Managing storage capacity and production set point for photovoltaic production plants

Relatore
Ch. Prof. Andrea Torsello

Laureando
Luca Zorzi
Matricola 810881

Anno Accademico
2011 / 2012
Abstract

The objective of this thesis is to realize an intelligent application that manages renewable energy production plants with the capability of storing energy into batteries, and in particular for the FIAMM Green Energy Island plant. Factors to be taken into account are mainly: energy market prices, energy consumption forecasting, energy production forecasting, state of charge of the batteries and signals of the smart grid manager. At each interval the data is updated and the model is verified, making the necessary modifications to the power curve in case of variations with respect to the forecasting made in a previous time. The application interacts with a SCADA system using a database.
Glossary

**AC:** Alternating Current. The movement of electric charge periodically reverses direction. Is the form in which electric power is delivered to businesses and residences.

**DBMS:** DataBase Management System. A software package with computer programs that controls the creation, maintenance, and use of a database.

**DC:** Direct Current. Unidirectional flow of electric charge. Is produced by sources such as batteries and solar cells.

**DSO:** Distribution System Operator. In the electricity supply industry is responsible for operating, ensuring the maintenance of and, if necessary, developing the distribution system in a given area and, where applicable, its interconnections with other systems.
Source: [http://userwikis.fu-berlin.de/display/energywiki/distribution+system+operator](http://userwikis.fu-berlin.de/display/energywiki/distribution+system+operator)

**EOC:** End Of Charge. Specific state in which batteries goes from 80% to 100% (in our case) at a given power input value.

**GME:** Gestore dei Mercati Energetici. Is vested with the economic management of the Electricity Market under principles of transparency and objectivity, with a view to promoting competition between producers and ensuring the availability of an adequate level of Reserve Capacity.
Source: [http://www.mercatoelettrico.org/En/Tools/Glossario.aspx](http://www.mercatoelettrico.org/En/Tools/Glossario.aspx)

**PLC:** Programmable Logic Controller. A digital computer used for automation of electromechanical processes, such as control of machinery on factory assembly lines. Is designed for multiple inputs and output arrangements, extended temperature ranges, immunity to electrical noise, and resistance to vibration and impact.
Source: [http://en.wikipedia.org/wiki/Programmable_logic_controller](http://en.wikipedia.org/wiki/Programmable_logic_controller)

**SCADA:** Supervisory Control And Data Acquisition. A type of industrial control system (ICS). Industrial control systems are computer controlled systems that monitor and control industrial processes that exist in the physical world.

**SOC:** State Of Charge. The percentage of charge of a battery or a battery pack.
# Contents

1 Introduction
   1.1 FIAMM Group ........................................... 4
   1.2 FIAMM SoNick batteries ................................. 4
   1.3 FIAMM Green Energy Island .............................. 4
   1.4 Overall functioning ..................................... 5
   1.5 Goal ................................................... 7
   1.6 Contents ............................................... 7

2 Power plant requirements  
   2.1 Curve of power ......................................... 8
   2.2 \(\cos \varphi\) ............................................. 9
   2.3 Power holes ............................................ 11
   2.4 Conclusions ............................................ 11

3 Software requirements  
   3.1 Requirements analysis ................................... 12
      3.1.1 Production ......................................... 12
      3.1.2 Consumption ....................................... 13
      3.1.3 Energy selling ...................................... 13
      3.1.4 Energy buying ...................................... 13
      3.1.5 Batteries and inverters ............................. 13
      3.1.6 Features to be implemented ........................ 14
      3.1.7 Regulation .......................................... 16
      3.1.8 Values forcing ...................................... 16
      3.1.9 Events and festivals ................................ 16
      3.1.10 Other elements .................................... 16
      3.1.11 Plugins ............................................ 17
      3.2 Requirements specification ............................. 17
         3.2.1 Functional requirements ........................... 18
         3.2.2 Non-functional requirements ...................... 21

4 Prototype  
   4.1 General architecture .................................... 23
   4.2 Configuration file ...................................... 25
   4.3 Database structure ..................................... 25
   4.4 Consumption forecast algorithm ......................... 26
   4.5 Midnight energy calculation ............................ 27
   4.6 Curve forecasting algorithm ............................ 27
Chapter 1

Introduction

Next ten-years scenario let us foresee that a strong increase of energy production from renewable sources will be promoted in Europe, Italy in particular. Production that will be related also to the so called grid parity: actual trends seems to lead to a parity between the cost per kWh produced from renewable and traditional sources in a matter of few years.

Eolic and solar power (with a potential of development higher than hydroelectric one) depends on the intensity of elements (wind and sun) whose are uncontrollable by nature. So production presents either macro peaks related to season changes, and daily micro peaks related to weather conditions up to day/night differences.

There is the need to align the production trend with the consumption trend, and to store energy. The simplest and technically feasible solution of lowing production would lead to a disuse of the source and less yield of the investment. Furthermore there is a problem with the continuity of supplying in places where it is not possible to connect to the general power grid, and no production means no energy at all. Capacity of storing energy and releasing it on moments where it is needed (when the price is higher due to the supply-and-demand law) becomes a crucial role. Using batteries designed and built for this purpose results to be the most efficient way, with respect to, for example, idric systems where water is pumped on basins. There are not only production but also distributive variances. If we are not able to manage production and consumption peaks, due to fundamental rule that electric energy have on human activities, we will be faced with the need of structuring the productive and distributive net to handle worst scenarios (negative peak of availability or positive peak of consumption) with a negative effect on investments, and on environment due to new power stations and delivering structures.

Source: [4]

It comes out that not only a well structured plant with renewable energy production sources and storing capacity is needed, but also intelligent algorithms have to be implemented to manage the plant. In fact many aspects have to be taken into account, as described in Chapter 3, and in the future probably even more sophisticated algorithms will be implemented as well as new generation’s batteries and renewable energy sources.

FIAMM Group decided to move ahead of the times and realized the so called FIAMM Green Energy Island, which fully implements this type of plant. The
intelligent algorithm, capable of making decisions on when and how much power to put in the energy grid, is the objective of this thesis.

1.1 FIAMM Group

Fabbrica Italiana Accumulatori Motocarri Montecchio (Italian Manufacturer Motorvehicle Batteries Montecchio) was born in 1942. At the beginning the company concentrates on the production of batteries for automobiles and the fork lift industry. Not long after that, production is started on stationary and semi-stationary batteries, vital to a country rebuilding its industries and railway network.

In the years of mass motorization, the brand begins to appear in automobile showrooms alongside those of the most important car makers. Also the production of acoustic devices begins.

Leaving out part of the history of the brand, nowadays FIAMM is present in sixty countries, with about seventy percent of its sales overseas. It has ten production plants (in Italy, USA, Czech Republic, Brazil, India and China) and more than twenty commercial and technical branches (Germany, United Kingdom, Czech Republic, Poland, Slovakia, Austria, France, USA, Spain, Brazil, Japan, Singapore, Korea, Malaysia, China, India and so on) and a wide network of importers and distributors. FIAMM produces starter batteries and may include among its top clients some of the most prestigious European car makers, including: OPEL-GM, Mercedes Benz, PSA (Peugeot-Citroen), Renault, Ferrari and Maserati. Furthermore produces horns (both for cars, heavy trucks and specialty equipment markets), antenna systems and industrial batteries.

Source: [2]

1.2 FIAMM SoNick batteries

FIAMM SoNick sodium nickel chloride batteries:

- High capacity (though smaller and lighter than traditional batteries)
- Reliable at high temperature (no cooling needed)
- High cycling capability (reliability guaranteed for many years)
- No maintenance
- No dangerous materials

Source: [3]

1.3 FIAMM Green Energy Island

FIAMM realized, in collaboration with Terni Energia, Elettronica Santerno, Galileia and ProSoft:

- First renewable energy plant in Italy, one of the first in Europe, equipped with an appropriate storage capacity (called energetic island)
Connected to the net of the production plant and so to the electric grid, with a peak power of 180 kW, 200 kWh per year

With a storage capacity of 230 kWh per day, equivalent to 85k kWh/year = 40% of total production (considering an entire cycle per day)

That aims to gain incentives for the correct forecast of production for the day after = +20% of the base incentive (for plants with power between 200 kW and 10 MW) for at least 300 days per year. Reliability obtained with a mix of fine-grain weather forecasts and the storage system. Extra incentives consists in 15-20.000 Euros per year.

4500 m$^2$ occupied by the plant
1150 m$^2$ of panels
Investment of 600k Euros (Panels, inverters, etc.) + 350k (Storage)
Power plant produces more than twice the consumption of the logistic plant
106 tons/year of CO$_2$ are saved
Surplus value is equivalent to 8/9 people employed per year
Environmental impact due to construction of new energy production plants is avoided

Source: [4]

1.4 Overall functioning

Figure 1.1 shows the overall architecture of the plant situated in Almisano (Vicenza, Italy). Here is a brief description on what is the role of every part of the system and how it works:

Production plant: consists on solar panels, but can be windmills or any source of renewable energy on which a forecasting of the production can be done

Energy storage: consists on a certain number of batteries packs those can store or release electric energy

Energy grid: is the actual energy grid at which every building is connected to, in the next future will become smart in the sense that the distribution system operator can send signals to regulate the flow of electric power

Consumption: actually consists on a logistic centre, but can be every kind of electric consumption on which it is possible to forecast the demand curve

PLC(s): can be one or more PLCs $^1$ dedicated to the acquisition of data from the other parts of the plant and with the job of setting the regulation

$^1$Programmable Logic Controller, hard real time system in which output results are produced in response to input conditions within a limited time
Figure 1.1: Overall architecture of the plant

- Server: the machine on which the algorithm runs, takes decisions based on the actual data and sends the result to the PLC(s) via a SCADA.²

- Regulation points: the points on which happen the regulation of the amount of power. Setting the first regulation point means to decide the "exact" amount of power outgoing to the grid, while setting the second regulation point is equivalent if the value of power consumption is known exactly. Several factors influence this operation, but what the plant does is trying to reach the given value.

The server runs both the algorithm developed in this thesis and the SCADA, which allows to interface with the PLCs. PLCs constantly read value from the plant and update them on the server, and also have the task to put in practice the decisions that the algorithm finds out. The server receives constantly the new values from the PLCs as well as the energy market prices, the production forecasting, the state of charge of the batteries, and computes the consumption forecasting. On the other hand, it sends values related to the regulations for the first point, which is the one that allows to decide exactly the energy to be sold.

The production plant produces power (energy) that can be used to charge the batteries, to satisfy the consumption or to be sold. Batteries can store energy or can be discharged to use the energy to cover the consumption or to be sold. They are grouped in packs, and every pack have a

---
²Supervisory Control And Data Acquisition, computer system that monitor and control industrial, infrastructure, or facility-based processes
certain number of batteries. The logistic centre consumes a certain amount of power, estimated via the implemented algorithm that is clearly explained in Chapter 6. Finally the smart energy grid is the external network at which energy is sent (sold) during the most effective periods (according to the decisions of the intelligent algorithm), moreover in every moment energy can be bought from the grid and can be used to satisfy consumptions or to charge the batteries. For more details on the functioning of the system and of the algorithm, see next chapters in which every aspect is explained.

1.5 Goal

The goal of this thesis is to develop an algorithm that manages the plant, in particular the sold energy and the storage capacity (the battery packs). The program has to schedule automatically the tasks without effort by the administrators of the system. The decisions have to be made on factors like weather and production forecasting, prices of the energy market and costs of the batteries. In particular the algorithm has to satisfy a predefined energy curve on a hourly basis, and to check for signals sent by the smart grid if there is the necessity to lower or increase the power in a certain period. More details can be found in Chapter 3.

1.6 Contents

In this first chapter motivations of the work are exposed, as well as who FIAMM Group is and how FIAMM Green Energy Island works. Chapter 2 contains the requirements FIAMM wants his energy island to implement. Chapter 3 lists the features that the intelligent program and algorithm have to implement. Chapter 4 describes all the aspects implemented in the prototype application, which was an intermediate step to the final product. In Chapter 5 it is described how a separate application saves in the database market prices for selling energy. From Chapter 6 to Chapter 10 all the aspects related to the final application are explained. Chapter 6 describes the algorithm used to compute the forecasting of the consumption of the logistic plant present in the energy island. In Chapter 7 are listed all the profiles (ways of functioning) that the algorithm implements. Chapter 8 describes the problem of forecasting the curve of power of the plant while Chapter 9 describes the problem of maintaining the declared curve. Chapter 10 describes the architecture and the general way of functioning of the final application. Finally Chapter 11 explains the results of the work made and contains the final considerations about the project.
Chapter 2

Power plant requirements

As said in the previous chapter, in the next future renewable energy plants will increase. The major problem is that the production from photovoltaic plants or windmills cannot be known exactly a priori and is not even constant. In order to avoid surplus or lack of power, and other particular electronic problems, on some points of the grid, all the so called islands will have to respect particular laws and rules defined by the smart grid manager. In general there is the need to cover some basic features that let every plant allow the grid manager to solve these issues about congestion on nodes of the grid. This is why FIAMM requires the Green Energy Island plant to implement three main requirements:

1. Declare the energy coming out from the plant the day after, and maintain it as close as possible
2. Maintain the largest possible value of $\cos \varphi$
3. Fill the "power holes"

The first requirement is the most important from the point of view of the smart grid, because once the consumptions on a certain node are forecasted and the production curve on the same node is done, the difference between the two gives the forecasting of how much power will lack or surplus. The second requirement involves both the production plant and the smart grid because the angle between active and reactive power is an important value when dealing with power supply, as described in Section 2.2. The third requirement is important mostly for the island since it allows to counter a possible lack of production or a surplus of consumption when the power (or energy) profile has to be kept as close as possible to the declared one.

The last two requirements are achieved by the inverters and the PLCs, so the first requirement is the one to consider for the purpose of the application that need to be developed.

2.1 Curve of power

The goal of the algorithm is to declare a curve of power (and consequently the curve of energy) coming out from the plant the day ahead. The real value of energy have to be constantly monitored and kept as close as possible to the
value declared the day before. In case the energy remains on a range of ±10% with respect to the expected value (on an hourly basis check), for a certain number of days per year, an economic bonus will be earned. (For more details see Chapter 3 Section 6)

Every day at a certain hour a curve of power is calculated for the day after. Once declared a value of the power, the SCADA reads it and provides it to the underlying system (the PLCs) which have the task to make sure the value of power outcoming the plant is exactly the forecasted value, using a certain combination of photovoltaic and stored energy.

It is also possible for the DSO \(^1\) to require a variation of the power putted on the net (via a signal). If the variation is putted into effects by the system, a bonus will be applied increasing the profit, otherwise no extra-profit will be given.

As described later in the thesis, there are many ways the plant can be used, but the main goal of a plant with storing capacity and an intelligent algorithm connected to a smart grid is to satisfy a previously declared curve of power, and respect the signals sent by the smart grid itself. How it is implemented, is explained in particular in Chapters 8 and 9 where the core of the algorithm is explained as well as how the profile (ways of functioning) are implemented.

### 2.2 \(\cos \phi\)

AC power flow has the three components:

- real power (also known as *active power*) \(P\), measured in Watts (W)
- apparent power \(S\), measured in Volt-Amperes (VA)
- reactive power \(Q\), measured in reactive Volt-Amperes (var)

The power factor is defined as:

\[
\frac{P}{S}
\]

In the case of a perfectly sinusoidal waveform, \(P\), \(Q\) and \(S\) can be expressed as vectors that form a vector triangle such that:

\[
S^2 = P^2 + Q^2
\]

If \(\phi\) is the phase angle between the current and voltage, then the power factor is equal to the cosine of the angle, \(|\cos \phi|\), and:

\[
|P| = |S||\cos \phi|
\]

Since the units are consistent, the power factor is by definition a dimensionless number between 0 and 1. When power factor is equal to 0, the energy flow is entirely reactive, and stored energy in the load returns to the source on each cycle. When the power factor is 1, all the energy supplied by the source is consumed by the load. Power factors are usually stated as "leading" or "lagging" to show the sign of the phase angle.

For a "visual" explanation, see Figure 2.1, which shows instantaneous and average power calculated from AC voltage and current with a zero power factor

\(^1\)Distribution System Operator, in the electricity supply industry
($\phi = 90, \cos \phi = 0$). The blue line shows all the power is stored temporarily in the load during the first quarter cycle and returned to the grid during the second quarter cycle, so no real power is consumed.

Figure 2.1: Instantaneous and average power calculated from AC voltage and current with a zero power factor ($\phi = 90, \cos \phi = 0$)

On the plant every pack of batteries is connected to the internal energy network via inverters, responsible to convert the AC power to DC power when batteries are charged, and viceversa from DC to AC when batteries are discharged. Objective of the system is to maintain the value of $\cos \phi$ as large as possible. Usually the lower bound for $\cos \phi$ is approximately 0.96. Since the inverters are already built and instructed to maintain that value as close as possible to 1, it is not necessary for the intelligent algorithm to take this problem into account. So there is no need to compute the curve of reactive power $Q$, but only the active power $P$ profile is required to be defined and checked. The remaining part of the task, consisting on keeping a large value for $\cos \phi$, is made by the inverters connected to the battery packs. Otherwise, if the inverters do not do this automatically, the algorithm should define both active and reactive power curve, so that the PLCs can send the values to the inverters and the plant can work efficiently.
2.3 Power holes

Last but not least, the system has to fill the power holes: the moments in which the production is not sufficient to cover the demand of power because of a temporary (in a matter of seconds or minutes) lack of sun or wind. This is done automatically using the stored energy, when a certain value is setted on the regulation point. The PLCs automatically recognize that the produced amount of power is lower than the required value, and use the stored power to reach the actual required value.

![Figure 2.2: Clouds obscuring the sun cause a power hole](image)

A typical situation that the plant has to deal with is the one illustrated in Figure 2.2, in which in a cloudy day there can be many power holes due to the clouds passing and obscuring the sun.

2.4 Conclusions

In conclusion, the task of keeping $\cos \varphi$ as large as possible is implemented by the inverters and the task of filling the "power holes" is implemented by the logic in the PLCs. The only task the algorithm have to implement is to declare a curve of power (energy) for the day ahead, and mantain it as close as possible to the declared one during the hours in which it is needed (see Chapter 3 on section about features to be implemented).
Chapter 3

Software requirements

In the previous chapter the need of a software capable of managing the plant in terms of set point to the smart grid and state of charge of the batteries has been defined. This chapter contains the requirements analysis in Section 3.1 and the requirements specification in Section 3.2 for that software. The requirements analysis explains in detail all the things that have to be considered for every particular aspect of the problem, defined during meetings with experts of FI-AMM and Galileia\(^1\). Furthermore are defined some characteristics regarding the functioning of the program. The requirements specification contains the formal specification of the functional and non-functional requirements using a simple schema defined at the beginning of the section. The book used as a reference is “Principi di Ingegneria del Software” by Roger S. Pressman [5]

3.1 Requirements analysis

A base for defining what are the aspects the application has to take into account can be found in BCCST10 [1], in which a first theoretical approach has been defined. In fact in the paper a feasibility study is made as well as the presentation of the operation profile for the energy storage system, accounting for its size, capacity and integral operational constraints.

In order to achieve the goal of implementing an intelligent algorithm that can decide in the right manner when and how much power have to go to the smart grid or to the batteries, all the following aspects have to be taken into account.

3.1.1 Production

On a daily basis Université de Grenoble sends to the plant server, via SFTP\(^2\), a file containing the estimated curve of power production of the photovoltaic plant.

The data is furnished in a formatted text file that constraints for every minute of the day the estimated amount of power produced by the production plant. This file is loaded only after aggregation of the data is done: the average power

---

\(^1\)http://www.galileia.it/

\(^2\)Secure File Transfer Protocol
for intervals of 10 minutes is computed and stored in the database. The modeling system they use is retrained using the data sent back from the plant representing the real curve of power production read by the SCADA system. Details about how this forecasting is made are not available.

### 3.1.2 Consumption

A logistics centre is situated next to the solar energy production plant. The centre needs an amount of power (or energy, equivalently) that can be supplied from the solar plant, if a sufficient quantity is produced or is present in the batteries, instead of buying it from the operator. The forecasting of the consumption curve is done on a daily basis, and the value of the curve is then stored in an appropriate table in the database. Details on the algorithm used to compute this forecasting can be found in Chapter 6.

### 3.1.3 Energy selling

Every day at about 11 AM the energy market prices are available for download on the GME website. Prices are setted from 00 AM to 12 PM of the day after, for intervals of an hour each. More details about this aspect can be found in Chapter 5. The real curve of sold power is stored in the database so that at the end of the day, a report on how much energy has been sold and at what price can be made.

### 3.1.4 Energy buying

For what concerns prices for buying energy, they are not defined yet.

### 3.1.5 Batteries and inverters

Batteries are managed as packs, everyone containing a certain number of SoNick batteries, and connected to a single inverter that can convert both from AC to DC and viceversa. SoNick batteries have to be used on a range varying from a minimum of 20% to a maximum of 80% of charge, except some special cases in which the charge can arrive to 100%. This last scenario happens only once every 60 hours, requiring the batteries to be charged from below 80% to 100%, in order to let them check their own state of health, and we will call it EOC.

Main characteristics of this type of batteries are:

- Nominal power: 23.5 kWh
- Power output: 9 kW (from 20% to 80% of SOC)
- Power input: 10 kW (from 20% to 80% of SOC), 3 kW (from 80% to 100% of SOC)
- DC/DC rountrip efficiency: 85-90%

---

3http://www.mercatoelettrico.org/
4End Of Charge
5State Of Charge
• AC/AC rountrip efficiency: 75-80%
• Service life cycles: 4500
• Cost: ≈ 250 Euros/MWh

Self-consumption is about 130 W per battery when flat (not used), but are not considered when the 5% of the actual input or output power is higher that 130 W.
The efficiency is computed as the efficiency of the battery times the inverter’s one, since they are connected in series. The former is around $\sqrt{0.9}$ for the input and the same value can be used for the output. The latter is represented in the Figure 3.1.

![Figure 3.1: Efficiency of the inverter](image)

The real state of charge of every pack of batteries is constantly updated and stored in the database, in order to make it available for the application, as well as the number of working hours.

The actual situation of the plant in Almisano (Italy) allows to put in end of charge only one pack of batteries at a time because of phisical limits that will be surpassed in the next few months. The modifications will allow to send more batteries in end of charge state if there is the need.

### 3.1.6 Features to be implemented

After listing all the aspects that have to be considered, the main features the application need to implement are:

• Download the data for the day ahead from the Gestore dei Mercati Energetici website
• Make forecasting of power production
• Forecast the curve of power consumptions
• Periodically read the actual state of charge of the batteries
• Declare the forecasted curve of power outcoming to the grid, based on the criteria chosen by the system administrator
• Check for possible signals by the DSO, and consider the possibility of adapting to those signals
• Take into account all the aspects reported on the sections of this chapter in order to make the decisions
• Make the program as pluggable and dynamic as possible in order to meet possible future requirements (i.e. new types or number of batteries, new objective functions, new constraints)

Furthermore the algorithm can be tuned in order to meet one or more of the following objectives:

• Profiliation: sell all the produced energy, store the surplus and use batteries to compensate underproduction
• Economical optimization: earn as much money as possible by selling energy
• Battery life optimization: do few number of cycles (switches between charge and discharge state) in order to preserve batteries integrity
• DSO signals: earn an extra bonus respecting signals requiring a variation of the power output to the grid
• Optimal SOC: leads the plant to a situation ideal for satisfying the objective defined on the next time slots

The objectives are called profiles, and a more detailed explanation on how they are implemented on the application is present in Chapters 8 and 9. When profillation is selected, the maintaining algorithm have to respect the criteria defined by D.M. 06-08-2010:

• hourly program (in terms of energy)
• from 8 AM to 8 PM
• margin of error equal to 10%
• for at least 300 days in a year

Overrunning more that 65 days in a year will lead to a loss of the extra money bonus. The overrun is counted when, even for a single hour, real energy differs from declared energy by at 10%. Following DSO signals will not lead to an overrun to be counted in the time-slot of the signal.

The result we want to obtain is that for every time slot, an objective (strategy, profile) is chosen, and the system uses that information to select the function and the constraints to use in order to obtain the best values for power curve and SOC of batteries.
3.1.7 Regulation

Regulation is made on 1st point, meaning that the real value of power coming out from the plant will be kept as close as possible to the setted one. In fact the algorithm tells to the SCADA, and the SCADA to the PLCs, what should be the value of the power outgoing to the external energy grid. If that value is higher than the production at that moment, then the PLCs will try to use the batteries to reach such amount of power, otherwise it will put as much power as possible. In the opposite case, if the production is higher than the set point, PLCs will try to store energy in batteries if possible. In every case the real amount of power is registered and stored in the database.

3.1.8 Values forcing

Forcing set point values have to be permitted. A manual value can be setted (from the SCADA) for the regulation point (forcing), so that the actual algorithm will recognize this situation without possibility of changing it. The only possibility is to recompute the best curve with respect to the actual configuration, and wait for it to be applied when forcing is finished. Forcing the set point is the equivalent of telling to the algorithm in some time slots what is the power to put out to the smart grid.

3.1.9 Events and festivals

Festivals and particular events, such as maintenance, have to be considered because the amount of energy produced or consumed can vary much. Festivals are known and the influence is made on the consumptions. Events such maintenance can be defined by technicians and the influence is made mainly on the production. Taking into account these situations in advance makes the algorithm a lot more efficient, probably avoiding some economic sanctions for non-estimated variations.

System administrators have to be given the possibility of inserting and forcing data for the hours or days ahead, regarding consumption, production or regulation (set point). Furthermore, a semi-automatic (or possibly completely automatic) mechanism for recognizing festivals has to be implemented, with the goal of dismissing this task to the administrator.

3.1.10 Other elements

- The system is always updated with actual “real-time” data from the PLCs. Measured SOC can differ from forecasted SOC, meaning that production and consumption forecasting were wrong (or something unexpected happened), leading to the need of an adaption of the system to the new situation. New power curve must be computed in order to adapt to the change aiming to meet the required objective

- A certain amount of energy should be kept into the batteries in the cases in which for the day ahead it is forecasted that production will not cover consumption during first hours of the night and morning
• Once the algorithm recognizes that an overrun is made for that day, it can take advantage of the situation doing other overruns to get closer to the given objective (for example, sell more energy to gain more money)

• EOC (End Of Charge) have to be managed in a smart way, for example executing it during the night without affecting at all the program chosen for the profillation during the central hours of the day

• The possibility of turning the production plant off cannot be considered. The only possibility is to reduce production of not more than \( \approx 10\% \), but this algorithm will not consider this option

• Buying energy at a low price, mainly during the night, to store it into the batteries, can be accounted

• System failure have to be managed in a way that as soon as the application is restarted, all the conditions preceding the crash have to be restored

• At the end of the day a report has to be produced with particular informations about the entire day and the situation since the beginning of the year

3.1.11 Plugins

Future modifications to how

• selling prices
• buying prices
• production forecasting
• consumption forecasting
• objectives and constraints are defined
• batteries and inverters are managed

have to be accounted, so the architecture of the application should make it easy to modify these aspects without the need to rewrite big parts of the program. Some aspects, for example how selling prices are defined, are already masked to the application because of the presence of the database that acts as the only place in which the state of the plant and of the algorithm is stored. For the remaining aspects, ”plugins” should be implemented: see Chapter 10 for more details about how the architecture of the application reflects this idea.

3.2 Requirements specification

In the previous section all the aspects needed for the application to work properly has been defined, as the requirements analysis states. In the process of software engineering, the following part is to specify the requirements, so let’s define the functional and non-functional requirements.
3.2.1 Functional requirements

Functional requirements are nothing but services and functions that the application will have to implement. Functional requirements may be calculations, data manipulation and processing, or other specific functionalities that define what a system have to do.

Functional requirements derived from the analysis seen in the first part of this chapter are defined below. The code in is the form FRxn, where FR stands for functional requirement, x can be present or not and represents a particular aspect of the application, and n is a sequence number (for each category). Categories are P for production, C for consumption, M for market, B for batteries and inverters, and is empty to indicate a general aspect.

Code: FRP1  
**Description:** Historical production power  
**Function:** Store the real power produced by the production plant  
**Motivation:** In order to implement requirement FRP2 and improve the forecasts

Code: FRP2  
**Description:** Production power forecasting  
**Function:** Forecast the power produced by the production plant  
**Motivation:** The set point forecasted for the plant will have to consider the power produced by the photovoltaic plant

Code: FRC1  
**Description:** Historical consumption power  
**Function:** Store the real power consumed by the logistic centre  
**Motivation:** In order to implement requirement FRC2 and improve the forecasts

Code: FRC2  
**Description:** Consumption power forecasting  
**Function:** Forecast the power consumed by the logistic centre  
**Motivation:** The set point forecasted for the plant will have to consider the power consumed by the logistic centre

Code: FRC3  
**Description:** Festivals consumption power forecasting  
**Function:** Forecast the power consumed by the logistic centre in festival days  
**Motivation:** The set point forecasted for the plant in the festival days can vary much with respect to the other days

Code: FRM1  
**Description:** Market selling prices  
**Function:** Store market prices for selling energy  
**Motivation:** The set point computed can take into account selling energy to the smart grid

Code: FRM2
**Description**: Market buying prices  
**Function**: Store market prices for buying energy  
**Motivation**: The set point computed can take into account buying energy from the smart grid

**Code**: FRB1  
**Description**: Historical state of charge  
**Function**: Store the real state of charge of the battery packs  
**Motivation**: In order to implement FRB2 and to let the core algorithm know at each time the updated value of state of charge

**Code**: FRB2  
**Description**: State of charge forecasting  
**Function**: Forecast the state of charge of the battery packs  
**Motivation**: The set point computed have to take into account the forecasted state of charge of the batteries

**Code**: FRB3  
**Description**: End of charge strategy  
**Function**: Forecast when every battery pack is in EOC  
**Motivation**: Every battery pack needs to update its own state of health

**Code**: FRB4  
**Description**: Minimum state of charge  
**Function**: Batteries cannot have less than 20% of charge  
**Motivation**: The lower bound for the batteries is setted to 20% of charge

**Code**: FRB5  
**Description**: Maximum state of charge  
**Function**: Batteries cannot have more than 80% of charge, except when in EOC  
**Motivation**: The upper bound for the batteries is setted to 80% of charge

**Code**: FRB6  
**Description**: Maximum power output  
**Function**: Batteries cannot be discharged with more than 9 kW  
**Motivation**: The upper bound for the batteries output is due to phisical limitations of the batteries themselves and of the inverters

**Code**: FRB7  
**Description**: Maximum power input  
**Function**: Batteries cannot be charged with more than 10 kW, except when in EOC  
**Motivation**: The upper bound for the batteries input is due to phisical limitations of the batteries themselves and of the inverters

**Code**: FR1  
**Description**: Power curve declaration  
**Function**: The forecasted curve for the day ahead is declared to the smart grid manager (DSO)
**Motivation:** The DSO will have to check if the curve is respected in the considered time slots

**Code:** FR2  
**Description:** Profiles  
**Function:** The algorithm allows the administrator to choose the way the plant acts  
**Motivation:** The administrator can change the way the plant uses the produced energy, the energy present in the batteries and what are the aspects to consider when selling energy

**Code:** FR3  
**Description:** Signals  
**Function:** The algorithm allows to recognize and evaluate when to follow a signal  
**Motivation:** The DSO can send signals on the smart grid, and the algorithm have to take into account the feasibility and the gain on following those signals

**Code:** FR4  
**Description:** D.M. 06-08-2010  
**Function:** The maintaining algorithm have to respect the criteria defined by D.M. 06-08-2010  
**Motivation:** When the proper profile is selected, the maintaining algorithm have to respect an hourly program (in terms of energy) from 8 AM to 8 PM, with a margin of error equal to 10% and for at least 300 days in a year

**Code:** FR5  
**Description:** Profiles strategy  
**Function:** The algorithm allows to change profile in every time slot  
**Motivation:** Different strategies can be chosen during different parts of the day

**Code:** FR5  
**Description:** Set point forcing  
**Function:** The algorithm allows the set point to be forced  
**Motivation:** In some situations the administrator can force the value of the set point

**Code:** FR6  
**Description:** Day report  
**Function:** At the end of each day a report is produced  
**Motivation:** The administrator has to check the situation of the plant, to control if something went wrong, how much power has been produced or sold, and other parameters

**Code:** FR7  
**Description:** System failure  
**Function:** System failure have to impact the less possible on the plant  
**Motivation:** In case of system failure the program has to recover as much information as possible to restart working at the task of forecasting or maintaining the curve of power
3.2.2 Non-functional requirements

Non-functional requirements are often called qualities of a system, because are used to specify criteria that can be used to judge the operation of a system rather than specific behaviors.

Non-functional requirements derived from the analysis seen in the first part of this chapter are defined below. The code in is the form NFRxn, where NFR stands for non-functional requirement, x can be present or not and represents a particular aspect of the application, and n is a sequence number (for each category). Categories are P for production, C for consumption, M for market, B for batteries and inverters, and is empty to indicate a general aspect.

**Code**: NFRP1  
**Description**: Historical production power interval  
**Function**: Real power has to be stored every 10 minutes

**Code**: NFRP2  
**Description**: Production power forecasting interval  
**Function**: Forecasting power has to be done for time-slots of 10 minutes

**Code**: NFRP3  
**Description**: Production power forecasting hours  
**Function**: Forecasting has to be done for the next day and the successive 8 hours

**Code**: NFRP4  
**Description**: Production power forecasting timing  
**Function**: Forecasting has to be done before the forecasting algorithm is executed

**Code**: NFRC1  
**Description**: Historical consumption power interval  
**Function**: Real power has to be stored every 10 minutes

**Code**: NFRC2  
**Description**: Consumption power forecasting interval  
**Function**: Forecasting power has to be done for time-slots of 10 minutes

**Code**: NFRC3  
**Description**: Consumption power forecasting hours  
**Function**: Forecasting has to be done for the next day and the successive 8 hours

**Code**: NFRC4  
**Description**: Consumption power forecasting timing  
**Function**: Forecasting has to be done before the forecasting algorithm is executed
Code: NFRM1
Description: Market selling prices timing
Function: Store market prices for selling energy have to be known before the algorithm is executed

Code: NFRM2
Description: Market buying prices timing
Function: Store market prices for buying energy have to be known before the algorithm is executed

Code: NFR1
Description: Profilation
Function: A profile in which all the produced energy is sold, the surplus is stored and batteries are used to compensate underproduction

Code: NFR2
Description: Economical optimization
Function: A profile in which the objective is to earn as much money as possible by selling energy

Code: NFR3
Description: Battery life optimization
Function: A profile in which the objective is to do few number of cycles (switches between charge and discharge state) in order to preserve batteries integrity

Code: NFR4
Description: Optimal SOC
Function: A profile that leads the plant to a situation ideal for satisfying the objective defined on the next time slots

Code: NF1
Description: Curve declaration
Function: Curve forecasting and declaration have to be done before of the daily deadline the DSO setted

Code: NF2
Description: Curve mantaining
Function: Curve mantaining has to be done every 10 minutes, 24h/24

Code: NF3
Description: Signals monitoring
Function: Signals have to be monitored constantly every 2 seconds
Chapter 4

Prototype

When the requirements analysis was almost completed, a first prototype application has been developed. The prototype application is in fact a toy application, written in Java, used to test some aspects of the system and the ability of defining the correct optimization with respect to the maximum earning of money. As described in this chapter, it does not implement all the requirements defined in the Chapter 3, but only part of them.

Section 4.1 describes the architecture and general functioning of the application while Section 4.2 describes the structure of the configuration file used. Section 4.3 shows the structure of the database, Section 4.4 briefly describes the algorithm used for the consumption forecasting and Section 4.5 explains how the algorithm deals with the possibility of keeping energy in the batteries for the first hours of the day ahead. In Section 4.6 is reported how the definition of the curve is made and in Section 4.7 is reported how the hourly energy amount is kept. Section 4.8 lists the libraries used to implement particular features. Section 4.9 described the results obtained and finally Section 4.10 contains the conclusions made after the development and deployment of this prototype.

4.1 General architecture

The application is stateless, and all the data needed is stored into the database, which structure is described later in this chapter, as for the configuration file. The main class of the program, called DecisionEngine, has the job of loading the configuration, the plugin manager, and scheduling the execution of the PluginManager, Core and Forecast classes. The former is scheduled every 10 minutes, the latter is scheduled once every day at a given time, for our purposes configured at 12:00 AM.

Forecast component executes first the forecasting of the consumptions for the day ahead, then the computation of the energy to be kept in the batteries at midnight, and finally constructs the problem to compute the forecasted curve of power for the day ahead. The plugins are used to define the constraints of the optimization problem, while the solver is called to solve the constructed one. The result is nothing but the curve to be declared, that is stored in the database as well as the trend of the SOC for the batteries, computed knowing the forecasted consumptions and approximating the interval over time to know
both total produced and consumed energy at every time of the day. Core component is responsible for maintaining, if possible, the hourly amount of energy declared during forecast. The actual SOC of the batteries, kept up to date by the SCADA, is read from the database, as well as the declared curve, and the curve maintaining problem is applied. Storing the new result in the database let the SCADA update the value of the set point in order to make the changes effective.

The Batteries class represents the SoNick batteries, and is used by the Forecast and Core classes to know what are the properties of the batteries, for example the input and output power.

The Solver uses the lp_solve library to solve the problem constructed each time, and returns the result that consists on a series of values of power which is exactly the forecasted output from the plant.

Finally the PluginManager does the following tasks: checks the physically available plugins, checks the enabled plugins (from the database), keeps the informations of the database coherent, and furnishes the informations to the Forecast and Core classes on which plugins to use.

More details on the algorithms and the operational problems used are described in the next sections.

Some assumptions have been done for simplicity:

- batteries are managed as if they are an only pack
- only nominal power, maximum input and output are considered
- no DSO signals are supported
• no cost per cycle is taken into account
• shutting the production plant off is not considered

Finally, the regulation is made on the 2nd regulation point, which is in general not a good solution because the amount of power putted into the grid is influenced by the amount of consumptions.

4.2 Configuration file

It is a Java properties file containing the following values:

• address, port, username and password for the connection to the DBMS
• name of the database to refer to
• names of the tables of the database
• hour on which the day-ahead forecasting have to be done
• minutes for the interval of the maintaining algorithm schedulation
• days to use for the consumption forecast (see the appropriate section)
• number of packs and batteries of the plant

4.3 Database structure

As already said, the database contains all the data of the stateless application. This allows to prevent data loss in case of system failure since all the data is stored into the PostgreSQL database via atomic transactions.

Only 6 small tables are needed:

• fotovoltaico: contains forecasted and real data about the production of power by the solar plant
• consumi: used to read and store forecasted and real consumptions
• borsaenergia: contains prices for selling and buying energy
• socbatterie: used to save real and forecasted values of state of charge
• setpoint: contains values used for the regulation point
• plugin: used to manage available and enabled plugins

All the ids are in the form "yyyyMMdd_HHmm", which allows to have an easy and uniform way of accessing values by date and time. The column forzato indicates when the value is forced by the administrator: in fact production, consumptions and setpoint can be forced. When setted to True, the related value cannot be overwritten by the program that uses it to force the algorithm.

\[\text{Database Management System}\]
\[\text{http://www.postgresql.org/}\]
to take that proper value into account instead of a forecasted one. Column 
reale contains real-time data kept up to date by the SCADA, and used by 
the algorithm for various tasks such as consumptions forecasting and actual 
SOC reading. Finally the setpoint table contains the declared curve of power 
(column dichiarato) which is kept fixed once the values are sent to the DSO, 
and potenzaimpianto is nothing but the real amount of power outgoing to the 
grid, which value can differ from the setted one (forced by the user or declared 
by the algorithm).

4.4 Consumption forecast algorithm

The implemented algorithm basically relies on the following assumptions

- past year during the same week the consumptions were similar to the 
  actual ones
- on the period prior to the actual one, consumptions were similar to actual 
  ones

A given number of days is used to select a window centered on past year’s date 
corresponding to tomorrow, and another window that ends exactly on tomorrow. 
On the selected weeks, for every time slot and every day equal to the one of 
tomorrow (example: for every Monday, if tomorrow is Monday), the average of 
the selected values is computed and saved on the database as forecasted value. 
Changing the number of days for the window size can make the algorithm adapt 
slower (higher number of days) or faster (fewer days) to eventual changes in the 
typical consumption curve of the plant for the given day. For example on a 
plant in which the consumptions does not vary much a larger window should be 
used (higher accuracy on the forecasting), while smaller window could be used 
in case of seasonal work, increasing accuracy on the short time periods.
4.5 Midnight energy calculation

Using historical data and forecastings, it is computed the amount of energy to keep into the batteries at the midnight, since forecasting is made on a daily basis and ends exactly at 12 PM.

The algorithm is very simple:

- first consumptions forecasting for the day after tomorrow is done, by the algorithm described above
- power production forecasting for the day after tomorrow is needed: it is computed with an algorithm very similar to the one used for consumption forecasting
- the difference between the total consumed and the total produced energy is made, for every time slot
- the highest negative difference is selected to compute the percentage of SOC needed to cover, if possible, that lack of energy
- this value is finally passed to the algorithm that adds it to the constraints of the problem, if the appropriate plugin is enabled (see next section).

4.6 Curve forecasting algorithm

The problem has been defined as a linear programming problem.

Since the goal is to maximize the profit by selling electric energy, the objective function is the overall gain expressed in terms of sum of the single gains.

We define:

- A as the array of bought energy
- B as the array of the quantities of incoming (positive values) and outgoing (negative values) energy for the batteries
- C as the array of forecasted consumption
- G as the array of the market (selling) prices for the electric energy
- L as the array of the buying market prices
- P as the array of forecasted production
- V as the array of sold energy

The number of values present in every array is the same. This number is defined as the number of samplings necessary to cover 24 hours, given the number of minutes which defines the interval that elapses between two consecutive updates of the system’s values.

In particular G and L do not change as the time passes, but the other arrays are kept up to date constantly since the values where forecasted earlier in time.

The values of C, G, L and P are given to the problem, so A, B and V are the variables. As already said, values of B > 0 indicates that the batteries are increasing their charge (energy is incoming into the batteries), values of B < 0
indicates that batteries are discharging. Values of all the other arrays are always ≥ 0.

The problem of maximizing the profit is so expressed as

$$\max \sum_{i=1}^{n} (0 \cdot b_i + g_i \cdot v_i - l_i \cdot a_i)$$

The constraint which the objective function is subject to are defined by the single plugins, so here is the list of the constraints implemented by every plugin.

### 4.6.1 Energetic balance

This first constraint is fundamental for the problem because defines the relation between all the variables of the problem and the given values

$$a_i - b_i - v_i = c_i - p_i \quad \forall i$$

### 4.6.2 Batteries constraints

First kind of constraints defined for batteries are upper and lower bound for the instantaneous value of electric current outcoming and incoming from the batteries, due to the specifications of the batteries and of the inverters.

$$b_i \leq \#\text{packs} \cdot \#\text{battsPerPack} \cdot \text{battMaxChrg} \quad \forall i$$

$$b_i \geq - (\#\text{packs} \cdot \#\text{battsPerPack} \cdot \text{battMaxDschrg}) \quad \forall i$$

There are other constraints that indicates the minumum and the maximum percentage of charge that has to be stored into the batteries in every moment of the day.

$$b_{iniz} + \frac{\text{mm}}{60} \sum_{j=1}^{i} b_j \geq 20\% \cdot \#\text{packs} \cdot \#\text{battsPerPack} \cdot \text{battNomEnrg} \quad \forall i$$

$$b_{iniz} + \frac{\text{mm}}{60} \sum_{j=1}^{i} b_j \leq 80\% \cdot \#\text{packs} \cdot \#\text{battsPerPack} \cdot \text{battNomEnrg} \quad \forall i$$

### 4.6.3 Production constraints

Since we suppose not to buy energy in order to store it into batteries, the value of energy incoming in the batteries cannot be higher than the produced quantity

$$b_i \leq p_i \quad \forall i$$

### 4.6.4 Midnight constraints

Usually during the first hours of the day there is no photovoltaic production since the sun has not risen yet. It is evident that saving part of the energy produced can be useful to cover the consumption that occurs during this part of the day. Notice that the percentage of charge into the batteries cannot exceed the specified maximum value, so it is limited just by this value.

$$b_{iniz} + \frac{\text{mm}}{60} \sum_{j=1}^{i} b_j \geq \min(\text{minEnrg} + \text{midEnrg}, \text{maxEnrg}) \quad \forall i$$

where

$$\text{minEnrg} = 20\% \cdot \#\text{packs} \cdot \#\text{battsPerPack} \cdot \text{battMaxEnrg}$$

$$\text{maxEnrg} = 80\% \cdot \#\text{packs} \cdot \#\text{battsPerPack} \cdot \text{battMaxEnrg}$$

and midEnrg is the value computed by the midnight energy calculation algorithm.
4.6.5 The LP problem

Writing down the entire LP problem, we obtain

\[
\begin{align*}
\text{min} & \quad 0^T B + G^T V - L^T A \\
\text{s.t.} & \quad V \geq 0 \\
& \quad A \geq 0 \\
& \quad A - B - V = C - P \\
& \quad B \leq 1 \cdot \#\text{packs} \cdot \#\text{battsPerPack} \cdot \text{battMaxChrg} \\
& \quad B \geq 1 \cdot -(\#\text{packs} \cdot \#\text{battsPerPack} \cdot \text{battMaxDschrg}) \\
& \quad 1^T B \geq 1 \cdot \min\left(\text{minEnrg} - b_{inz}\right) \\
& \quad 1^T B \leq 1 \cdot \min\left(\text{maxEnrg} - b_{inz}\right) \\
& \quad B \leq P \\
& \quad 1^T B \geq 1 \cdot \min\left(\text{minEnrg} + \text{midEnrg}, \text{maxEnrg}\right) - b_{inz}) \\
\end{align*}
\]

4.7 Curve maintaining algorithm

The objective of this algorithm is to keep, if possible, the value of sold energy in a range of ±10% with respect to the declared value, in a hourly basis. Constantly monitorizing the state allows the application to recompute the power curve, if necessary. Real power curve is used to compute the amount of energy already sold in the actual hour, while the forecasted curve of power allows to compute the remaining part of energy to be sold. If the sum of these two values differs too much from the expected one, a revision of the power curve for the next part of the hour has to be done.

In particular the difference of energy is computed as

\[ \text{diff} = \text{decl} - (\text{sold} + \text{forecasted}) \]

and the gap to be covered is nothing but that value plus (if \( \text{diff} < 0 \)) or minus (if \( \text{diff} > 0 \)) 10\% \cdot \text{decl}.

If the difference is positive, the new set point has to be higher than previously forecasted one, otherwise it will be lower than forecasted one. Once the computation of the new set point is made, also the forecasted value of state of charge of the batteries is made, and if it exceeds the minimum and maximum percentages (namely, 20\% and 80\%), a possible failure in maintaining the declaration is accounted.

Finally for the previous hours is always made a check on the meet of the goal of maintaining the real energy in a range not far from the declared one of more that ±10%.

4.7.1 The LP problems

Writing down the LP problem for a positive value of \( \text{diff} \), we obtain

\[
\begin{align*}
\text{max} & \quad 1^T X \\
\text{s.t.} & \quad 1^T(X - X_{prev}) = \frac{60}{mm} \cdot \text{diff} \\
& \quad X \geq X_{prev} \\
\end{align*}
\]

While for a negative value of \( \text{diff} \) we have

\[
\begin{align*}
\text{max} & \quad 1^T X \\
\text{s.t.} & \quad 1^T(X - X_{prev}) = \frac{60}{mm} \cdot \text{diff} \\
& \quad X \leq X_{prev} \\
\end{align*}
\]
4.8 Libraries used

Since the application is written in Java, there is the need to use some external libraries to implement special purpose components, such as a charting class that allows to easily control if the result is always the expected one.

**JFreeChart**
Free 100% Java chart library that makes it easy to display professional quality charts. It is distributed under the terms of the GNU Lesser General Public Licence (LGPL), which permits use in proprietary applications.

http://www.jfree.org/jfreechart/

**Java Plug-in Framework (JPF)**
JPF provides a runtime engine that dynamically discovers and loads "plugins". A plug-in is a structured component that describes itself to JPF using a "manifest". It is released under LGPL license.

http://jpf.sourceforge.net/

**lp_solve**
lp_solve is a Mixed Integer Linear Programming (MILP) solver, or more specifically, lp_solve is a free linear (integer) programming solver based on the revised simplex method and the Branch-and-bound method for the integers. Solves pure linear, (mixed) integer/binary, semi-continuous and special ordered sets (SOS) models. Via the Branch-and-bound algorithm, it can handle integer variables, semi-continuous variables and Special Ordered Sets. Finally has no limit on model size. It is released under LGPL (GNU lesser general public) license.

http://lpsolve.sourceforge.net/5.5/

**PostgreSQL JDBC Driver**
Allows Java programs to connect to a PostgreSQL database using standard, database independent Java code. It is a pure Java (Type IV) implementation. It is distributed under the BSD license, same as the server. The simplest explanation of the licensing terms is that you can do whatever you want with the product and source code as long as you don’t claim you wrote it or sue the developers of the driver.

http://jdbc.postgresql.org/

4.9 Results

Everytime an algorithm or an application is developed, the interesting part is to check if the obtained results reflects the requirements and the expectations that leaded to that project. This section presents some results obtained with the algorithm explained in this chapter and shows that the expected results are met.

The data used for these examples consists of real data for both the forecasted production curve and the market prices, while the consumption curve has been created on purpose.
The Figure 4.3 shows the set point obtained for a random day, in which it is notable that the consumptions are always “covered” by the forecasted curve of power, meaning that this first goal is met. Furthermore more energy is sold in the moments in which the price is higher, related also to the availability of the energy itself, so to maximize the money earned, meeting the second main goal. The state of charge of the batteries is always kept between the 20% and the 80% which are the boundaries, as described previously. In this example the initial state of charge of the batteries is supposed to be 65% and the value of energy to keep in the batteries at midnight has been setted to the minimum SOC possible which is 20%.

Figure 4.4 can be used for a comparison with the previous one since the consumption and production forecasted curves are the same, but the market prices are changed. The difference in the set point is due to the change of the hours in which it is more convenient to store the energy in the batteries rather than selling it, and viceversa.

For the declaration of the curve the results are correct, but let’s show the correctness also of the maintaining algorithm using the time slot between the 15:00 and the 16:00, and supposing to be at about 15:30. Figure 4.5 shows the forecasted situation in which the forecasted set point represents also the declared one. In the case in which during the first part of the hour the set point has been lower than expected, it has to be corrected (if possible) for the second part of the hour. See Figure 4.6. The opposite case is represented in Figure 4.7 in which the set point is lowered in the second part due to a overselling of energy during the time slots in the first part of the hour. This second case presents a problem: the batteries cannot be charged to absorb all the surplus of energy produced. The algorithm recomputes the correct set point, leaves the forecasted state of charge unchanged, and notifies the situation to the administrator writing it both in the standard output (system console) and in the log file. In the charts only the new forecasted set point is shown, while the real power outgoing from the plant in the first part of the hour is omitted in order to obtain clearer charts.

The importance of the charts is in fact to show that the expected results are obtained for the given algorithm implementing only part of the total amount of requirements present in Chapter 3.

4.10 Conclusions

This prototype application has been useful because let us learn the aspects of the problem in a more detailed way, so that the development of the optimization algorithm for the final application started from the optimization problem defined in this prototype. Moreover this in fact already working application was useful to test the communication with the SCADA system, and let us solve bugs before testing and deploying the final application. Finally the results obtained let us see the feasibility of the application, even if they do not cover yet all the requirements and the constraints that will be implemented in the final algorithm.

The next chapter describes the application used by the system to download the market data from the GME website. The same application is used also for the final application that will be implemented using the notions explained in the following chapters.
Figure 4.3: Example result for a random day
Figure 4.4: Example result for a random day with changed market prices
Figure 4.5: Forecasted situation for a random day between 15:00 and 16:00

Figure 4.6: Correction from 15:30 in case of highered set point
Figure 4.7: Correction from 15:30 in case of lowered set point
Chapter 5

Market prices application

A special purpose separated application has been developed with the goal of downloading the market prices from the GME website and making them available on the database, on which the application can read and make use of them.

This application is used by the prototype described in the previous chapter, and also by the final application described in Chapter 10.

Differently from the prototype and the final application, whom are written in Java, this application is written in Python using mechanize \(^1\) and ZipFile modules. When the program is launched by the SCADA, it does the following steps:

1. creation of a browser and access to the GME website
2. submission of the form putting the date of tomorrow in the proper fields
3. download of the zip file and extraction of the contained xml file
4. processing of the xml file in order to obtain the values in €/kWh instead of €/MWh
5. saving of the data into a formatted text file which data will be loaded into the database to make it available to the algorithm

This is one of the fundamental parts since the price of the energy is at the base of the algorithm. The fact that it is separated from the program containing the algorithm itself makes it independent. Furthermore the data is loaded into a proper database table so that the independency of the algorithm is total because the way data is loaded into the database is not important. Finally if there is any change in the website, in the format the data is furnished, or in the market itself (depending on the region and state on which the plant is installed), this does not change the way the algorithm reads the data from the table.

Figure 5.1 shows the iter used from the system to load the data into the database.

\(^1\)http://wwwsearch.sourceforge.net/mechanize/
Figure 5.1: Loading of market data into the database
Chapter 6

Consumption

Market prices are one of the fundamental values the final algorithm needs to take into account. They are defined by the GME and downloaded using the application described in Chapter 5. One of the other fundamental variables are the consumption values. The algorithm needs to know in advance the consumptions, forecasting the curve of power for the day ahead. The forecasting can be done since the real consumptions are stored in the database for each time-slot of the day, and this historical data is used by the forecasting algorithm in the way described in this chapter.

The consumption forecasting algorithm is based on simple assumptions

- on the same day of the weeks, the consumptions are similar
- on the days preceding the one for which the forecasting has to be made, the consumptions were similar
- on the past year, in the same period, the consumptions were similar to the ones in this period

So that the parameters that can be used to tune the algorithm are

- window size for this year’s period
- window size for the preceding period in the past years
- window size for the succeeding period in the past years
- number of past years to consider

On the selected weeks, for every time slot and every day equal to the one of tomorrow (example: for every Wednesday, if tomorrow is Wednesday), the average of the selected values is computed and saved on the database as forecasted value.

Changing the number of days for the window size can make the algorithm adapt slower (higher number of days) or faster (fewer days) to eventual changes in the typical consumption curve of the plant for the given day. For example on a plant in which the consumptions does not vary much a larger window should be used (higher accuracy on the forecasting), while smaller window could be used in case of seasonal work, increasing accuracy on the short time periods.
Figure 6.1: Consumption algorithm schema

Figure 6.1 shows briefly on a schema the periods selected by the algorithm, and the days selected in those ranges of days.

Pseudocode of the consumption forecasting algorithm is defined in Algorithm 1, which is nothing but the computation of the average values, for the selected days and for each time slot. \( \text{prev}[t] \) identifies the forecasting for a given time-slot \( t \), \( d[t] \) is the real consumption value for the day \( d \) at the time-slot \( t \), and \( |d| \) is the number of days selected.

**Algorithm 1** Consumption algorithm pseudocode

```
for all time slot t do
    prev[t] ← 0
    for all selected day d do
        prev[t] ← prev[t] + d[t]
    end for
    prev[t] ← prev[t]/|d|
end for
```

6.1 Festivals

As stated in Chapter 3, a semi-automatic mechanism to recognize festivals is implemented since these events usually influence consumptions a lot.

The mechanism implemented is very easy since it is similar to the one used for the non-festival days.

In the database there is a list that contains for every festival, identified with its own name, and for every year, the exact date it occurs. Given this list it is easy to check if the day for which the forecasting has to be made is a festival day or not, simply checking if there is an event which date is the given one. Once discovered the date is a festival, the same festival’s dates are selected from the past years. On all the past year’s dates, for every time slot, a ratio of the
value of the festival day and the other equivalent days (for example all the other Thursdays) is computed. If the data is present and the ratios can be computed, that ratio will be kept, otherwise a default ratio will be used. Finally, given the ratio, the forecasting is made using the ratio to multiply the average value of the equivalent days in the period preceding the actual day.

A sketch of the algorithm is presented, from which it is easy to notice that Algorithm 1 is a particular case of Algorithm 2. As in the former one, prev[t] identifies the forecasting for a given time-slot t, d[t] is the real consumption value for the day d at the time-slot t, and |d| is the number of days selected. ratio is the array of the ratio values, count is the array of the days counter and average is the array with the average values of this year’s equivalent days consumptions. Finally festival[y][t] identifies the real consumption value for the given festival in the year y at time slot t.

Algorithm 2 Festivals consumption algorithm pseudocode

```plaintext
for all time slot t do
    ratio[t] ← 0
    count[t] ← 0
    average[t] ← 0
end for

for all previous year y do
    for all selected day d do
        for all time slot t do
            ratio[t] ← ratio[t] + festival[y][t] / d[t]
            count[t] ← count[t] + 1
        end for
    end for
end for

for all selected day d do
    for all time slot t do
        average[t] ← average[t] + d[t]
    end for
end for

for all time slot t do
    prev[t] ← average[t] / |d|
    if count[t] > 0 then
        prev[t] ← ratio[t] · prev[t]
    end if
end for
```

The algorithm for the consumption forecasting is based on simple assumptions and even how it works is very simple. The prototype implements an algorithm very similar, but does not contain the management of the festivals that will be implemented in the final application.
Chapter 7

Profiles

Chapter 6 shows how the consumption curve forecasting is made, based on historical data stored in the database by the SCADA. The application will use those data to know in advance the values to use in the declaration and maintaining problems, explained in the next chapters. This chapter shows the various ways the algorithm can be tuned, in order to meet the needs of the administrator.

Various so called profiles are defined for the algorithm, and every profile aims to a different objective. Profiles used for declaration of power curve are usually different for the maintaining ones, since the goal is different: the former ones in general define an optimal curve, the latter ones aim to keep the real values as close as possible to the forecasted ones.

Since the D.M. 06-08-2010 requires an hourly program (in terms of energy), from 8 AM to 8 PM, for at least 300 days in a year with a margin of error equal to 10%, then the profiles called "profilation" will aim to this goal.

Another profile have to be defined for managing the situation between 8 PM and 8 AM in order for the batteries to reach a SOC in which the profilation have more probability to be kept.

Other profiles can be used to gain the maximum possible amount of money, selling energy when it is more cost effective, or to save the batteries, doing as few cycles as possible with the batteries.

The signal of the DSO have to be accounted but in the next chapters will be explained how they are treated, so no particular profiles are used for this purpose because of the unknown value and timing of them.

Finally on every time slot a different profile can be selected in order to meet the requirements specified by the plant administrator.

The profiles defined are the following ones, but every different combination of values for the coefficients of the functions defined in the next chapters allows to obtain different objectives, so that an "infinite" number of profiles could be theoretically defined.

- Profilation: when forecasting is used to define a curve of power which is theoretically equal to the difference between the produced (forecasted) and the consumed (forecasted, too) power. This should be the main purpose of the plant: use the batteries only to cover the power holes, and sell all the unconsumed energy. When in maintaining, is used to keep the hourly
amount of energy in the range of ±10% with respect to the forecasted (and declared) values.

- Max gain: defines the curve of power that allows to earn as much money as possible selling the energy mostly during the hours in which the selling price is higher.

- Battery save: aims to do as few battery cycles as possible in order to make the life of the batterie longer.

- Optimal SOC: is used to let the batteries reach a state of charge which is good for letting the profilation be effective the following hours. It takes into account the variance of the forecasting (for both production and consumption), and uses it to know what is the percentage of the batteries for which the profilation can be mantained, covering overproduction or power holes.

### 7.1 Usage of profiles

Despite the fact profiles can be chosen at every time by plant administrator ("forcing" the profile itself for some time slots), an automatic system of chosing proper profiles for every time slot is defined.

In particular the typical situation that can be used is the following:

- Profillation: every day from 8 AM to 8 PM, both for declaration and mantaining (green color)

- Optimal SOC: every day from 8 PM to 8 AM, both for declaration and mantaining (dark blue color), and also for mantaining in case of profilation failure (fallback)

- Max gain: every day from 0 AM to 12 PM, both for declaration and mantaining, at the end of the year after the 300 years of mantained profilation has been reached (blue color)

Figure 7.1 shows how profiles are used in the different parts of the year and of the day.

![Figure 7.1: Periods for the profiles](image)

In this chapter the ways of functioning (the possible objectives) of the algorithm have been listed. The typical schedulation of them is also explained,
even if it can be changed by the system administrator basing on the needs. As described in Chapter 10, there will be a profile manager in the final application that will manage the profiles the way it has been described here. Next chapters will contain the definitions of the declaration (forecasting) and maintaining problems, as well as how the profiles are implemented in there.
Chapter 8

Declaration

The forecasting and declaration process is very important because it defines the conditions that the plant and the algorithm itself will have to face with during the following hours. It consists on forecasting the set point curve (power outgoing in the smart grid) and the state of charge of the batteries, given some fundamental parameters such as

- the forecasting of production power curve for the entire period considered
- the forecasting of consumption power curve for the entire period considered
- the forecasted state of charge of the batteries at the starting of the forecasting period
- the market prices for both selling and buying energy during the period for which the forecasting is made

All the above listed variables, as well as other parameters that will be described, will contribute for the computation of the power curve. Every parameter influences the algorithm in a way that depends on the type of profile chosen in every time slot. For example the market prices are very important if the algorithm is setted to earn money selling energy but do not count if the algorithm is setted to save the life of the batteries.

The declaration is a task that has implied the forecasting, since the former consists in declaring the power curve for period between the 8 AM and the 8 PM of the day ahead, every day at a given hour, in a way to respect the D.M. 06-08-2010. In order to obtain a more precise declaration and a higher probability of being able to respect it the next day, the forecasting is made for the entire day ahead and for the following 8 hours, for a total of 32 hours. All the above listed parameters need to be computed before the power curve forecasting process, for the entire period considered.

In Section 8.1 the optimization problem for the forecasting is explained, in Section 8.2 the resolution method for the given problem is shown and Section 8.3 shows how the problem’s coefficients are used to tune it for the various profiles.
8.1 Optimization problem

The optimization problem is an operational research problem consisting of an objective function which value has to be minimized (or maximized) with respect to the given parameters and subject to the given constraints. In this case the objective function has to be maximized since it represents a gain in terms of money.

The objective function is a composed function that is the summation of

- the money to earn selling the energy: every kW sent out from the plant has a certain value
- the money to spend buying the energy: every kW bought by the plant has a certain value
- the cost of using the batteries: batteries have a known cost and maximum number of cycles they can do
- the cost of the energy loss due to inefficiency: power (energy) dissipated by batteries or inverters is a loss of money
- the cost of the energy used to keep the batteries at correct temperature: batteries are provided with resistors that keeps the temperature high when batteries are not used much
- the gain on reaching an optimal state of charge: if the state of charge of the batteries is in a given range, the probability of being able to satisfy the D.M. 06-08-2010 will be higher. The range depends on factors like the variance computed for the forecasted production and consumption

The resulting objective function is

\[
\max \sum_{j=1}^{n} g(j)v_j - \sum_{j=1}^{n} c(j)a_j
\]

\[
- \sum_{i=1}^{k} \sum_{j=1}^{n} \text{batt}(i,j) - \sum_{i=1}^{k} \sum_{j=1}^{n} \text{cost1}(i,j) \cdot (1 - \text{eff}(i,j))
\]

\[
- \sum_{i=1}^{k} \sum_{j=1}^{n} \text{disp}(i,j)
\]

\[
- \sum_{j=1}^{n} \text{cost3}(j) \cdot \text{strategy}(j, \text{end}, \text{idealEnergy})
\]

Subject to the following constraints

Energy balance: all the produced or bought power has to go into batteries, be consumed or be sold.

\[
p_j + a_j = c_j + v_j + \sum_{i=1}^{k} \text{eff}(i,j) \cdot Bi_j \quad \forall j = 1 \ldots n
\]

Energy bought or sold is always a positive amount.

\[
v_j \geq 0 \quad \forall j = 1 \ldots n
\]

\[
a_j \geq 0 \quad \forall j = 1 \ldots n
\]

The batteries have power bounds both in charge and discharge mode.

\[
Bi_j \leq \text{maxChargePower}(i,j) \quad \forall j = 1 \ldots n, i = 1 \ldots k
\]

\[
Bi_j \geq -\text{maxDischargePower}(i,j) \quad \forall j = 1 \ldots n, i = 1 \ldots k
\]

The batteries have state of charge bounds, as seen in Chapter 3.

\[
b_0(i) + \sum_{j=1}^{n} \text{eff}(i,j) \cdot Bi_j \geq 20%\text{nominalEnergy}(i) \quad \forall i = 1 \ldots k, p = 1 \ldots n
\]
\[ b_0(i) + \frac{mn}{60} \sum_{j=1}^{p} \text{eff}(i,j) \cdot B_{ij} \leq 80\% \text{nominalEnergy}(i) \ \forall i = 1 \ldots k, p = 1 \ldots n \]

This basic problem lets the algorithm choose the profile just changing the values of the coefficients. We will see later in this chapter how they are implemented.

Some particular cases have to be considered:

- if the i-th pack of batteries is in EOC at time-slot j, then the second constraint for batteries power has to be changed with
  \[ B_{ij} = \text{EOChargePower}(i) \ \forall j = 1 \ldots n, i = 1 \ldots k \]
- if the set point is forced (by the administrator) for some time slots j, then \( v_j \) is substituted with the given value instead of being a variable to be computed.

The definitions used are the following

- \( k \) = number of battery packs
- \( n \) = number of time slots
- \( mn \) = number of minutes of every time slot
- \( g(j) \) = energy selling price at j-th time slot
- \( v_j \) = power to put out at j-th time slot
- \( c(j) \) = energy buying price at j-th time slot
- \( a_j \) = power to let in at j-th time slot
- \( \text{batt}(i,j) = \text{cost of using } j\text{-th battery pack at } i\text{-th time slot} \)
  \[ = 0.25 \cdot \text{eff}(i,j) \cdot |B_{ij}| + \text{cycleCost}(i) \cdot f(i,j) \]
  Note that the number 0.25 is due to the fact that batteries are known to cost about 250 Euros/MegaWatt, which is equal to 0.25 Euro/kiloWatt
- \( f(i,j) \) = \[ \begin{cases} 1 & \text{if sign}(i,j-1) \neq \text{sign}(i,j) \\ 0 & \text{otherwise} \end{cases} \]
- \( \text{cycleCost}(i) \) = cost of a single cycle of the batteries, is computed easily given the cost of the batteries in the i-th pack and the maximum number of cycles they can do
- \( \text{sign}(i,j) = \begin{cases} 1 & \text{if } B_{ij} > 0 \\ 0 & \text{if } B_{ij} = 0 \\ -1 & \text{if } B_{ij} < 0 \end{cases} \)
- \( B_{ij} \) = power to put in (if \( > 0 \)) or to put out (if \( < 0 \)) from i-th battery pack at j-th time slot
- \( \text{cost1}(i,j) = \text{cost due to inefficiency in } i\text{-th battery pack (and inverter) at } j\text{-th time slot} \)
• \( \text{eff}(i,j) = \text{efficiency of batteries and inverters} = \text{effBatt}(i,j) \cdot \text{effInv}(i,j) \)

• \( \text{effBatt}(i,j) = \begin{cases} 
\approx \sqrt{0.9} & \text{if } B_{ij} > 0 \text{ (charging)} \\
1 & \text{if } B_{ij} = 0 \\
\approx \sqrt{0.9} & \text{if } B_{ij} < 0 \text{ (discharging)} 
\end{cases} \)

• \( \text{effInv}(i,j) = \begin{cases} 
1 & \text{if } B_{ij} = 0 \\
0.95 & \text{if } |B_{ij}| \geq 10\% \text{convMaxLoad}(i) \\
0.95 \cdot \frac{|B_{ij}|}{\text{invMaxLoad}(i)} & \text{if } |B_{ij}| < 10\% \text{convMaxLoad}(i) 
\end{cases} \)

• \( \text{invMaxLoad}(i) = \text{maximum power load of the i-th inverter} (\text{associated to the i-th battery pack}) \)

• \( \text{cost}2(i,j) = \text{cost due to dispersion and self-consumption in i-th battery pack at j-th time slot} \)

• \( \text{disp}(i,j) = \text{dispersion and self-consumption in i-th battery pack at j-th time slot} = \begin{cases} 
\frac{1}{100}|B_{ij}| + 0.13 \cdot \text{battNum}(i) & \text{if } 5\%|B_{ij}| \leq 0.13 \cdot \text{battNum}(i) \\
0 & \text{otherwise} 
\end{cases} \)

• \( \text{battNum}(i) = \text{number of batteries in the i-th pack} \)

• \( \text{cost}3(j) = \text{coefficient for the cost on not meeting the required state of charge of the batteries at the j-th time slot} \)

• \( \text{strategy}(j, \text{end}, \text{idealEnergy}) = \text{cost on not meeting the required state of charge of the batteries at the j-th time slot} = \begin{cases} 
0 & \text{if } j \neq \text{end} \\
|\sum_{i=1}^{k}(b_{0}(i) + \frac{mm}{} \sum_{l=1}^{j} \text{eff}(i, l) \cdot B_{il}) - \text{idealEnergy}| & \text{otherwise} 
\end{cases} \)

• \( \text{idealEnergy} = \text{energy that should be in the batteries at the given time slot} = \sum_{i=1}^{n} \text{nominalEnergy}(i) - \text{prodVarPos} + \text{prodVarNeg} + \text{consVarPos} - \text{consVarNeg} \)

• \( \text{prodVarPos} = \text{positive variance of production} (\text{value } \geq 0) \text{ computed on forecasting} \)

• \( \text{prodVarNeg} = \text{negative variance of production} (\text{value } \geq 0) \text{ computed on forecasting} \)

• \( \text{consVarPos} = \text{positive variance of consumption} (\text{value } \geq 0) \text{ computed on forecasting} \)

• \( \text{consVarNeg} = \text{negative variance of consumption} (\text{value } \geq 0) \text{ computed on forecasting} \)

• \( p_{ij} = \text{forecasted production at time-slot } j \)

• \( c_{ij} = \text{forecasted consumption at time-slot } j \)

• \( \text{nominalEnergy}(i) = \text{nominal energy of i-th battery pack}, \text{can be easily computed as } \text{battNum}(i) \text{ times the nominal energy of a single battery} \)
EOC\text{chargePower}(i) = \text{value of charging power to be used for i-th battery pack in case of end of charge}

b_0(i) = \text{forecasted amount of energy present in the i-th batteries pack before the first time slot}

An assumption made when defining the problem is that an heuristic strategy can be used to manage the batteries. Even if heuristics give not the best results in general, in this case it can be very close to the optimal one. The assumption consists on supposing that only a single battery pack can be sent to the end of charge state at a time. Together with this, the other assumption is that it is always convenient to send batteries in EOC during the night. The algorithm will so find all the packs available during the day and deal with the profiling more efficiently. Finally when the average state of charge has to reached at the end of a period, the pack sent in EOC is not counted for the average percentage meeting the ideal energy value.

The state kept for the problem consists on a vector that contains all the following elements

- \( v = \{ v_j \mid j = 1 \ldots n \} = \text{sold power curve} \)
- \( a = \{ a_j \mid j = 1 \ldots n \} = \text{bought power curve} \)
- \( B = \{ B_{ij} \mid j = 1 \ldots n, i = 1 \ldots k \} = \text{energy present into the battery packs} \)

The solution is nothing but the combination of values for the state that returns the best result for the objective function subject to the given constraints.

The resolution method used to obtain the result is explained in the next section.

### 8.2 Resolution

All the parts of the objective function, except the sign function, are linear or linearizable. Also the constraints are all linear. The method proposed to solve the problem is not difficult to understand.

First it is assumed that the problem is in the standard form: all the functions are in the standard functioning way, meaning that the standard value is setted on every function. The cost per cycle of the batteries is then left apart, taking into account only the remaining part of the problem. The obtained problem and constraints are linear, so that with one or more cycles of the symplex method the best solution is found. Once obtained this partial solution, all the functions are binded to the correct value with respect to the obtained result, and the solution is then recomputed. It is still a partial solution, so there is the need to take into account the cost of the cycles of the batteries. By default the cost is setted to zero, then a fixed-point iteration method is used to improve the solution and find the local best solution resetting at each iteration the points in which the cycles happen and recomputing the function value.

The solution obtained is not the global best solution because it is an heuristic method that allows to obtain the solution easily.

The book used as a reference for the resolution methods is "Linear and Nonlinear Programming" by David G. Luenberger [6].
8.3 Implementation of profiles

Every profile defines a different way for the algorithm to work. The formulation of the optimization problem allows to switch between the profiles at each time slot simply changing the coefficients of the functions. Here is a list of the profiles and how the coefficients have to be setted to implement that specific one. For every profile is then reported an example of power curve resulting from the forecasting of the algorithm in that configuration, so that a visual comparison can be made.

- **Profilation**: the objective is to not use batteries so that the set point curve corresponds to the production at which the consumption is subtracted. In this profile $g(j)$ and $c(j)$ represent the real values, $\text{batt}(i,j)$ returns an higher value than the defined one, while $\text{cost1}(i,j)$, $\text{cost2}(i,j)$ and $\text{cost3}(j)$ are all zeros. Otherwise it can be obtained by forcing $B_{ij} = 0$, setting $v_j = p - c$ if $p \geq c$ and $a_j = c - p$ if $c > p$.

- **Max gain**: $g(j)$ and $c(j)$ are unchanged as well as $\text{batt}(i,j)$, $\text{cost1}(i,j)$ and $\text{cost2}(i,j)$, while $\text{cost3}(j)$ is zero. This is the way the objective function computes the maximum real gain in terms of money.

- **Battery save**: $g(j)$ is lowered while $c(j)$ is unchanged as well as $\text{cost1}(i,j)$ and $\text{cost2}(i,j)$. $\text{batt}(i,j)$ return an higher value, as for profilation, and $\text{cost3}(j)$ is zero.

- **Optimal SOC**: like Max gain, but $\text{cost3}(j)$ is a strictly positive value because the importance for the batteries to reach a given state of charge has to be accounted.

Some examples of theoretical results are presented in Figure 8.1, 8.2 and 8.3. The data that has been used is the same of Figure 4.4 in the prototype chapter. Examples are made on a 24 hours basis instead of 32, as it will be done on the final application, but the concepts applied are the same.

Figure 8.1 shows the difference between the Max Gain profile and the Optimal SOC one. Black line represents the set point for the Max Gain profile and red line is the SOC of the batteries (average between the two packs of the plant of the island). The difference with the power curve and SOC curve of the Optimal SOC profile can be seen on the right most part of the chart, at the final hours of the day, when the amount of energy ideal for finishing the day was setted. Less energy is sold, and also some energy is bought, in order to achieve a state of charge which is equal to 71%. In the Max Gain case no objective for state of charge is setted so at the end of the period the percentage can be lowered to 20%, the minimum allowed by the batteries.

Figure 8.2 shows the difference between the Max Gain profile and the Profilation one. Black line represents the set point for the Profilation profile. This curve is nothing but the difference between the production and the consumption, at every interval. When the difference is negative or zero, the set point curve is setted to zero because no energy can be sold. Even if the state of charge for Profilation profile is not represented, batteries can not satisfy the energy requirement during the hours of no production so that a certain amount of energy must be bought from the smart grid. The difference between the two set point
curves is very high because of the use of the batteries in the Max Gain profile and the disuse of them in the other profile.

Figure 8.3 finally shows the difference between the Optimal SOC profile and the Profilation one. The result obtained is actually very close to the one of Figure 8.2.

The visual representation of the major differences between profiles gives a better idea of what every profile should do. The way the plant works is defined by the profiles, chosen at each time by the profiles management (part of the application, already described) or by the system administrator.

8.4 Conclusions

The forecasting problem has been described in this chapter. The requirements analysis lead to the definition of the problem seen. All the variables used by the problem are listed. How every profile is implemented is described in the apposite section. The theoretical resolution method should then lead to a forecasted curve that represents the one that best fits with the given data, even if it is not the global but only local best solution. Only after the implementation the real results will be obtained so that the correct functioning of the problem will be checked. The next chapter will describe a similar problem used for the mantaining of the curve.
Figure 8.1: Max Gain and Optimal SOC profiles
Figure 8.2: Max Gain and Profilation profiles
Figure 8.3: Optimal SOC and Profilation profiles
Chapter 9
Maintaining

The maintaining process is as important as the forecasting and declaration one, seen in the previous chapter. This process aims to keep the forecasted situation as close as possible to the forecasted one, both in terms of sold energy and in terms of state of charge of the batteries. The objective is defined by the profile chosen, but usually the goal is not to exceed a given range for the hourly amount of energy sold. It is a "real-time" process in which the situation is constantly monitorized and the eventual corrections are reported to the system according to the profile chosen.

The fundamental parameters are the same used for the declaration process plus the ones listed below

- the real production power curve for some time-slots before the starting one
- the real consumption power curve for some time-slots before the starting one
- the real state of charge of the batteries at the starting of the considered period
- the declared power curve for the entire period considered

In Section 9.1 the optimization problem for the computation of the maintaining curve is explained, in Section 9.2 the resolution method for the given problem is shown and Section 9.3 shows how the problem’s coefficients are used to tune it for the various profiles. Section 9.4 explains how the signals by the DSO are managed.

9.1 Optimization problem

The resulting objective function is

$$\begin{align*}
\max & \sum_{j=1}^{n} g(j) \cdot v_j - \sum_{j=1}^{n} c(j) \cdot a_j \\
- & \sum_{i=1}^{k} \sum_{j=1}^{n} \text{batt}(i, j) - \frac{mm}{60} \cdot \sum_{i=1}^{k} \sum_{j=1}^{n} \text{cost1}(i, j) \cdot (1 - \text{eff}(i, j)) \\
- & \frac{mm}{60} \cdot \sum_{i=1}^{k} \sum_{j=1}^{n} \text{cost2}(i, j) \cdot \text{disp}(i, j)
\end{align*}$$
\[-\sum_{j=1}^{n} \text{cost3}(j) \cdot \text{strategy}(j, \text{end}, \text{idealEnergy}) \]
\[-\sum_{h=1}^{\lfloor n \cdot \text{mm}/60 \rfloor} \sum_{j=1}^{n} \text{cost4}(h) \cdot \text{profilation}(h) \]

Only the objective function has been reported since the only differences with the declaring problem is in there. In particular has been added a new function that allows to take into account the maintaining of the hourly amount of energy as far as it is concerned in the D.M. 06-08-2010.

The definitions not present in the previous chapter are the following:

- \(\lfloor n \cdot \text{mm}/60 \rfloor\) = number of hours considered, included the one in which the actual time-slot is in
- \(\text{cost4}(h)\) = cost on missing the profilation on a given hour
- \(\text{profilation}(h)\) = step function that tells if the curve is in the range or not
  \[= \begin{cases} 
  0 & \text{if } \text{real}(h) + \text{prev}(h) \geq \frac{90}{100} \cdot \text{decl}(h) \text{ and } \text{real}(h) + \text{prev}(h) \leq \frac{110}{100} \cdot \text{decl}(h) \\
  1 & \text{otherwise} 
  \end{cases} \]
- \(\text{real}(h)\) = energy putted out from the plant in the given hour
  \[= \begin{cases} 
  \frac{\text{mm}}{60} \sum_{j=\lfloor -\text{mm}/60 \rfloor + 1}^{0} \text{v}_{\text{real}}(j) & \text{if } h = 1 \\
  0 & \text{otherwise} 
  \end{cases} \]
- \(\text{prev}(h)\) = forecasted energy to be putted out from the plan in the remaining time-slots of the given hour
  \[= \begin{cases} 
  \frac{\text{mm}}{60} \sum_{j=1}^{\lfloor \text{mm}/60 \rfloor} \text{v}_{\text{prev}}(h-1) \frac{\text{mm}}{60} + j & \text{if } h = 1 \\
  \frac{\text{mm}}{60} \sum_{j=1}^{\lfloor \text{mm}/60 \rfloor} \text{v}_{\text{prev}}(h-1) \frac{\text{mm}}{60} + j - \frac{\text{mm}}{60} & \text{otherwise} 
  \end{cases} \]
- \(\text{decl}(h)\) = declared amount of energy to be putted out from the plant in the given hour
  \[= \begin{cases} 
  \frac{\text{mm}}{60} \sum_{j=1}^{\lfloor \text{mm}/60 \rfloor} \text{decl}(h-1) \frac{\text{mm}}{60} + j & \text{if } h = 1 \\
  \frac{\text{mm}}{60} \sum_{j=1}^{\lfloor \text{mm}/60 \rfloor} \text{decl}(h-1) \frac{\text{mm}}{60} + j - \frac{\text{mm}}{60} & \text{otherwise} 
  \end{cases} \]

All the considerations made in section 8.1 are valid also here. The unique situation that this algorithm cannot handle is when a signal arrives: this particular scenario is explained in section 9.4.

9.2 Resolution

The resolution process is the same that is explained in Section 8.2 because the problem is almost identical with the exception of the adding of a linear function, solvable with the symplex method.

The book used as a reference for the resolution methods is "Linear and Nonlinear Programming" by David G. Luenberger [6].

9.3 Implementation of profiles

As for the forecasting problem, every profile defines a different way for the algorithm to work. The formulation of the optimization problem allows to switch between the profiles at each time slot simply changing the coefficients of
the functions. Here is a list of the profiles and how the coefficients have to be setted to implement that specific one.

- Profilation: the objective is to keep the real hourly amount of energy as close as possible to the declared one. In this profile \( g(j), c(j) \) and \( \text{batt}(i,j) \) represent the real values while \( \text{cost1}(i,j), \text{cost2}(i,j) \) and \( \text{cost3}(j) \) are all zeros. \( \text{cost4}(h) \) is a value greater than zero and have to be predominant on the other cost functions.

- Max gain: \( g(j) \) and \( c(j) \) are unchanged as well as \( \text{batt}(i,j), \text{cost1}(i,j) \) and \( \text{cost2}(i,j) \), while \( \text{cost3}(j) \) and \( \text{cost4}(h) \) are zero. This is the way the objective function computes the maximum real gain in terms of money.

- Battery save: \( g(j) \) is lowered while \( c(j) \) is unchanged as well as \( \text{cost1}(i,j) \) and \( \text{cost2}(i,j) \). \( \text{batt}(i,j) \) return a higher value, as for profilation, and \( \text{cost3}(j) \) is zero as \( \text{cost4}(h) \).

- Optimal SOC: like Max gain, but \( \text{cost3}(j) \) is a strictly positive value because the importance for the batteries to reach a given state of charge has to be accounted.

### 9.4 Signals

The system constantly monitors for incoming signals sent by the DSO: it is a real-time process that does not involve the optimization problem explained previously. When a signal arrives, the application reacts evaluating the possible gain and the feasibility of responding correctly to the signal itself. Since the type, the duration and all the other aspects of the signals are not known yet, nothing has been developed except the general approach to use.

### 9.5 Conclusions

As for the declaration problem, this one is directly derived from the requirements analysis and is very similar to that one. Also in this case theoretically will give the local optimal solution, not the global one. The results that will be obtained after the implementation will allow to make corrections if necessary. Implementation of profiles is also described, and very similar to the one reported in the previous chapter. For the signals there is a bigger problem: due to the unknown form of them, the duration, and other parameters, they are not treated but will be in the next implementations on a proper part of the final application, as described in the next chapter.
Chapter 10

Application

The final application implements all the features seen in Chapter 3 that are needed to let the algorithm work properly to manage the plant as required from FIAMM. In Chapters from 5 to 9 some parts of the application has been explained in more detail, so that in this chapter an overview of the architecture of the application, and his functioning, is explained.

In Section 1 the database structure is shown, in Section 2 the architecture is discussed and in Section 3 it is shown how all the parts of the application are used and interacts to implement the required features.

10.1 Database

In this section the structure of the database is presented.
First the tables schema is represented in Figure 10.1, where the field names as well as their types are listed. The fields with a key at their left represent the columns that composes the primary keys of the related table.
After that, there are some tables in which it is described the purpose of everyone of the fields. Usually the "id" field is omitted in the description, since it is easy to understand it is the identifier of a single tuple. All the ids are in the form "yyyyMMdd_HHmm", as seen in the prototype. They identify the time-slot for the related data, for example id "20120801_1450" identifies the data of the interval from 14:50 to 15:00 of the 1st day of August 2012.
<table>
<thead>
<tr>
<th>Table</th>
<th>Columns</th>
</tr>
</thead>
<tbody>
<tr>
<td>profiles</td>
<td>id, forecasting, forecastingforced, maintaining, maintainingforced, fallback, fallbackforced, signals</td>
</tr>
<tr>
<td>batteries</td>
<td>pack, id, forecastedenergy, realenergy, forecasteddisc, realdisc, forecasteddoc, realec</td>
</tr>
<tr>
<td>setpoint</td>
<td>id, forecasted, real, forced, declared, power</td>
</tr>
<tr>
<td>consumption</td>
<td>id, forecasted, real, forced</td>
</tr>
<tr>
<td>production</td>
<td>id, forecasted, real, forced</td>
</tr>
<tr>
<td>prices</td>
<td>id, selling, buying</td>
</tr>
<tr>
<td>Column name</td>
<td>Description</td>
</tr>
<tr>
<td>----------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>forecasting</td>
<td>the profile that has to be used to forecast the curve of power</td>
</tr>
<tr>
<td>forecastingforced</td>
<td>if setted, means that no other profile can be chosen by the automatic profile manager for forecasting the curve because the administrator forced the algorithm to use the indicated one</td>
</tr>
<tr>
<td>maintaining</td>
<td>the profile that has to be used to constantly keep the power (or energy) profile the desired one</td>
</tr>
<tr>
<td>maintainingforced</td>
<td>as for the forecastingforced, but related to the maintaining profile</td>
</tr>
<tr>
<td>fallback</td>
<td>the profile to be used in case the maintaining profile chosen is profilation, and the profilation fails (i.e. during an hour, the energy putted out is out of the range of ±10% with respect to the declared value</td>
</tr>
<tr>
<td>fallbackforced</td>
<td>as for the forecastingforced, but related to the fallback profile</td>
</tr>
<tr>
<td>signals</td>
<td>if setted, means that DSO signals have to be considered, the possible gain and the feasibility have to be verified, and finally the signal can be followed or not</td>
</tr>
</tbody>
</table>

Table 10.1: profiles table

<table>
<thead>
<tr>
<th>Column name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>pack</td>
<td>identifies the number of the batteries pack, since there are many in the plant</td>
</tr>
<tr>
<td>forecastedenergy</td>
<td>the forecasted amount of energy stored in the batteries at a given time</td>
</tr>
<tr>
<td>realenergy</td>
<td>the real amount of energy stored in the batteries at a given time</td>
</tr>
<tr>
<td>forecastedsoc</td>
<td>the forecasted state of charge of the batteries at a given time</td>
</tr>
<tr>
<td>realsoc</td>
<td>the real state of charge of the batteries at a given time</td>
</tr>
<tr>
<td>forecastedeoc</td>
<td>if setted, means that the algorithm forecasted that the pack of batteries is in end of charge in this time slot</td>
</tr>
<tr>
<td>realeoc</td>
<td>if setted, means that the pack of batteries is really in end of charge in this time slot</td>
</tr>
</tbody>
</table>

Table 10.2: batteries table
Table 10.3: setpoint table

<table>
<thead>
<tr>
<th>Column name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>forecasted</td>
<td>the forecasted amount of power (set point) putted out of the plant at a given time-slot</td>
</tr>
<tr>
<td>real</td>
<td>the real set point of the plant at a given time-slot</td>
</tr>
<tr>
<td>forced</td>
<td>if setted, means that the algorithm cannot change the real set point</td>
</tr>
<tr>
<td>declared</td>
<td>the declared (to the DSO) amount of power putted out of the plant at a given time-slot</td>
</tr>
<tr>
<td>power</td>
<td>the real amount of power putted out of the plant at a given time-slot (can differ from the real set point)</td>
</tr>
</tbody>
</table>

Table 10.4: consumption table

<table>
<thead>
<tr>
<th>Column name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>forecasted</td>
<td>the forecasted amount of power consumed by the logistic center at a given time-slot</td>
</tr>
<tr>
<td>real</td>
<td>the real amount of power consumed by the logistic center at a given time-slot</td>
</tr>
<tr>
<td>forced</td>
<td>if setted, means that the forecasted value cannot be changed by the consumption forecasting algorithm</td>
</tr>
</tbody>
</table>

Table 10.5: buying table

<table>
<thead>
<tr>
<th>Column name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>forecasted</td>
<td>forecasted value of power to be bought from the net</td>
</tr>
<tr>
<td>real</td>
<td>real value of power bought from the net</td>
</tr>
</tbody>
</table>

Table 10.6: production table

<table>
<thead>
<tr>
<th>Column name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>forecasted</td>
<td>the forecasted amount of power produced by the solar plant at a given time-slot</td>
</tr>
<tr>
<td>real</td>
<td>the real amount of power produced by the solar plant at a given time-slot</td>
</tr>
<tr>
<td>forced</td>
<td>if setted, means that the forecasted value cannot be changed by any external algorithm</td>
</tr>
</tbody>
</table>

Table 10.7: prices table

<table>
<thead>
<tr>
<th>Column name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>selling</td>
<td>price at which the energy is sold, at a given time-slot</td>
</tr>
<tr>
<td>buying</td>
<td>price at which the energy is bought, at a given time-slot</td>
</tr>
<tr>
<td>Column name</td>
<td>Description</td>
</tr>
<tr>
<td>-------------</td>
<td>-------------</td>
</tr>
<tr>
<td>name</td>
<td>name that identifies the festival</td>
</tr>
<tr>
<td>year</td>
<td>year at which the festival is referred to</td>
</tr>
<tr>
<td>date</td>
<td>exact date at which the festival occurs in the given year</td>
</tr>
</tbody>
</table>

Table 10.8: festivals table

### 10.2 Architecture

![Class diagram](image)

- **DecisionEngine**: main class that schedules executions of the various plugins at the correct time
- **Logger**: logs the actions and errors of the program
- **DbManager**: manages all the actions to do on the database
- **ConsumptionForecast**: plugin which calculates the forecasting for the consumptions, and manages also the festival days
- **CurveForecast**: plugin which computes the forecasted curve of power, based on the profiles chosen
- **CurveMaintain**: plugin which recomputes the curve of power every time it is scheduled, based on the profiles chosen
- **SignalsManager**: plugin that manages the signals of the DSO and decides when to follow the signals or not
• ProfilesManager: plugin that manages the profiles to use
• ReportManager: plugin used to produce a daily report
• Solver: class that is used to solve the problems constructed by the Curve-Forecast and CurveMantain algorithms

10.3 General functioning

The main class of the program is the DecisionEngine class, which has two tasks. The first one is to manage the plugins, since the plugins can be changed "on the fly" by the system administrator in case an updated plugin is developed or one with a different behavior has to be used. Given a scheduling period, the main class checks for changes in the plugins and reloads the modified ones so that the next execution will use them. The second task is to schedule, with the correct timing, the execution of the plugins that compose the application. For example, the plugin for the reports has to be scheduled once per day not much time after midnight.

The ConsumptionForecast is executed once per day and implements the algorithm for the consumption forecast seen in Chapter 6. The data needed is read using the DbManager, that is used also to store the result of the forecasting once it is done. Also, the festivals are managed, and an automatic algorithm to compute and store the festival days in a year can be implemented if specified.

Also, the CurveForecast is executed once per day, not before the consumptions forecasting has been done. This class implements the algorithm seen in Chapter 8 since the tasks are both to forecast the power curve and the battery state as well as declaring the set point to the smart grid manager. This last function is not implemented yet because there is not a smart grid implemented, but future changes will not affect much the functioning because it is an external plugin to the application.

The CurveMantain plugin implements the problem seen in Chapter 9 and is scheduled with a time period of 10 minutes, 24 hours a day. The task is to maintain the best curve based on the profile chosen, updating the values on the database so that the SCADA can send them to the PLCs.

Both CurveForecast and CurveMantain classes use the same Solver class that implements the heuristic resolution method explained in Section 8.2. The problems composed each time are solved and the results are simply sent back so that the values are used in the correct way by the calling functions.

SignalsManager is the plugin to which the function of reading and evaluating the signals sent by the smart grid manager is associated. As already stated in Section 9.4, the way signals arrives, what is the duration of the signals, and other aspects, are not known yet so that in this first implementation no code is present. Once implement in the next future, the task will be of sending to the SCADA the real time value of set point, so that the scheduling has to happen in a matter of seconds or less.

The class ProfilesManager has the task of managing the usage of the profiles. It is scheduled once every 10 minutes and decides for every time slot what profile has to be used for the maintaining and what for the declaration. In particular in the case in which the profilation fails for a given hour, this plugin will choose the best profile to use from that moment up to the end of the day, since profilation
is no more needed and another objective (for example saving batteries) can be pursued.

The ReportManager produces once per day, a few minutes after midnight, the daily report in which all the important parameters of the plant are saved. The output is a pdf file that contains charts representing the forecasted and the real situation, as well as other informations like the number of profiling days already met from the first day of the year. All these informations are useful to the system administrator to tune the algorithm, to decide the strategy to use, and can also be used to refactor the optimization problems in order to correct inconsistencies.

Logger is not a plugin, but has an important task: it registers all the actions that the application performs and the eventual errors and exceptions. It is useful mainly for the future correction of bugs present in the software and to check that everything is working as expected. The output is stored in a simple text file with a level of information that is sufficient for the administrator.

As already said, all the values in the database that represents real values measured from the various parts of the plant are stored in the database in a way that is transparent to this application. They are simply accessible using the DbManager, which job is also to allow the various classes to store or update values.
Chapter 11

Conclusions

The FIAMM Green Energy Island is one of the first plants in the world that produces renewable energy with the possibility of storing it. Thanks to the batteries, the energy is available not only when the sun is up but whenever it is required. To allow the plant working the best way for both the smart grid manager and the investor, an intelligent application for managing the storage has to be developed. The objective of this thesis is to develop an algorithm that can support the decisions of the plant administrator in terms of functioning of the plant, requiring few effort or no effort at all. The resulting application should let the administrator choose the way of functioning once, and then just check on a daily basis if the system responds as expected.

The prototype implemented does not take into account all the aspects defined in Chapter 3, but has been useful for many purposes. It is a partial implementation of the requirements allowing to forecast a set point and to maintain the hourly amount of energy as close as possible to the forecasted one. The results obtained let us foresee the feasibility of a system like the required one, and also let us face and solve some problems in advance before the final application is developed and deployed.

The optimization problems described in Chapters 8 and 9 are directly derived from the implementation of the requirements analysis and the technical specifications. The coefficients associated to the single functions that compose the objective function allow the algorithm to work in many ways, while the constraints represent the physical boundaries of the plant. The resolution method is an heuristic that gives a non global optimal result, but on the other hand is pretty simple.

Theoretically the results that can be obtained by using this work can cover all the desired profiles for the way of functioning, and are even customizable with no much effort simply changing the profile manager plugin.

The strategy for the usage of the battery packs and the end of charge is also an heuristic but it is a good one because no more than one pack per time can be sent in EOC. Even the fact that this state is setted only at night gives the chance of doing it without interfering with the profiling, in which any variation can lead to an uncounted day for reaching the required number of 300 per year.

Finally the structure of the application allows to make several changes to the various parts in a semi real-time fashion, just substituting the plugins.
Bibliography


