];
505
506 _.map(data_obj.lines_pq_12[line_index], function (data ,
      minute_index) {
507 dataP1_line_12[minute_index] = data.P1;
508 dataP2_line_12[minute_index] = data.P2;
509 dataP3_line_12[minute_index] = data.P3;
510 })
511 _.map(data_obj.lines_pq_21[line_index], function (data ,
      minute_index) {
512 dataP1_line_21[minute_index] = data.P1;
513 dataP2_line_21[minute_index] = data.P2;
514 dataP3_line_21[minute_index] = data.P3;
515 })
516 _.map(data_obj.lines_vi[line_index], function (data ,
      minute_index) {
517 dataI1_line[minute_index] = data.I1;
518 dataI2_line[minute_index] = data.I2;
519 dataI3_line[minute_index] = data.I3;
520 })
521
522 for (var i = 0; i < dataP1_line_12.length; i++) {
523 dataLosses_1.push(dataP1_line_12[i] + dataP1_line_21[i]);
524 dataLosses_2.push(dataP2_line_12[i] + dataP2_line_21[i]);

```

```

525 dataLosses_3.push(dataP3_line_12[i] + dataP3_line_21[i]);
526 }
527 console.log(dataLosses_1[0])
528
529 // bucle para hacer el acumulado de potencias
530 for (var i = 0; i < dataP1_line_12.length; i++) {
531 dataP_acumulada.push(dataP1_line_12[i] + dataP2_line_12[i] +
    dataP3_line_12[i]);
532 }
533 // console.log(dataP_acumulada)
534 dataP_acumulada.sort(function (a, b) { return b - a }); //
    la funcion sort() sirve para ordenar arrays, en este caso
    de mayor a menor
535
536
537 var t_line = _.range(1440);
538 // Graficas de potencia de cada una de las lineas (tomadas en
    terminal 1)
539 var P_lines = new Dygraph(
540 document.getElementById("P"),
541 _.zip(t_line, dataP1_line_12, dataP2_line_12, dataP3_line_12
    ),
542 {
543 labels: ['Minute', 'Power A(kW)', 'Power B(kW)', 'Power C(kW)
    '],
544 animatedZooms: true,
545 strokeWidth: 1.2,
546 title: 'Active Power (all phases)',
547 gridLineColor: 'rgba(0,0,0,0)',
548 highlightCircleSize: 2,

```

```

549 axes: {
550 x: {
551 axisLabelFormatter: function (t_line) {
552 return numeral(t_line * 60).format("00:00:00").slice(0, -3);
553 }
554 }
555 },
556 //strokeWidth: 0.5,
557 //strokeBorderWidth: isStacked ? null : 1,
558 highlightSeriesOpts: {
559 strokeWidth: 2,
560 strokeBorderWidth: 1,
561 highlightCircleSize: 3
562 },
563 });
564 // Con esta funcion al seleccionar una fase , la grafica sale
      destacada
565 var onclick_P_lines = function (ev) {
566 if (P_lines.isSeriesLocked()) {
567 P_lines.clearSelection();
568 } else {
569 P_lines.setSelection(P_lines.getSelection(), P_lines.
      getHighlightSeries(), true);
570 }
571 };
572 P_lines.updateOptions({ clickCallback: onclick_P_lines },
      true);
573 P_lines.setSelection(false, 's010');
574 // Graficas de perdidas en las lineas
575 var P_losses = new Dygraph(

```



```

576 document.getElementById("Q"),
577 _.zip(t_line , dataLosses_1 , dataLosses_2 , dataLosses_3),
578 {
579 labels: ['Minute ', 'Losses A(kW) ', 'Losses B(kW) ', 'Losses C
      (kW) '],
580 animatedZooms: true ,
581 strokeWidth: 1.2 ,
582 title: 'Losses (all phases)',
583 // gridLineColor: 'rgba(0,0,0,0)',
584
585 highlightCircleSize: 2,
586 axes: {
587 x: {
588 axisLabelFormatter: function (t_line) {
589 return numeral(t_line * 60).format("00:00:00").slice(0, -3);
590 }
591 }
592 },
593 // strokeWidth: 0.5 ,
594 // strokeBorderWidth: isStacked ? null : 1,
595
596 highlightSeriesOpts: {
597 strokeWidth: 2,
598 strokeBorderWidth: 1,
599 highlightCircleSize: 3
600 }
601 }
602 );
603 // Con esta funcion al seleccionar una fase , la grafica sale
      destacada

```

```

604 var onclick_P_losses = function (ev) {
605   if (P_losses.isSeriesLocked()) {
606     P_losses.clearSelection();
607   } else {
608     P_losses.setSelection(P_losses.getSelection(), P_losses.
        getHighlightSeries(), true);
609   }
610 };
611 P_losses.updateOptions({ clickCallback: onclick_P_losses },
        true);
612 P_losses.setSelection(false, 's001');
613
614 // grafica de la corriente de las lineas
615 var I_line = new Dygraph(
616   document.getElementById("V"),
617   _.zip(t_line, dataI1_line, dataI2_line, dataI3_line),
618   {
619     labels: ['Minute', 'Current A(A)', 'Current B(A)', 'Current
        C(A)'],
620     animatedZooms: true,
621     strokeWidth: 1.2,
622     title: 'Currents (all phases)',
623     // gridLineColor: 'rgba(0,0,0,0)',
624     highlightCircleSize: 2,
625     axes: {
626       x: {
627         axisLabelFormatter: function (t_line) {
628           return numeral(t_line * 60).format("00:00:00").slice(0, -3);
629         }
630       }

```

```

631 },
632 //strokeWidth: 0.5,
633 //strokeBorderWidth: isStacked ? null : 1,
634
635 highlightSeriesOpts: {
636   strokeWidth: 2,
637   strokeBorderWidth: 1,
638   highlightCircleSize: 3
639 }
640 }
641 );
642 // Con esta funcion al seleccionar una fase , la grafica sale
        destacada
643 var onclick_I_line = function (ev) {
644   if (I_line.isSeriesLocked()) {
645     I_line.clearSelection();
646   } else {
647     I_line.setSelection(I_line.getSelection(), I_line.
        getHighlightSeries(), true);
648   }
649 };
650 I_line.updateOptions({ clickCallback: onclick_I_line }, true
        );
651 I_line.setSelection(false, 's001');
652
653 // grafica de potencia acumulada
654 new Dygraph(
655   document.getElementById("delta"),
656   _.zip(t_line, dataP_acumulada),
657   {

```

```

658 labels: ['Minute', 'Active Power Consumed(kW)'],
659 animatedZooms: true,
660 strokeWidth: 1.5,
661 title: 'Cumulative Power',
662 // gridLineColor: 'rgba(0,0,0,0)'
663 }
664 );
665
666 } //end of select line
667
668 //SELECCION DE NODO
669 //Desde el boton
670 function SelectNodeFromButton() {
671 load_index = +$("#load_index").val()
672 //console.log(load_index)
673 selectLoad(load_index);
674 }
675
676 // Funcion selectNode: muestra stroke al seleccionar
        cualquier carga y tambien muestra las graficas de cada
        uno de los nudos
677 function selectLoad(load_index) {
678 if (selected_load != null) {
                                                    // Permite quitar
        el stroke de una carga ya seleccionada
679 d3.select("#load_" + selected_load).attr("stroke", "none")
        // haciendo click en una nueva
680 }
681 selected_load = load_index;

```

```

        // Permite mostrar un nuevo stroke al hacer click
682 d3.select("#load_" + selected_load).attr("stroke", "hsla
        (221, 66%, 33%, 0.59)").attr("stroke-width", 2.5) //
        en una nueva carga
683 $("#load_index").val(load_index); // Al hacer click en un
        nodo muestra su numero en el boton del panel
684 $("#line_index").val(null); // resetear el boton de lineas
685 // console.log(data_obj.Loads[load_index].phases);
686 if (load_index != 56) {
687 if (data_obj.Loads[load_index - 1].phases == "A") {
688 $("#sell").val("Phase: A");
689 }
690 else if (data_obj.Loads[load_index - 1].phases == "B") {
691 $("#sell").val("Phase: B");
692 }
693 else if (data_obj.Loads[load_index - 1].phases == "C") {
694 $("#sell").val("Phase: C");
695 };
696 }
697 if (load_index == 56) {
698 $("#sell").val("Transformer");
699 }
700 console.log(load_index);
701 //DATOS DE LAS CARGAS
702 var dataP = [];
703 var dataQ = [];
704 var dataP_gpv = [];
705 var dataQ_gpv = [];
706 var dataV_1 = [];
707 var dataV_2 = [];

```

```

708 var dataV_3 = [];
709 var dataAngle_1 = [];
710 var dataAngle_2 = [];
711 var dataAngle_3 = [];
712 //DATOS DEL NODO SLACK
713 //Potencias de las tres fases
714 var dataP_1_slack = [];
715 var dataP_2_slack = [];
716 var dataP_3_slack = [];
717 var dataQ_1_slack = [];
718 var dataQ_2_slack = [];
719 var dataQ_3_slack = [];
720 //Tensiones de las tres fases
721 var dataV_1_slack = [];
722 var dataV_2_slack = [];
723 var dataV_3_slack = [];
724 //Corrientes de las tres fases
725 var dataI_1_slack = [];
726 var dataI_2_slack = [];
727 var dataI_3_slack = [];
728 //DATOS DE LAS LINEAS
729
730 //Con la funcion _.map se recorre los datos importados y se
      guardan en las variables antes declaradas
731 _.map(data_obj.pq[load_index], function (data , minute_index)
      {
732 dataP[minute_index] = data.P1;
733 dataQ[minute_index] = data.Q1;
734 })
735 _.map(data_obj.gpv_pq[load_index], function (data ,

```

```

        minute_index) {
736 dataP_gpv[minute_index] = -1 * (data.P1);
737 dataQ_gpv[minute_index] = -1 * (data.Q1);
738 })
739
740 _.map(data_obj.vi_1[load_index], function (data ,
        minute_index) {
741 dataV_1[minute_index] = data.V1_1;
742 dataAngle_1[minute_index] = data.VAngle1_1;
743 })
744 _.map(data_obj.vi_2[load_index], function (data ,
        minute_index) {
745 dataV_2[minute_index] = data.V1_2;
746 dataAngle_2[minute_index] = data.VAngle1_2;
747 })
748 _.map(data_obj.vi_3[load_index], function (data ,
        minute_index) {
749 dataV_3[minute_index] = data.V1_3;
750 dataAngle_3[minute_index] = data.VAngle1_3;
751 })
752 _.map(data_obj.slack_pq, function (data , minute_index) {
753 dataP_1_slack[minute_index] = data.P1;
754 dataP_2_slack[minute_index] = data.P2;
755 dataP_3_slack[minute_index] = data.P3;
756 dataQ_1_slack[minute_index] = data.Q1;
757 dataQ_2_slack[minute_index] = data.Q2;
758 dataQ_3_slack[minute_index] = data.Q3;
759 })
760 _.map(data_obj.slack_vi, function (data , minute_index) {
761 dataV_1_slack[minute_index] = data.V1;

```

```

762 dataV_2_slack[minute_index] = data.V2;
763 dataV_3_slack[minute_index] = data.V3;
764 dataI_1_slack[minute_index] = data.I1;
765 dataI_2_slack[minute_index] = data.I2;
766 dataI_3_slack[minute_index] = data.I3;
767 })
768 //GRAFICAS PARA LAS CARGAS
769 var t = _.range(1440);
770 if (load_index != 56) {
771 // Grafica de la potencia Activa (consumida y generada)
772 new Dygraph(
773 document.getElementById("P"),
774 _.zip(t, dataP, dataP_gpv),
775 {
776 labels: ['Minute', 'Active Power Consumed(kW)', 'Active
          Power Generated(kW)'],
777 animatedZooms: true,
778 strokeWidth: 1.5,
779 title: 'Active Power',
780 axes: {
781 x: {
782 axisLabelFormatter: function (t) {
783 return numeral(t * 60).format("00:00:00").slice(0, -3);
784 }
785 }
786 },
787 // gridLineColor: 'rgba(0,0,0,0)'
788 }
789 );
790 // Potencia Reactiva

```



```

791 new Dygraph(
792 document.getElementById("Q"),
793 _.zip(t, dataQ),
794 {
795 labels: ['Minute', 'Reactive Power(kvar)'],
796 animatedZooms: true,
797 strokeWidth: 1.5,
798 title: 'Reactive Power',
799 axes: {
800 x: {
801 axisLabelFormatter: function (t) {
802 return numeral(t * 60).format("00:00:00").slice(0, -3);
803 }
804 }
805 },
806 // gridLineColor: 'rgba(0,0,0,0)'
807 }
808 );
809 // Tensiones de las tres fases
810 var g = new Dygraph(
811 document.getElementById("V"),
812 _.zip(t, dataV_1, dataV_2, dataV_3),
813 {
814 labels: ['Minute', 'Voltages A(V)', 'Voltages B(V)', '
      Voltages C(V)'],
815 animatedZooms: true,
816 strokeWidth: 1.2,
817 title: 'Voltages (all phases)',
818 // gridLineColor: 'rgba(0,0,0,0)',
819 highlightCircleSize: 2,

```

```

820 //strokeWidth: 0.5,
821 //strokeBorderWidth: isStacked ? null : 1,
822 axes: {
823 x: {
824 axisLabelFormatter: function (t) {
825 return numeral(t * 60).format("00:00:00").slice(0, -3);
826 }
827 }
828 },
829 highlightSeriesOpts: {
830 strokeWidth: 2,
831 strokeBorderWidth: 1,
832 highlightCircleSize: 3
833 }
834 }
835 );
836 // Con esta funcion al seleccionar una fase , la grafica sale
      destacada
837 var onclick_g = function (ev) {
838 if (g.isSeriesLocked()) {
839 g.clearSelection();
840 } else {
841 g.setSelection(g.getSelection(), g.getHighlightSeries(),
      true);
842 }
843 };
844 g.updateOptions({ clickCallback: onclick_g }, true);
845 g.setSelection(false , 's005');
846
847 //Angulo de desfase de la fase conectada a la carga

```

```

848 // Condiciones para que aparezca en el grafico la fase de la
      carga que esta conectada
849 if (data_obj.Loads[load_index - 1].phases == "A") {
850 new Dygraph(
851 document.getElementById(" delta " ),
852 _.zip(t, dataAngle_1),
853 {
854 labels: ['Minute', 'Angle A'],
855 animatedZooms: true,
856 strokeWidth: 1.5,
857 title: 'Voltage Angle',
858 axes: {
859 x: {
860 axisLabelFormatter: function (t) {
861 return numeral(t * 60).format("00:00:00").slice(0, -3);
862 }
863 }
864 },
865 // gridLineColor: 'rgba(0,0,0,0)'
866 }
867 );
868 }
869 else if (data_obj.Loads[load_index - 1].phases == "B") {
870 new Dygraph(
871 document.getElementById(" delta " ),
872 _.zip(t, dataAngle_2),
873 {
874 labels: ['Minute', 'Angle B'],
875 animatedZooms: true,
876 strokeWidth: 1.5,

```

```

877 title: 'Voltage Angle',
878 axes: {
879 x: {
880 axisLabelFormatter: function (t) {
881 return numeral(t * 60).format("00:00:00").slice(0, -3);
882 }
883 }
884 },
885 // gridLineColor: 'rgba(0,0,0,0)'
886 }
887 );
888 }
889 else if (data_obj.Loads[load_index - 1].phases == "C") {
890 new Dygraph(
891 document.getElementById("delta"),
892 _.zip(t, dataAngle_3),
893 {
894 labels: ['Minute', 'Angle C'],
895 animatedZooms: true,
896 strokeWidth: 1.5,
897 title: 'Voltage Angle',
898 axes: {
899 x: {
900 axisLabelFormatter: function (t) {
901 return numeral(t * 60).format("00:00:00").slice(0, -3);
902 }
903 }
904 },
905 // gridLineColor: 'rgba(0,0,0,0)'
906 }

```

```

907 );
908 }
909 }
910
911 //GRAFICAS DEL NUDO SLACK
912 else if (load_index == 56) {
913 var t_1 = _.range(1440);
914 // Graficas de potencia activa
915 var P_slack = new Dygraph(
916 document.getElementById("P"),
917 _.zip(t_1, dataP_1_slack, dataP_2_slack, dataP_3_slack),
918 {
919 labels: ['Minute', 'Active Power A(kW)', 'Active Power B(kW)
          ', 'Active Power C(kW)'],
920 animatedZooms: true,
921 strokeWidth: 1.2,
922 title: 'Active Power (all phases)',
923 // gridLineColor: 'rgba(0,0,0,0)',
924 highlightCircleSize: 2,
925 axes: {
926 x: {
927 axisLabelFormatter: function (t) {
928 return numeral(t * 60).format("00:00:00").slice(0, -3);
929 }
930 }
931 },
932 //strokeWidth: 0.5,
933 //strokeBorderWidth: isStacked ? null : 1,
934
935 highlightSeriesOpts: {

```

```

936 strokeWidth: 2,
937 strokeBorderWidth: 1,
938 highlightCircleSize: 3
939 }
940 }
941 );
942 // Con esta funcion al seleccionar una fase , la grafica sale
          destacada
943 var onclick_P_slack = function (ev) {
944 if (P_slack.isSeriesLocked()) {
945 P_slack.clearSelection();
946 } else {
947 P_slack.setSelection(P_slack.getSelection(), P_slack.
          getHighlightSeries(), true);
948 }
949 };
950 P_slack.updateOptions({ clickCallback: onclick_P_slack },
          true);
951 P_slack.setSelection(false, 's005');
952 // Graficas de potencia reactiva
953 var Q_slack = new Dygraph(
954 document.getElementById("Q"),
955 _.zip(t_1, dataQ_1_slack, dataQ_2_slack, dataQ_3_slack),
956 {
957 labels: ['Minute', 'Reactive Power A(kvar)', 'Reactive Power
          B(kvar)', 'Reactive Power C(kvar)'],
958 animatedZooms: true,
959 strokeWidth: 1.2,
960 title: 'Reactive Power (all phases)',
961 // gridLineColor: 'rgba(0,0,0,0)',

```

```

962
963 highlightCircleSize: 2,
964 axes: {
965 x: {
966 axisLabelFormatter: function (t) {
967 return numeral(t * 60).format("00:00:00").slice(0, -3);
968 }
969 }
970 },
971 //strokeWidth: 0.5,
972 //strokeBorderWidth: isStacked ? null : 1,
973
974 highlightSeriesOpts: {
975 strokeWidth: 2,
976 strokeBorderWidth: 1,
977 highlightCircleSize: 3
978 }
979 }
980 );
981 // Con esta funcion al seleccionar una fase, la grafica sale
          destacada
982 var onclick_Q_slack = function (ev) {
983 if (Q_slack.isSeriesLocked()) {
984 Q_slack.clearSelection();
985 } else {
986 Q_slack.setSelection(Q_slack.getSelection(), Q_slack.
          getHighlightSeries(), true);
987 }
988 };
989 Q_slack.updateOptions({ clickCallback: onclick_Q_slack },

```

```

        true);
990 Q_slack.setSelection(false, 's005');
991 // Graficas de tensiones(modulo)
992 var V_slack = new Dygraph(
993 document.getElementById("V"),
994 _.zip(t_1, dataV_1_slack, dataV_2_slack, dataV_3_slack),
995 {
996 labels: ['Minute', 'Voltage A(V)', 'Voltage B(V)', 'Voltage
          C(V)'],
997 animatedZooms: true,
998 strokeWidth: 1.2,
999 title: 'Slack Voltages (all phases)',
1000 // gridLineColor: 'rgba(0,0,0,0)',
1001 highlightCircleSize: 2,
1002 axes: {
1003 x: {
1004 axisLabelFormatter: function (t) {
1005 return numeral(t * 60).format("00:00:00").slice(0, -3);
1006 }
1007 }
1008 },
1009 // strokeWidth: 0.5,
1010 // strokeBorderWidth: isStacked ? null : 1,
1011
1012 highlightSeriesOpts: {
1013 strokeWidth: 2,
1014 strokeBorderWidth: 1,
1015 highlightCircleSize: 3
1016 }
1017 }

```



```

1018 );
1019 // Con esta funcion al seleccionar una fase , la grafica sale
        destacada
1020 var onclick_V_slack = function (ev) {
1021 if (V_slack.isSeriesLocked()) {
1022 V_slack.clearSelection();
1023 } else {
1024 V_slack.setSelection(V_slack.getSelection(), V_slack.
        getHighlightSeries(), true);
1025 }
1026 };
1027 V_slack.updateOptions({ clickCallback: onclick_V_slack },
        true);
1028 V_slack.setSelection(false, 's005');
1029 // Graficas de corrientes(modulo)
1030 var I_slack = new Dygraph(
1031 document.getElementById("delta"),
1032 _.zip(t_1, dataI_1_slack, dataI_2_slack, dataI_3_slack),
1033 {
1034 labels: ['Minute', 'Current A(A)', 'Current B(A)', 'Current
        C(A)'],
1035 animatedZooms: true,
1036 strokeWidth: 1.2,
1037 title: 'Slack Currents (all phases)',
1038 // gridLineColor: 'rgba(0,0,0,0)',
1039
1040 highlightCircleSize: 2,
1041 axes: {
1042 x: {
1043 axisLabelFormatter: function (t) {

```

```

1044 return numeral(t * 60).format("00:00:00").slice(0, -3);
1045 }
1046 }
1047 },
1048 //strokeWidth: 0.5,
1049 //strokeBorderWidth: isStacked ? null : 1,
1050
1051 highlightSeriesOpts: {
1052   strokeWidth: 2,
1053   strokeBorderWidth: 1,
1054   highlightCircleSize: 3
1055 }
1056 }
1057 );
1058 // Con esta funcion al seleccionar una fase, la grafica sale
      destacada
1059 var onclick_I_slack = function (ev) {
1060   if (I_slack.isSeriesLocked()) {
1061     I_slack.clearSelection();
1062   } else {
1063     I_slack.setSelection(I_slack.getSelection(), I_slack.
      getHighlightSeries(), true);
1064   }
1065 };
1066 I_slack.updateOptions({ clickCallback: onclick_I_slack },
      true);
1067 I_slack.setSelection(false, 's005');
1068 };
1069
1070 }; // Fin de SelectNode()

```

```

1071 //SELECCION DE LA BATERIA
1072
1073 function SelectBat(bat_index) {
1074   if (selected_bat != null) {
1075     // Permite quitar
1076     // el stroke de una bateria ya seleccionada
1077     d3.select("#battery_" + selected_bat).attr("stroke", "black")
1078     .attr("stroke-width", 0.5) // haciendo click en una
1079     // nueva
1080   }
1081   selected_bat = bat_index;
1082
1083   // Permite
1084   // mostrar un nuevo stroke al hacer click
1085   d3.select("#battery_" + selected_bat).attr("stroke", "hsla
1086   (221, 66%, 33%, 0.59)").attr("stroke-width", 2) // en una
1087   // nueva bateria
1088   $("#load_index").val(bat_index); // Al hacer click en un
1089   // nodo muestra su numero en el boton del panel
1090   $("#line_index").val(null); // resetear el boton de lineas
1091   $("#sell").val(null); // resetear el boton de la fase
1092
1093   // Datos de las baterias
1094   var datakWh = [];
1095   var dataPIn = [];
1096   var dataPOut = [];
1097   var dataP = [];
1098   var dataP_gpv = [];
1099   var dataPneta = [];
1100
1101   _ .map(data_obj.pq[bat_index], function (data, minute_index)

```

```

    {
1092 dataP[minute_index] = data.P1;
1093 })
1094 _.map(data_obj.gpv_pq[bat_index], function (data ,
    minute_index) {
1095 dataP_gpv[minute_index] = -1 * (data.P1);
1096 })
1097 for (var i = 0; i < dataP.length; i++) {
1098 dataPneta.push(dataP[i] - dataP_gpv[i]);
1099 }
1100 _.map(data_obj.bat_p[bat_index], function (data ,
    minute_index) {
1101 datakWh[minute_index] = data.kWh;
1102 })
1103 _.map(data_obj.bat_p[bat_index], function (data ,
    minute_index) {
1104 dataPIn[minute_index] = data.kWIn;
1105 })
1106 _.map(data_obj.bat_p[bat_index], function (data ,
    minute_index) {
1107 dataPOut[minute_index] = data.kWOut;
1108 })
1109 var t = _.range(1440);
1110 // Potencia consumida y generada
1111 new Dygraph(
1112 document.getElementById("P") ,
1113 _.zip(t, dataP, dataP_gpv),
1114 {
1115 labels: [ 'Minute ', 'Active Power Consumed(kW) ', 'Active
    Power Generated(kW) ' ],

```

```

1116 animatedZooms: true ,
1117 strokeWidth: 1.5 ,
1118 title: 'Active Power Consumed and Generated' ,
1119 axes: {
1120 x: {
1121 axisLabelFormatter: function (t) {
1122 return numeral(t * 60).format("00:00:00").slice(0, -3);
1123 }
1124 }
1125 },
1126 // gridLineColor: 'rgba(0,0,0,0)'
1127 }
1128 );
1129 // Potencia global neta
1130 new Dygraph(
1131 document.getElementById("Q") ,
1132 _ .zip(t, dataPneta) ,
1133 {
1134 labels: ['Minute', 'Net Active Power(kW)'] ,
1135 animatedZooms: true ,
1136 strokeWidth: 1.5 ,
1137 title: 'Net Active Power' ,
1138 axes: {
1139 x: {
1140 axisLabelFormatter: function (t) {
1141 return numeral(t * 60).format("00:00:00").slice(0, -3);
1142 }
1143 }
1144 },
1145 // gridLineColor: 'rgba(0,0,0,0)'

```

```

1146 }
1147 );
1148 // Carga de la bateria
1149 new Dygraph(
1150 document.getElementById("V"),
1151 _.zip(t, datakWh),
1152 {
1153 labels: ['Minute', 'Battery Charge(kWh)'],
1154 animatedZooms: true,
1155 strokeWidth: 1.5,
1156 title: 'Battery Charge',
1157 axes: {
1158 x: {
1159 axisLabelFormatter: function (t) {
1160 return numeral(t * 60).format("00:00:00").slice(0, -3);
1161 }
1162 }
1163 },
1164 // gridLineColor: 'rgba(0,0,0,0)'
1165 }
1166 );
1167 // Potencia cedida y consumida por la bateria
1168 new Dygraph(
1169 document.getElementById("delta"),
1170 _.zip(t, dataPIn, dataPOut),
1171 {
1172 labels: ['Minute', 'Active Power IN(kW)', 'Active Power OUT(
kW)'],
1173 animatedZooms: true,
1174 strokeWidth: 1.5,

```

```

1175 title: 'Active Power IN and OUT',
1176 axes: {
1177 x: {
1178 axisLabelFormatter: function (t) {
1179 return numeral(t * 60).format("00:00:00").slice(0, -3);
1180 }
1181 }
1182 },
1183 // gridLineColor: 'rgba(0,0,0,0)'
1184 }
1185 );
1186
1187 };//Fin de SelectBat()
1188
1189 // POTENCIAS GLOBALES
1190 function DibujaPotenciasGlobales() {
1191 var dataPglo = new Array(1440).fill(0);
1192 var dataPgenglo = new Array(1440).fill(0);
1193 var dataPnetaglo = [];
1194 var dataPacumuladaglo = [];
1195
1196 var dataP1_line_12 = new Array(1440).fill(0);
1197 var dataP2_line_12 = new Array(1440).fill(0);
1198 var dataP3_line_12 = new Array(1440).fill(0);
1199 var dataP1_line_21 = new Array(1440).fill(0);
1200 var dataP2_line_21 = new Array(1440).fill(0);
1201 var dataP3_line_21 = new Array(1440).fill(0);
1202 var dataLosses_1 = [];
1203 var dataLosses_2 = [];
1204 var dataLosses_3 = [];

```

```

1205
1206 for (var i = 0; i < 1440; i++) {
1207 for (var j = 1; j < 906; j++) {
1208 dataP1_line_12[i] = data_obj.lines_pq_12[j][i].P1 +
      dataP1_line_12[i];
1209 dataP2_line_12[i] = data_obj.lines_pq_12[j][i].P2 +
      dataP2_line_12[i];
1210 dataP3_line_12[i] = data_obj.lines_pq_12[j][i].P3 +
      dataP3_line_12[i];
1211 dataP1_line_21[i] = data_obj.lines_pq_21[j][i].P1 +
      dataP1_line_21[i];
1212 dataP2_line_21[i] = data_obj.lines_pq_21[j][i].P2 +
      dataP2_line_21[i];
1213 dataP3_line_21[i] = data_obj.lines_pq_21[j][i].P3 +
      dataP3_line_21[i];
1214 }
1215 }
1216
1217 for (var i = 0; i < 1440; i++) {
1218 dataLosses_1[i] = dataP1_line_12[i] + dataP1_line_21[i];
1219 dataLosses_2[i] = dataP2_line_12[i] + dataP2_line_21[i];
1220 dataLosses_3[i] = dataP3_line_12[i] + dataP3_line_21[i];
1221 }
1222
1223 for (var i = 0; i < 1440; i++) {
1224 for (var j = 1; j < 56; j++) {
1225 dataPglo[i] = data_obj.pq[j][i].P1 + dataPglo[i];
1226 }
1227 }
1228

```



```

1229 for (var i = 0; i < 1440; i++) {
1230 for (var j = 1; j < 56; j++) {
1231 dataPgenglo[i] = -1 * (data_obj.gpv_pq[j][i].P1) +
        dataPgenglo[i];
1232 }
1233 }
1234
1235 // bucle para hacer el acumulado de potencias
1236 for (var i = 0; i < dataPglo.length; i++) {
1237 dataPacumuladaglo.push(dataPglo[i] - dataPgenglo[i]);
1238 dataPnetaglo.push(dataPglo[i] - dataPgenglo[i]);
1239 }
1240 // console.log(dataP_acumulada)
1241 dataPacumuladaglo.sort(function (a, b) { return b - a });
        // la funcion sort() sirve para ordenar arrays, en este
        caso de mayor a menor
1242
1243 var t = _.range(1440);
1244 // Potencia global consumida y generada
1245 new Dygraph(
1246 document.getElementById("P"),
1247 _.zip(t, dataPglo, dataPgenglo),
1248 {
1249 labels: ['Minute', 'Active Power Consumed(kW)', 'Active
        Power Generated(kW)'],
1250 animatedZooms: true,
1251 strokeWidth: 1.5,
1252 title: 'Active Power Consumed and Generated',
1253 axes: {
1254 x: {

```

```

1255 axisLabelFormatter: function (t) {
1256 return numeral(t * 60).format("00:00:00").slice(0, -3);
1257 }
1258 }
1259 },
1260 // gridLineColor: 'rgba(0,0,0,0)'
1261 }
1262 );
1263 // Potencia global neta
1264 new Dygraph(
1265 document.getElementById("Q"),
1266 _.zip(t, dataPnetaglo),
1267 {
1268 labels: ['Minute', 'Net Active Power(kW)'],
1269 animatedZooms: true,
1270 strokeWidth: 1.5,
1271 title: 'Net Active Power',
1272 axes: {
1273 x: {
1274 axisLabelFormatter: function (t) {
1275 return numeral(t * 60).format("00:00:00").slice(0, -3);
1276 }
1277 }
1278 },
1279 // gridLineColor: 'rgba(0,0,0,0)'
1280 }
1281 );
1282 // Potencia global neta acumulada
1283 new Dygraph(
1284 document.getElementById("V"),

```

```

1285  _ .zip(t, dataPacumuladaglo),
1286  {
1287  labels: ['Minute', 'Cumulative Net Active Power(kW)'],
1288  animatedZooms: true,
1289  strokeWidth: 1.5,
1290  title: 'Cumulative Net Active Power',
1291  // gridLineColor: 'rgba(0,0,0,0)'
1292  }
1293  );
1294  // Perdidas globales
1295  new Dygraph(
1296  document.getElementById("delta"),
1297  _ .zip(t, dataLosses_1, dataLosses_2, dataLosses_3),
1298  {
1299  labels: ['Minuto', 'Global Losses A(kW)', 'Global Losses B(
1300           kW)', 'Global Losses C(kW)'],
1301  animatedZooms: true,
1302  strokeWidth: 1.5,
1303  title: 'Global Losses(all phases)',
1304  axes: {
1305  x: {
1306  axisLabelFormatter: function (t) {
1307  return numeral(t * 60).format("00:00:00").slice(0, -3);
1308  }
1309  },
1310  // gridLineColor: 'rgba(0,0,0,0)'
1311  }
1312  );
1313

```

```

1314 } //Fin de dibujar potencias globales
1315
1316 // SLIDER
1317 $(function () {
1318   var handle = $("#custom-handle");
1319   $("#slider").slider({
1320     create: function () {
1321       handle.text($(this).slider("value"));
1322     },
1323     slide: function (event, ui) {
1324       handle.text(numeral(ui.value * 60).format("00:00:00").slice
         (0, -3));
1325       position = Math.round(ui.value);
1326       valorslider = ui.value;
1327       actualizeSlider();
1328       //console.log(position);
1329     },
1330     min: 0,
1331     max: 1439
1332   });
1333 });
1334
1335 // Funcion para hacer que el slider se mueva automaticamente
    .
1336 var simulation_interval;
1337 function startSimulation() {
1338   minute_index = valorslider;
1339   simulation_interval = setInterval(Simulate, 200);
1340 }
1341 function Simulate() {

```

```

1342
1343 $("#slider").slider("value", valorslider);
1344
1345 position = minute_index;//Math.round(valorslider / 100 *
        1440);
1346 actualizeSlider();
1347 //valorslider = valorslider + 1;
1348 valorslider = minute_index;
1349 //Introducir reset cuando valorslider es 1440
1350 if (minute_index >= 1439) {
1351 $("#slider").slider("value", 0);
1352 valorslider = 0;
1353 minute_index = 0;
1354 stopSimulation();
1355 $("#pause").hide();
1356 $(".button3").show();
1357 }
1358 var handle = $("#custom-handle");
1359 handle.text(numeral(valorslider * 60).format("00:00:00").
        slice(0, -3));
1360 minute_index++;
1361 }
1362
1363 function stopSimulation() {
1364
1365 clearInterval(simulation_interval);
1366 }
1367
1368 //ACTULIZESLIDER() Variables para asociar el slider
1369

```

```

1370 function actualizeSlider() { // leer para cada
    instante del slider los valores P1 de todas las cargas
1371 // //Para resetear los simbolos de carga de la bateria
1372 // for (var j = 1; j < 56; j++) {
1373 //     d3.select("#bat_" + j)
1374 //         .text(null)
1375 // }
1376 for (var j = 1; j < 56; j++) {
1377
1378 sliderP[j - 1] = data_obj.pq[j][position].P1;
1379 // console.log(data_obj.pq[0]); // Para escoger el
    archivo pq el primer indice debe ser el 1
1380 // console.log(data_obj.Loads[0]) // Para escoger el
    archivo Loads el primer indice debe ser 0
1381 // console.log(sliderP[0]);
1382 sliderPgen[j - 1] = -1 * (data_obj.gpv_pq[j][position].P1);
1383 sliderPneta[j - 1] = sliderP[j - 1] - sliderPgen[j - 1]; //
    potencia neta Pc-Pgen
1384 // Arrays para las baterias
1385 sliderBat[j - 1] = (data_obj.bat_p[j][position].kWh);
1386 sliderBatState[j - 1] = (data_obj.bat_p[j][position].State);
1387
1388 if (data_obj.Loads[j - 1].phases == "A") {
1389 sliderV[j - 1] = data_obj.vi_1[j][position].V1_1;
1390 }
1391 else if (data_obj.Loads[j - 1].phases == "B") {
1392 sliderV[j - 1] = data_obj.vi_2[j][position].V1_2;
1393 }
1394 else if (data_obj.Loads[j - 1].phases == "C") {
1395 sliderV[j - 1] = data_obj.vi_3[j][position].V1_3;

```

```

1396 }
1397
1398 if (botonP == 1) {
1399 d3.select("#load_" + j)
1400 .attr("stroke", "green")
1401 .attr("stroke-width", (sliderP[j - 1]) * 2.5) // Dibujar
        stroke en la carga j con width en funcion a P1
1402 }
1403
1404 if (botonPgen == 1) {
1405 d3.select("#load_" + j)
1406 .attr("stroke", "green")
1407 .attr("stroke-width", (sliderPgen[j - 1]) * 4) // Dibujar
        stroke en la carga j con width en funcion a P1 generada
1408 }
1409
1410 if (botonPneta == 1) {
1411 if (sliderPneta[j - 1] > 0) {
1412 d3.select("#load_" + j)
1413 .attr("stroke", "green")
1414 .attr("stroke-width", (sliderPneta[j - 1]) * 2.5) // Dibujar
        stroke en la carga j con width en funcion a P1 neta
1415 } else {
1416 d3.select("#load_" + j)
1417 .attr("stroke", "blue")
1418 .attr("stroke-width", (-4 * (sliderPneta[j - 1]))) // Dibujar
        stroke en la carga j con width en funcion a P1 neta
1419 }
1420 }
1421

```

```

1422 if (botonV == 1) { // rangos para
        representar los valores de tension
1423 if (sliderV[j - 1] > 240 * 1.1) {
1424 d3.select("#load_" + j)
1425 // .transition()
1426 // .duration(100)
1427 .attr("stroke", "hsla(26, 100%, 54%, 1)") // El primer %
        indica la intensidad del color, el segundo % la
        transparencia
1428 .attr("stroke-width", 8)
1429 }
1430 if (sliderV[j - 1] > 240 * 1.02 && sliderV[j - 1] < 240 *
        1.1) {
1431 d3.select("#load_" + j)
1432 // .transition()
1433 // .duration(100)
1434 .attr("stroke", "hsla(28, 50%, 70%, 1)")
1435 .attr("stroke-width", 8)
1436 }
1437 if (sliderV[j - 1] < 240 * 1.02 && sliderV[j - 1] > 240 *
        0.98) {
1438 d3.select("#load_" + j)
1439 // .transition()
1440 // .duration(100)
1441 .attr("stroke", "white")
1442 .attr("stroke-width", 0)
1443 }
1444 if (sliderV[j - 1] < 240 * 0.98 && sliderV[j - 1] > 240 *
        0.9) {
1445 d3.select("#load_" + j)

```



```

1446 // .transition ()
1447 // .duration (100)
1448 .attr ("stroke", "hsla (200, 50%, 70%, 1)")
1449 .attr ("stroke-width", 8)
1450 }
1451 if (sliderV [j - 1] < 240 * 0.90) {
1452 d3.select ("#load_" + j)
1453 // .transition ()
1454 // .duration (100)
1455 .attr ("stroke", "hsla (249, 100%, 19%, 1)")
1456 .attr ("stroke-width", 8)
1457 }
1458 }
1459 if (sliderPgen [j - 1] > 0) {
1460 d3.select ("#gen_" + j)
1461 .attr ("fill", "green")
1462 .attr ("stroke-width", 1.5)
1463 .attr ("stroke", "hsla (73, 66%, 33%, 0.61)")
1464 }
1465 if (sliderPgen [j - 1] == 0) {
1466 d3.select ("#gen_" + j)
1467 .attr ("fill", "black")
1468 .attr ("stroke-width", 0)
1469 .attr ("stroke", "black")
1470 }
1471 // Baterias: eleccion del simbolo segun el estado de carga
1472 //           if (sliderBat [j - 1] >= 13) {
1473 //           d3.select ("#bat_" + j)
1474 //           .text ("\uf240")
1475 //           } else if (sliderBat [j - 1] > 7.5 && sliderBat [j

```

```

- 1] == !0) {
1476 //           d3.select("#bat_" + j)
1477 //           .text("\uf241")
1478 //           } else if (sliderBat[j - 1] > 5.5 && sliderBat[j -
           1] == !0) {
1479 //           d3.select("#bat_" + j)
1480 //           .text("\uf242")
1481 //           } else if (sliderBat[j - 1] <= 5.5 && sliderBat[j
           - 1] == !0) {
1482 //           d3.select("#bat_" + j)
1483 //           .text("\uf243")
1484 //           }
1485
1486 // Modificar el ancho del rectangulo con la carga.
1487 var axis_charge = d3.scale.linear().domain([0,13]).range
           ([0,12]);
1488 d3.select("#bat_" + j)
1489 .attr("width", axis_charge(sliderBat[j - 1]))
1490
1491 if(sliderBat[j - 1] == 0){
1492 d3.select("#batterytext_" + j)
1493 .text(null)
1494 }else{
1495 d3.select("#batterytext_" + j)
1496 .text(Math.round (sliderBat[j - 1] / 13 * 100) + '%')
1497 }
1498
1499
1500
1501

```

```

1502 // Baterias: estado de la bateria; descargando = 1, cargando
        = -1
1503 if (sliderBatState[j - 1] == 1) {
1504 d3.select("#bat_" + j)
1505 .attr("fill", "red")
1506 } else if (sliderBatState[j - 1] == -1) {
1507 d3.select("#bat_" + j)
1508 .attr("fill", "green")
1509 } else if (sliderBatState[j - 1] == 0) {
1510 d3.select("#bat_" + j)
1511 .attr("fill", "deepskyblue")
1512 }
1513
1514 }
1515 }
1516
1517 function DibujaP() { //funcion para dibujar P al
        pulsar el boton Power
1518 botonV = 0;
1519 botonPgen = 0;
1520 botonPneta = 0;
1521 botonP = 1;
1522 $("#legend-voltage").hide();
1523 $("#legend-netpower").hide();
1524 $("#legend-conpower").show();
1525 $("#legend-genpower").hide();
1526 d3.selectAll(".node")
1527 .attr("stroke", "white")
1528 .attr("stroke-width", 0)
1529 }

```

```

1530
1531 function DibujaPgen() {                               //funcion para dibujar P
           al pulsar el boton Power
1532 botonV = 0;
1533 botonPgen = 1;
1534 botonPneta = 0;
1535 botonP = 0;
1536 $("#legend-voltage").hide();
1537 $("#legend-netpower").hide();
1538 $("#legend-conpower").hide();
1539 $("#legend-genpower").show();
1540 d3.selectAll(".node")
1541   .attr("stroke", "white")
1542   .attr("stroke-width", 0)
1543 }
1544
1545 function DibujaPneta() {                               //funcion para dibujar
           P al pulsar el boton Power
1546 botonV = 0;
1547 botonPgen = 0;
1548 botonPneta = 1;
1549 botonP = 0;
1550 $("#legend-voltage").hide();
1551 $("#legend-conpower").hide();
1552 $("#legend-genpower").hide();
1553 $("#legend-netpower").show();
1554 d3.selectAll(".node")
1555   .attr("stroke", "white")
1556   .attr("stroke-width", 0)
1557 }

```

```

1558
1559 function DibujaV() { //funcion para dibujar V al
        pulsar el boton Voltage
1560 botonV = 1;
1561 botonPgen = 0;
1562 botonPneta = 0;
1563 botonP = 0;
1564 $("#legend-voltage").show();
1565 $("#legend-netpower").hide();
1566 $("#legend-conpower").hide();
1567 $("#legend-genpower").hide();
1568 d3.selectAll(".node")
1569   .attr("stroke", "white")
1570   .attr("stroke-width", 0)
1571 }
1572
1573
1574 function uploadZip(event) {
1575   var file = event.target.files[0]; // file uploaded
1576   zip.loadAsync(file).then(function (zip) {
1577
1578     console.log(zip.files);
1579     zip_files = zip.files;
1580     async.parallel(read_fcns, function (err, data) {
1581       data_obj = data;
1582       // Aqui tenemos los datos
1583       console.log(data_obj);
1584       console.log(data_obj.slack_pq[0].P1);
1585       // console.log(data_obj)
1586       //-----Creacion del mapa

```

```

-----//
1587 // drawNodes ()
1588 var w = $("#map").width (); // estas son las
    dimensiones del mapa
1589 var h = $("#map").height ();
1590 var padding = 15;
1591
1592
1593 // Creacion de svg
1594 var svg = d3.select("#map")
1595 .append("svg")
1596 .attr("width", w)
1597 .attr("height", h)
1598 .call(d3.behavior.zoom() // Creacion del
    zoom
1599 .scaleExtent([1, 6])
1600 .on("zoom", function () {
1601 svg.attr("transform", "translate(" + d3.event.translate + ")
    " + " scale(" + d3.event.scale + ")")
1602 })))
1603 .append("g")
1604
1605
1606 // Creacion de las escalas
1607 var xScale = d3.scale.linear()
1608 .domain([d3.min(data_obj.Buscoords, function (d) { return d.
    x; }), d3.max(data_obj.Buscoords, function (d) { return d
    .x; })])
1609 .range([padding, w - padding]);
1610

```

```

1611 var yScale = d3.scale.linear()
1612 .domain([d3.min(data_obj.Buscoords, function (d) { return d.
      y; }), d3.max(data_obj.Buscoords, function (d) { return d
      .y; })])
1613 .range([h - padding, padding]);
1614
1615 var layer_lines = svg.append("g");
1616 var layer_nodes = svg.append("g");
1617 var layer_battery = svg.append("g");
1618 var layer_legend = svg.append("g");
1619
1620 // Draw the line
1621
1622 layer_lines.selectAll(".line")
1623 .data(data_obj.Lines)
1624 .enter()
1625 .append("line")
1626 .attr("class", "line")
1627 .attr("id", function (line, line_index) { return ("line_" +
      (line_index + 1)); })
1628 .attr("x1", function (line) { return xScale(_.findWhere(
      data_obj.Buscoords, { Busname: line.Bus1 }).x); }) //
      el findWhere se utiliza para sacar los datos de
      coodenadas de
1629 .attr("y1", function (line) { return yScale(_.findWhere(
      data_obj.Buscoords, { Busname: line.Bus1 }).y); }) //
      del archivo Buscoords, ya que en Lines solo tenemos el
      origen y
1630 .attr("x2", function (line) { return xScale(_.findWhere(
      data_obj.Buscoords, { Busname: line.Bus2 }).x); }) //

```

```

        destino de las lineas
1631 .attr("y2", function (line) { return yScale(_.findWhere(
        data_obj.Buscoords, { Busname: line.Bus2 }).y); })
1632 .attr("stroke-width", 1.5)
1633 .attr("stroke", "darkcyan")
1634 .on("click", function (line, line_index) { selectLine(
        line_index + 1) })
1635
1636 // Definir y destacar los puntos con carga y el trafo
1637
1638
1639 // var loads_points;
1640
1641 layer_nodes.selectAll(".node")
1642 .data(data_obj.Loads)
1643 .enter()
1644 .append("circle")
1645 .attr("class", "node")
1646 .attr("id", function (load, load_index) { return ("load_" +
        (load_index + 1)); }) // El load index empieza en 0
1647 .attr("cx", function (load) { return xScale(_.findWhere(
        data_obj.Buscoords, { Busname: load.Bus }).x); })
1648 .attr("cy", function (load) { return yScale(_.findWhere(
        data_obj.Buscoords, { Busname: load.Bus }).y); })
1649 .attr("fill", "rgba(220,20,60,0.7)")
1650 .attr("r", 4)
1651 // .on("click", function (bus_name, index) { selectNode(
        bus_name.Bus, index) }) // con la funcion selectNode al
        hacer click en cualquier carga
1652 .on("click", function (load, load_index) { selectLoad(

```



```

        load_index + 1) }) // con la funcion selectNode al hacer
        click en cualquier carga
1653 // te devolveria el bus seleccionado y un contador
1654
1655 var bus_trafo = _.findWhere(data_obj.Buscoords, { Busname: 1
        })
1656 layer_nodes.append(" circle ")
1657 .attr(" class ", "node ")
1658 .attr(" id ", "load_56 ")
1659 .attr(" cx ", xScale(bus_trafo.x))
1660 .attr(" cy ", yScale(bus_trafo.y))
1661 .attr(" fill ", "yellow ")
1662 .attr(" r ", 4)
1663 //.on(" click ", function () { selectNode(bus_trafo.Busname,
        55) }) // Asociamos la carga 55 al trafo
1664 .on(" click ", function () { selectLoad(56) }) //
        Asociamos la carga 56 al trafo
1665
1666 // Incluir imagen de un GENERADOR si existe en ese nodo.
1667 // Sera un simbolo de fontawesome.
1668 layer_nodes.selectAll(". nodesymbol ")
1669 .data(data_obj.Loads)
1670 .enter ()
1671 .append(" text ")
1672 .attr(" class ", "nodesymbol ")
1673 .attr(" id ", function (gen, index) { return ("gen_" + (index
        + 1)); }) // Atribuir a cada figura de panel un id, para
        luego utilizarlo en la funcion actualizarSlider()
1674 .attr(" x ", function (load) { return xScale(_.findWhere(
        data_obj.Buscoords, { Busname: load.Bus }) .x - 1); })

```

```

1675 .attr("y", function (load) { return yScale(_.findWhere(
        data_obj.Buscoords, { Busname: load.Bus }).y - 3.5); })
1676 .attr("text-anchor", "start") // horizontal
1677 .attr("alignment-baseline", "central") // vertical
1678 .text(function (gen, index) {
1679 if (data_obj.gpv_pq[index + 1][0].P2 == 0) {
1680 return "\uf185";
1681
1682 }
1683 })
1684 .attr('font-family', 'FontAwesome')
1685 .attr('font-size', '11px')
1686 // Incluir imagen de una BATERIA si existe en el nodo
1687 //         layer_nodes.selectAll(".batsymbol")
1688 //         .data(data_obj.Loads)
1689 //         .enter()
1690 //         .append("text")
1691 //         .attr("class", "batsymbol")
1692 //         .attr("id", function (battery, index) {
return ("bat_" + (index + 1)); }) // Atribuir a cada
figura de bateria un id, para luego utilizarlo en la
funcion actualizarSlider()
1693 //         .attr("x", function (load) { return xScale
(_.findWhere(data_obj.Buscoords, { Busname: load.Bus }).x
- 1); })
1694 //         .attr("y", function (load) { return yScale
(_.findWhere(data_obj.Buscoords, { Busname: load.Bus }).y
+ 1.5); })
1695 //         .attr("text-anchor", "start") // horizontal
1696 //         .attr("alignment-baseline", "central") //

```

```

        vertical
1697 //                .text(function (bat , index) {
1698 //                    if (data_obj.bat_p[index + 1][0].State
        == !2) {
1699 //                        return "\uf242";
1700 //                    }
1701 //                })
1702 //                .attr('font-family', 'FontAwesome')
1703 //                .attr('font-size', '10px')
1704 //                .on("click", function (bat , bat_index) {
        SelectBat(bat_index + 1) });
1705
1706
1707 //prueba de dibujar rectangulo y moverlo con la carga de la
        bateria
1708
1709 //                var axis_charge = d3.scaleLinear().domain
        ([0,13]).range([0,4]);
1710
1711 const dx = 2;
1712 const dy = -10;
1713 const bat_width = 12;
1714 const bat_height = 6;
1715
1716 layer_nodes.selectAll(".nodecharge") // Dibujar rectangulo
        interior de carga
1717 .data(data_obj.Loads)
1718 .enter()
1719 .append("rect")
1720 .attr("class", "nodecharge")

```

```

1721 .attr("id", function (battery , index) { return ("bat_" + (
        index + 1)); }) // Atribuir a cada figura de bateria un
        id, para luego utilizarlo en la funcion actualizarSlider
        ()
1722 .attr("x", function (load) { return xScale(_.findWhere(
        data_obj.Buscoords, { Busname: load.Bus }).x) + dx; })
1723 .attr("y", function (load) { return yScale(_.findWhere(
        data_obj.Buscoords, { Busname: load.Bus }).y) + dy; })
1724 // .attr("width", 2) //20% inicial
1725 .attr("width",function (bat , index) {
1726 if (data_obj.bat_p[index + 1][0].State != 2) {
1727 console.log(data_obj.bat_p[index + 1][0].kWh);
1728 return data_obj.bat_p[index + 1][0].kWh;
1729
1730 }
1731 })
1732 .attr("height", bat_height)
1733 .attr("stroke", "none")
1734 .attr("stroke-width", 0.5)
1735 // .attr("fill", "green")
1736 .attr("fill",function (bat , index) {
1737 if (data_obj.bat_p[index + 1][0].State != 2) {
1738 return "deepskyblue";
1739 }
1740 })
1741 .on("click", function (bat , bat_index) { SelectBat(bat_index
        + 1) });
1742
1743 layer_nodes.selectAll(".nodebattery") // Dibujar
        rectangulo exterior de marco

```

```

1744 .data(data_obj.Loads)
1745 .enter()
1746 .append("rect")
1747 .attr("class", "nodebattery")
1748 .attr("id", function (battery, index) { return ("battery_" +
    (index + 1)); })
1749 .attr("x", function (load) { return xScale(_.findWhere(
    data_obj.Buscoords, { Busname: load.Bus }).x) + dx; })
1750 .attr("y", function (load) { return yScale(_.findWhere(
    data_obj.Buscoords, { Busname: load.Bus }).y) + dy; })
1751 .attr("width", bat_width)
1752 .attr("height", bat_height)
1753 // .attr("stroke", "black")
1754 .attr("stroke", function (bat, index) {
1755 if (data_obj.bat_p[index + 1][0].State != 2) {
1756 return "black";
1757 }
1758 })
1759 .attr("stroke-width", 0.5)
1760 .attr("fill", "none")
1761 // .attr("fill", function (bat, index) {
1762 // if (data_obj.bat_p[index + 1][0].State
    == !2) {
1763 // return "rgba(0,0,0,0)";
1764 // }
1765 // })
1766 .on("click", function (bat, bat_index) { SelectBat(bat_index
    + 1) });
1767
1768 layer_nodes.selectAll(".nodebatteryline") //

```

Dibujar linea exterior de marco

```
1769 .data(data_obj.Loads)
1770 .enter()
1771 .append("rect")
1772 .attr("class", "nodebatteryline")
1773 .attr("id", function (battery, index) { return ("battery_" +
    (index + 1)); })
1774 .attr("x", function (load) { return xScale(_.findWhere(
    data_obj.Buscoords, { Busname: load.Bus }).x) + dx +
    bat_width; })
1775 .attr("y", function (load) { return yScale(_.findWhere(
    data_obj.Buscoords, { Busname: load.Bus }).y) + dy + 1.5;
    })
1776 .attr("width", 0.6)
1777 .attr("height", 3)
1778 // .attr("stroke", "black")
1779 .attr("stroke", function (bat, index) {
1780 if (data_obj.bat_p[index + 1][0].State != 2) {
1781 return "black";
1782 }
1783 })
1784 .attr("stroke-width", 0.6)
1785 .attr("fill", "none")
1786
1787 layer_nodes.selectAll(".nodetext")           // Escribir el
    valor en kWh de la bateria
1788 .data(data_obj.Loads)
1789 .enter()
1790 .append("text")
1791 .attr("class", "nodetext")
```

```

1792 .attr("text-anchor", "middle")
1793 .attr("alignment-baseline","central")
1794 .attr("id", function (battery , index) { return ("
        batterytext_" + (index + 1)); })
1795 .attr("x", function (load) { return xScale(_.findWhere(
        data_obj.Buscoords , { Busname: load.Bus }).x) + dx +
        bat_width/2; })
1796 .attr("y", function (load) { return yScale(_.findWhere(
        data_obj.Buscoords , { Busname: load.Bus }).y) + dy +
        bat_height/2; })
1797 .attr("fill","black")
1798 .text(function (bat , index) {
1799 if (data_obj.bat_p[index + 1][0].State != 2) {
1800 var a = data_obj.bat_p[index + 1][0].kWh/13*100
1801 return a.toFixed(0)+"%"
1802 }
1803 })
1804 .attr('font-size', '0.3em')
1805 });
1806 });
1807 }

```