Politecnico di Torino

Politecnico di Torino

Corso di Laurea in Ingegneria Energetica e Nucleare
A.A. 2022/2023
Sessione di Laurea Novembre 2022

Strategies for citizen engagement on renewable energy plants

A case study on Pantelleria Island

Relatori:
Prof. Giuliana Mattiazzo
Dott. Riccardo Novo
Dott. Enrico Giglio
Dott. Claudio Moscoloni
Dott. Daniele Schiera

Candidato:
Federico Gallarato

"Se uno vive in una comunità e pensa di poter fare a meno degli altri,
o è bestia, o è Dio"

Aristotele

Acknowledgements

In Italiano, per poter parlare meglio a chi mi hanno accompagnato in questo viaggio.

Innanzitutto ci tengo a ringraziare il MORE Lab e la professoressa Mattiazzo per l'opportunità e le conoscenze che ho acquisito nello svolgere questa tesi. Grazie soprattutto ai correlatori Riccardo Novo, Daniele Schiera, Enrico Giglio e Claudio Moscoloni per la curiosità e le motivazioni che mi hanno instillato ad ogni incontro, e per l'infinita pazienza nel seguirmi nello svolgimento del progetto. Anche se non c'è stata l'opportunità confrontarsi più spesso, ringrazio il prof. Jens Lowitzsch, fonte di ispirazioni e riflessioni in ogni discussione.

Ringrazio infinitamente, anche se può venirmi difficile a parole, la mia famiglia per il supporto, l'amore e l'appoggio incondizionato durante tutti questi anni: questo traguardo sarebbe arrivato senza di voi. È per mamma, è per papà, è per Elena, è per zia Ivana, per tutto quello che avete fatto per me. È per Lucy che mi ha pagato gli studi in cambio di un pezzo di mela, ed è per Pina che basta la sua presenza per rendermi felice. È per tutti i parenti di Mondovì sono sempre felici di sentirmi anche se li sento poco, è per i parenti di Mompantero, le radici che mi seguono da lontano e mi accolgono sempre a braccia aperte. È per i parenti che non ci sono più ma che sento sempre vicini, "Sei stato promosso? Studia!"

È per le persone che ho incontrato lungo il cammino e vorrei mi accompagnassero ovunque e per sempre. Copio una di queste e non faccio la lista dei nomi, così non dimentico nessuno, ma ricordo ciò che mi porto dietro più volentieri.

È per ringraziare le persone con cui si è andati alla scoperta dei pub per poi attraccare sempre in quello più comodo; con cui si è andati al mare tra scorci di Thailandia, pizze in spiaggia al tramonto e bagni di notte; con cui si è girato per le montagne a cercare formaggi, droni e torrenti. Persone con cui andare a vedere le partite, distruggere i termosifoni e festeggiare in casa; con cui imparare a vivere da soli e ad apprezzarlo, crescere come non pensavo di riuscirci. Persone con cui imparare a fare cocktail raffinati o mescolare due cose insieme, cosa conta è stare bene. E i sushi, la nduja, i tartufi, il cioccolato e quegli spaghetti con i gamberi troppo buoni. È per una festa di laurea fatta a rate dopo una proclamazione via mail. Per chi pensavo non mi parlasse più e sono felice che sia tornata, per chi sento vicino anche se ora vivono lontano. Per chi ha condiviso serate di gioco finite a tarda notte, meme, video e le cose più casuali, dai piccioni e i gabbiani, le scimmiate, i balli e i fantatornei. E soprattutto grazie a chi mi ha continuato a supportare, motivare ed ascoltare in questi anni. Senza di voi non ce l'avrei fatta.

È per Torino, la città più bella in cui potessi pensare di andare a studiare.

Non me lo dico spesso, però ora me lo merito: "and last but not least... I wanna thank me"

Abstract

The European Union is facing a serious energy crisis. With more than 80% of its oil and gas imported from Russia, gas prices have reached an all-time high following the invasion of Ukraine and subsequent sanctions, affecting the energy security of many citizens. This has also led to discontent against the big energy companies over their profits, as well as high inflation in all member states. Furthermore, with global temperatures rising dangerously, it is crucial to move forward to ensure the possibility of fighting climate change. One solution could lie in a reorganisation of the energy market, allowing citizens to join together in Renewable Energy Communities with the aim of a decentralized electricity production locally with the direct participation of the members.

The aim of this work is to analyse the process of engagement in the creation of a Renewable Energy Community, especially at the moments when a preliminary assessment is needed to understand the feasibility of the community and its potential achievements.

The proposed solution is to develop a tool with the goal of simulating electricity sharing between members of an energy community during a solar year. The end users of the tool have to provide an input file that gives an overview of the consumers, prosumers, generation plants and public storages present in the community with some brief information such as technical data or family composition. Then the software performs the calculations of the annual consumption and production profile, the use of the batteries, the shared electricity among the members and the electricity exchanged with the public grid.

The evaluation of the results is done by comparing four scenarios with different but realistic characteristics. The first is a general community composed of a mixture of consumers, prosumers, a central production facility and a centralized storage, while the second differs only in the absence of the storage. The third describes a situation where the end users are all consumers, while in the fourth those are only of prosumers. The results highlight some important aspects that could provide useful guidance in the engagement phase. One of them is that the absence of energy storage reduces the amount of time a community can be self-sufficient, as this is only guaranteed for about 43% of the time, much less than in the other scenarios where self-sufficiency is achieved for at least 78% of the time. In particular, in the third scenario, where the users are all consumers, 94% of the best self-sufficiency time is achieved, which corresponds to 344 days.

In addition, the evaluation of the incentives and the expenses showed that in that scenario are generated more than 13500 [€] in incentives, with more than 10000 [€] of revenues to redistribute among members or in the community.

Contents

Introduction

Research background and motivation

The transition to global sustainability in social, economic and environmental terms is defined by the Sustainable Development Goals or Global Goals,the global guidelines that United Nations member states must follow to move toward a better world [1]. In particular, the aspect related to the access to energy has a great impact on people's lives. Among the seventeen SDGs, there are three that relate specifically to this topic.

The first goal, that is most related to the work of this thesis, is number 7: *Affordable and Clean Energy*. To give some perspective, 10% of the world's population still does not have access to electricity, which among those who do have access is responsible for 73% of greenhouse gases emitted by human activity. In 2017, only 17.5% of electricity was generated from renewable sources, with 11.5 million people employed. So, it is easy to understand that their development would benefit many people, especially in developing countries [1].

The second goal that has an impact on this thesis is number 11: *Sustainable cities and communities*. It is estimated that by 2050, more than 6.5 billion people will live in urban areas, more than two-thirds of all humanity. With sustainable cities and communities, it will be possible to provide people with benefits such as safe and affordable housing, public transportation, better city planning, and more civic participation. [1]

The third item, number 13: *Climate Action*, addresses the need to fight climate change. Every country, no one excluded, will be affected, and the consequences can be tragic. From geopolitical conflicts due to climate-related events to extreme weather conditions caused by temperature changes around the world. The goal is to limit global warming to below 1.5°C and reduce CO2 emissions by 45% by 2030, while achieving net-zero emissions before 2050 [1].

The introduction of Renewable Energy Communities will have a direct impact on these targets. Namely, these will increase and promote the production of electricity from renewable

sources locally, with power plants distributed and democratically owned, rather than concentrated and owned by a few oligopolistic companies. The ability to operate and influence at the local level gives citizens and small businesses the opportunity to participate and not be excluded for social, economic or racial reasons.

This work is intended to contribute to the research and development of technologies that can help follow these guidelines. In particular, it is intended to contribute to the creation of an ideal sustainable model that will lead to the successful implementation of renewable energy communities.

Structure of the thesis

The aim of this thesis is to create an application code capable of simulating the energy flows generated in a Renewable Energy Community over the course of a year, with an evaluation of the differences between the traditional consumption and production method and a use of the benefits available in a community. Chapter 1 gives a general overview of electricity generation in Europe and the definition of Renewable Energy Communities, and presents the legal framework and some existing examples. Next, Chapter 2 explains the theoretical aspects needed for the work in the different phases that characterise it, and is followed by the explanation of the simulation in Chapter 3. The resulting results are presented in Chapter 4, while Chapter 5 describes the conclusions and the future perspective of the project and of the Renewable Energy Communities.

CHAPTER 1

Introduction to Renewable Energy Communities

In this first chapter of the thesis, an introduction to the Renewable Energy Communities is given in order to understand the current situation in the European market from both a production and legislative perspective. In the Section 1.1, the current state of electricity production is analysed, comparing European production with that of Italy and Pantelleria, and listing some of the challenges that the energy world faces today. Then, the Section 1.2 presents the European and Italian initiatives to fight climate change, while the following Section 1.3 takes a closer look at the definitions, regulations and examples related to Renewable Energy Communities. The last Section 1.4 looks at the aspects behind the creation of a Renewable Energy Community, from the legal framework to the steps to follow in the creation phase.

1.1 Electricity generation

For an introduction to Renewable Energy Communities in Europe, it is important to first get an overview of the current situation in terms of installed capacity and final electricity production by source.

1.1.1 Electricity production in the EU

The first point to assess is the installed capacity of electricity production, which can be seen in Table 1.1. Here the situation of electricity production in the European Union, in Italy and in Pantelleria is compared. In Table 1.1 can be seen the electricity produced.

Looking at and comparing the shares of fossil fuels consumed, the opportunities for growth in renewable energy across the continent are great. Especially in Pantelleria, where renewable energy sources are almost untapped, it is possible to take action and at the same time have the opportunity to adopt the most advanced technologies and be a pioneering example.

Source	European Union [MW]	Shares [%]	Italy [MW]	Shares [%]	Pantelleria [MW]	Shares [%]
Fossil fuels	333449	38.7	55930	51.0	23	96.8
Nuclear	111240	12.9	0.0	0.0	0.0	0.0
Renewables & biofuels	416133	48.3	53769	49.0	0.752	3.2
of which						
hydro	148613	17.3	22393	20.4	0.0	0.0
geothermal	861	0.1	767	0.7	0.0	0.0
solar	110591	12.8	20107	18.3	0.720	3.0
wind	155545	18.1	10230	9.3	0.032	0.2
others	523	0.1	272	0.2	0.0	0.0
Total	860822	100.0	109699	100.0	32.75	100.0

Table 1.1: Comparison between the installed capacities for electricity production in European Union, Italy and Pantelleria [MW] [2].

Source	European Union [GWh]	Shares [%]	Italy [GWh]	Shares [%]	Pantelleria [GWh]	Shares [%]
Fossil fuels	1209510	41.2	173309	59.8	39.07	98.7
Nuclear	761943	25.9	0.0	0.0	0.0	0.0
Renewables & biofuels	966520	32.9	116161	40.2	0.498	1.3
of which						
hydro	370234	12.6	50502	17.4	0.0	0.0
geothermal	6655	0.2	6105	2.1	0.0	0.0
solar	113026	3.8	22654	7.8	n.d.	n.d.
wind	320613	10.9	17716	6.1	n.d.	n.d.
others	156992	5.3	19184	6.6	0.0	0.0
Total	*2937973*	*100.0*	*289470*	*100.0*	*39.564*	*100.0*

Table 1.2: Comparison between the electricity produced in European Union, Italy and Pantelleria [MW] [2].

To get an idea of how Pantelleria is doing in terms of electricity generation, it is possible to check out the Transition Agenda [3]. In total there are 6 Diesel generators, which account for 23 [MW] of installed capacity, which is 96.8% of the total capacity. As for renewable energy, there are two wind turbines, one with a capacity of 30 [kW] and the other with only 2 [kW], representing a total of 0.2% of the nominal capacity. Finally, there are 720 [kW] photovoltaic panels in many small distributed installations, which account for the remaining 3.2% of the installed capacity. It is also worth noting that the Diesel generators account for 98.7% of total electricity generation, which means that they are still the preferred technology even when renewable energy is available.

In the Italian scenario, which unfortunately does not use nuclear energy to generate electricity, investments in renewable energy are necessary and are on the agenda of every political party, of course with different points of view. After Italy withdrew from the use of nuclear energy following the referendum in 1987 and 2011, many candidates have promoted the use of nuclear

energy for the first time, but it is not foreseeable if and when these reforms will be approved and initiated.

1.1.2 Challenges for the energy world

Energy poverty

According to the United Nations, the fight against energy poverty will be one of the greatest challenges in the coming years. This refers to the lack of access to energy at a sustainable price in the form of electricity, cooking fuels and technologies [4]. Globally, about 1.2 billion people suffer from a lack of electricity, while up to 40% of people do not have access to clean fuels. Reducing inequalities in the world related to this factor can lead to a reduction in inequalities and injustices and thus also contribute to the development of the other Sustainable Development Goals [1].

The aspects that need to be worked on in the future are the quality of energy access, the creation of renewable energy strategies, the democratisation of the energy sector and the support of low-income citizens to invest in renewable energy [5].

Price volatility

The recent conflict between Russia and Ukraine, which began in February 2022, together with the residues of the pandemic that started in 2019, have highlighted the difficulties that the European energy sector may face in the event of geopolitical and social problems. The retail price of electricity has increased by more than 200% in some states, causing a general energy crisis in Europe.

The Figure 1.1 shows the fluctuation of electricity prices compared to some European countries.

The volatility of energy prices caused many problems for citizens, industry and the nations' reserves and showed the need to overcome certain constraints and increase investment in renewable energy.

Year	I [€/kWh]	II [€/kWh]	III [€/kWh]	IV [€/kWh]
2018	0.2062	0.1898	0.2022	0.2176
2022	0.4603	0.4134	0.4151	0.6601

Table 1.3: Prices [€/kWh] of electricity for a 2,700 [kWh/y] consumption at 3 [kW] [7]

In Table 1.3 it is possible to compare the quarterly retail price of electricity provided to a family with an annual consumption of 2700 [kWh] and a power of 3 [kW]. This year's prices are more than twice as high, which is a problem for low-income families.

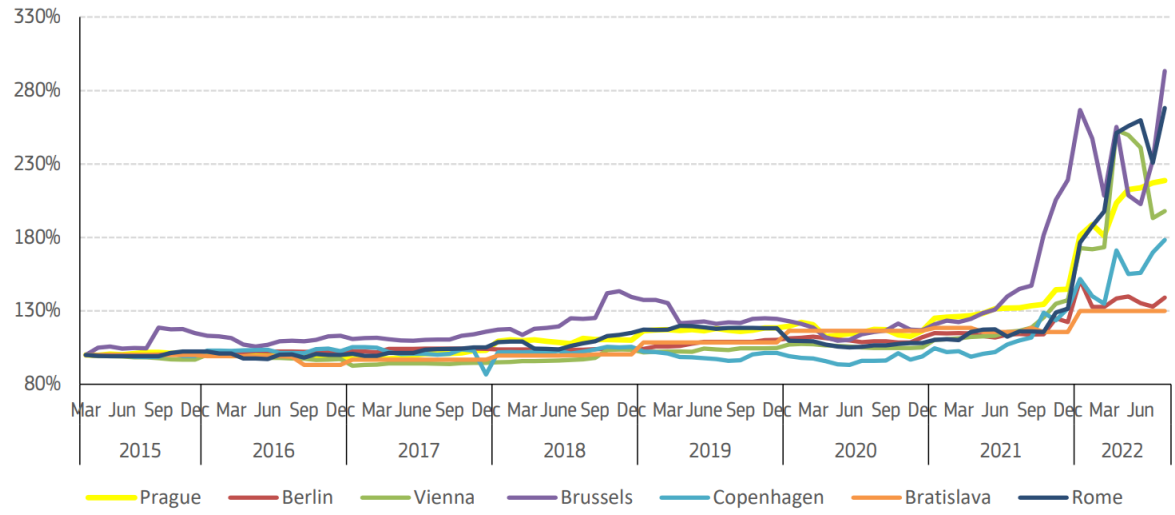

Figure 1.1: Trend of the electricity prices in EU from 2015 to 2022 [6].

Pollution and global warming

As shown in Section 1.1 and in the Table 1.2, the European scenario is driven by 1,209,510 [GWh] of electricity generated by fossil fuels, which corresponds to 41.2% of the total production, and contributes to the emission of 3.7 [Gt] of CO2 yearly [8].

A report from the European Environmental Agency [9], has estimated that the growth in global emissions from electricity generation will outpace the growth in emissions from other sectors over the next two decades. The links between carbon dioxide levels and the global warming are more than well known and this is also an aspect that world leaders are trying to address to avoid a global threat that has the potential to be catastrophic.

1.2 European and Italian initiatives

In recent years, the European Union has begun to develop a legal framework aimed at enforcing changes in members' regulations in order to take action against the problems of climate change and improve citizens' quality of life from an energy perspective.

1.2.1 European initiatives

In the EU, laws are not made directly by the European Parliament, but they provide guidelines for nations to make their laws. In the end, each country will have a different law trying to achieve the effect desired by the central authority. In 2016, for example, the Clean Energy Package [10] was developed to focus on five main aspects:

1. Energy safety
2. Internal energy market

3. Energy efficiency
4. Decarbonization
5. Research and innovation

These five themes are combined with another plan to reduce CO2 emissions and fossil fuel dependency, known as the European Green Deal, which were unveiled in 2019. The main objectives are to achieve net zero greenhouse gas emissions by 2050 and to decouple economic growth and resource use, guaranteeing the participation and inclusion of all citizens in this pact [11].

Its goal is to provide:

- Fresh air, clean water, healthy soil and biodiversity
- Renovated and energy efficient buildings
- Healthy and affordable food
- More public transport
- Cleaner energy and cutting-edge clean technological innovation
- Longer lasting products that can be repaired, recycled and reused
- Future-proof jobs and skills training for the transition
- Globally competitive and resilient industries

Each sector will be involved and supported to achieve these goals through initiatives, policies and financing that each country must adopt to reach these goals before the 2050 deadline.

1.2.2 Italian initiatives

To achieve this result, the Italian government took action by implementing laws and subsidising many sectors to help citizens, industry and communities move forward in line with European directives. The initiative influencing and guiding the transition is the Piano Nazionale Ripresa e Resilienza, PNRR, which is funded with European funds from the program *Next Generation EU* [12]. Some notable incentives might include:

- Superbonus 110%: With this huge financial manoeuvre, the government incentivised the renovation of buildings by granting a tax deduction to those who improved the energy efficiency of households through measures such as thermal insulation, installation of photovoltaic panels or heat pumps [13].

- Hydrogen Valley: Thanks to these funds, a national plant for the production of electrolyzers will be created, which should reach a capacity of 1 [GW] by 2026 [14].
- Industria 4.0: This initiative aims to help industries improve their machinery, management and research and development activities in order to renew and increase the competitiveness of Italian industry [15]
- Promotion of Renewable Energy Communities and Collective Self-Consumption: The objective is to support the creation of Renewable Energy Communities and Collective Self-Consumption and provide incentives to produce, consume and exchange clean energy from renewable sources locally so that their members pay less for it [16].

Going into further detail, in Pantelleria, the goal is further explained in their Energy Transition Agenda [3] and can be summarised in six points, which are listed below. The administration's goal is to develop all of these points in order to progress towards an island with net zero emissions:

1. Energy efficiency: Improve the energy efficiency of buildings by promoting their renovation and the use of better technologies, such as heat pumps.
2. Electricity generation from renewable sources: Increase electricity generation from renewable sources instead of the old diesel power plant.
3. Decentralised electricity generation and building self-sufficiency: Encourage the spread of rooftop solar to improve household self-sufficiency.
4. Electrification of transport: Replace old buses and public administration vehicles with new electric vehicles and install charging stations.
5. Installation of storage systems: Regulate electricity flows using production forecasts and demand-side management to avoid losses or blackouts.
6. Pantelleria Renewable Energy Community: Starting with pilot projects, identifying a business model and creating a local energy community.

It can also be said that thanks to the PNRR and its structure, some of these objectives can be financed more quickly, speeding up the implementation of initiatives with European funding that would otherwise depend entirely on local government or private initiatives.

1.3 Renewable Energy Communities

Among the possible solutions that propose innovative solutions to the problems described earlier, one of the most promising is the creation of Renewable Energy Communities to help

citizens consume their own electricity and increase the overall production of electricity from renewable sources.

1.3.1 European directives

What is exactly a Renewable Energy Community is defined by the Clean Energy Package, which is part of the Green Deal mentioned earlier. It is a set of rules aimed at having a positive impact on the environment, citizens and the economy [17]. A "Renewable Energy Community" is defined in Article 2.16 as: "*a legal entity:*

- *which, in accordance with the applicable national law, is based on open and voluntary participation, is autonomous, and is effectively controlled by shareholders or members that are located in the proximity of the renewable energy projects that are owned and developed by that legal entity;*
- *the shareholders or members of which are natural persons, small/medium enterprises or local authorities, including municipalities;*
- *the primary purpose of which is to provide environmental, economic or social community benefits for its shareholders or members or for the local areas where it operates, rather than financial profits.*"

Moreover, in Article 2.11 is also defined the "Citizen Energy Community" as:
"*a legal entity:*

- *is based on voluntary and open participation and is effectively controlled by members or shareholders that are natural persons, local authorities, including municipalities, or small enterprises;*
- *has for its primary purpose to provide environmental, economic or social community benefits to its members or shareholders or to the local areas where it operates rather than to generate financial profits;*
- *may engage in generation, including from renewable sources, distribution, supply, consumption, aggregation, energy storage, energy efficiency services or charging services for electric vehicles or provide other energy services to its members or shareholders.*"

Legal framework

A big step towards the legal definition and conditions defining renewable energy communities is the *European Renewable Energy Directive II*, also known as *RED II*, which introduces and proposes the framework for Renewable Energy Communities in Article 22, with a legal perspective that allows RECs to be established and owned as companies by individuals, public administrations and businesses.

With this regulation, Member States commit to producing 32% of final energy consumption from renewable sources by 2030. Member States must draw up a national plan to achieve

this target, which may include financing and incentivising the purchase of renewable energy installations and the promotion of renewable energy for heating and cooling, amounting to at least 14% of total consumption [18].

1.3.2 Existing communities in the world

Around the world, there are already some examples of functioning energy communities that have been in operation for a few years.

Extra-European examples

A non-European example of an island that has managed to produce 100% renewable energy generation is Kodiak Island in Alaska. There, a community model was developed with public and private participation, adopted first for the establishment of a wind farm and then for all other facilities by issuing bonds to finance a public trust fund. It is the only example applied on an island that demonstrates the potential of community participation, and the results obtained show that it can lead to great successes over time [19].

Another example is Rio Grande Energia, a company founded in the Brazilian state of Rio Grande do Sul. Founded in 1999, the company initially owned one hydroelectric plant and gradually acquired others up to six plants, which today supply 20,000 members in the surrounding area [20].

Denmark

It is common for reluctant citizens to protest against the installation of wind farms, but in Denmark a great result was achieved when it was found that after the community-owned turbines were installed, there were no protests, but instead a lot of support from the citizens. An example of this phenomenon occurred in 2010 when it was proposed in Hvide Sande to build a wind farm 80% owned by a community foundation and 20% owned by other local investors [21].

Germany

The energy market in Germany is characterised by the fact that around 40% of the total capacity is owned by citizens and communities.

In the municipality of Schonau, a group of 650 citizens established a community in 1994 with an initial investment of the equivalent of 4 million Euros, which started operations in 1996. In the years that followed, it grew to its current value of 43 million euros, has over 5,000 members and serves over 160,000 households [22].

Netherlands

In the Netherlands, it has been common in recent years for municipalities to purchase solar panels or wind turbines, despite their natural gas deposits in the sea

The largest community is composed by around 5,000 citizens who, in cooperation with the public and a German turbine manufacturer, financed the construction of a 103 [MW] wind farm with 34 turbines.

1.3.3 Italian legislation

Given the current situation in Italy, the government has taken steps to define the first legal framework, for the first time in the PNRR. One of the aspects being promoted is the proliferation of Renewable Energy Communities (REC) and Collective Self-Consumption (CSC) to increase widespread electricity generation on a broad basis [16].

The first legal framework was published in 2019 with Law 162/2019, the so-called "Decreto Milleproroghe"

In addition to a number of environmental measures, such as the aforementioned Superbonus 110, there was the first experimental law defining RECs.

As it was the first time this issue was dealt with, the main constraints were that the members had to be close to each other and connected to the same medium voltage cabin and that the power generation could not exceed 200 kW. These aspects were a limiting factor as they could not have created larger communities [23].

The law, later summarised in its successive adaptations and recapped by [20], defines some technical limitations for the establishment of a renewable energy community:

- Rated power: Production facilities shall not have a total nominal power of more than 1 [MW]
- Location: The production facilities and the households must be connected to the grid via the same primary high-voltage cabin.
- Existing installations: These may opt to participate in new energy communities, but their capacity must not exceed 30% of the total nominal capacity.

It also defined the incentives, set for 20 years from the date of establishment of the community, which regulate and encourage the production and sharing of electricity from the communities:

- Premium tariff of 110 [€/MWh] for RECs, or 100 [€/MWh] for Collective Self-Consumption, for the electricity shared in the community.
- Selling tariff for the excess electricity sold to the grid with the "ritiro dedicato" tariff. It can also be decided to take the minimum guaranteed price, fixed at of 40.7 [€/MWh] for

photovoltaic plant. In Table 1.4 are listed the minimum guaranteed prices for the renewable sources.

- Restitution of 8 [€/MWh] by ARERA of the shared electricity to award the benefits given to the system avoiding transmission losses.

With these incentives, both the electricity sold and the electricity shared is paid back, but the difference in value is meant to motivate members to share more rather than resell.

Source	Energy acquired yearly [MWh]	Guaranteed price [€/MWh]
Biogas and biomasses	< 2,000	96.2
Gas from depuration	< 1,500	51.1
Wind power	<1,500	40.7
Geothermal	<1,500	53.4
Hydroelectric	< 250	158.9
	250 - 500	109.0
	500 - 1,000	68.5
	1,000 - 1,500	59.2
Others	<1,500	40.7

Table 1.4: Minimum guaranteed price depending on the renewable energy sources [24]

Thanks to these incentives, many new churches have been planted or are in the process of being planted. It is estimated that more than 100 new communities have been established in the last 2 years, with the possibility that there will be even more in the years to come [25].

1.3.4 Existing communities in Italy

As indicated in a report by the environmentalist association Legambiente [25], 2 renewable energy communities are currently active and operating, while at the time of the report 16 are in the planning phase and another 7 are in the preparation phase. There are also 15 collective self-consumption systems that demonstrate how new technologies can meet people's needs. The two energy communities that are currently active and working are in Naples and Magliano Alpi.

San Giovanni Teduccio

The first is located in the neighbourhood of San Giovanni a Teduccio in the east of the city of Naples and uses a 53 [kW] photovoltaic system. Its aim is to reduce energy poverty among families in the area, who will benefit from the positive aspect of being part of the community. Economically, the savings are estimated at €300,000 over 25 years, both from the incentives and the energy savings.

Magliano Alpi

The second case, involving a 20 [kW] photovoltaic system, is an example of how the municipality can help and serve the community by using public funds to cover the installation costs. The system provides electricity to the town hall where the panels are installed, the nearby school, library and gymnasium, and five neighbouring families. Benefits of the installation include meeting 40% of the electricity demand, a charging station for electric cars and the installation of smart metres to monitor consumption for energy flow analysis and service management.

Other examples

In the same report are also included projects that are almost completed, such as:

- Savona self-sufficient port: With a 4 [MW] photovoltaic plant, it can meet up to 95% of the port's annual electricity needs. It can also supply electricity to the cruise ships in the port by storing the excess electricity in a 20 [MWh] storage facility.
- D'Annunzio University: In Chieti, the project of a 100% renewable energy university campus is currently being implemented with internal funds and crowdfunding by students, professors and other employers. Measures are still being evaluated, but include power generation systems and thermal efficiency as well as technical support from university staff.
- Agro-energy community: This initiative, proposed by Coldiretti Veneto and also adopted by Coldiretti Puglia, makes it possible to generate about 1,800 [MWh] of electricity per year thanks to the equipment installed in the users' farms, which also supplies the regional and local Coldiretti offices. The innovative technology used at REC is a blockchain-based monitoring protocol that automatically records transactions and calculates the electricity inputs and outputs of each user.

There are several other examples in the literature of islands without an interconnected grid that have chosen to generate 100% of their electricity from renewable energy sources that could guide Pantelleria's decision-making, such as those mentioned in Section 1.3.2.

1.4 Creation of a REC

A Renewable Energy Community is a legal entity whose structure must be defined by its internal regulation, policies and financing, and each of these elements should be carefully considered in order to create the best possible structure.

1.4.1 Financial structure

One of the aspects that strongly shapes REC in terms of community impact and economic outcome is the way it is funded by the users or the public entities involved.

The first option is to set up a *Cooperative*: They are widespread in Italy in various fields, from employment services to cultural associations. Their aim is to work for a collective enterprise with the values defined by the International Cooperative Alliance. The basic principles are voluntarism, self-management and distribution of profits among members independent of other organisations [26]. This organisation is the one that best implements the concept of energy democracy, since the owners are also the consumers, without external interference, and most importantly, when it comes to making decisions, each member has the same rights and importance, regardless of investment, public importance or experience. It also creates and redistributes value, with a social meaning as it is inclusive and open to pursue its goal. The active participation required to the members is certainly a positive aspect, especially in small cooperatives where it is possible to know and manage the desires, willingness and problems of individual members.

In larger organisations, however, it is important to maintain control. Effective management that can handle problems with social knowledge is the key to avoiding internal conflicts or technical problems for which there is no one within the membership with the expertise to solve them. It is also possible for local governments to receive public funds, but since the voting mechanism is democratic, their weight in the decision-making process is the same as that of the individual citizen, and this can become risky because their interests may not be guaranteed. So the solution could be a non-repayable loan if the purpose is only to help the cooperative receive funds, or some kind of voting agreement, but that may not be the best democratic solution for the community. Another aspect to consider with a cooperative fund for energy communities is that the scalability will be difficult with a large number of members, especially if there are several neighbouring cooperatives: There could be competition between them, with members who might switch from one to another just because they are pursuing their personal interests rather than the big picture [22].

Another way is to form a *CSOP*. This stands for *Community Stock Ownership Plan*, which is a way to raise money and invest in the creation and purchase of shares in an Renewable Energy Community that can provide various benefits to the participants. This requires setting up a trust fund, preferably under the supervision of a competent authority and managed by an independent trustee. They can then take out a loan, which is repaid over time with the CSOP's earnings. Unlike cooperatives, shareholders can decide how much money they want to invest without being restricted by a fixed entry fee. Also, the number of shares the user owns can vary depending on various factors defined and limited by the community's regulations and its

ultimate purpose. For example, a community that wants to tackle fuel poverty could choose a structure that provides shares to vulnerable users with facilitated financing schemes that are repaid over the years, so that they can obtain their shares in an easier and cheaper way than in "traditional" ways. Another advantage is that the public administration and external members can invest and influence the decision-making process, so that the community is treated more like a company than a cooperative.

Of course, the weight that individual members have can be changed according to the purpose of the community, the internal guidelines or the amount of money invested. Of course, care must be taken that the distribution of shares is not in the hands of a few, otherwise it will be almost the same as the current days, where the leadership of the sector is in the hands of the larger companies. Instead, the community can decide how to set ownership limits or how to limit the influence of unwary investors or institutions.

In a scheme proposed by professor Jens Lowitzsch [22], the formation phase involving all parts involved in the CSOP is shown schematically.

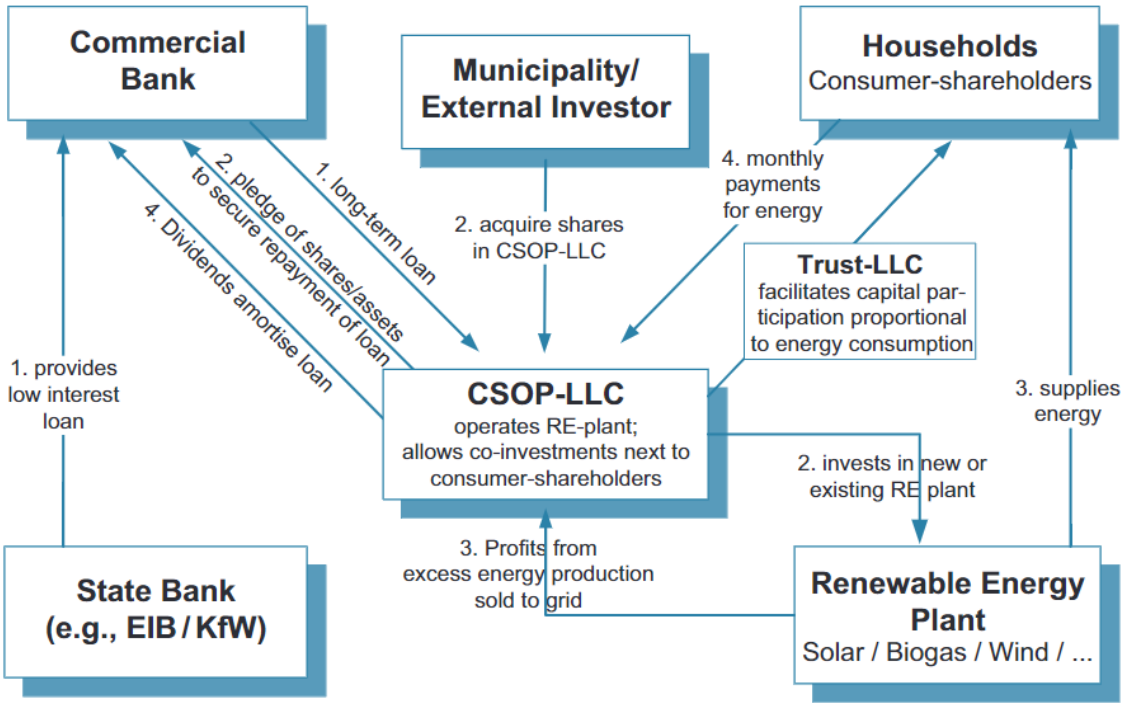

Figure 1.2: Fluxes of finances in a CSOP [22].

The first basic thing is the establishment of a Limited Liability Company, *CSOP-LLC* , which will manage the aspects of the community and be the unifying element between all the entities involved, which are:

- Municipality/External Investors: These elements buy shares to invest and make decisions in the council of the LLC. The public administration has the possibility to be part of the

community as a consumer with one or more buildings as well if it is located in the area of interest connected to the same primary cabin.

- Households: the key element that will be part of the community. They pay their energy bills monthly as they would with a normal energy company, but in this case they have the opportunity to earn from the energy they share and sell, as well as dividends from the energy sold from the plant they have invested in. This aspect could be managed by a Trust-LLC set up with the aim of managing the community economically. This category includes not only citizens' households, but also industries, businesses or any kind of commercial activity interested in the investment.
- Commercial Bank: it provides a long-term loan to the community by taking out a low-interest loan from the State Bank to finance the construction of the plant and all facilities. In return, they get their money back over time with interest, assets or dividends. Some commercial banks could also participate with their philanthropic foundations by providing sunk investments or low-interest funds that could help the community.
- Renewable Energy Plant: financed by the investments of the Trust- LLC, will return to them the potential profits from the sale of surplus electricity to the grid and of course provide energy to the users.

Advantages of CSOP

Among the advantages that a CSOP financing system can offer, as mentioned by Lowitzsch [22], is the possibility of a stable and constant presence of shareholders who are either direct members of the community or participate as external investors. This offers the possibility to have better access to funding when needed and to distribute the income more evenly among the shareholders. Another notable advantage is that the decision-making process can be directly controlled by the members, thus influencing the quality of the service provided. This type of financing structure can also enable the participation of local communities or larger institutions at regional and national level, banks or companies that can provide social, economic and technical assistance and also receive benefits.

Disadvantages of CSOP

According to the book by Lowitzsch [22], the presence of oligopolistic energy companies in the market could be a dangerous aspect if the CSOP becomes popular and visibly more beneficial. Their lobbying potential could put pressure on governments, so that a compromise with them might be necessary to include them as shareholders or investors in the participation process. Some other aspects that could lead to bad feelings among citizens are poor management, which

could significantly increase administrative and operational costs, or inappropriate policies that lead to dissatisfaction among citizens.

1.4.2 Steps to follow

A report by the Italian organisation Confartigianato describes, with a guide, the technical and bureaucratic steps to be followed in establishing the community [27]:

1. Planning: In this first phase, the costs and the expected environmental, economic and social benefits are analysed. In addition, the legal assets and the preliminary composition of the shareholders are determined.

2. Programming: In this second step, the economic resources and internal administrative rules of the community are assessed and defined. It is important to look for possible administrative constraints and their removal, and to find possible people to recruit as members of the community.

3. Design: In this third phase, based on the preliminary analysis carried out in the first step, a detailed hourly assessment of the members' consumption is carried out, as well as the definition and dimensioning of the distributed and concentrated facilities to be installed at the site.

4. Realisation: In this fourth step, all permits required for the realisation of the power plants and their installation are applied for. In addition, the organisation created in the planning phase is legally established here.

5. Management: the community needs administrative management for relations with incoming and outgoing users, financial management that plays with the revenues and manages their redistribution, and technical management for the operation and maintenance of the power plants that optimises them according to the needs of the community are defined and initialized. The latter must install the storage facilities and teach citizens about demand management.

This work aims to fit into the first phase and the third phase, where the analysis of the environmental and economic aspects, as well as the detailed hourly assessment of the consumption and sizing of the plants can be carried out.

CHAPTER 2

Methodology

The second chapter aims to provide an overview of the technological aspects that influence the behaviour and governance of a renewable energy community.

It begins with Section 2.1, which illustrates the priorities in energy distribution among community members in the case of a deficit and in the case of an electricity surplus by users. This is followed by Section 2.2, which analyses the aspect of energy flows by assessing the factors that contribute to household consumption. Then, following that line will be analyzed in Section 2.3the aspect of electricity generation through solar energy is analysed, and finally, in Section 2.4, it is illustrated how this electricity can be stored in batteries to be used when needed.

2.1 The REC hierarchy

After explaining in Chapter 1 what a Renewable Energy Community is, it is necessary to determine which settings should be chosen for this project. It is important to define a hierarchy for managing energy flows in order to maximise self-consumption and incentives while reducing the need for imports from the grid.

The current Italian regulation does not yet prescribe a specific hierarchy or overall target for the community, and the community must decide for itself. Instead, only electricity shared between members is taken into account. Therefore, the strategy chosen for this work was made with the aim of adopting a balanced approach that involves citizens by allowing them to control their demand, without taking into account the economic situation of the members or their individual investments, and is summarised in the Figure 2.1below.

As can be seen, the first priority level is the promotion of self-consumption that is the use of the self-generated electricity and the storage of the surplus in the reservoirs, if available. The second level is the sharing of electricity, that is the exchange between consumers with

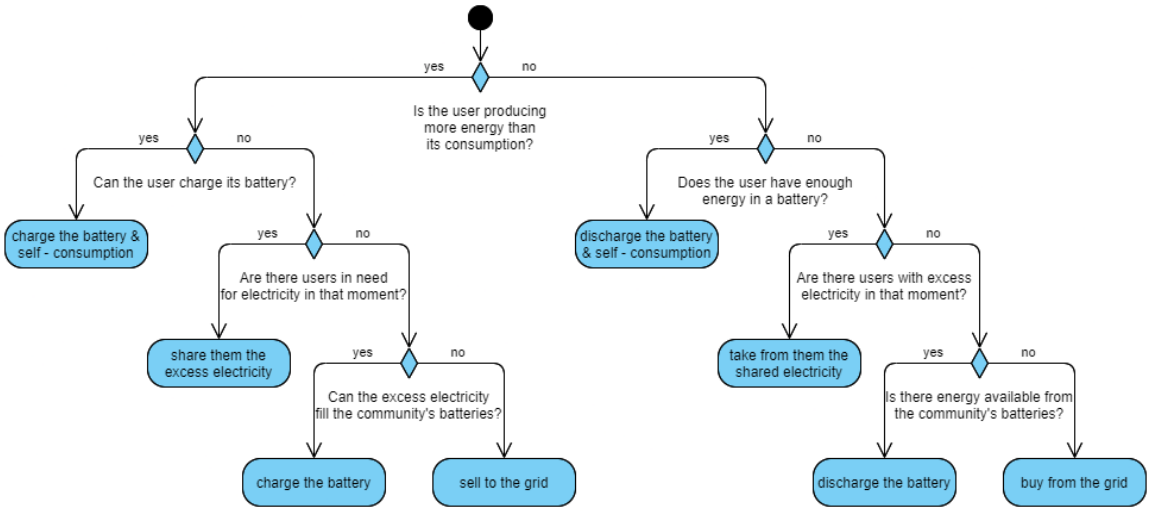

Figure 2.1: Flow chart explaining the hierarchy of the sharing principles.

demand and consumers with surplus. The third level includes the community's centralised and decentralised batteries, which shop the surplus energy when it is not shared or release it for the benefit of users with demand. The last level is the exchange with the grid, buying or selling electricity when the batteries are full or empty.

In this system, self-consumption and sharing take precedence over exchange with the grid, so that the community aims to be self-sufficient. There are cases where elements that are present in this system are missing, such as decentralised plants or batteries. In these cases, the flowchart will not change, but some options will not be available, such as:

- No distributed production: In this case there are no prosumers, self-consumption is not taken into account, but there will be a community plant that produces electricity and shares it, bypassing the first level.
- No distributed batteries: This solution, which could be more beneficial to users from an economic point of view, does not consider the first level in energy storage, with sharing being more important than self-consumption.
- No concentrated production: If there are no common central facilities, electricity is generated by the individual users who consume it themselves and share it among themselves.
- No concentrated batteries: In this case, there could be more exchanges with the grid when demand or production is higher and cannot be stored.
- No grid exchange: This option aims at an off-grid, self-sufficient community that is able to consume what it produces without other energy coming in or going out.

In conclusion, the chosen community will have all the components necessary to give an overview of how each thing will affect the overall balance.

2.2 Consumption profile generation

One of the two main key aspects of the code is the consumption profile, which defines the energy consumption of the house in each timestamp over a whole year. Below is an example of what a consumption profile may look like in the Table 2.1.

User ID	Timestamp	Consumption kWh
6	18/09/2022 18:00	1.3520
6	18/09/2022 18:15	0.9684
6	18/09/2022 18:30	0.6727
6	18/09/2022 18:45	1.1252

Table 2.1: Example, for one hour, of a consumption profile

As shown, it can be quite easy to see and monitor the consumption. The devices available in a household are those that condition the profile according to usage. This can be continuous or random usage.

- Continuous use: This applies to appliances that use a constant amount of energy throughout the day. An example of this is security cameras, sensors or control devices. There are also devices that turn on or off periodically as needed, but with a regular frequency, such as refrigerators or heat pumps.
- Random use: This category is for devices that are not used regularly, but can be used for a short or long time depending on the device.

When simulating consumption, the use of appliances is governed by statistics that are not changed from the original programme in order to preserve the original study behind it.

2.2.1 Solar radiation

As Richardsonpy explains [28], this will have a particular impact on the use of electricity for lighting. Indeed, the likelihood of the lights being switched on depends not only on the number of people in the house, but also on the outdoor lighting that the house emits. So if you add the average solar radiation calculated on a monthly basis to the occupancy profile, the result will be different in the different simulations

The impact on the community is remarkable, both on the consumption and production side. The incoming radiation from the sun is absorbed by the building in the form of thermal energy and light. The former conditions the use of electrical heating devices, if any, such as heat pumps,

air conditioners, heaters and refrigerators. The second aspect concerns the use of lamps and their operating time On the production side, on the other hand, the sun's rays are transformed into energy that is used by members when a photovoltaic system is used to generate electricity, if they opt for this type of technology [29].

2.3 Production profile generation

The other important aspect of the code is the production profile. It provides information on how much energy the plant has produced at a given time. In the following, Table 2.2 will give and example of how does a consumption profile looks like.

User ID	Timestamp	Production kWh
3	23/09/2022 15:00	1.2220
3	23/09/2022 15:15	1.1035
3	23/09/2022 15:30	0.9950
3	23/09/2022 15:45	0.7252

Table 2.2: Example, for one hour, of a production profile

When it comes to electricity generation in RECs, the source of energy should be selected depending on the final purpose and budget of the community

Photovoltaic panels and wind turbines are a common choice because they are cheaper, more scalable and more accessible than a hydroelectric plant or a nuclear power plant.

Specifically, this work uses only photovoltaic panels to generate electricity, which can be installed on rooftops or freestanding on the ground.

In the first case, by installing the panels on the roofs, it is possible to have direct production by the user for self-consumption without using land, but occupying the otherwise unused space on the roofs. The advantages of a stand-alone system, on the other hand, are the greater concentration of production and the ability to orient the panels to capture more sunlight. In addition, the land on which the plant is installed can also be used for agriculture and, in some cases, arable farming.

2.3.1 Photovoltaic power generation

The technology chosen for this project, which simultaneously offers renewable energy generation at moderate cost and the possibility of scaling from large to small systems, is photovoltaics. It is the second largest renewable energy source in Italy in terms of installed capacity and the most widely used for domestic installations [30].

It works thanks to the solar radiation hitting a silicon surface called a photovoltaic cell. This radiation causes the photoelectric effect inside the two layers of the cell, which triggers an electron migration that creates a potential difference, as shown schematically in Figure 2.2.

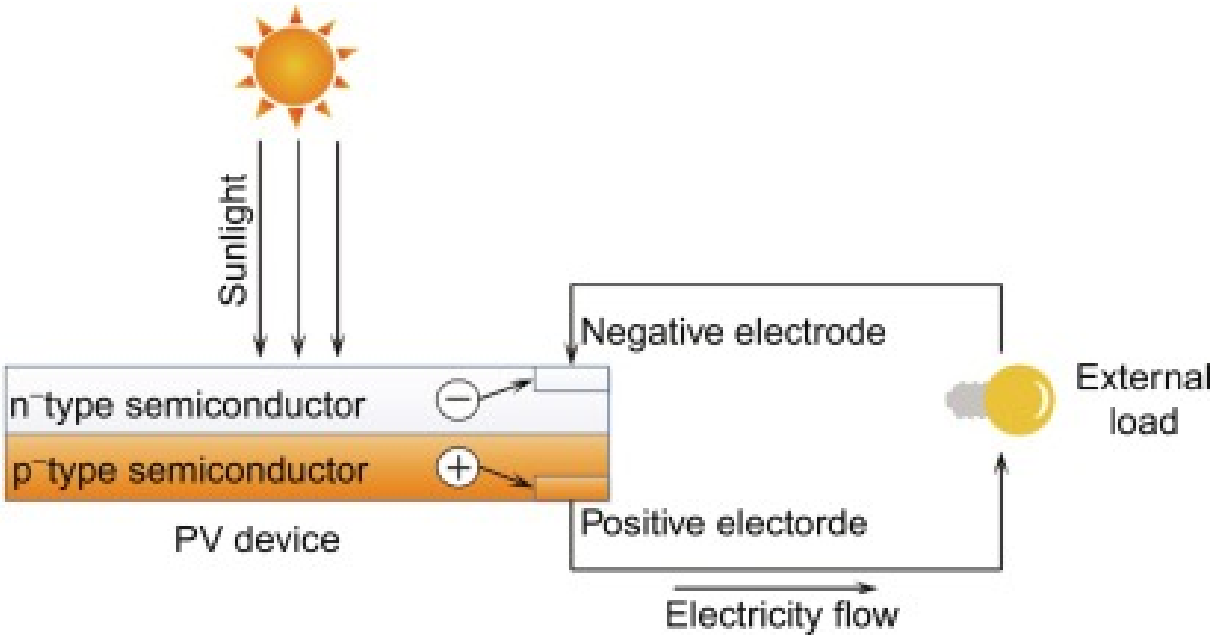

Figure 2.2: Scheme of the photovoltaic cell [31].

The direction from which the sun's rays strike the panel affects the amount of electricity generated. The ideal condition is when the Sun's rays strike the surface of the panel perpendicularly, but due to the Sun's movement this is constantly changing, so production fluctuates throughout the day. However, it is possible to install alignment devices that track the surface to the position of the Sun to maximise production.

The orientation can be done using a helioscope that tracks sunlight and tells a mechanism the position of the Sun to align the panels to maximise production.

This option is used more for ground-mounted systems than for rooftop systems because the slope and roof may not be suitable for alignment, or because the weight of the mechanism moving the panels puts additional stress on the roofs and cannot be tolerated by the structures.

By looking at the way the panel will move it is possible to classify the orienting devices:

1. Fixed: As with the roof panels, these cannot move. Therefore, the tilt and azimuth angles must be optimised to achieve the best possible production during the year or to maximise energy production at certain times of day

2. 1 axis: To improve on the first solution, the possibility of rotating around an axis was introduced to better follow the sunlight and produce more energy during the day. The right choice of axis is also important, because by choosing one or the other it is possible to increase productivity in certain parts of the day. It can rotate to:

    - Vertical axis: In this case it will simply rotate around itself, so the angle of inclination must also be optimised. However, it is not necessary to set an azimuth angle as this will change throughout the day. It can be seen this in Figure 2.3, on the right hand

side.

- Horizontal axis: The orientation of the axis also plays a role. Usually it is the North-South orientation, so that the panel is tilted to the west or east, increasing production in the morning and evening. Another option is the East-West orientation, where the panels face more north or south. This is useful in equatorial zones where the sun changes its vertical position depending on the season and the panels can be better exposed throughout the year. It can be seen in Figure 2.3, on the left hand side.
- Inclined axis: This solution offers a compromise between the vertical and the horizontal axis and utilises more energy during the day and throughout the year. As a rule, the axis in this case is aligned in a North-South direction, so that better production is achieved in the morning and evening.

3. 2 axis: This solution is the most expensive as it requires software to calculate the best combination of the two rotations to achieve the highest possible production. However, with this solution it is possible to produce more energy than with the others, as it is tilted in the best possible way at every moment of the day. Figure 2.4 illustrates some possibilities.

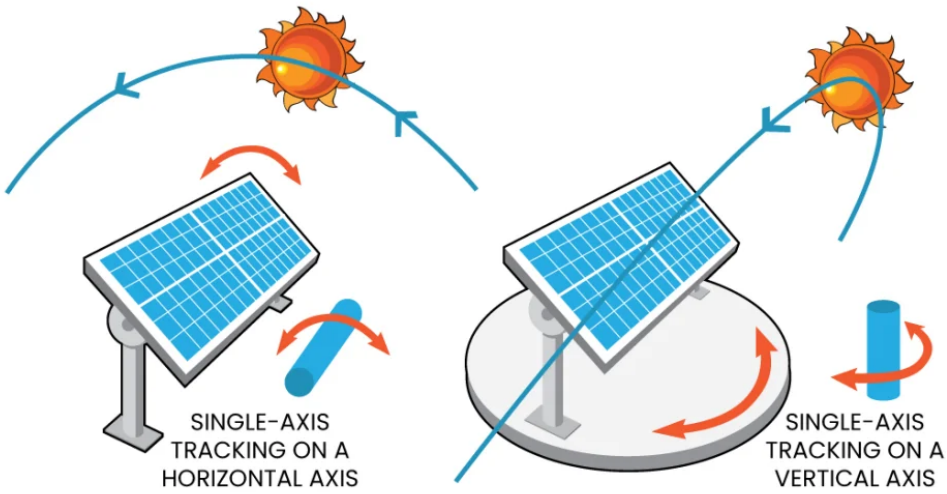

Figure 2.3: Examples of rotation along a single axis [32].

Italian solar potential

Thanks to its geographical location and climate, Italy has good potential for using photovoltaic technology to generate electricity.

The Figure 2.5 shows the annual sum of global irradiation expressed in [kWh/m^2], which is the total irradiation that hits a horizontal surface. By orienting the panels at an optimal angle, the potential increases considerably.

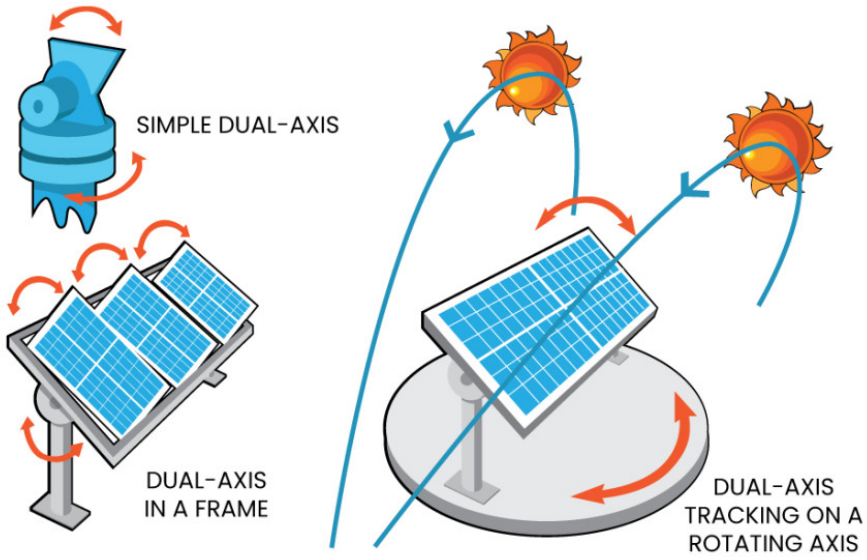

Figure 2.4: Examples of rotation along dual axis [33].

It has been estimated by [35] that potentially up to 200 [TWh] electricity per year could be generated from photovoltaic systems, which would represent 69% of total consumption compared to the 289 [TWh] electricity per year visible in Table 1.2.

2.4 Batteries profiles

One of the greatest challenges in the increased use of renewable energies is storage management due to their low predictability and non-constant generation. Indeed, electricity generation must match consumption to avoid blackouts or unstable parameters.

Storage options depend on the size, the purpose, the time period for which the energy needs to be stored and many other factors that can also vary from site to site.

Some notable example of storages that can be used in a Renewable Energy Community might be:

- Electrochemical devices: Batteries that can be used as storage for a community, providing energy for all members or for each individual. This is also the storage chosen for this work because it has the advantage of advanced state of the art technology, its scalability and its readiness. The disadvantages are the pollution that can occur if it is not properly recycled and the possibility of hazardous material leaking.

- Hydrogen system: A fuel cell is used to generate electricity from gas such as hydrogen or methane, producing water or carbon dioxide and water. When the cell needs to store electricity, it can use the excess electricity to produce hydrogen or methane. The advantage is that the gas can easily be stored in tanks and even sold. The disadvantage is the high

cost and a state of the art that is not as advanced as batteries.

- Water pumped hydroelectricity: This system uses a natural or artificial basin that is filled with water by pumping it using the surplus electricity and emptying it when needed. The advantage is the large capacity available and the possibility of seasonal storage that does not suffer losses over time, but it is limited by the geomorphological nature of the area and the need for large investments.

The storage chosen for this work is the domestic battery, due to its advanced technology state of the art, its scalability and its readiness.

The parameters that define the storages are:

- Capacity, expressed in [Wh] or in [Ah], define the total amount of energy that can be stored in the battery.
- Power, expressed in [W], define the maximum amount of power that can be provided from the battery to the user or from the production plant to the device.
- Depth of Discharge, expressed in [%], and it is the fraction between the amount of energy that is stored in a given moment in the battery, and the total capacity that can potentially be stored in it.

Any battery can be defined by these three parameters, but the difference lies in how the technology chosen affects the equation that determines the behaviour of the battery over time. The reference values used to measure the parameters were those of electrochemical storage devices, commonly referred to as batteries because they are most commonly used. The positive aspect that characterises the storage definition with these three parameters is that it can also be used to describe other technologies such as pumped hydro storage or fuel cells, but in this case it does not provide further details that define these technologies, such as the rotation speed of the turbine or the hydrogen and oxygen flows.

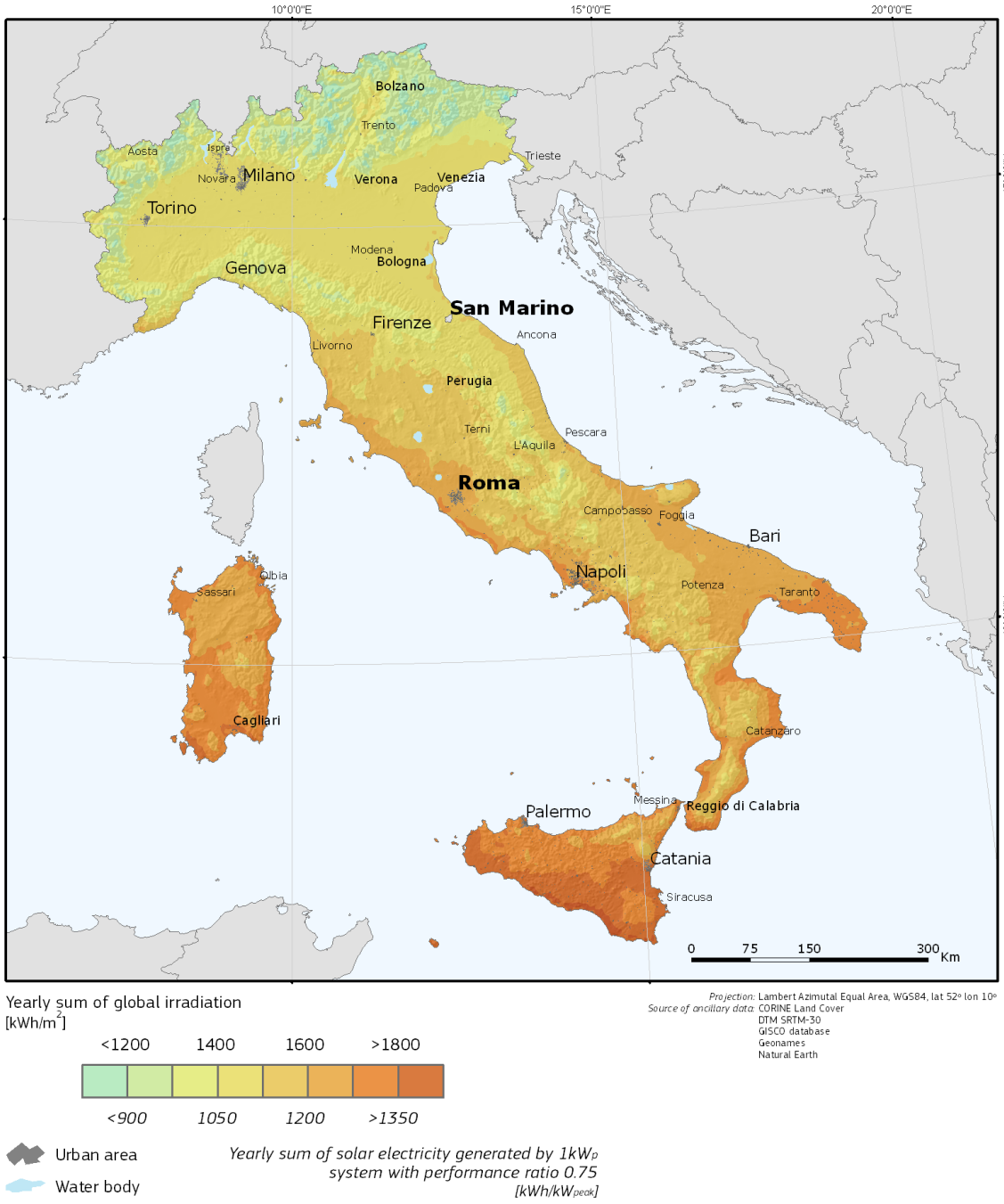

Figure 2.5: Yearly sum of global irradiation, horizontal angle [kWh/m^2] [34]

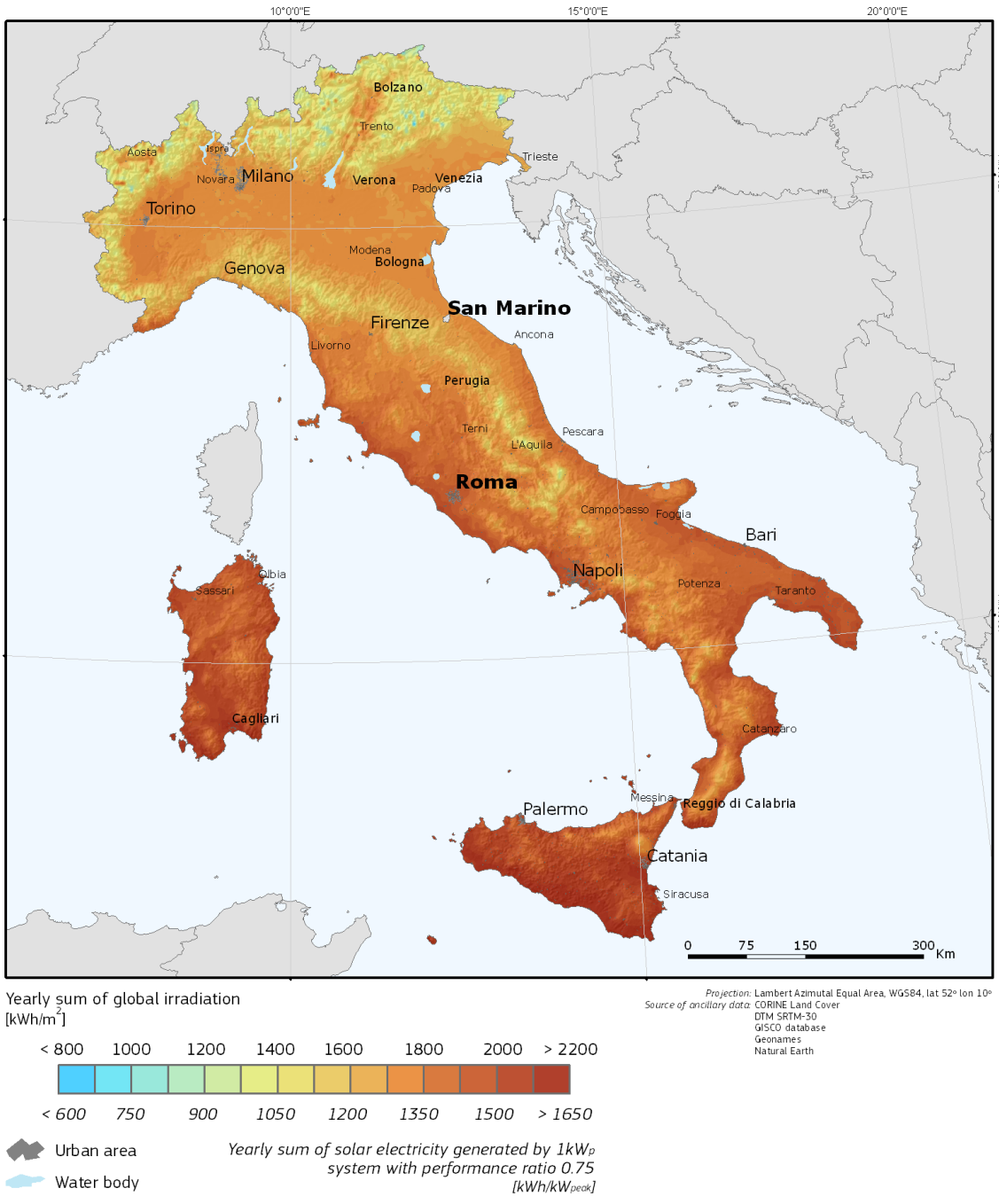

Figure 2.6: Yearly sum of global irradiation, optimized angle [kWh/m^2] [34]

CHAPTER 3

REC Simulation

This chapter is about exhausting and analysing the simulation process phase by phase. For each of these phases, the underlying algorithm is explained, as well as the technical decisions that were made to achieve a reliable final result.

First, Section 3.1 analyses the input conditions that define the four scenarios that are compared to evaluate the performances of the different configurations. Section 3.2 then deals with the generation of the consumption profiles, from the statistical analysis of the devices to the randomisation of the average profile This is followed by Section 3.3, which describes the process of creating the code that calculates the production of the solar panels, and Section 3.4, which describes the behaviour of the storage units and their equations. Finally, Section 3.5 describes the activity diagram that illustrates the phases that characterise the algorithm that simulates the behaviour of the whole community during one year.

In the end, the reader has both the theoretical knowledge provided in Chapter 2 and the technical knowledge to fully understand the process behind this work.

3.1 Input file

To run the simulation, a starting point is required where the parameters are defined characterising the REC and used by the software for the calculations. The end user only has to collect all the necessary information and fill in the file, upload it again and then receive the simulation with which to evaluate the feasibility of the community.

The input file is presented to the user as an Excel file, where each row represents a user and each sheet represents the user's categories. As for the requested information, it can be categorised according to the following criteria:

- Family composition
- Production plant parameters

- Storage characteristics

Some of this information may not be needed depending on the user's category and is summarised in the Table 3.2.

Category	Family composition	Production plant parameters	Storage characteristics
Consumers	Requested	Not requested	Optional
Prosumers	Requested	Requested	Optional
Production Plants	Not requested	Requested	Not requested
Public storage	Not requested	Not requested	Requested

Table 3.1: Summary of the categories and their requested information.

An input file is needed to create the code, and since the composition of the community that will be located on Pantelleria has not yet been officially determined, a preliminary file is in the writing. The parameters are chosen to provide realistic data according to the energy authorities and available technologies. It consists of 5 consumers, 5 prosumers, two production plants and one public storage.

1. Family members: The prosumers and the consumers each consist of five members, so that each family may consist of a different number of persons. Regarding the number of elements that spend more time in the house, this parameter was chosen randomly, but it was ensured that there were two equal cases for the users and the prosumers.

2. Annual electrical consumption: This parameter was chosen following [36], which defines the average consumption of Italian families according to the number of their members. Even if it is not the exact value, it does not deviate by more than 300 [kWh].

3. Production plant: For the prosumers, the solar panels on the roofs were chosen as they represent the best example of decentralised electricity generation. For the power plants, the free-standing plants were chosen instead to test their functionality as well. In terms of rated power, a maximum power of 6 [kW] was chosen for the rooftop plants, while the freestanding plants are 3-6 times larger than that of the building.

4. Storage: The required parameters [37], [38], [39] were selected based on some commercially available batteries. The way each family has been assigned to a battery is completely random.

   As for the collective storage, instead,it was dimensioned as a group of batteries so that its performance and capacity could be easily calculated.

The meaning of each parameter is explained in the previous chapter, while the way they are used is described in the following chapters.

Model	Capacity [kWh]	Instant continuous power [kW]
Tesla Powerwall 2	14	5
LG Chem	10	5
Sonnen Hybrid 9.53	15	3.3

Table 3.2: Summary of the domestic batteries datasheet

The case simulated and analysed by the tool is presented below. The purpose is to have four possible options that could correspond to the real situations proposed in the establishing a Renewable Energy Community process.

The users involved in the creation of the community are 25 families. Their composition, working attitude and annual consumption will not change in the 4 different setups, but there could be some variations in production or storage depending on the case study. In Table 3.3 the families are listed.

User ID	Family members	Non-working members	Working members	Yearly consumption [kWh]
1	1	0	1	1300
2	4	1	3	3500
3	3	1	2	3200
4	5	3	2	5300
5	2	0	2	2500
6	4	2	2	3700
7	2	2	0	2700
8	3	3	0	3300
9	1	1	0	1500
10	5	1	4	5200
11	1	1	0	1400
12	2	1	1	2600
13	3	2	1	3200
14	4	4	0	3800
15	5	2	3	5300
16	1	1	0	1600
17	4	2	2	3800
18	3	2	1	3300
19	5	2	3	5200
20	2	2	0	2800
21	3	1	2	3100
22	5	4	1	5400
23	1	0	1	1400
24	4	0	4	3600
25	2	1	1	2600

Table 3.3: Composition of the 25 families designed for the simulation

The following, is defined for each case the design choices and the composition of the elements that make up the community.

The criterion followed in the design of the storage facilities and the power plants is that the communities should be as similar as possible in terms of the total installed capacity of the batteries and the total nominal power of the photovoltaic plants, in order to allow a reliable comparison between them.

3.1.1 Case 1: Blended Community

Prosumers and Consumers

The community consists of 15 consumers, those with *User ID* from 1 to 15, and 10 consumers, those with *User ID* from 16 to 25. By this division of users it results that for consumers there are three families with 1 member, three families with 2 members, three families with 3 members, etc., while for prosumers there are only two families per category. As far as the number of employed and non-employed members is concerned, the selection was made completely at random. Even if it were possible to include one member for each subdivision of the category, i.e. 20 in total, it was preferred to try it in a more casual way to avoid lengthy simulations.

Storages

In this situation, the storage facilities are owned by the community, as a single storage facility, with the possibility for some users to have their own battery. In this case, the batteries were randomly allocated to avoid over- or under-allocation, ensuring that at least one user per category has a battery.

For what concerns the batteries information, the choice was:

User ID	Instant power [kW]	Capacity [kWh]
2	3.3	15
3	5	14
4	3.3	15
5	5	10
7	5	14
9	5	14
14	3.3	15
16	5	14
17	5	20
20	3.3	15
21	3.3	15
22	5	14
24	3.3	15
27	9.9	60

Table 3.4: Characteristics of the storages present in the community

As defined in the previous subdivision of members, the storages with ID between 1 and

15 are the consumers', while the storages with 16 to 25 are the prosumers'. In total, there are 250 [kWh] of installed capacity among the members, of which 60 [kWh] are for the community battery, 97 [kWh] for the consumers and 93 [kWh] from the prosumers.

The last one, number 27, is the community storage. It has a higher capacity and a higher instantaneous power, as it must be able to provide enough energy to several users when needed.

Production plants

This category lists all the production plants available in the community, both those that come from the prosumers and those that are shared by the members.

User ID	Rated power [kW]	Mounting type	Tracking	Tilt angle [°]	Azimuth [°]
16	2	building	none	30	nan
17	4	building	none	30	nan
18	4	building	none	30	nan
19	5	building	inclined axis n/s	nan	nan
20	3	building	none	nan	nan
21	3	building	none	nan	nan
22	5	building	none	35	10
23	3	building	none	35	5
24	4	free	vertical axis	nan	nan
25	2	building	none	30	0
26	20	free	two axis	nan	nan

Table 3.5: Characteristics of the production plants present in the community

In this case, ID number 26 corresponds to the plant of the community. As far as the type of mounting is concerned, the majority was considered to be fixed rooftop solar, which is most commonly used by families. However, there are some exceptions that allow for other options, such as tracking on a flat roof for user 19 or freestanding in the backyard for user 24. The common installation was designed to be 20 [kW] because it does not require a tender and can be easily financed. In total, 55 [kW] are installed, with 35 [kW] added to the prosumer installations.

3.1.2 Case 2: No-Storage Community

For the second situation, it was decided to assess the impact of storage on sharing and balance with the grid. To achieve this, it was decided to exclude any user from the possibility of using storage. In this way, it is only possible to share energy directly between users and assess its impact and exchange with the grid

Prosumers and Consumers

The structure of users and prosumers is the same as in Section 3.1.1, with 15 consumers and 10 prosumers, with the same consumption and family composition. Their list can be seen in the Table 3.3.

Storages

Not available, neither in the possession of the users nor in the possession of the community. It is expected that this decision will affect the amount of energy shared as well as the self-sufficiency of the community.

Production plants

The production plants, both those owned by individual users and those concentrated, are the same as in Case 1. For reference, the Table 3.5 gives a recap.

3.1.3 Case 3: All-Consumers Community

The concept followed in designing this situation was to simulate a situation that could correspond to the case where users are not willing to install the solar panel and become prosumers, preferring instead to remain simple consumers. In this way, it is also possible to simulate collective self-consumption, where the users are the inhabitants of a building and there is only one production plant, possibly on the roof.

Prosumers and Consumers

Unlike the first two case study, in this situation all 25 members are considered consumers. Their family composition or consumption does not differ, so that a comparison is possible that defines the previously described scenarios as well as possible.

Storages

When investigating the sizing of a storage facility, the options should be explored and simulated to match the instantaneous demand and the total demand at rest. However, this process is for optimising community's costs and balances and is beyond the scope of this paper. Since in Case 1 the total storage capacity was 250 [kWh], it was decided to keep 250 [kWh] as the reference value for the storage system.

User ID	Instant power [kW]	Capacity [kWh]
27	50	250

Table 3.6: Characteristics of the storage used in the community

Production plants

Since this community consists only of consumers, it is not necessary to define the power plants for prosumers as far as the production plant is concerned. On the other hand, the nominal power of the production plant was set at 55 [kW] in order to match the nominal power of Case 1. In this way, the production should be almost the same and it will be possible to compare the two situations. Unlike in this case study, where the production is partly distributed among the prosumers' households, in this case there are three power plants with different characteristics.

User ID	Rated power [kW]	Mounting type	Tracking	Tilt angle [°]	Azimuth [°]
26	20	free	two axis	nan	nan
27	15	free	none	nan	nan
28	15	free	vertical axis	nan	nan

Table 3.7: Characteristics of the production plants present in the community

It was decided to design three smaller plants instead of one in order to simulate the situations where there is not enough space for everyone in a single factory, but also to simulate the possible relocation or more production facilities that could occur in a larger scenario with many more consumers relying on a smaller number of factories.

3.1.4 Case 4: All-Prosumers Community

Prosumers and Consumers

Unlike in the previous Case 3, in this scenario all the users produce their own electricity with their solar panels, so there are no consumers. As for the family composition and consumption, the values have remained the same and can be seen in the Table 3.3.

Storages

To maintain the same total capacity as in Case 1, the 25 members each have a storage with a capacity of 10 [kWh]. The composition is summarised in Table 3.8.

In this setup, there is an overall capacity of 250 [kWh], matching the one for case 1.

User ID	Instant power [kW]	Capacity [kWh]
1	5	10
2	5	10
3	5	10
4	5	10
5	5	10
6	5	10
7	5	10
8	5	10
9	5	10
10	5	10
11	5	10
12	5	10
13	5	10
14	5	10
15	5	10
16	5	10
17	5	10
18	5	10
19	5	10
20	5	10
21	5	10
22	5	10
23	5	10
24	5	10
25	5	10

Table 3.8: Characteristics of the storages present in the community in case 4

Production plants

In this case, the community production facility is not present, but each user produces his own energy. In order to correspond to the nominal power of case 1, a total nominal power of 55 [kW] was chosen. This was done because it was not coherent to have plants with less than one kilowatt or, on the contrary, large plants that alone would have covered the needs of more families. So it was decided to allocate a 1.5 [kW] plant for 1 and 2 person families, while it was increased to 2 [kW] for families with 3 residents. For the 4 and 5 person families it was left at 3 [kW]. In the end, the final composition was as shown in Table 3.9.

Regarding case 1, the decision was made that the building-integrated photovoltaic systems are predominant in order to simulate the most common situation that defines prosumers in a community with their panel and self-consumption.

3.1.5 Recap and comparison

The following Table 3.10, summarises the case studies comparing their aspects and characteristics:

User ID	Rated power [kW]	Mounting type	Tracking	Tilt angle [°]	Azimuth [°]
1	1.5	building	none	30	10
4	3	building	none	35	-5
3	2	building	none	25	0
5	3	building	none	nan	nan
2	1.5	free	horizontal axis n/s	nan	nan
4	3	building	none	nan	nan
2	1.5	building	none	35	nan
3	2	building	none	35	0
1	1.5	building	none	35	0
5	3	building	none	30	10
1	1.5	free	two axis	nan	nan
2	1.5	building	none	30	5
3	2	building	none	30	-5
4	3	building	none	30	-10
5	3	building	none	30	nan
1	1.5	building	none	30	nan
4	3	building	none	30	nan
3	2	building	none	30	nan
5	3	building	inclined axis n/s	nan	nan
2	1.5	building	none	nan	nan
3	2	building	none	nan	nan
5	3	building	none	35	10
1	1.5	building	none	35	5
4	3	free	vertical axis	nan	nan
2	1.5	building	none	30	0

Table 3.9: Characteristics of the production plants present in the community in case 4

In the end, the total consumption remains the same for all 25 users at 81,300 [kWh]. How this is redistributed depends on the case. The total value of the capacity of all batteries remains the same at 250 [kWh], as does the nominal power for electricity generation, which is 55 [kW]

The lack of storage capacity is expected to affect the self-sufficiency of the municipality, as does in the opposite way the high number of prosumers in case 4. In case 3, the presence of only consumers could have a better impact on processing, as fewer calculations need to be made and the grid can be better managed due to the lower number of installations.

3.2 Consumption simulation

When it comes to evaluating electricity consumption, one can find many articles in the literature that analyse this starting from a selected group of people and studying the results of their smart metres. However, these evaluations do not focus on people's behaviour and the electrical devices in their home, but mainly on the pure results that are measured and analysed.

As this work aims to simulate the electricity consumption of a REC, another aspect of the

Parameter	Case 1 Blended	Case 2 No-Storage	Case 3 All-Consumers	Case 4 All-Prosumers
Number of consumers	15	15	25	0
Annual load [kWh]	48,500	48,500	81,300	0
Storage capacity [kWh]	97	0	0	0
Number of prosumers	10	10	0	30
Annual load [kWh]	32,800	32,800	0	81,300
Storage capacity [kWh]	93	0	0	250
Installed power [kW]	35	35	0	55
Plant rated power [kW]	20	20	55	0
Total REC rated power [kW]	55	55	55	55
Storage power [kW]	6.6	0	50	0
Storage capacity [kWh]	60	0	200	0
Total REC capacity [kWh]	250	250	250	250

Table 3.10: Comparison between the design choice for the case study

research that needs to be taken into account is that each user has their own electricity profile that varies throughout the day due to their own unique routine and devices that characterise it. Therefore, instead of having a predefined profile, it is better to have an output that varies consumption and provides users with a realistic pattern without compromising their uniqueness.

Thus, since there is no previous work that is suitable for the specific needs of this program, it was decided to create the user profiles directly in the code. The main constraints to achieve this were:

- Realism: The consumption profile must follow a realistic trend both during the day and during the year.
- Causality: Since each user has a different behaviour, usage and consumption profile, each must be different from the others, but realism must be maintained.
- Proportionality: If users are willing to give their annual consumption as input, their output profile must be scaled proportionally to this input.

The consumption profile is a parameter that concerns consumers and prosumers. For what concerns the other two categories, there may be some consumption during the lifetime, for example to orientate oneself or to monitor the state of the batteries. However, as it varies from model to model, it is also assumed to have a zero value in this work for sake of simplicity. To define a profile, are used three parameters:

1. User ID: This is a unique number assigned individually to each user and serves as a reference number in each data frame
2. Timestamp: This parameter contains the date and hour when the electricity was consumed. The timestamp has a statutory interval of a quarter of an hour and is valid for the whole

year. It indicates the fifteen minutes following the time defined by the timestamp. It therefore begins on 01/01/2015 at 00:00 and ends on 31/12/2015 at 23:15.

The choice of 2015 as the base year was made due to the availability of data from PVGIS, which, as will be explained below, ends in 2016. However, as it is a leap year, it was preferred to choose the previous year in order to have a normal year with 365 days.

In total, the number of timestamps will be:

$$n = 4 * 24 * 365 = 35040 \tag{3.2.1}$$

3. Energy Consumption: expressed in kWh, it indicates the energy consumed in the relative timestamp.

Finally, once it is defined what a consumption profile is and what it looks like, the next step is to create it. The tool chosen, which allows the creation of profiles based on different parameters suitable for the final objective of this work, is the tool Richardsonpy published by [28].

3.2.1 Richardsonpy script

On its dedicated page on github.com [28] it is possible to download the packages created by Richardsonpy that generate the consumption profile with the user-defined boundary condition.

There are two versions of the same program, one as Python code and the other as Excel files. The required inputs are:

1. Time: Since there are differences in the profile depending on the time of the year in which the simulation is requested, the month in which it takes place and the day must be specified. In particular, the day can be either during the week or at the weekend, as there are differences in terms of appliances used, lighting or household occupancy.

2. Appliances: Depending on the number of electrical appliances present in the house, this results in a certain power, which is influenced by their base load or power demand.

3. Occupancy profile: The consumption calculator simulates the energy consumption of households in which one to five people live. However, the presence of people in the house determines the consumption of the previously mentioned electrical appliances. Therefore, it is necessary to know how many people occupy the house during each timestamp.

4. Solar radiation: It is also needed to know the external lighting and radiation. This is necessary because the external light affects the use of the lamps and the radiation affects the heating devices in the house.

The latter three parameters will be defined in detail in the next sections, explaining their principles and the choice to include them in this work.

Appliances

The study on electrical appliances aims to obtain a realistic profile of the electrical appliances that Italian families own, depending on how many residents live in the house. The source of the data is a national survey made by ISTAT in 2013 [40].

In this survey, Italian citizens filled out a survey in which they answered questions on various topics related to electricity consumption, for example:

- Number of people composing the family.
- Dimensions and location of the house.
- Thermal insulation around the building.
- Technology used to generate heat and sanitary water and energy source used by those.
- Electrical appliances in the house
- Amount of energy sources used during the whole year

When filtering the data, some considerations must be made in order to obtain only the data needed for the application area:

1. Location: Even though the project is located in Pantelleria, no distinction has been made between the different possible zones from which the families originate. In this way, it is possible to obtain an average value on a national basis that can be used in any possible place where a new community could be established and is reliable in any case
2. Family filtering: With this survey it is possible to group families according to the number of members, from one to six members. However, since the Richardsonpy tool only allows the calculation of up to five members in the households, the parameters belonging to the 6-person families were merged with the 5-person families. In this way, the more numerous families are also represented and can participate in the formation of the community without distinction.

In the end, the final filtered database was used to determine whether a particular device was owned by the majority and therefore included in the database as being present in the household.

The data collected from this survey is presented in an Excel spreadsheet, with the numbers representing the response. For example, for the question "Is there an electric cooker in the house?" the answer is 1 if the question is answered in the affirmative and 2 if the answer is in the

negative. Other multiple choice questions use a scale from 1 to the number of choices available. A device is considered to be owned by the category if the majority, 50%+1, of families from that category own it.

The Richardsonpy script uses a table in which the household is characterised by the presence of the appliances from a list and needs to know whether the appliance is present or not.

The considerations made to match the requested table to the table available from the survey are:

- Upright freezer: Since it is almost the same appliance device as the *Chest freezer*, it is also evaluated so negatively.
- Answering machine: Even though the spread of mobile phones has reduced the use of answering machines, according to [41] these devices can be compared in terms of consumption with a W-LAN router, which is present in 79% of all households, as can be seen from [42].
- Cassette/ CD player, Clock, Cordless telephone, Hi-Fi: All are considered positive, even if they do not appear in the survey, as they are very widely used.
- Fax: It is considered absent because it is considered outdated technology and therefore no longer widespread.
- Printer: even if it is not directly listed in the survey, it was considered as an item of *personal* computer and its results are copied for this item.
- TV 1, TV 2, TV 3: These devices were calculated indirectly, as the questionnaire asked citizens about the number of TV sets in the house. So to determine the number, the arithmetic mean was calculated and the number of televisions was determined.
- VCR/DVD, TV receiver box: These items were considered positive because nowadays they are usually included in the TV set or are present anyway.
- DESWH, E- INST, Electric shower, storage heaters, Other electric space heating: all these devices are considered negative because, looking at the data, the majority of families do not use electricity to heat water or air

In the following Table 3.13, the percentage of prevalence of the devices in the survey, calculated by counting the positive responses in relation to the total number of responses.

As far as the number of television sets is concerned, the survey did not determine the presence of the individual sets, but the average number of television sets. Based on this figure, it was possible to determine the presence of televisions as required by the submission:

The results are listed in the Table 3.13, which defines the presence or absence of the appliance depending on the number of people in the house.

Device	1 person [%]	2 people [%]	3 people [%]	4 people [%]	5 people [%]
Chest freezer	19.1	32.9	37.4	37.9	45.3
Fridge freezer	93.3	93.6	94.1	95.4	97.3
Refrigerator	98.9	99.9	99.9	100.0	100.0
Iron	92.7	98.5	98.6	99.3	98.5
Vacuum	65.5	76.3	81.2	85.7	84.0
Personal Computer	22.6	40.2	72.3	84.9	82.1
Hob	4.4	4.4	3.8	4.4	6.1
Oven	63.1	73.3	79.0	83.4	81.5
Microwave	31.2	39.3	48.2	50.7	52.1
Kettle	15.3	18.8	24.5	27.7	26.3
Small cooking group	17.1	26.2	33.3	39.6	37.7
Dish washer	24.7	40.8	50.1	55.4	56.2
Tumble dryer	0.7	1.6	4.3	6.8	7.2
Washing machine	95.3	98.1	98.6	99.1	99.3
Washer dryer	2.0	2.2	2.8	2.9	4.1

Table 3.11: Percentage of penetration of the electrical devices [40].

	1 person	2 people	3 people	4 people	5 people
TV Owned	1.53	1.81	2.08	2.21	2.27

Table 3.12: Mean number of television per family [40].

Occupancy Profiles

The importance of the occupancy profile generator, as explained earlier, is that it provides the code with the number of people who are in the dwelling at the simulated time. It can be described as a table that basically counts the number of active occupants in the household at a given time.

Even though the code refers to "active" as a person who is in the house and interacting with the appliances, this term can lead to misunderstandings when translated into Italian. This is because this definition is also used for people who are frequently outdoors and can therefore cause confusion among users.

To overcome this linguistic issue, the definitions were changed in:

1. Working inhabitants: This category includes family members who are out of the house most of the day. In the simulation, profiles were selected that were not in the house both in the morning and in the afternoon to simulate a person going to work. Of course, this does not mean that non-residents do not fit into this category, but if they are out of the house during the day, they do fit into this category. An example would be a student who goes to school in the morning and goes out in the evening.

2. Non-Working inhabitants: This category includes the other people, i.e. those who spend most of the day at home. Just as before, the name of the category does not mean that all

Device	1 person	2 people	3 people	4 people	5 people
Chest freezer	No	No	No	No	No
Fridge freezer	Yes	Yes	Yes	Yes	Yes
Refrigerator	Yes	Yes	Yes	Yes	Yes
Upright freezer	No	No	No	No	No
Answer machine	Yes	Yes	Yes	Yes	Yes
Cassette/CD player	Yes	Yes	Yes	Yes	Yes
Clock	Yes	Yes	Yes	Yes	Yes
Cordless telephone	Yes	Yes	Yes	Yes	Yes
Hi-Fi	Yes	Yes	Yes	Yes	Yes
Iron	Yes	Yes	Yes	Yes	Yes
Vacuum	Yes	Yes	Yes	Yes	Yes
Fax	No	No	No	No	No
Personal computer	No	No	Yes	Yes	Yes
Printer	No	No	Yes	Yes	Yes
TV 1	Yes	Yes	Yes	Yes	Yes
TV 2	Yes	Yes	Yes	Yes	Yes
TV 3	No	No	No	No	No
VCR/DVD	Yes	Yes	Yes	Yes	Yes
TV Receiver box	Yes	Yes	Yes	Yes	Yes
Cooking hob	No	No	No	No	No
Oven	Yes	Yes	Yes	Yes	Yes
Microwave	No	No	Yes	Yes	Yes
Kettle	No	No	No	No	No
Small cooking group	No	No	No	No	No
Dish washer	No	No	Yes	Yes	Yes
Tumble dryer	No	No	No	No	No
Washing machine	Yes	Yes	Yes	Yes	Yes
Washer dryer	No	No	No	No	No
DESWH	No	No	No	No	No
E-INST	No	No	No	No	No
Electric shower	No	No	No	No	No
Storage heaters	No	No	No	No	No
Other electric space heating	No	No	No	No	No

Table 3.13: Summary of the domestic batteries datasheet

people who do not work fall into this category, e.g. those who work from home and live in the house.

Ultimately, there are 20 possible profiles characterised by the number of inhabitants and the number of those described as "working". In the Table 3.14 all profiles are listed with their identification code.

Profiling can be done in the Excel spreadsheet by simply entering the number of people in the house and choosing between weekday and weekend. When you run the simulation, a profile is created that can be used in the further steps. However, there are some issues that need to be addressed to ensure that the output is reliable and realistic.

As mentioned earlier, users are characterised both by the number of people living in the

Code	Family members	Non-Working	Working
0	1	1	0
1	1	0	1
2	2	2	0
3	2	1	1
4	2	0	2
5	3	3	0
6	3	2	1
7	3	1	2
8	3	0	3
9	4	4	0
10	4	3	1
11	4	2	2
12	4	1	3
13	4	0	4
14	5	5	0
15	5	4	1
16	5	3	2
17	5	2	3
18	5	1	4
19	5	0	5

Table 3.14: List of the possible profiles available

house and by those who are frequently at home. This tool, on the other hand, is only defined by the first one, so there may be different outputs that do not match the desired user. To solve this problem, the creation of the profile was done for each individual user and then summed.

The system works simply and can be summarized in three steps:

1. Check how the user is defined by the number of inhabitants and working members.

2. Generate a profile representing the first family member and count the hours they are at home. If it does not match their definition, the random generation is repeated until it gives a suitable result. On the other hand, if it is good, the results are saved and the next user is generated. In order to define the working and non-working users correctly, the hours at home were counted: If the household is occupied for more than 10 hours, it is considered as a non-working member, in the other case it is the other way round.

3. Once that all the profiles are created, the sum is made timestamp by timestamp to obtain the final profile

These profiles can already be used in code if there is only one user in the household. To create profiles representing the more numerous families, a single profile was simply created to match the definition of each user, working or not, and summed timestamp by timestamp to get the profile for the day.

An example can be seen in Table 3.15, in which is shown the occupation profile of each single

user as well as the final one. A value equal to 1 means that the user is at home in that moment, while a 0 means that it is not at home in that timestamp.

Time	User 1	User 2	User 3	User 5	Total
00:00	1	1	1	1	4
00:10	1	1	1	1	4
00:20	1	1	1	1	4
00:30	1	1	0	1	3
00:40	0	0	0	1	1
– –	-	-	-	-	-
23:20	0	0	1	0	1
23:30	0	1	1	1	3
23:40	0	1	1	1	3
23:50	0	1	1	1	4
Total individual hours at home	15	14:40	13:30	6:50	–

Table 3.15: Detail of the creation of the profile

From this table it can also be seen that there is a large difference in time spent at home, which confirms the threshold of 10 hours at home as the minimum to be classified in the "working" category. Thus, this profile represents a family in which only 1 member is usually not at home, while the other three are.

In the following Figure 3.1, Figure3.2, Figure3.3, Figure3.4, Figure3.5 located in the end of this chapter, all the occupancy profiles for the 20 families profiles are showed.

Solar Radiation

Among the factors that have an influence on the use of the devices there is solar radiation.

The solar radiation parameters can be found on the PVGIS website. The chosen location from which to extract the radiation data was, in this case, the island of Pantelleria. Although the code would work for other locations, a code that uses radiation as an input when calculating each user's consumption would certainly be suitable for a more accurate program, but in this case this was considered too in-depth and beyond the scope of the work. The second aspect that is affected by solar radiation is the heating demand. However, as mentioned in the section on household appliances, heating sources other than electric were used on average at the time of the survey, so electricity consumption is not affected by this.

When downloading the profile, the solar radiation is given with an interval of one hour. To comply with the code requiring an interval of one minute between measurements, the values were linearly interpolated between the individual hours. The hourly data of the global radiation obtained by PVGIS monthly are visible in Table **??**.

As can be seen, global radiation has higher values in the summer months and lasts longer. This affects the production of the solar modules, which perform best during these months.

Hour	JAN	FEB	MAR	APR	MAY	JUN	JUL	AUG	SEP	OCT	NOV	DEC
00:00	0	0	0	0	0	0	0	0	0	0	0	0
01:00	0	0	0	0	0	0	0	0	0	0	0	0
02:00	0	0	0	0	0	0	0	0	0	0	0	0
03:00	0	0	0	0	0	0	0	0	0	0	0	0
04:00	0	0	0	0	0	0	0	0	0	0	0	0
05:00	0	0	0	0	0	0	0	0	0	0	0	0
06:00	0	0	0	22	56	65	58	34	7	0	0	0
07:00	0	3	83	159	210	214	206	187	159	114	18	0
08:00	144	191	274	350	397	402	409	398	357	316	246	159
09:00	331	376	472	537	589	588	612	610	555	505	427	344
10:00	464	536	635	692	745	739	775	769	705	651	564	474
11:00	568	651	743	786	832	851	890	894	800	739	656	560
12:00	610	706	793	844	868	893	939	942	835	751	671	592
13:00	576	656	748	812	840	867	914	919	819	696	629	553
14:00	493	597	680	729	733	771	825	823	704	603	525	459
15:00	382	463	543	578	578	626	677	660	541	441	370	327
16:00	217	293	366	391	404	446	484	458	360	255	178	163
17:00	11	109	173	196	214	246	269	241	156	50	0	0
18:00	0	0	6	33	53	72	80	56	8	0	0	0
19:00	0	0	0	0	1	13	12	0	0	0	0	0
20:00	0	0	0	0	0	0	0	0	0	0	0	0
21:00	0	0	0	0	0	0	0	0	0	0	0	0
22:00	0	0	0	0	0	0	0	0	0	0	0	0
23:00	0	0	0	0	0	0	0	0	0	0	0	0

Table 3.16: Hourly global radiation [34]. All data are expressed in $[W/m^2]$

3.2.2 Post processing and randomization

Once an occupancy profile and a device profile were available for each user, a consumption profile was created for each month, both for weekdays and weekends. Basically, the algorithm simulates the profiling 10 times. After that, the average is calculated and then used as input for the second randomisation phase.

The steps to be followed are listed below:

1. Firstly, must be inserted the parameters determining the users and the time, so:
   - Number of users in the house
   - Month of the year
   - Weekday or Weekends
   - Occupation profile
   - Appliances profile
2. Run the simulation
3. Save the produced dataframe in another file

4. Repeat step 2 and 3 for nine more times
5. Calculate the average consumption from the 10 simulations
6. Insert the average consumption in the input file, which will collect all the average in the different cases

In total, there will be 480 dataframes, two per each month per each user.

The input file is used in the next phase to create each user's individual profile. Since the input file contains only different daily profiles, the first task of the code is to create an annual data frame from 1 January 2015, the reference year, at 00:00 to 31 December 2015, at 23:45. Then the production profiles are converted from a 10-minute time span from the input file to a 15-minute time span to fit the annual time frame.

This is done using a line-by-line iteration that copies the load values from the input file to the new data frame at each timestamp, corresponding to the respective day and month. The end result is a data frame that contains the annual consumption day by day. At this point, however, the profiles will change monthly rather than daily. In order to obtain a unique variation of the consumption, its value is randomly determined by multiplying the original value by a randomly generated value called *Delta* in the equation 3.2.2.

$$Delta = Base\ variation + Non\ working\ people * Individual\ variation \tag{3.2.2}$$

With the two parameters influencing *Delta* are:

- Base variation: this value represents the minimum margin guaranteed to change the load when no one is using an appliance in the house and represents the random consumption of the base load.
- Individual variation: this value, multiplied by the number of users in the house, increases the span from which you select the number to get a higher number of possible results representing the variations caused by more people in the house.

These two values can be between 0 and 100 and represent a percentage. Both the parameter *Base variation* and the parameter *Individual variation* parameter are set to 10, which means that without any activity in the house, the baseload can vary at most between $\pm$ 10% and also each user will affect it at most by $\pm$ 10%. The positive aspect of this method is that the load can vary greatly in all ranges between - *Delta* and + *Delta*, which meets the requirements for a random but reliable profile.

3.3 Production simulation

For photovoltaic production, which characterises both prosumers and power plants, the [34] website is used to obtain the data.

3.3.1 PVGIS

Instead of directly using the PVGIS website and calculating the production profiles one by one, which would take time and effort and is not easy for less expert user, the API (Application Program Interface) is used, which gives the same result but in an automatic way, communicating directly from the code to the website.

The code has been set up to receive a data frame for each user who produces, whether a prosumer or a producer, indicating the energy produced in each quarter of an hour for the whole year. The information required when filling in the input file is:

- Coordinates: Latitude and Longitudes, expressed in degrees [°] with positive values indicating North and East
- Peak Power Production: Expressed in [kWh], indicating the nominal power of the whole plant. It does not make a distinction on the number of panels or their single nominal power
- Mounting Type: The place where it is installed. It can be:
  - Free: it indicates the free-standing devices installed directly on the ground.
  - Building: it indicates the panels installed on the rooftops.
- Tracking: the panels could have the option to rotate to follow the sun and get better Sun exposure. The choice must be made to maximise production while keeping costs low, depending on the geographical location and exposure. You can choose between:
  - None: it must be selected if the panel is fixed and cannot rotate.
  - Horizontal axis N/S: in this case, there is a mechanism that allows the rotation of the structure on the horizontal axis that is laying in the North/South direction. In this way, it is possible to face better Eastward and Westward
  - Two Axis: this possibility permits to rotate in the two main directions, following the sun in the best way possible and maximising its exposition.
  - Vertical Axis: when the panel can rotate on its vertical axis, this option must be chosen. In this setup, the panel will track the sunlight maximising the exposition during the morning and during the evening.
  - Inclined Axis N/S: the concept is the same as the Horizontal Axis N/S, but in this case the rotation axis is not horizontal but inclined to a certain degree.

- Tilt Angle: This value indicates the angle between the horizontal plane and the panel plane. A higher tilt angle maximises exposure when the Sun's position is lower in the morning and evening, while a lower value increases productivity during midday.
- Azimuth: This value specifies the angle between the normal direction of the panel perpendicular to the plane and the South direction. By changing this parameter, it is possible to maximize the production in the morning or in the evening.

All the above parameters must be filled in when the file is compiled. However, for the last two, there is a possibility that some of them are not yet known. In this case, the PVGIS software performs the optimisation of the angles, depending on the tracking parameter.

1. If the panel is fixed, by inserting NaN in Tilt Angle, it will optimize it, while if both are NaN, it will optimize both. If the Tilt Angle is defined but the Azimuth one is not, it is not coded to optimize that one.
2. If it is Horizontal Axis N/S, the Azimuth optimization won't be necessary, but only the Tilt Angle one
3. If it is Two Axis, the optimization will be done by the tracking type itself, so it won't be necessary to define or optimize the Tilt Angle or the Azimuth.
4. If it is Vertical Axis, the Azimuth optimization won't be necessary, but only the Tilt Angle one
5. If it is Inclined Axis N/S, the Azimuth optimization won't be necessary, but only the Tilt Angle one

This valuable opportunity offers the chance to better understand and optimise production facilities to generate more electricity and add value to the community over the years.

The parameters previously mentioned are those required to run the PVGIS programme and will change depending on the community. There are also other variables that are needed for the software but cannot be changed because they are either difficult to know or have been fixed during the coding process to get consistent results and fixed values that are consistent with the other phases of the process, such as:

- Base Year: 2015. This year was chosen because of two reasons: firstly, because the irradiation database is available up to 2016; secondly, because 2016 was a leap year and the simulation was chosen to be done in a standard year of 365 days.
- PVCalculation=1. This is the way the PVGIS API is told to calculate the productivity as well as the irradiation. If it was set to zero, it would only calculate the irradiation.

- System Loss = 0.14. This value is mandatory for the calculation of the productivity, but since it could be difficult to be found by the user, it was chosen to keep the default number already set in the browser version of PVGIS.
- Optimalinclination: if this parameter is set to 1, it will calculate and optimize the Tilt Angle parameter. It won't be inserted in the input file by the user because its value will be changed by the code depending on the parameters inserted, as explained before.
- Optimalangles: It works like the previous one, but in this case if set by the code to 1 it will calculate and optimize both the Tilt Angle and the Azimuth angle

Once these parameters are defined, it is possible to calculate productivity for a year by creating a link that is sent to the PVGIS servers and provides the results, as explained in the next subsection.

3.3.2 Productivity Calculation

Once all the parameters are known, the function works to determine the productivity of the plant. The API communicates with the website via a link that provides all the information to the website:

ttps://re.jrc.ec.europa.eu/api/seriescalc?lat=36.82913&lon=11.93668&pvcalculation=1&peakpower=5&loss=14&trackingtype=1&optimalinclination=1&mountingtype=free&startyear=2015&endyear=2015.

The parameters are inserted in the link in this way:

1. *https://re.jrc.ec.europa.eu/api/seriescalc*: this first part is fixed and is the basic address plus the information to give the result of as a series of hourly measurements.
2. *lat=36.82913&lon=11.93668*: it indicates the coordinates, in degrees [°].
3. *pvcalculation=1*: without this command, the output would only be the solar radiation without the productivity calculation.
4. *peakpower=5&loss=14*: with these parameters, are imposed the nominal power, in [kW], and the losses, in percentage.
5. *trackingtype=1*: the tracking type is selected in this line, but instead of writing the name of the orientation technology, it is coded by a number:
    - 0 = None
    - 1 = Horizontal axis N/S
    - 2 = Two axis
    - 3 = Vertical axis

- 5 = Inclined axis N/S

6. *optimalinclination=1&optimalangles=1*: if these parameters are set to one, the server will also give the optimization as explained before.

7. *mountingtype=free*: to indicate the location of the plant, with *free* meaning for stand-alone and *building* representing the rooftop solar.

8. *startyear=2015&endyear=2015*: this is used to select the timespan, in years, from which extract the data.

After the program has converted the data into a raw file that also contains other information, the second part of the programme cleans the file so that only the data needed for the calculations and the creation of the data frame of the installation are retained.

[[inserire qui o come allegato uno dei raw file?]]

3.4 Batteries simulation

Finally, for the batteries, it was decided not to create a separate function, but to integrate it into the main structure of the code.

This decision was necessary because their behaviour during the year is not constant and predetermined, but depends on the conditions at the beginning of the timestamp and on the production and consumption in each specific period.

The input provided are:

1. Instant power: listed as *storage power kW*

2. Capacity: listed as *storage energy kWh*

3. Depth of Discharge: listed as *storage dod*

The first two parameters, instantaneous power and capacity, remain constant for each user during the year. It has been chosen to not consider equipment decay, as it is defined and changes according to technology and use. This would have required another detailed study outside the scope of this paper

From the instantaneous power, it is easy to calculate the amount of energy that is used in the calculations and can be used in the timestamp. It is calculated as:

$$E_{inst}[kWh] = P_{max}[kW] * \frac{t[min]}{60[min]} \tag{3.4.1}$$

Where:

- E is the previously mentioned deployable energy

- P is the instant power from the input file
- t is the sampling time, in this case constant at 15 minutes

Commercial equipment does have the ability to charge and discharge at different rated power, but for ease of assessment this was considered to be the same in both cases.

Then the parameters to be considered for the calculation change depending on whether the battery is charging or discharging.

Discharge phase

In the discharge process the energy that can be provided will depend on the depth of discharge, and can be described by equation 3.4.2.

$$E_{available} = P_{max} * DOD \tag{3.4.2}$$

To avoid an additional calculation for each user in each time frame, the code simply copies the previous value of available energy instead of copying the parameter DOD and then calculating it, giving the same result as before but slowing down the simulation time.

The available energy is then compared to the instant power. If:

$$E_{available} \geq E_{inst} \tag{3.4.3}$$

it means that all the instant power can be provided, while in the contrary it can only be discharged the available energy.

Charge phase

In this case it is not considered the energy available but the space that can be occupied by surplus energy. In this case it can be evaluated as follows:

$$E_{space} = P_{max} * (1 - DOD) = P_{max} - E_{available} \tag{3.4.4}$$

In this case, the space is compared to the instantaneous power, and if

$$E_{space} \geq E_{inst} \tag{3.4.5}$$

This means that the entire instantaneous power from the surplus energy can be stored in this timestamp, while on the contrary only the space could be filled

3.5 Activity Diagram

The creation and coding of complex programmes is a lengthy process that requires much study, not only of the subject itself, but also of the structure and stages that must be followed to obtain the final result. In many cases, the idea behind a programme is easy to imagine, but the implementation and optimisation can be overly complicated and require a defined algorithm that is not easy to understand if not properly visualised and described.

In computer science, there is a useful tool that is often used when defining a program: the activity diagram. This is basically a flowchart that shows each step of the program from a dynamic perspective, activity by activity. Although it looks like a flowchart, it differs from it in that it describes actions and sequences and how they relate to each other. For example, they can be executed in parallel instead of one after the other, conditioned by a logical comparison or by other actions.

Each block used in the diagram has its own meaning and contains a description of its activity. In particular, the most common shapes are:

- Circular: it is used to describe the initial and the final points, giving for example starting parameters or showing the results.
- Squared: these blocks represent the action required from the code. Inside there can be a comment on the action and its formula.
- Rhomboid: when there is a decision to be done by the code using a Boolean comparison by two parameters, this block indicates two possibilities, in case the Boolean value is true or false.

In the end, when the activity diagram is ready, it is easier to work on the code by remembering its structure and finally conveying the idea behind it. Since this program is a complex code, the activity diagram is quite large and does not fit on a single page. In this work, the solution for presentation is to show and explain sections of the code and then move on to the next part, highlighting the connections between them.

In the Figure 3.6 the colours used are explained to help you understand the code:
In which:

- General parameter: these blocks represent setting or general values not referred to users or to the community.
- User parameter: these blocks are indicating the calculation of parameters for each user, so will be different for everyone.
- REC parameter: these blocks are used to highlight the parameters calculated for the whole community, usually as the result of sum of user parameter.

- Storage parameter: these blocks represent the calculation of parameters referred to the community's storage

Now, it is possible to move forward to the analysis of the activity diagram.

Part 1: Self-consumption calculation

The first calculation are about the possibility to recharge the battery, if present, or to use it to overcome the electricity deficit, and it is visible in Figure 3.7.

Where:

- *t_i* is the index, which is the timestamp and is iteratively changed at the end of the cycle.
- *previous_energy_in_battery_t_kWh* is the amount of energy that was available in the battery at the time the timestamp started. It is defined as:

$$previous\_energy\_in\_battery\_t\_kWh = storage\_energy\_kWh * previous\_dod \tag{3.5.1}$$

  in the event that the timestamp is the first and must be calculated using the values from the input file:

  - *storage_energy_kWh*, which is the capacity of the battery considered constant during the simulation.
  - *previous_dod*, this is the depth of discharge of the battery at the time when the previous timestamp ended.

  If it is not the first timestamp, this value is updated by copying the previous one, as explained at the end of the cycle

- *energy_in_battery_t_kWh* is the energy remaining in the battery at the end of the timestamp. It is set equal to *previous_energy_in_battery_t_kWh* to be able to change it without changing the initial value and is updated during the time frame.
- *previous_space_in_battery_t_kWh* is the remaining storage in the battery at the beginning of the timestamp and is calculated as follows:

$$previous\_space\_in\_battery\_t\_kWh = storage\_energy\_kWh * energy\_in\_battery\_t\_kWh \tag{3.5.2}$$

- *raw_user_balance* is the energy balance of each user without taking into account storage or redistribution, by simply calculating the difference between production and load. If it is positive, it means that the user has a surplus, and if it is negative, it means that he is in deficit.

- *max_energy_chargeable_t_kWh* this parameter defines the maximum amount of energy that can be charged from the renewable source in the battery in a single time stamp, as explained in paragraph 3.4.
- *max_energy_dischargeable_t_kWh* this parameter defines the maximum amount of energy that can be discharged from the battery to the household in a single time stamp, as explained in paragraph 3.4.
- *flag* is a numerical parameter used exclusively to classify the situation specifically for a single user in a given timestamp. Its value can be a value between 1 and 4, assigned according to the case:
  - *flag = 1* is assigned when:

$$raw\_user\_balance\_t\_kWh \geq 0 \tag{3.5.3}$$

    and

$$max\_energy\_chargeable\_t\_kWh \geq raw\_user\_balance\_t\_kWh \tag{3.5.4}$$

    That represent the self-consuming user that has excess space in the battery and stores that in it without sharing to others.
  - *flag = 2* is assigned when:

$$raw\_user\_balance\_t\_kWh \geq 0 \tag{3.5.5}$$

    and

$$max\_energy\_chargeable\_t\_kWh < raw\_user\_balance\_t\_kWh \tag{3.5.6}$$

    This situation occurs either when the battery is almost full and the surplus electricity cannot be completely stored in the battery, or when there is no battery but the prosumer generates his own energy.
  - *flag = 3* is assigned when:

$$raw\_user\_balance\_t\_kWh < 0 \tag{3.5.7}$$

    and

$$max\_energy\_dischargeable\_t\_kWh \geq |raw\_user\_balance\_t\_kWh| \tag{3.5.8}$$

    In this case, this flag is assigned when a user consumes energy from his battery itself and does not need external intervention.

– *flag* = *4* is assigned when:

$$raw\_user\_balance\_t\_kWh < 0 \tag{3.5.9}$$

and

$$max\_energy\_dischargeable\_t\_kWh < |raw\_user\_balance\_t\_kWh| \tag{3.5.10}$$

This last case occurs either when a user is using energy from his own battery but it is not sufficient to satisfy all his needs, or when the user has no battery and buys energy from others.

- *battery_balance_t_kWh* is the current flow as seen by the battery. If it is positive, it means that the battery is being charged, and if it is negative, it means discharge. Its value depends in particular on the value of the flag:
  – if *flag* = *1*:

$$battery\_balance\_t\_kWh = raw\_user\_balance\_t\_kWh \tag{3.5.11}$$

because in this situation all the surplus energy is going to recharge the battery.
  – if *flag* = *2*:

$$battery\_balance\_t\_kWh = max\_energy\_chargeable\_t\_kWh \tag{3.5.12}$$

because only a part of the surplus energy can be used to recharge the battery.
  – if *flag* = *3*

$$battery\_balance\_t\_kWh = raw\_user\_balance\_t\_kWh \tag{3.5.13}$$

because all the energy needed is obtained by discharging the battery.
  – if *flag* = *4*

$$battery\_balance\_t\_kWh = max\_energy\_dischargeable\_t\_kWh \tag{3.5.14}$$

because only the amount of energy that can be discharged will be actually used from the user.

- *user_and_battery_balance_t_kWh* is the flow of electricity from the household's point of view, taking into account both the user's and the battery's influence. As with the balance of the battery, the value depends on the flag. In particular:
  – if *flag* = *1*:

$$user\_and\_battery\_balance\_t\_kWh = 0 \tag{3.5.15}$$

because in this situation there is self-consumption storing the energy, therefore it is not exchanged with outside.

– if *flag* = *2*:

$$user\_and\_battery\_balance\_t\_kWh = raw\_user\_balance\_t\_kWh - max\_energy\_chargeable\_t\_kWh \tag{3.5.16}$$

because it represents the part of energy that cannot be stored because the battery is full or because it is not present.

– if *flag* = *3*

$$user\_and\_battery\_balance\_t\_kWh = 0 \tag{3.5.17}$$

because all the energy needed is obtained from self-consumption and does not need energy from outside.

– if *flag* = *4*

$$user\_and\_battery\_balance\_t\_kWh = raw\_user\_balance\_t\_kWh + max\_energy\_dischargeable\_t\_kWh \tag{3.5.18}$$

because it represents the part of deficit that cannot be supplied using the battery.

- *space_in_battery_t_kWh* represents the amount of space that is available in the battery, and it is obtaining by subtracting the battery balance to the previous value, representing the variation of the charge or discharge, depending on the case.

Part 2: Introduction of the community's battery

The second part is about the community's battery and how it is recharged using the excess electricity, and it is visible in Figure 3.8.

In particular, in this part it is not only about the user with calculation to be done user by user, but instead it is referred to the community's battery and the whole REC. The parameters involved are:

- *rec_raw_balance_t_kWh* that is the difference, timeframe by timeframe, of all the production and all the consumption.
- *users_and_batteries_energy_surplus_t_kWh* which represent the sum of all the positive values of *user_and_battery_balance*, to consider the surplus.
- *users_and_batteries_energy_deficit_t_kWh* which, at the contrary, represents the sum of all the negative values of *user_and_battery_balance*, to consider all the deficit.

- *users_and_batteries_balance_t* is the balance that considers the users and the battery. If it is positive, it means that all the users are are in surplus, while if it is negative means that all the users are in deficit.
- *flag* as explained for part 1, the flag is a numeric parameter that identify the situation and simplify the code.
  - *flag* = *5* is assigned when:

$$users\_and\_batteries\_balance\_t\_kWh \geq 0 \tag{3.5.19}$$

    and

$$max\_energy\_chargeable\_t\_kWh \geq users\_and\_batteries\_balance\_t\_kWh \tag{3.5.20}$$

    similarly to flag = 1 in part 1, in this situation the whole community is in surplus and the battery has enough space to fill all in it.
  - *flag* = *6* is assigned when:

$$users\_and\_batteries\_balance\_t\_kWh \geq 0 \tag{3.5.21}$$

    and

$$max\_energy\_chargeable\_t\_kWh < users\_and\_batteries\_balance\_t\_kWh \tag{3.5.22}$$

    similarly to flag = 2 in part 1, in this situation the whole community is in surplus and the battery but the battery does not have enough space store all it, and part is shared outside.
  - *flag* = *7* is assigned when:

$$users\_and\_batteries\_balance\_t\_kWh < 0 \tag{3.5.23}$$

    and

$$max\_energy\_dischargeable\_t\_kWh \geq users\_and\_batteries\_balance\_t\_kWh \tag{3.5.24}$$

    similarly to flag = 3 in part 1, in this situation the whole community is in deficit but the battery is able to overcome it and supply all.

– *flag = 8* is assigned when:

$$users\_and\_batteries\_balance\_t\_kWh < 0 \tag{3.5.25}$$

and

$$max\_energy\_dischargeable\_t\_kWh < users\_and\_batteries\_balance\_t\_kWh \tag{3.5.26}$$

similarly to flag = 4 in part 1, in this situation the whole community is in deficit but the battery is not able to overcome it and supply all, and will be needed external sharing.

- *battery_balance_t_kWh* is the same as the one calculated in part one, but this time it is not used the single user's balance but the whole community's in that timestamp.
- *energy_in_battery_t_kWh* is the amount of energy that remains in the battery after the charge or discharge in the timestamp

The procedure observed for part 2 can be considered similar to the one in part 1, except that it is not applied on each user but instead on all the community

Part 3: Sharing and redistributing

In the last part, the purpose is to calculate how much energy is shared and how much energy is redistributed.

It is visible in Figure 3.9.

Where:

- *total_shared_energy_user2storage* is the total amount of energy in surplus that is shared from the users and stored in the community's battery. It is equal to *battery_balance_t_kWh* only if the battery balance is higher than zero, meaning that it is recharging.
- *total_shared_energy_user2user* is the total amount of energy shared from the users in surplus to the users in need. It is equal to 0 if the surplus is equal to zero; it is equal to the whole surplus if it is lower than the deficit; it is equal to the deficit if it is higher than the deficit.
- *total_grid_exchange* represent the amount of energy that it is transferred from the REC to the grid. It is equal to:

$$total\_grid\_exchange = users\_and\_battery\_balance\_t\_kWh - battery\_balance\_t\_kWh \tag{3.5.27}$$

meaning that the overall balance is influenced by the energy exchanged with the storage.

- *potential_redistribution_t_kWh* is the maximum possible amount of energy that can be stored in each user's batteries, and it is equal to:

$$potential\_redistribution\_t\_kWh = max\_energy\_chargeable\_t\_kWh - battery\_balance\_t\_kWh \tag{3.5.28}$$

  meaning that the maximum amount of energy that can be redistributed is the maximum amount of energy that can normally be charged but adding the battery balance.

- *max_redistribution* is the sum of all the *potential_redistribution_t_kWh*, so the total amount of energy that can be potentially redistributed.

- *redistributed_energy* is the amount of energy that is in the end redistributed on a REC scale, reducing the energy exchanged with the grid. Its value can be:

  – equal to *0* if the energy exchanged with the grid is not higher than zero, meaning that there is a deficit or simply the recharging of the battery.

  – equal to *total_grid_exchange* if $0 < total\_grid\_exchange < max\_redistribution$, meaning that the energy that there is enough space in the batteries to collect the excess energy that otherwise would be sold to the grid.

  – equal to *max_redistribution* if $total\_grid\_exchange > max\_redistribution$, meaning that all the surplus can provide energy to the distributed batteries and in the meantime sell it to the grid

- *final_grid_exchange* is the final amount of energy exchanged to the grid after the redistribution. It is:

  1. equal to *total_grid_exchange*, so it stays untouched, if the *total_grid_exchange* is not positive, so in the case there isn't any redistribution
  2. equal to *0* if $0 < total\_grid\_exchange < max\_redistribution$, meaning that all the surplus is redistributed and therefore there is not energy exchanged with the grid.
  3. equal to *total_grid_exchange - max_redistribution* in case $total\_grid\_exchange > max\_redistribution$ because there is excess electricity that cannot be stored further.

- *total_shared_out_and_redistributed_energy* is the sum of the *redistributed_energy*, the *total_shared_energy_user2user* and the *total_shared_energy_user2storage*. It represents the total amount of energy that is shared among all the members and the storage, without distinction. This parameter is the one that will receive the subsidies, according to the Italian law.

At the end of this part, it is calculated the final amount of energy that is exchanged with the grid and the final amount of energy that is shared among members.

Part 4: individual contribution

This last part aims to calculate the amount of energy that is shared or exchanged with the grid by each users. The Figure 3.10 represents it.

In which:

- *fraction_of_free_space* is the percentage of free space that is available in the single battery compared to the total space available. It is calculated as:

$$fraction\_of\_free\_space = potential\_redistribution\_t\_kWh / max\_redistribution \quad (3.5.29)$$

- *user_redistributed_energy_t_kWh* is the final amount, expressed in [kWh], of energy that the user takes in after the redistribution, and it is equal to the total redistributed energy multiplied bt the fraction of free space.
- *final_battery_balance_t_kWh* is the updated and definitive value of the battery balance that takes into account the self-produced electricity and the redistribution, and it is calculated as the battery balance added to the redistributed energy.
- *final_user_and_battery_balance_t_kWh* is the updated value of the household's balance considering the self-consumption, the self-production and the redistribution. It is calculated adding the the redistributed electricity to the previous value.
- *final_energy_in_battery* is the amount of energy that is left in the storage at the end of the timestamp, and it is calculated as the previous value of the final energy adding the battery balance.
- *final_space_in_battery* is the amount of energy that is missing from the battery at the end of the timestamp, and it is calculated as the difference between the full capacity and the energy left in the battery.
- *fraction_of_energy_to_grid_%* is the percentage of surplus energy that is sent to the grid. It is calculated as the amount of energy that is sent to the grid by the community divided by all users' surplus.
- *fraction_of_energy_from_grid_%* is the percentage of needed energy that is taken from the grid. It is calculated as the amount of energy that is taken from the grid by the community divided by all users' deficit.
- *fraction_of_shared_out_energy_%* is the percentage of surplus energy that is shared to other users, and it is calculated as the total shared energy by the community divided by the total surplus.

- *fraction_of_shared_in_energy_%* it is the percentage of shared energy that is taken from other users, and it is calculated as the complementary part of the energy from grid.
- *energy_shared_out_t_kWh* is the amount of energy in kWh that the single user is sharing to the users, and it is calculated as the fraction of energy shared out multiplied to the household's balance
- *energy_shared_in_t_kWh* is the amount of energy in kWh that the single user is taking from other users, and it is calculated as the fraction of energy shared in multiplied to the household's balance
- *energy_sold_to_grid_t_kWh* is the amount of energy in kWh that the single user is sharing to the grid, and it is calculated as the fraction of energy sold out multiplied to the household's balance
- *energy_sold_to_grid_t_kWh* is the amount of energy in kWh that the single user is taking from the grid, and it is calculated as the fraction of energy bought in multiplied to the household's balance

At this point, every useful parameter is calculated and it is possible to move forward to the next last part.

Part 5: iteration

The iteration phase is the part of the code that is used to copy the values from the current timestamp that need to be iterated in the successive timestamps.

This process is done simply by copying the values useful for iteration into the corresponding values of the next timestamp.

These values are:

- *storage_dod* the depth of discharge is not used directly in the calculations except for the first few lines, but is extremely useful in the review phase to see if there are any discrepancies and to see the behaviour of the battery over time if desired.
- *previous_energy_in_battery_t_kWh*, which records the amount of energy remaining in the battery to continue with the battery balance calculations.

At the end, the values calculated in the timestamp are copied to the corresponding values marked as "previous" in the next timestamp, so that it is possible to restart the cycle and continue the calculations.

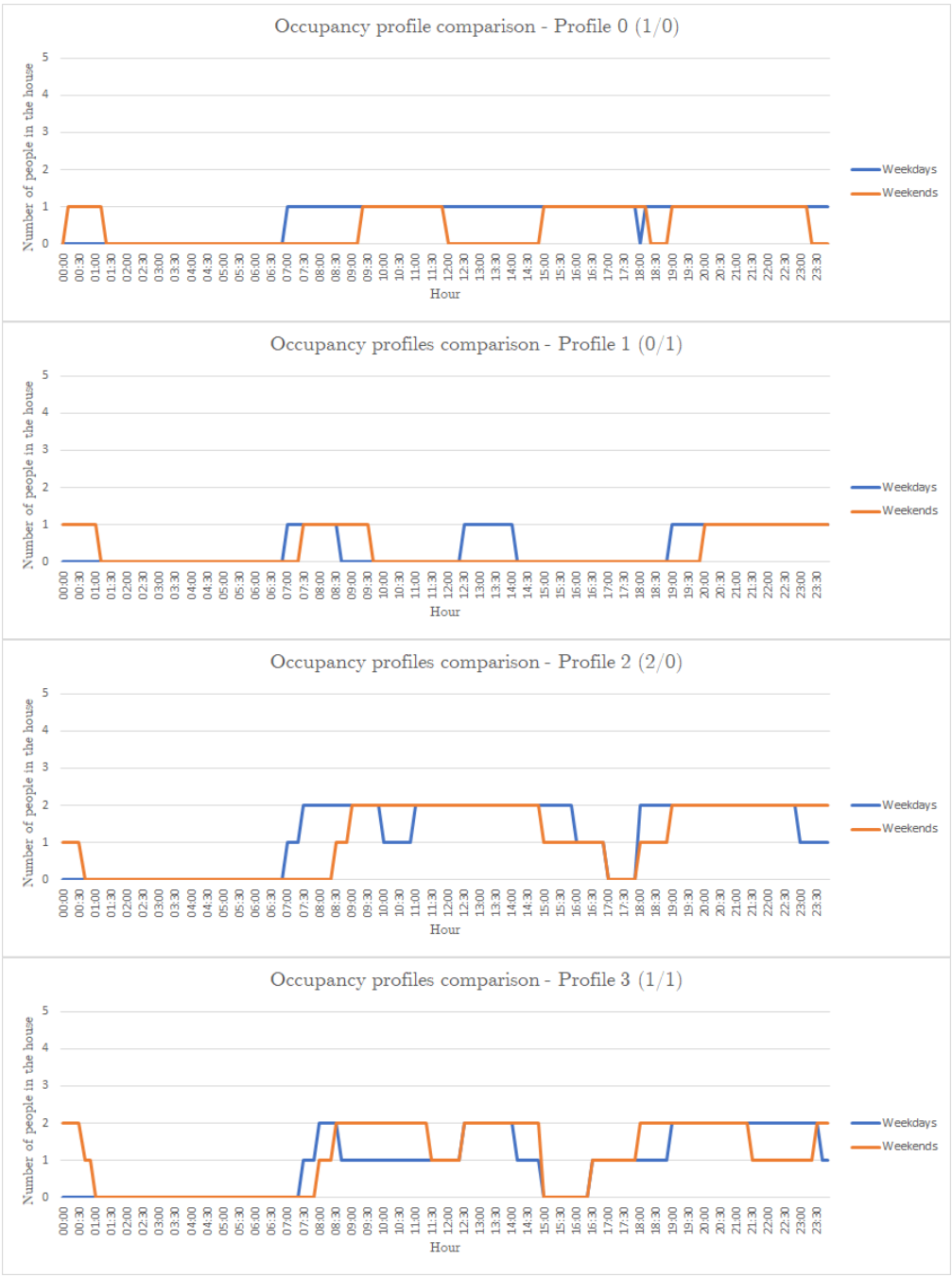

Figure 3.1: Hourly trend of the occupancy for the first four family categories

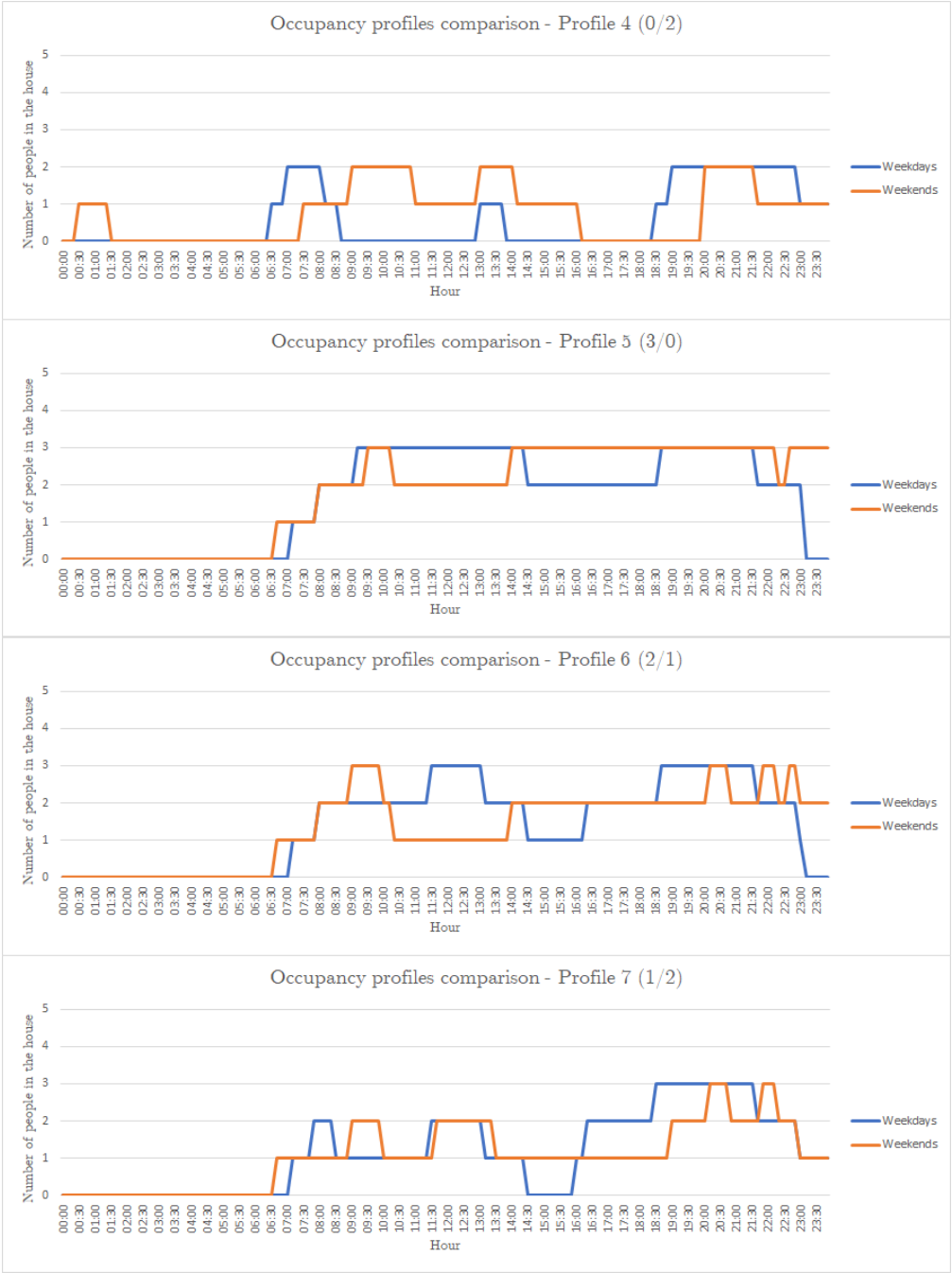

Figure 3.2: Hourly trend of the occupancy for the second four family categories

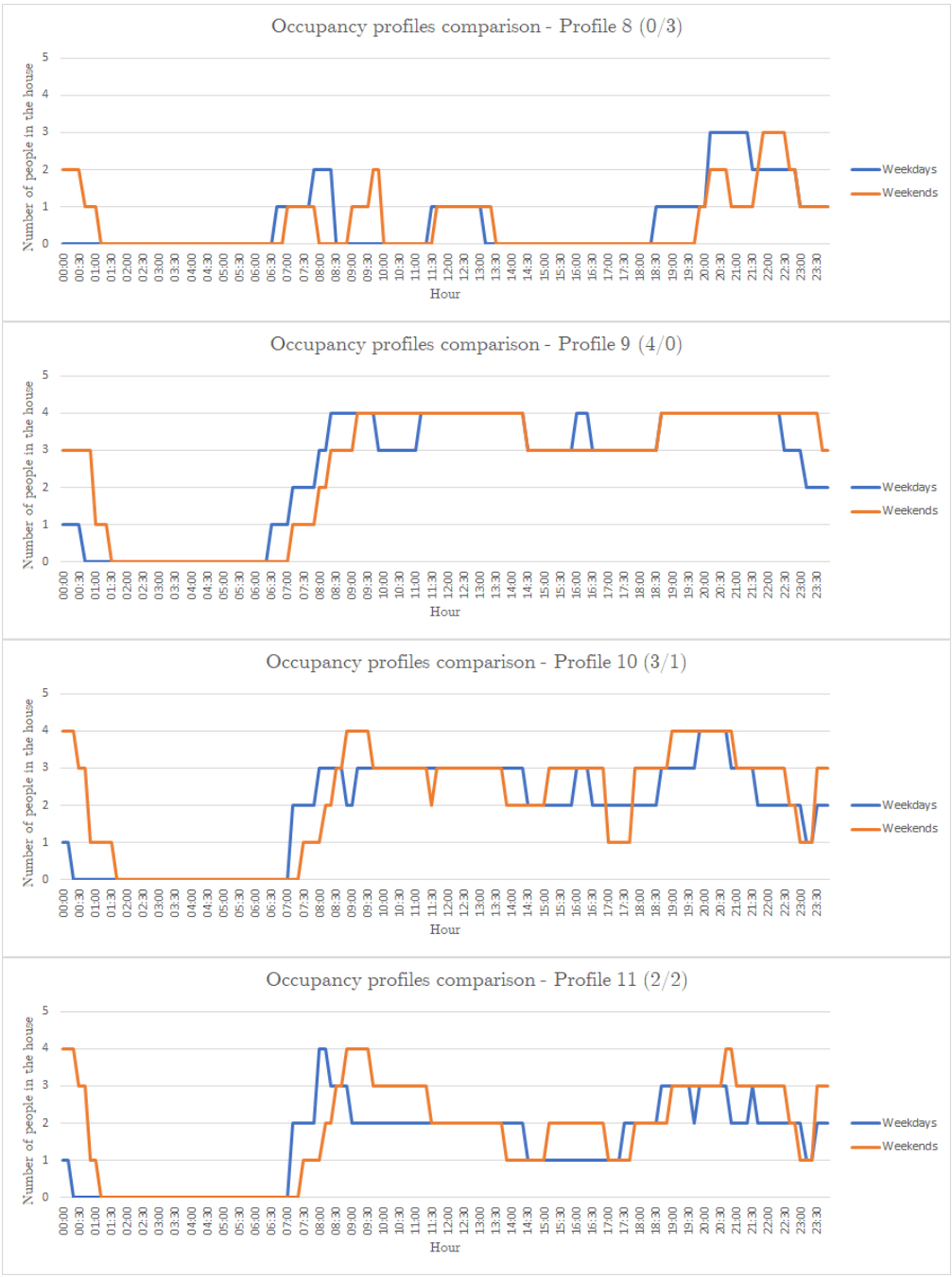

Figure 3.3: Hourly trend of the occupancy for the third four family categories

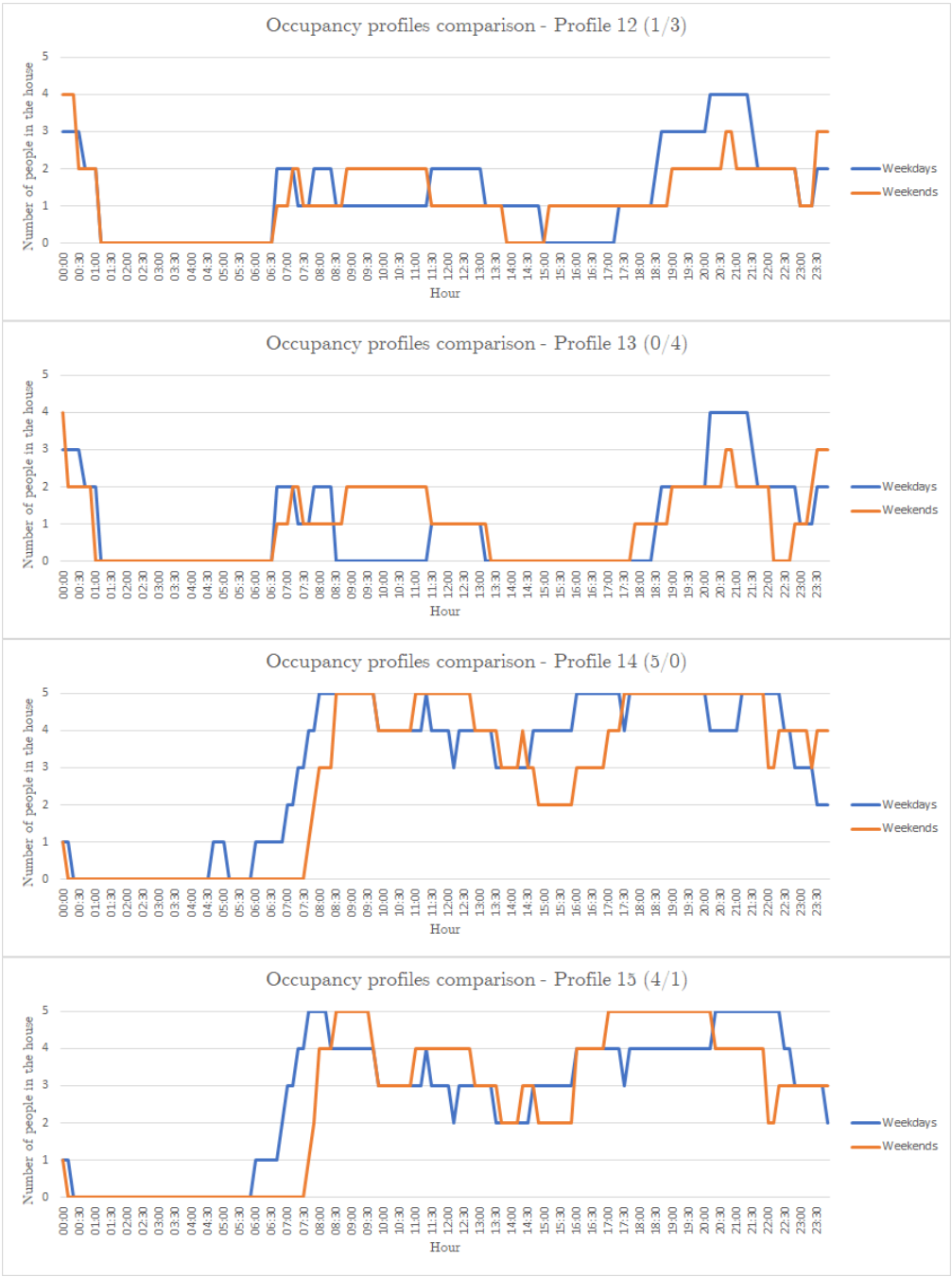

Figure 3.4: Hourly trend of the occupancy for the fourth four family categories

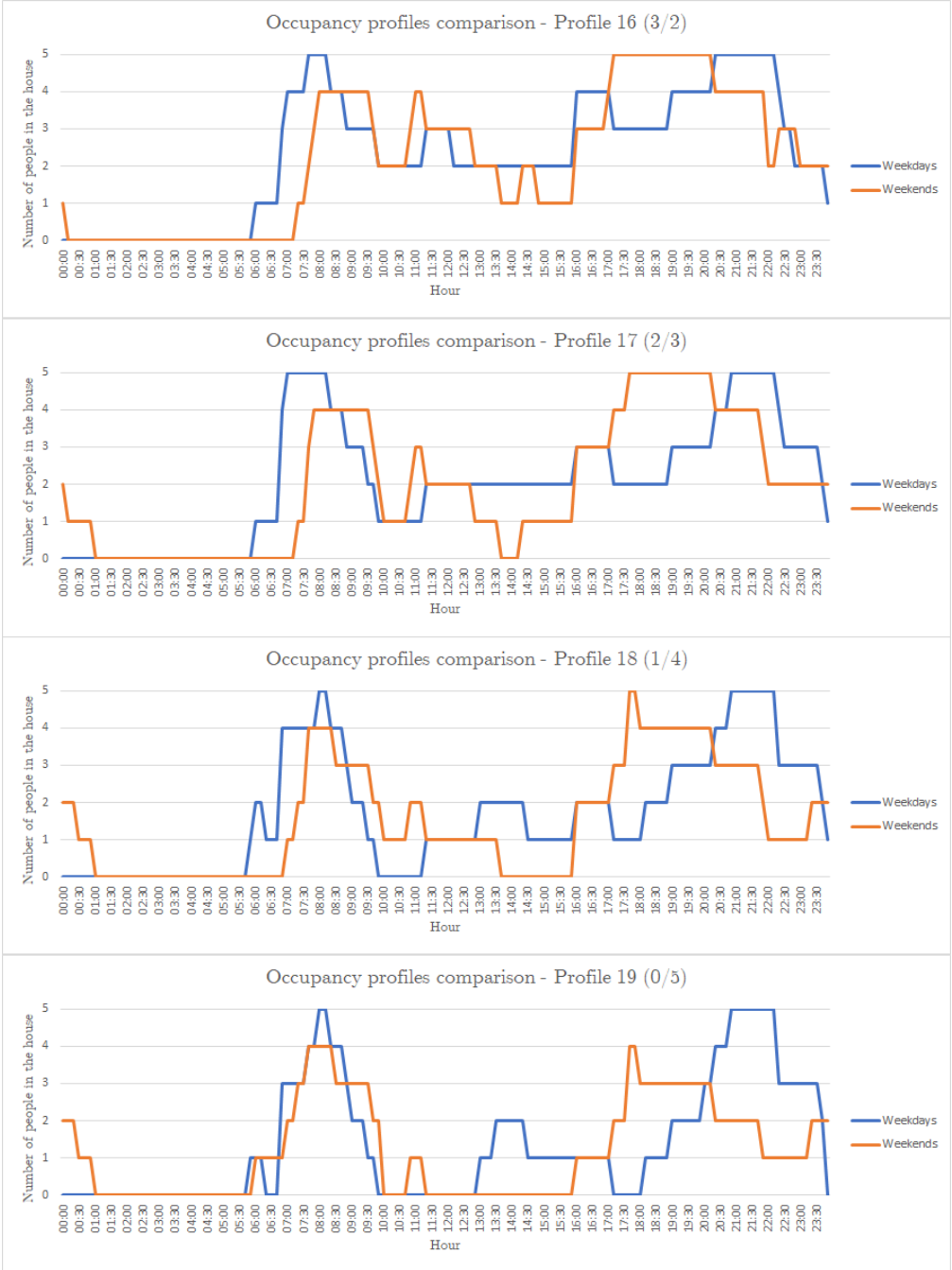

Figure 3.5: Hourly trend of the occupancy for the fifth four family categories

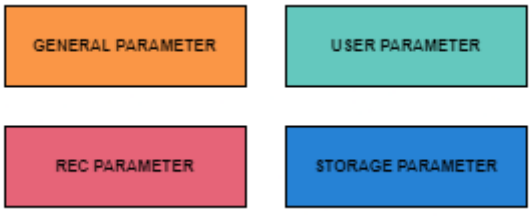

Figure 3.6: Representation of the colours used in the activity diagram

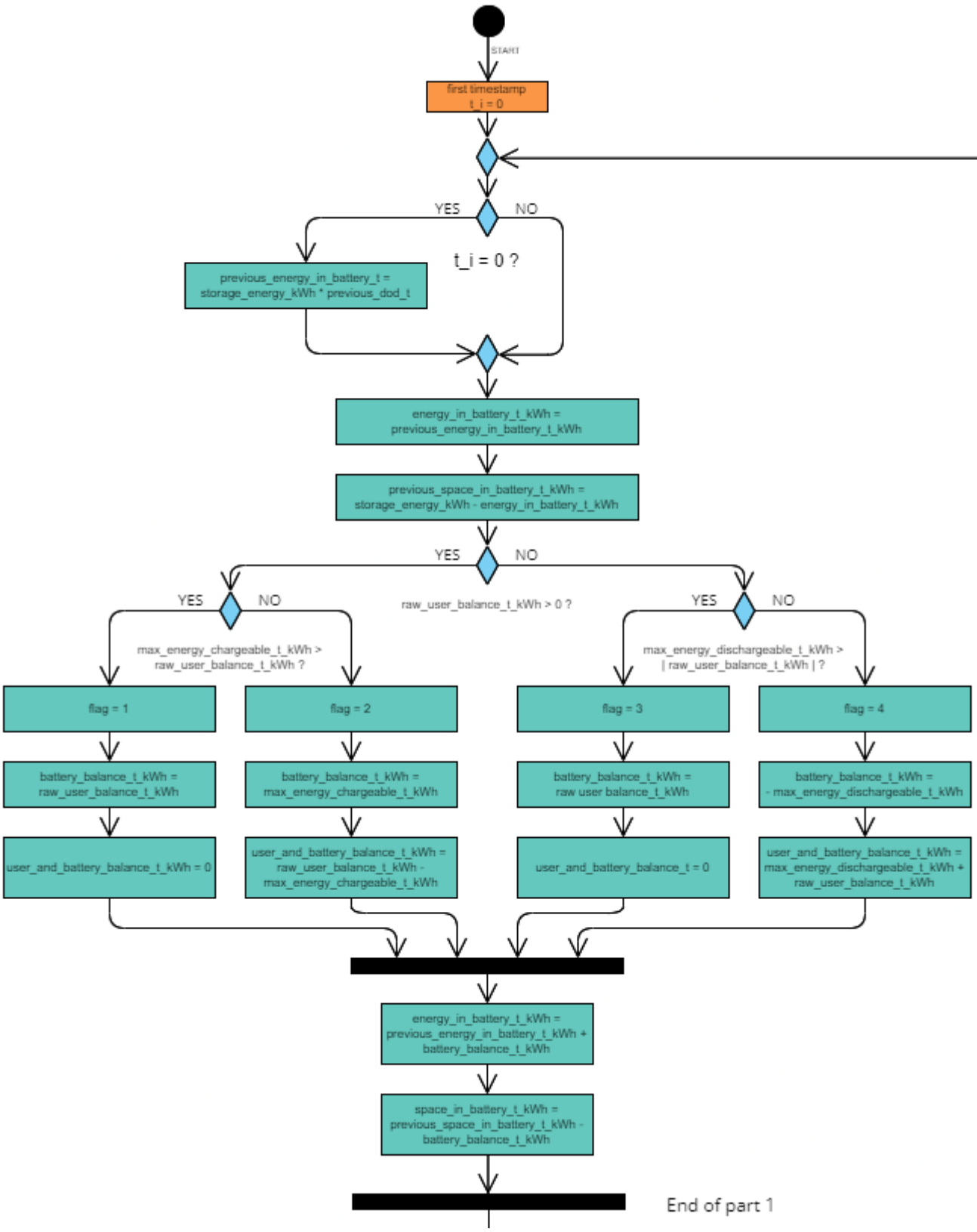

Figure 3.7: First part of the Activity Diagram

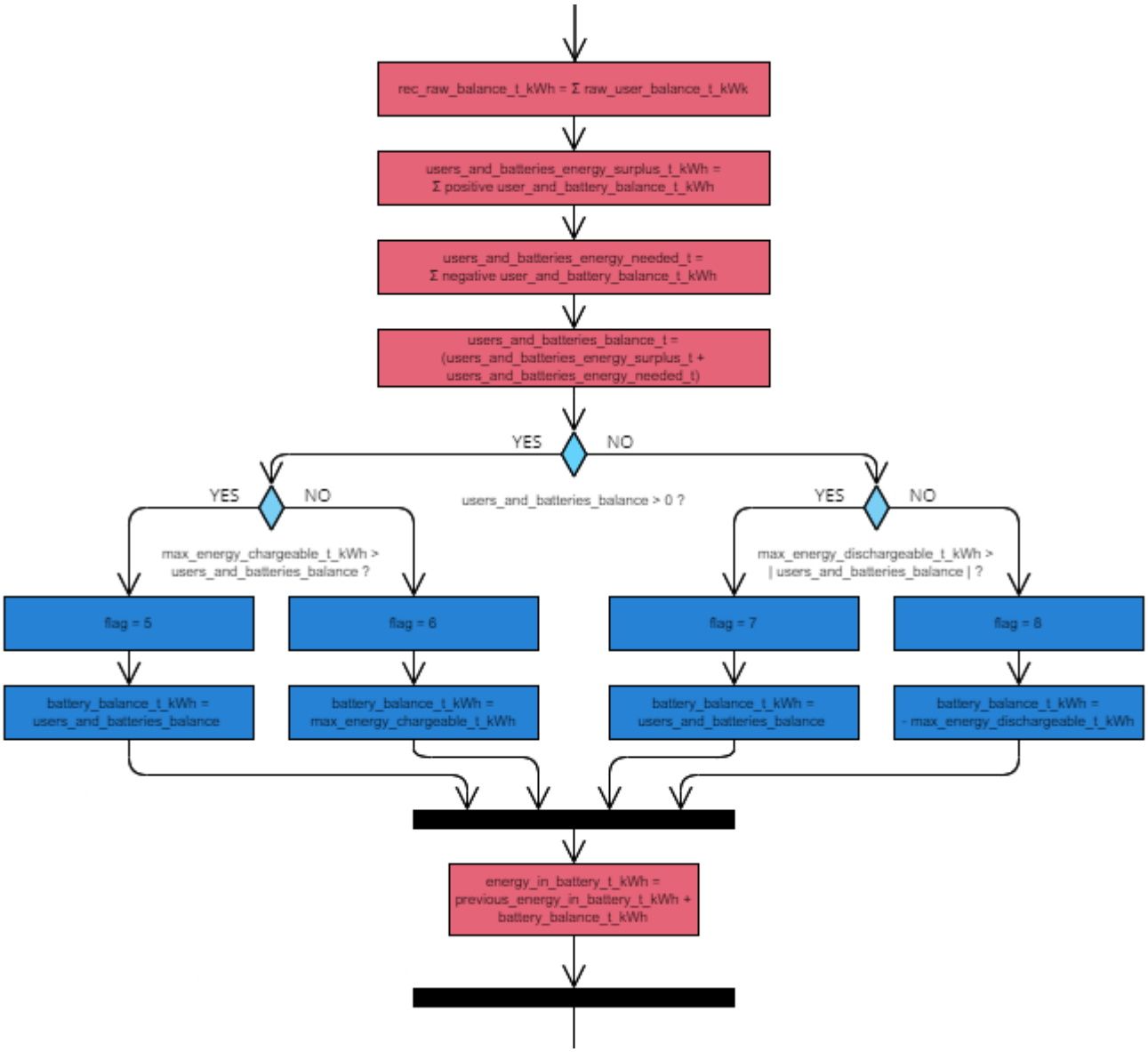

Figure 3.8: Second part of the Activity Diagram

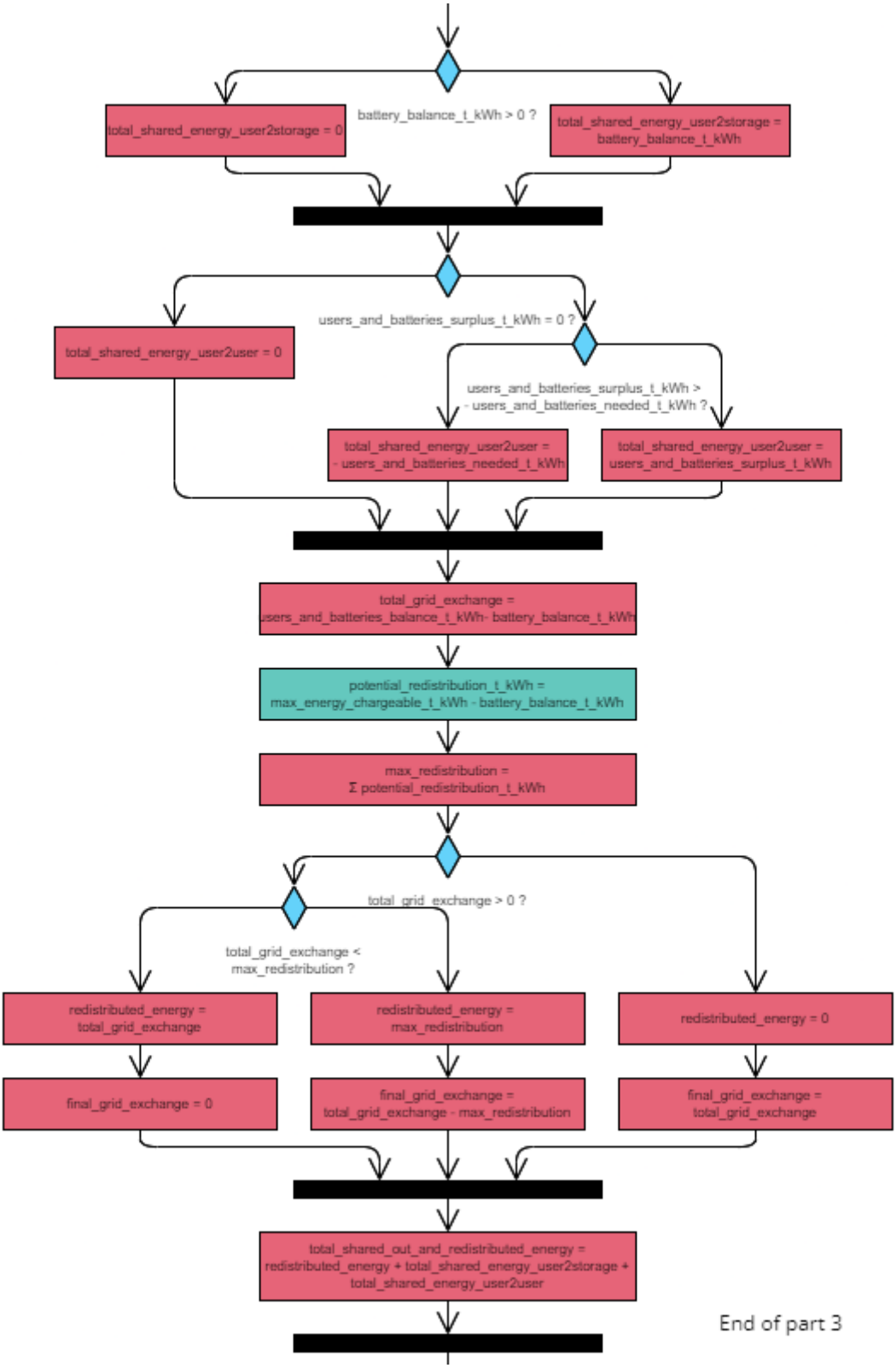

Figure 3.9: Third part of the Activity Diagram

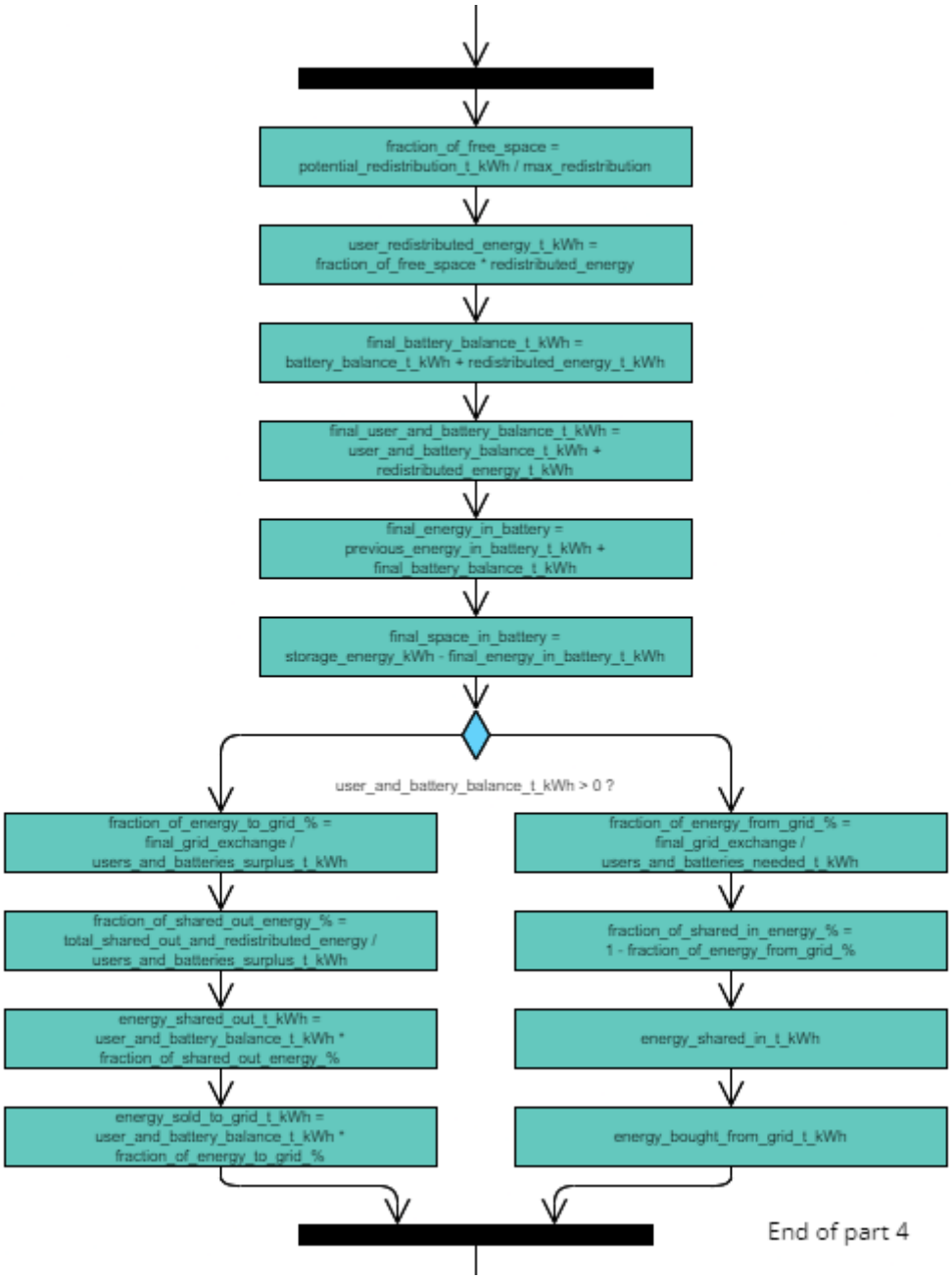

Figure 3.10: Fourth part of the Activity Diagram

CHAPTER 4

Results

This chapter aims to give an overview of the results obtained after the simulations. It starts by presenting the results of the energy flows and balances for each month and year in the different case studies. It then compares, comments on and evaluates the results from the users' and the municipality's point of view in order to understand the impact of the design choices made.

4.1 Results presentation

The decision to make a comparison between different situations should serve to give an overview of some of the many possibilities that may arise in the decision-making phase of establishing a community. The aim of this chapter is not to give an answer to the question of which case is absolutely better, but to highlight the features and positive aspects of each case.

In every case study will be analyzed data for:

- Net balance, comparing the production and the consumption.
- Grid exchanges, comparing the electricity sold and bought from the grid.
- Shared electricity, comparing the amount of electricity shared with the centralized storage, the other users' storages and for the other users' instant consumption
- Self-consumption, evaluating the amount of electricity that is self consumed and stored in their storages by the prosumers, if present.

These parameters are the most important in shaping a community and its policies.

4.1.1 Case 1: Blended Community

In this case study, as described in Section 3.1.1, the community consists of a heterogeneous group that includes all the categories: 15 consumers, 10 prosumers, a 20 [kW] photovoltaic system and a 30 [kWh] storage.

Net balance

In Table 4.1 it are listed the monthly and the total production, consumption and net balance.

Month	Production [kWh]	Consumption [kWh]	Balance [kWh]
1	6117	-6947	-830
2	5430	-6314	-885
3	8068	-6962	1105
4	10566	-6695	3871
5	11224	-6969	4255
6	11542	-6620	4921
7	11983	-6761	5222
8	10751	-6891	3859
9	9344	-6642	2702
10	7562	-7006	555
11	6087	-6624	-537
12	6790	-6863	-72
Total	105468	-81299	24168

Table 4.1: Net balance with monthly and total calculation in case 1

Predictably, the amount of electricity produced is higher in the summer months and lower in the winter months. The month with the highest production is July with 11983 [kWh] and the month with the lowest production is February with 5430 [kWh]. As far as the balance is concerned, the winter production is not enough to cover the demand. In February there is a deficit of -885 [kWh], while in July more is produced, so the balance is +5222 [kWh].

Exchanges with the grid

By observing the Table 4.2, are listed all the exchanges with the grid.

When evaluating the energy sold to the grid, it is immediately noticeable that the month in which the energy sold is the lowest is February (261 [kWh]), and the month with the highest electricity sold is again July (5226 [kWh]), as this parameter is related to production. Regarding the electricity purchased from the grid, the most positive aspect is that the municipality does not purchase electricity from the grid between June and September. However, there is a higher demand in the winter months, especially in February, when 1147 [kWh] are drawn from the grid, with a net balance of -886 [kWh].

Month	Sold to grid [kWh]	Bought from grid [kWh]	Grid balance [kWh]
1	281	-1135	-854
2	261	-1147	-886
3	1418	-410	1007
4	3915	-49	3866
5	4254	0	4254
6	4918	0	4918
7	5226	0	5226
8	3865	0	3865
9	2706	0	2706
10	1049	-405	644
11	222	-759	-537
12	295	-425	-130
Total	28410	-4330	24079

Table 4.2: Monthly and total exchanges with the grid in case 1

Shared electricity

In Table 4.3 are listed all the values of the various parameters that compose the shared electricity.

Month	Shared to storage [kWh]	Redistributed [kWh]	Instant sharing [kWh]	Total share [kWh]
1	1460	1260	1188	3909
2	1236	1136	1065	3436
3	1605	1834	1188	4627
4	1490	1992	1171	4653
5	1578	2103	1206	4888
6	1407	2057	1199	4663
7	1408	2134	1225	4767
8	1544	2140	1156	4840
9	1602	1962	1069	4633
10	1682	1640	1110	4431
11	1523	1263	1085	3871
12	1710	1664	1091	4464
Total	18245	21185	13753	53182

Table 4.3: Monthly and total shared electricity among the community in case 1

As far as the shared electricity is concerned, it can be noted that there is not much difference between the highest and lowest values in each category, which means that the amount is almost constant during the year. In February, a lower amount of shared energy is registered, mainly due to the lower number of days in this month: 1235 [kWh] for storage sharing, 1136 [kWh] for other users' storage sharing and 1065 [kWh] for instantaneous sharing by other users, for a total amount of 3436 [kWh]. However, the peaks occur in different months: The month with the highest storage sharing is December with 1710 [kWh], the month with the highest redistribution

more redistribution is August, with 2140 [kWh], and the month with the highest instant sharing is July with 1225 [kWh]. The month with the highest total sharing is May, with 4888 [kWh] electricity sharing. The trend throughout the year shows that the values of shared electricity remain almost constant throughout the year, as the values are not that far apart. The peak value of energy shared to the storage in winter is explained by the fact that the users produce less and therefore the storage helps more and is discharged more by the users.

Self-consumed energy

The ten prosumers among the members are those who produce their own electricity and storing in their storages, if any. Table 4.4 shows the monthly and total amount of electricity shared.

Month	Self-consumed and stored [kWh]
1	1928
2	1732
3	2024
4	1999
5	2083
6	1961
7	1990
8	2046
9	2006
10	2082
11	1993
12	2031
Total	23875

Table 4.4: Self-consumed and stored in batteries electricity for case 1

As imaginable, the self-consumed electricity does not fluctuate much over the course of the year, because once prosumers fill their storage, they simply consume the electricity immediately and the surplus is shared.

4.1.2 Case 2: No-Storage Community

In this scenario, the municipality is the same as in case 1 from the previous Section 4.1.1, but the batteries are considered not to be present, as might be the case in a community that does not want to install them for various reasons, such as economic, cost or other personal reasons it has decided.

Net balance

In the following Table 4.5 is recapped the yearly production and consumption of the community.

Month	Production [kWh]	Consumption [kWh]	Balance [kWh]
1	6117	-6947	-830
2	5430	-6315	-885
3	8068	-6962	1106
4	10567	-6696	3871
5	11225	-6970	4255
6	11542	-6620	4922
7	11983	-6761	5221
8	10752	-6892	3860
9	9344	-6642	2702
10	7563	-7007	556
11	6087	-6625	-537
12	6790	-6863	-73
Total	105468	-81300	24168

Table 4.5: Net balance with monthly and total calculation in case 2

Since the conditions that make up the community, in terms of the number of users, prosumers and the power plant, are the same as in case 1, the results in terms of production and consumption are also the same. The description goes back to Section 4.1.1.

Exchanges with the grid

Unlike the previous situation, where there are stores belonging to the members of the community, the grid exchanges are listed in Table 4.6.

Month	Sold to grid [kWh]	Bought from grid [kWh]	Grid balance [kWh]
1	3418	-4248	-830
2	2964	-3848	-885
3	5007	-3901	1106
4	7234	-3363	3871
5	7675	-3420	4255
6	8059	-3138	4922
7	8408	-3187	5221
8	7302	-3442	3860
9	6197	-3494	2702
10	4584	-4028	556
11	3443	-3980	-537
12	3973	-4045	-73
Total	68264	-44094	24168

Table 4.6: Monthly and total exchanges with the grid in case 2

The aspect that immediately catches the eye is that the values defining exchanges with the net are much higher than those seen in case 1 at Section 4.1.1. Again, exports to the grid are lowest in February with 2964 [kWh], while they are highest in July with 8048 [kWh]. Following the opposite trend, the possibility to produce energy and consume it immediately is higher in summer, so that less imports are needed in June, namely only 3138 [kWh], while more is needed in winter, namely 4248 [kWh] in January. This resulted in the highest balance being reached in July with 5222 [kWh] and the lowest in February with -885 [kWh].

Shared electricity

This case study is characterised by the absence of storage, which results in the absence of shared and redistributed electricity, as can be seen in Table 4.7.

Month	Shared to storage [kWh]	Redistributed [kWh]	Instant sharing [kWh]	Total share [kWh]
1	0	0	1677	1677
2	0	0	1486	1486
3	0	0	1889	1889
4	0	0	2045	2045
5	0	0	2177	2177
6	0	0	2156	2156
7	0	0	2228	2228
8	0	0	2128	2128
9	0	0	1901	1901
10	0	0	1790	1790
11	0	0	1596	1596
12	0	0	1705	1705
Total	0	0	22778	22778

Table 4.7: Monthly and total shared electricity among the community in case 2

Since only electricity is shared immediately, these are values that only occur when one user or the facility is in surplus and others are in need. The dimensioning of each plant in this situation is fundamental to maximise this parameter. Following the trend of production, the peak value is 2228 [kWh] in July and the lowest value is 1486 [kWh] in February.

Self-consumed energy

With ten prosumers among the members, but without their storage, the amount of self-consumed and stored electricity is expected to be less. Table 4.8 shows the monthly and total amount of electricity shared.

As can be imagined, the self-consumed electricity is lower and does not fluctuate much throughout the year, as in the previous scenario, because the prosumers simply consume the electricity immediately and the surplus is shared, bypassing the storage phase.

Month	Self-consumed and stored [kWh]
1	1022
2	980
3	1172
4	1288
5	1372
6	1327
7	1347
8	1322
9	1247
10	1189
11	1049
12	1113
Total	14429

Table 4.8: Self-consumed and stored in batteries electricity for case 2

4.1.3 Case 3: All-Consumers Community

To recall the settings in this situation, each user is only a consumer who draws his electricity from the shared installation and avoids producing it himself and storing it in the individual battery, using the shared installation instead, as would be the case with Collective Self-Consumption or in special situations where there is no space on the roof for solar panels or other technologies such as wind power are used.

Net balance

Since the families and consumption of the members remain unchanged, it is assumed that their needs remain almost the same, while the difference is made up by the solar system, as can be seen in Table 4.9. In fact, as explained in Section 3.1.3, there are three concentrated installations instead of one.

Predictably, there is maximum and minimum production in June and February, respectively at 12595 [kWh] and at 5922 [kWh]. These results are consistent with the peaks of the final balance, which are 58340 [kWh] in July and -667 [kWh] in November.

Exchanges with the grid

About the import/export with the grid, in the Table 4.10 the exchanges during the year are listed. The presence of a single plant should have advantages in terms of grid balance from the distributor's point of view, as productivity is easier to predict compared to multiple panels.

The most promising outcome of this scenario is that in the summer months between May and September, the community does not need electricity, is self-sufficient and can sell any surplus on the market. The month in which more energy is sold is July (5838 [kWh]), while less is

Month	Production [kWh]	Consumption [kWh]	Balance [kWh]
1	6685	-6947	-263
2	5922	-6315	-392
3	7921	-6962	958
4	10654	-6696	3958
5	11403	-6970	4433
6	12033	-6620	5413
7	12595	-6761	5834
8	10916	-6892	4024
9	9340	-6642	2698
10	7683	-7007	676
11	5957	-6625	-667
12	7129	-6863	266
Total	108238	-81300	26938

Table 4.9: Net balance with monthly and total calculation in case 3

Month	Sold to grid [kWh]	Bought from grid [kWh]	Grid balance [kWh]
1	465	-703	-238
2	308	-700	-392
3	1258	-449	809
4	3966	-13	3953
5	4432	0	4432
6	5409	0	5409
7	5838	0	5838
8	4029	0	4029
9	2702	0	2702
10	1072	-250	822
11	203	-870	-667
12	325	-150	175
Total	30007	-3135	26872

Table 4.10: Monthly and total exchanges with the grid in case 3

sold in November (2030 [kWh]). In this month, the demand for electricity from the grid is also the highest at 870 [kWh]. As for the balance, the peaks are conceivable in July and November with 5838 [kWh] and -667 [kWh] respectively. However, the most promising aspect of this configuration is a positive grid balance in December caused by low requests from the grid.

Shared electricity

Since there is no distributed storage, the values for redistributed energy are zero, so this possibility must be excluded. All shared power is either stored in the battery or immediately shared with the members, as can be seen in the Table 4.11.

What happens in this scenario when it comes to shared electricity is that the values are similar, which means that it is distributed almost equally among the members and to the battery.

Month	Shared to storage [kWh]	Redistributed [kWh]	Instant sharing [kWh]	Total share [kWh]
1	3579	0	2641	6220
2	3222	0	2393	5615
3	3728	0	2934	6663
4	3384	0	3303	6687
5	3462	0	3508	6971
6	3120	0	3504	6624
7	3150	0	3607	6757
8	3435	0	3452	6887
9	3492	0	3147	6638
10	3755	0	2856	6611
11	3203	0	2552	5754
12	4056	0	2749	6805
Total	41586	0	36646	78232

Table 4.11: Monthly and total shared electricity among the community in case 3

In particular, the month in which the total amount of electricity shared is higher is May with 6971 [kWh] and the month with the lowest distribution is February with 5615 [kWh].

Self-consumed energy

In this case study, the absence of prosumers implies the absence of self-consumed energy if one uses the previous definition, which includes only prosumers. However, depending on the rules of the community, a certain amount of electricity generated by the central plants can be defined as self-consumption, but this aspect goes beyond the objective of this work.

4.1.4 Case 4: All-Prosumers Community

In this situation, where a community consists of prosumers without a shared facility or storage, users produce their own electricity, store it and share their surplus with the others when they need it, as explained in Section 3.1.4.

Net balance

Regarding the net balance, the difference is defined by the plant because each user has its own plant and is producing its own electricity. In Table 4.12 can be seen the summary of each month's balance.

The trend of the previous cases is confirmed: in July and February 10180 [kWh] and 4739 [kWh] are produced, respectively, while the final balance in July is 3418 [kWh] and in January -1696 [kWh].

Month	Production [kWh]	Consumption [kWh]	Balance [kWh]
1	5251	-6947	-1696
2	4739	-6315	-1576
3	7529	-6962	567
4	9562	-6696	2866
5	9962	-6970	2992
6	9915	-6620	3295
7	10180	-6761	3418
8	9603	-6892	2712
9	8567	-6642	1925
10	6927	-7007	-79
11	5666	-6625	-959
12	5958	-6863	-905
Total	93859	-81300	12560

Table 4.12: Net balance with monthly and total calculation in case 4

Exchanges with the grid

In Table 2 can be seen a summary of the exchange with the public network. It should be remembered that in this scenario there is neither a central power plant nor a central storage facility, which means that each user only has the option of generating their own energy or purchasing it from the others.

Month	Sold to grid [kWh]	Bought from grid [kWh]	Grid balance [kWh]
1	56	-1761	-1705
2	55	-1627	-1571
3	994	-544	451
4	2911	-48	2862
5	2993	-1	2992
6	3292	0	3292
7	3419	0	3419
8	2717	0	2717
9	1929	0	1929
10	583	-556	27
11	9	-972	-962
12	3	-933	-930
Total	18961	-6442	12521

Table 4.13: Monthly and total exchanges with the grid in case 4

It can also be seen that, also due to the highest capacity among the four scenarios, the community is self-sufficient in the summer months and does not need to import electricity from the public grid from May to August, and the low values in the neighbouring months also define a longer period of self-sufficiency. In winter, especially in January, imports are highest at 1761 [kWh] and exports, especially in December, are lowest for the year at 3 [kWh]. In summer, in

July, monthly production is highest at 3419 [kWh], which is also the balance due to zero imports.

Shared electricity

As visible in Table 4.14, the energy that in the end is shared to the community's storage is null, as imaginable by knowing that the community was designed with diffused storages.

Month	Shared to storage [kWh]	Redistributed [kWh]	Instant sharing [kWh]	Total share [kWh]
1	0	286	343	629
2	0	284	340	624
3	0	624	1618	2242
4	0	568	3478	4046
5	0	646	3640	4286
6	0	629	3921	4550
7	0	649	4068	4717
8	0	602	3319	3920
9	0	599	2529	3128
10	0	531	1114	1644
11	0	410	420	830
12	0	435	438	873
Total	0	6263	25228	31489

Table 4.14: Monthly and total shared electricity among the community in case 4

The main goal of a community of prosumers is self-consumption, being able to use the generated electricity immediately and to shop the surplus. Redistribution from users to other users' storage is also reduced, as this is not the priority, but self-consumption and immediate sharing are the main focus. In July there are the most shares with 4717 [kWh], and in February the fewest with 624 [kWh].

Self-consumed energy

In this scenario, the composition consisting only of prosumers is expected to significantly increase the values of own consumption. Table 4.15 shows the monthly and total self-consumption values and the stored energy.

In the winter months, the self-consumed and stored energy is much higher than in the summer months. This aspect is explained by the fact that the amount of energy that is dispatched, as seen in the previous subsection, is lower due to the prioritisation of self-consumed and stored energy. In summer, the storage is replenished in a shorter time, both by the user's panels and by the surplus stored in the batteries by the other users. In this way, there is more time during the day when the surplus electricity is sold instead of stored, resulting in a lower value.

Month	Self-consumed and stored [kWh]
1	4566
2	4060
3	4292
4	2606
5	2682
6	2073
7	2043
8	2966
9	3510
10	4700
11	4826
12	5082
Total	43406

Table 4.15: Self-consumed and stored in batteries electricity for case 4

4.2 Results comparison

The aim of this section is to propose a comparison between the four scenarios. To this end, the results are presented both numerically, in terms of the amount of electricity shared or exchanged with the grid, and relatively, as a percentage of electricity generated and consumed. In this way, results can be compared that would otherwise not be so easy to evaluate.

Allocation of the produced energy

This section is about comparing and evaluating the end use of the electricity generated. It can be used in three ways: for self-consumption, for sharing among members and for sale to the grid. The Table 4.16 shows the amounts of energy in [kWh] and the percentage of the total electricity generated.

Parameter	Case 1 Blended	Case 2 No-Storage	Case 3 All-Prosumers	Case 4 All-Consumers
Total produced energy [kWh]	105,467	105,469	108,238	93,860
Shared [kWh]	53,183	22,777	78,231	31,490
Shared %	50	22	72	34
Sold [kWh]	28,411	68,263	30,007	18,964.2
Sold %	27	65	28	20
Self-consumed [kWh]	23,875	14,429	0	43,406
Self-consumed %	23	14	0	46

Table 4.16: Destination of the electricity produced by the community

In considering the data, some considerations are suggested for each case study.

- Shared electricity: The solution with the highest percentage of shared electricity in the community is the third solution, "All-Consumers Community" which assures 72 % of the

total electricity as shared. When it comes to the potential revenue from the incentives, this solution is the one that can guarantee more revenue from this point of view, as will be seen later in the dedicated subsection. It should also be borne in mind that the aforementioned revenues must be distributed among the members according to their financing plan. The first scenario, "Blended Community" also offers a high percentage of shared energy, 50 % of the total, due to the large presence of consumers and storage.

- Sold electricity: In the second option, where there is no storage in which to keep the electricity, users are forced to sell it, exactly the 65 % of the total production. The benefits of this option depend on the tariffs at which the energy can be sold. The fourth scenario, with prosumers producing and storing energy, allows less energy to be sold, only 20 % of the total, favouring sharing and self-consumption.
- Self-Consumption: Again, this parameter is used to compare the scenarios. The fourth scenario seems to favour this aspect more, as 46 % of the energy produced is shared, while the second scenario does not favour this at all and only allows 14 % for self-consumption.

From the community's point of view, the community's purpose could lead to a decision based on these parameters. If the ultimate goal is to create more incentives to redistribute to members, the third option is the one that better enables this. On the other hand, if the objective is to save energy and encourage self-consumption, the fourth option is more appropriate. From a community perspective, the purpose of the community could lead to a decision based on these parameters. If the ultimate goal is to create more incentives for redistribution to the members, the third option is the one that better enables this. On the other hand, if the goal is to save energy and encourage self-consumption, the fourth option is more appropriate.

Grid balances

As far as the exchange with the grid is concerned, another aspect that could be evaluated is the energy sold to the grid and the energy bought from it. These two values are evaluated as a proportion of the total energy produced and sold, from which the external impact on the community can be seen. In the Table 4.17 the percentages are compared to allow a better understanding.

As can be seen from the previous comments, the second scenario is the one in which more is exchanged than in the others, buying the 54 % of the electricity needed and selling the 65 % of the produced, which makes it a particular option that requires certain constraints and incentives. In contrast, among the other three scenarios, the third scenario is the one that has the best net balance, allowing the community to require less than the others and sell more. The first option is also interesting from that point of view, reducing the amount of energy needed from outside more than what a community of only prosumers would do.

Parameter	Case 1 Blended	Case 2 No-Storage	Case 3 All-Consumers	Case 4 All-Prosumers
Sold electricity [kWh]	28411	68263	30,007	18964
% of sold electricity	27	65	28	20
Bought electricity [kWh]	4331	44094	3135	6441
% of bought electricity	5	54	4	8

Table 4.17: Percentages of the produced electricity that is sold and of the consumed electricity that is bought

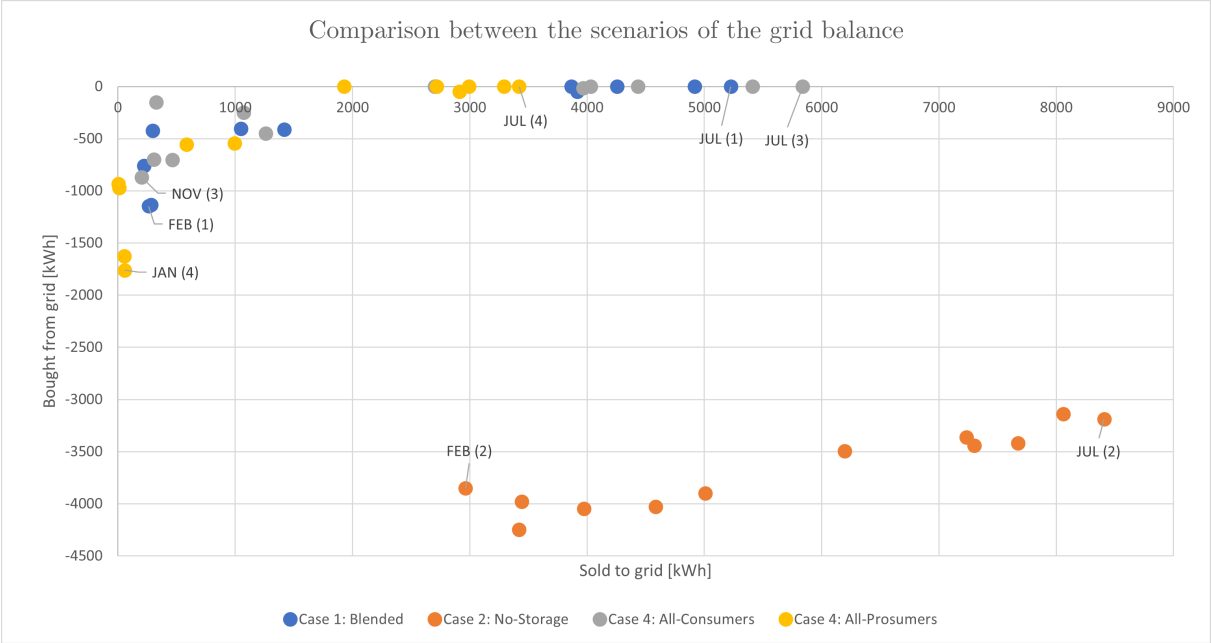

Figure 4.1: Comparison between each months' import and export from the grid

In Figure 4.1 can be seen the performance of each month using a scatter plot where the x-axis represents export to the grid and the y-axis represents import from the grid. The most remarkable result is the distance between the orange dots representing the No-Storage scenario and the other dots, which shows that a larger amount of energy is exchanged in each month. If you look at the points of the other three scenarios, you will see that the grey points representing the "All- Consumers" scenario are further along the x-axis, showing that in this case fewer imports are needed, and that the yellow points representing the "All-Prosumers" scenario have the lowest exports and the highest imports in the winter months.

Community efficiency

The parameter that should define the independence potential of a community is the length of time it is self-sufficient and does not need electricity imports from the grid. It is assessed by this parameter, which is calculated as the ratio between the time the community is able to be self-sufficient with the energy generated and stored divided by the total time.

It differs from the simple production/consumption balance, as the former does not depend

on time, but only on the final value. The simple balance does not take into account the impact of the batteries and the peaks in the production phase that can occur when the electricity is produced with solar energy, for example, but on the contrary counts the hours when the municipality does not import energy. It is a parameter that can be scaled from the daily period to the annual period. The following Table 4.18 show the number of hours and the percentage of the total number of hours in the month in which the municipality is self-sufficient.

Month	Case 1		Case 2		Case 3		Case 4	
	[Hours]	[%]	[Hours]	[%]	[Hours]	[%]	[Hours]	[%]
1	471	63	228	31	630	85	294	39
2	408	61	199	30	566	84	253	38
3	621	84	294	3	664	89	561	75
4	703	98	350	49	715	99	695	97
5	744	100	385	52	744	100	740	99
6	720	100	389	54	720	100	720	100
7	744	100	409	55	744	100	744	100
8	744	100	371	50	744	100	744	100
9	720	100	330	46	720	100	719	100
10	632	85	289	39	716	96	528	71
11	531	74	252	35	588	82	426	59
12	611	82	266	36	711	96	415	56
Total	7648	87	3760	43	8261	94	6836	78

Table 4.18: Summary of the monthly and yearly hours in which the community is self-sufficient

Again, the community with more hours of self-sufficiency is the third which can self-sustain itself for 94 % of the time. In general, self-sufficiency in cases 1, 3 and 4 is more than 95 % from April to September, but the difference is in the winter months. Number 3 maintains even longer moments of independence and never drops below 80 % at any time.

Figure 4.2 shows a monthly comparison between the efficiency for each scenario. The orange bars referring to the second case "No-Storage" are clearly the lowest in each month, but in winter the yellow bars for the "All-Prosumers" are also quite low, despite their high performance in summer. The good results of the third scenario "All- Consumers" are illustrated by the high bars in each month, showing their high efficiency.

Economic evaluation

The other aspect that influences the decision-making process in planning a Renewable Energy Community is the economic side. As mentioned in Chapter 1.3.3, the incentives are given to the community as a whole, which then has the duty to distribute them to the members according to their share or contribution.

The Italian scheme, as described in Chapter 1.3.3, pays 110 [€/MWh] for the shared energy, plus 8 [€/MWh] from ARERA for the benefit of the system, plus the price for purchasing the

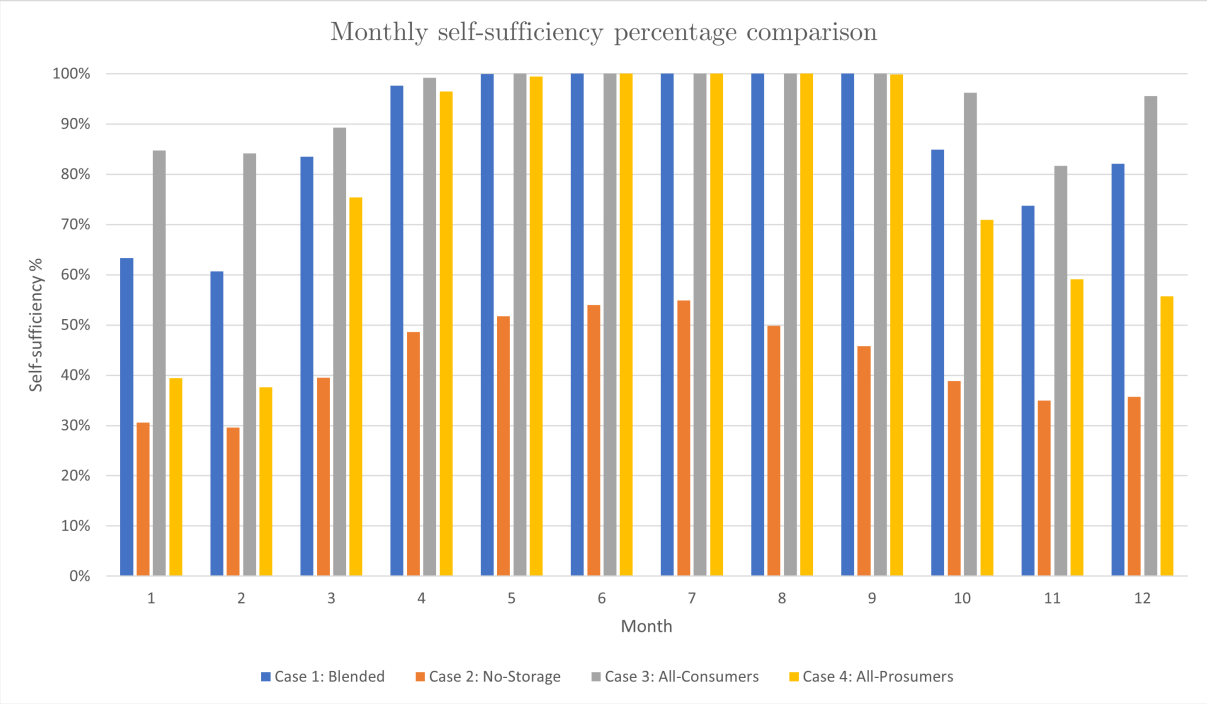

Figure 4.2: Comparison of the monthly efficiency in each four cases

electricity. This last price was chosen as the guaranteed minimum price of 40.7 [€/MWh] in 2022 due to the high volatility of the energy market.

Parameter	Case 1 Blended	Case 2 No-Storage	Case 3 All-Consumers	Case 4 All-Prosumers
Total produced energy [kWh]	105,469	105,469	108,238	93,860
Shared [kWh]	53,183	22,777	78,231	31,490
Remuneration [€]	8,440.08	3,614.67	12,418.32	4,997.50
Sold [kWh]	28,411	68,263	30,007	18,964
Remuneration [€]	1,156.34	2,778.30	1,220.29	771.85
Self-consumed [kWh]	23,875	14,429	0	43,406
Remuneration [€]	3,788.94	2,289.89	0	6,888.45
Total remuneration [€]	13,385	8,683	13,637	12,658
Average remuneration [€/MWh]	126.91	82.33	125.99	134.86

Table 4.19: Evaluation of the incentives on the balance of the produced energy

The information contained in Table 4.19 provide another parameter that the community can use to evaluate community performance. The results of the third scenario are those that offer the highest remuneration due to a higher participation and a higher amount of energy sold to the grid. The final 13,636.61 € that the community would receive at the end of the year must then be returned to the community in accordance with their ordinance.

Figure 4.3 shows the income item that makes up the national incentives. Shared electricity is the highest item for every scenario except the fourth, showing that they rely primarily on income from consuming the electricity themselves. The case that relies more on revenue for

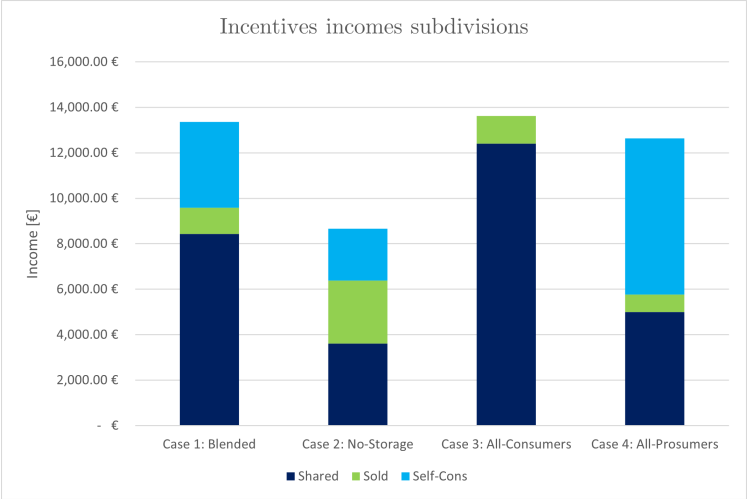

Figure 4.3: Subdivision of the incomes in the

sales to the network is the second. The lack of storage greatly increases the likelihood of a deficit being filled via the grid, and the fluctuation price of electricity can mean that a community has no positive impact or even loses money.

As expected, the second scenario was the one that allowed for less revenue due to low self-consumption and sharing. Choosing a different price paid by the government for the electricity sold would have changed the total remuneration significantly.

It is also possible to compare the electricity saved that would have been purchased each month without the use of a community with the four scenarios, using the prices listed in Table 1.3.

Scenario	2018 [€]	2022 [€]	2022 max [€]
Case 1	15698.03	37501.71	50807.90
Case 2	7588.16	18127.69	24559.69
Case 3	15941.75	38083.94	51596.72
Case 4	15267.30	36472.70	49413.77
Price [€/kWh]	0.204	0.487	0.660

Table 4.20: Saved money for electricity purchase

This comparison was made by calculating the difference between the total amount of electricity that would have been purchased under normal conditions without a community and the money spent to purchase the electricity needed in the presence of a community. The result was that, as expected, the second scenario does not provide too much savings compared to the other scenarios, even though it still gives users an advantage.

Another possible evaluation, that gives more an idea of the advantage of being in a community, is a balance between the incentives and the expenses for electricity, listed in Table 4.21.

	Case 1 [€]	Case 2 [€]	Case 3 [€]	Case 4 [€]
Incentives	13364.26	8661.76	13614.96	12639.02
2018 (mean) Expenses	-883.1	-8992.97	-639.38	-1313.85
Difference	*12481.15*	*-331.21*	*12975.58*	*11325.18*
2022 (mean) Expenses	-2109.68	-21483.7	-1527.45	-3138.7
Difference	*11254.57*	*-12821.94*	*12087.51*	*9500.32*
2022 (max) Expenses	-2858.23	-29106.45	-2069.41	-4252.36
Difference	*10506.03*	*-20444.69*	*11545.55*	*8386.66*

Table 4.21: Balance of the communities depending on the electricity purchase prices.

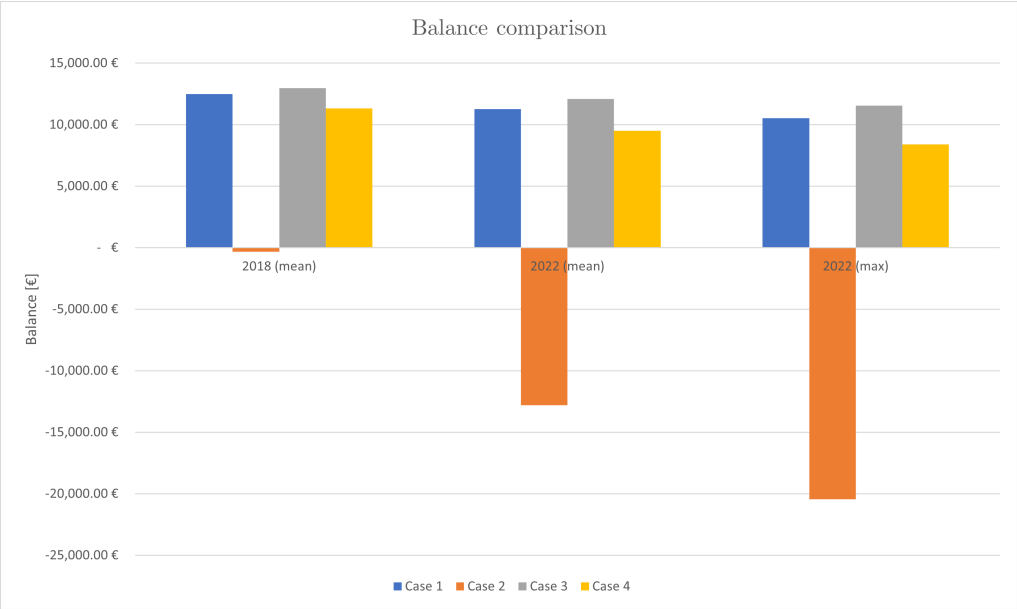

Figure 4.4: Visual comparison between the balance with different pricing

In this final comparison, it is also noticeable from Table 4.21 and Figure 4.4 that in Case 2, the reduction in incentives for shared and self-consumed energy due to the lack of storage not only leads to lower revenues, but also does not generate profits for the users. At the contrary, the increase of the energy prices still generates an income among the members of the community, with the third scenario being the most profitable one. Of course, reducing expenses is always a benefit, but the possibility of earning an income by selling energy to the grid can always make a difference, especially when dealing with small, isolated communities or low-income families.

CHAPTER 5

Conclusions

In this final chapter, it will be firstly given a brief overview of the work itself and the results obtained. Then, the perspective of social and economical benefits will be explained, before taking a look at the future challenges that may arise, as well as an overview of the possible improvements that can be made to the tool to add more features and possibilities.

The work began with a definition of the current situation of electricity generation in the European Union, Italy and Pantelleria. It also analyse the regulatory aspects conditioning the establishment of a Renewable Energy Community. Next, in the second chapter, are discussed the aspects that influence the behaviour and governance of a Renewable Energy Community, defining the principles of energy sharing. The second part explains the technical aspects related to the technologies involved in the community. The third chapter defines the four scenarios used for the simulation and the code itself, explained starting from the individual functions to calculate the energy consumed and produced. Then the behaviour of the batteries and the equations controlling it are analysed and finally are described in detail the code, the logical decisions and equations used to calculate the final balance of the community. Finally, in the fourth chapter, the results obtained in each scenario are listed, described and compared. In the end, in this chapter, are given comments and conclusions from this work.

5.1 Benefits for the Communities

The crucial importance of the engagement phase in the process of creating energy communities is influenced by how these initial steps are carried out. Being able to simulate and see what the results might look like is an important help for users who are ready to move forward in energy generation.

The four scenarios have been developed to have the possibility to cover some of the possible solutions that can be proposed in the engagement phase.

In the first scenario, *Blended Community*, the users are consumers and prosumers, some of them with individual storage, all supplied and backed up by a central power plant and storage. The results do not show best performance in any particular category, but neither do they have any particular problem. In 87% of the time, or about 317 days, the community is self-sufficient. The incentives received amount to about 13400 [€], about 127 [€/MWh], thanks to the high amount of energy shared, 53.2 [MWh] and energy sold, 28.4 [MWh].

The second, the *No-Storage Community*, has the same characteristics as the first, but has no storage at all, either individual or communal. The result is a community that can only be self-sufficient for the 43% of the year, 157 days, which corresponds to the time when users produce more than the consumption. The result is that more electricity has to be imported from the grid, about 44.0 [MWh], but also more is sold to the grid, 68.3 [MWh]. The incentives earned due to the low share of electricity are also the lowest of all, less than 8700 [€], only 82 [€/MWh]

In the third, the *All-Consumers Community*, all users are consumers who get their electricity from a central power plant and store it in a central storage facility. This results in the highest time of self-sufficiency, namely 8261 hours, which corresponds to 343 days. In this scenario, the highest amount of shared energy was also achieved, resulting in a total compensation of more than 13600 €, which corresponds to 126 [€/MWh]. The high self-sufficiency time also contributed to the lowest electricity imports, 3.1 [MWh], from the grid and the second highest sales, 30.0 [MWh].

Finally, in the *All-Prosumers Community*, all users produce and store electricity in their homes without using a central plant or storage. In this case, the shared and sold electricity is not as high as in the other scenarios, 31.5 [MWh] and 19.0 [MWh] respectively, but registered 43.4 [MWh] of self-consumed electricity contributing to provide around 12600 [€], around 135 [€/MWh]. In terms of self-sufficiency, the result is not as promising as the others, as it only lasts 78% of the time, or about 285 days.

In the end, the calculation of the balance that would result if the expenditure on electricity was subtracted from the incentives showed that some savings were possible in each case study. With the exception of the second scenario, the community is always able to earn around 10000[€], even with the high prices occurred in 2022.

In summary, this work was also a practical demonstration of how everyone can have the opportunity to earn and ensure energy safety while engaging in a community. Ultimately, the ability to simulate, analyse and compare the possible configurations can help renewable energy communities decide how to build the community in the engagement phase and how to realise its potential and benefits in economic and environmental terms.

5.2 Challenges for the Communities

The challenges that Renewable Energy Communities will face over the years are many and varied, and the outcome is also influenced by how these are designed.

Firstly, a poor design could lead to dissatisfaction among citizens in the long run and not entice them to continue and establish more communities. Secondly, if their numbers increase greatly and they collude with each other, there could be unhealthy competition and important aspects such as social outcomes might not be achieved. The scalability of the communities established is important to connect the communities and attract more members, and therefore also needs to be considered. From a technical point of view, the management of the grid could be difficult due to the potentially high number of small decentralised producers, and a new organisation of energy distribution could require further studies.

Finally, it have to be considered the influence that national and international policies will have on the new communities in the form of incentives and policies that could facilitate or suddenly hinder their development, as well as lobbying from the large energy companies that would be economically harmed by a self-sufficient society.

5.3 Improvements and future developments

Although this work is able to provide an analysis of the most common scenarios, potentially there could be more solutions and more possibilities. The first improvement that would complete and expand the potentialities of this program is the addition of more user typologies, such as commercial activities or public buildings, which would offer the simulation of their behaviour throughout the year giving more depth to the program.

Another aspect that can be worked on to improve the capabilities of this program is the inclusion of other power generation technologies such as wind turbines, wave power plants or hydro turbines. Also on the storage side, a possible improvement is the inclusion of other technologies, such as the behaviour of a fuel cell used to store and generate electricity using hydrogen, or a pumped hydro storage power plant that can work for the whole community.

Users could also have the possibility to add their own device list, occupancy profile or directly their consumption profile, to adapt their behaviour in a tailored simulation. From their point of view, an integrated financial instrument that can calculate economic aspects such as the initial investment, mortgages, incentives or revenues could also be a crucial aspect that could motivate the creation and exploitation of the true potential of the Renewable Energy Communities during the engagement phase.

Appendices

APPENDIX A

Appendix A

This appendix is used to describe and define the code of the software defined in the previous chapters with the Python syntax used in coding. Section A.1 shows in detail the first function that calculates the consumption simulation as described in 3.2, while the following Section A.2 describes the code for the second function that uses PVGIS to calculate the productivity simulation. This is followed by Section A.3, which presents the first part of the main code, while the clue part, which is also described by the Activity Diagram in Section 3.5, is shown in detail in Section A.4.

A.1 Consumption simulation

This part of the Appendix section illustrates the function used in Section 3.2 to generate the simulation of the yearly production.

A.1.1 Import libraries, files and folders

The first part introduces all the variables, files and packages used in the code

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import datetime
import os
import random

def hh_cons_rand(settings, folders, rec_members, cons_load_df, pros_load_df,
    pros_storage_df, cons_storage_df, rec_storage_df):

    # Files
    inhabitants_csv = os.path.join(folders
```

```
        ['households_consumption'],'inhabitants.csv')
    refload_csv = os.path.join(folders
        ['households_consumption'],'reference_load_occupation.csv')
    outputloadhh_cons_csv = os.path.join(folders
        ['output_folder'],'households_yearly_load_cons.csv')
    outputloadhh_pros_csv = os.path.join(folders
        ['output_folder'],'households_yearly_load_pros.csv')
    inputrec_xlsx = settings['input_file']
```

A.1.2 Read and manipulate database

This part creates the Pandas dataframes for the consumers and the prosumers, and start the indexing to divide weekdays and weekends.

```
    #%% READ DATABASE AND INPUTS

    inhabitants_df = pd.read_csv(inhabitants_csv,sep=';')
    refload_df = pd.read_csv(refload_csv,sep=';')
    # input_df_old = pd.read_excel(inputconshh_xlsx)
    input_consumers_df = rec_members['Consumers']
    input_prosumers_df = rec_members['Prosumers']
    input_consumers_df = pd.read_excel(inputrec_xlsx, 'Consumers', skiprows = 1)
    input_prosumers_df = pd.read_excel(inputrec_xlsx, 'Prosumers', skiprows = 1,
        usecols = [i for i in range(5)])
    input_df = pd.concat([input_consumers_df, input_prosumers_df])


    #%% MANIPULATE DATABASE

    # Add sampled interval
    refload_df['ref_sample'] = np.ceil(refload_df['minute'].div(settings
        ['sampling_minutes']))

    # Resample
    samp_load = refload_df.groupby([refload_df.index//settings
        ['sampling_minutes'],refload_df['WE'],refload_df['ref_sample']]).mean()
    samp_load = samp_load.drop(columns=['minute','id'])
    samp_load['WE'] = samp_load.index.get_level_values(1)
    samp_load['ref_sample'] = samp_load.index.get_level_values(2)
    samp_load.index = samp_load.index.droplevel(level=[1,2])
```

A.1.3 Power load for every consumer

This part of the code contains a loop that is able by iterating for each timestamp to copy the values from the database of average consumption to the correspondant day and hour.

```
    #%% CREATE POWER LOAD FOR EVERY REC CONSUMER

    df_hh_load_cons = pd.DataFrame(columns=["member_id","timestamp","load_kW"])

    for index, row in input_consumers_df.iterrows():

        load_memb_cons = pd.DataFrame(columns=["member_id","timestamp","load_kW"])

        # Timestamp
        load_memb_cons['timestamp'] = pd.date_range('01/01/'+str(settings
            ['reference_year'])+' 00:00','31/12/'+str(settings['reference_year'])+
                ' 23:59',freq=str(settings['sampling_minutes'])+'min')

        # ref_sample calculation
        load_memb_cons['ref_sample'] = (load_memb_cons['timestamp'].dt.minute +
            60*load_memb_cons['timestamp'].dt.hour)/settings['sampling_minutes']+1

        # Member id
        load_memb_cons = load_memb_cons.assign(member_id = row
            ['member_id'].astype(int))
        # load_memb = load_memb.assign(member_id = row['member_id'])

        # Inhabitants id
        in_id = inhabitants_df[(inhabitants_df['n_inhabitants_tot']==row
            ['family_members']&\(inhabitants_df['n_non_working_inhabitants']==
            row['non_working_inhabitants'])]['id'].values[0]
        load_memb_cons = load_memb_cons.assign(inhabitants_id = in_id)

        # Include columns to enable join
        load_memb_cons['month'] = load_memb_cons['timestamp'].dt.month
        load_memb_cons['WE'] = 0*(load_memb_cons['timestamp'].dt.dayofweek<=4) +
            1*(load_memb_cons['timestamp'].dt.dayofweek>4)

        # Merge load_memb and samp_load
        load_memb_cons = load_memb_cons.merge(samp_load,how='left',left_on=
            ['inhabitants_id','month','WE','ref_sample'],right_on=
            ['inhabitants_id','month','WE','ref_sample'])

        # load_kW
        load_memb_cons['load_kW'] = load_memb_cons['load_W']/1000
        load_memb_cons = load_memb_cons.drop(columns=['load_W'])

        # Weight if necessary
        if np.isfinite(row['yearly_consumption_kWh']):
            load_memb_cons['load_kW'] = load_memb_cons['load_kW']/load_memb_cons
            ['load_kW'].sum()*row['yearly_consumption_kWh']*4

        # Update dataframe
        df_hh_load_cons = df_hh_load_cons.append(load_memb_cons)

    # Export
```

```
52    df_hh_load_cons.to_csv(outputloadhh_cons_csv,sep=';')
```

A.1.4 Randomization for consumers

This part is the one that randomize the consumption values as explained in Section 3.2.2 referred to the consumers dataframe.

```
1     #%% RANDOMIZATION FOR CONSUMER
2
3     df_hh_load_cons['delta'] = settings['base_variation'] +
4         df_hh_load_cons['non_working'] * settings['individual_variation']
5     df_hh_load_cons['randNumCol'] = (np.random.randint(-df_hh_load_cons
6         ['delta'],df_hh_load_cons['delta']+1, size=len(df_hh_load_cons))/100)+1
7     df_hh_load_cons['variation_\%'] = (df_hh_load_cons['randNumCol']-1)*100
8     df_hh_load_cons['random_load_kW'] = df_hh_load_cons['randNumCol']*
9         df_hh_load_cons['load_kW']
```

A.1.5 Power load for every prosumer

This part is equal to the one shown in Section A.1.3, but instead of copying the values of the profiles into the data frame of the consumers, this time they are appended to the data frame of the prosumers.

```
1     #%% CREATE POWER LOAD FOR EVERY REC PROSUMER
2
3     df_hh_load_pros = pd.DataFrame(columns=["member_id","timestamp","load_kW"])
4
5     for index, row in input_prosumers_df.iterrows():
6
7         load_memb_pros = pd.DataFrame(columns=["member_id","timestamp","load_kW"])
8
9         # Timestamp
10        load_memb_pros['timestamp'] = pd.date_range('01/01/'+str(settings
11            ['reference_year'])+' 00:00','31/12/'+str(settings['reference_year'])+
12                ' 23:59',freq=str(settings['sampling_minutes'])+'min')
13
14        # ref_sample calculation
15        load_memb_pros['ref_sample'] = (load_memb_pros['timestamp'].dt.minute +
16            60*load_memb_pros['timestamp'].dt.hour)/settings['sampling_minutes']+1
17
18        # Member id
19        load_memb_pros = load_memb_pros.assign(member_id = row
20            ['member_id'].astype(int))
21        # load_memb = load_memb.assign(member_id = row['member_id'])
22
23        # Inhabitants id
```

```
        in_id = inhabitants_df[(inhabitants_df['n_inhabitants_tot']==row
            ['family_members']&\(inhabitants_df['n_non_working_inhabitants']==
            row['non_working_inhabitants'])]['id'].values[0]
        load_memb_pros = load_memb_pros.assign(inhabitants_id = in_id)

        # Include columns to enable join
        load_memb_pros['month'] = load_memb_pros['timestamp'].dt.month
        load_memb_pros['WE'] = 0*(load_memb_pros['timestamp'].dt.dayofweek<=4) +
            1*(load_memb_pros['timestamp'].dt.dayofweek>4)

        # Merge load_memb and samp_load
        load_memb_pros = load_memb_pros.merge(samp_load,how='left',left_on=
            ['inhabitants_id','month','WE','ref_sample'],right_on=
            ['inhabitants_id','month','WE','ref_sample'])

        # load_kW
        load_memb_pros['load_kW'] = load_memb_pros['load_W']/1000
        load_memb_pros = load_memb_pros.drop(columns=['load_W'])

        # Weight if necessary
        if np.isfinite(row['yearly_prosumption_kWh']):
            load_memb_pros['load_kW'] = load_memb_pros['load_kW']/load_memb_pros
            ['load_kW'].sum()*row['yearly_prosumption_kWh']*4

        # Update dataframe
        df_hh_load_pros = df_hh_load_pros.append(load_memb_pros)

    # Export
    df_hh_load_pros.to_csv(outputloadhh_pros_csv,sep=';')
```

A.1.6 Randomization for prosumers

This part is the one that randomize the prosumption values as explained in Section 3.2.2 referred to the prosumers dataframe.

```
    #%% RANDOMIZATION FOR PROSUMER

    df_hh_load_pros['delta'] = settings['base_variation'] +
        df_hh_load_pros['non_working'] * settings['individual_variation']
    df_hh_load_pros['randNumCol'] = (np.random.randint(-df_hh_load_pros
        ['delta'],df_hh_load_pros['delta']+1, size=len(df_hh_load_pros))/100)+1
    df_hh_load_pros['variation_\%'] = (df_hh_load_pros['randNumCol']-1)*100
    df_hh_load_pros['random_load_kW'] = df_hh_load_pros['randNumCol']*
        df_hh_load_pros['load_kW']
```

A.1.7 Creation of the database

This last part of the code is done to create the databases that will be used by the main part of the code to perform all the calculations.

```
    #%% CREATE THE DATABASE

    pros_load_df['member_id'] = df_hh_load_pros['member_id']
    cons_load_df['member_id'] = df_hh_load_cons['member_id']
    pros_load_df['timestamp'] = df_hh_load_pros['timestamp']
    cons_load_df['timestamp'] = df_hh_load_cons['timestamp']
    pros_load_df['load_kW'] = df_hh_load_pros['load_kW']
    cons_load_df['load_kW'] = df_hh_load_cons['load_kW']

    #%% IMPORT THE INITIAL BATTERY PARAMETERS

    input_prosumers_storage_df = pd.read_excel(inputrec_xlsx, 'Prosumers',
        skiprows = 1, usecols = [0, 13, 14, 15])
    pros_storage_df['member_id'] = input_prosumers_storage_df['member_id']
    pros_storage_df['storage_power_kW'] = input_prosumers_storage_df
        ['storage_power_kW']
    pros_storage_df['storage_energy_kWh'] = input_prosumers_storage_df
        ['storage_energy_kWh']
    pros_storage_df['storage_dod'] = input_prosumers_storage_df['storage_dod']

    input_consumers_storage_df = pd.read_excel(inputrec_xlsx, 'Consumers',
        skiprows = 1, usecols = [0, 5, 6, 7])
    cons_storage_df['member_id'] = input_consumers_storage_df['member_id']
    cons_storage_df['storage_power_kW'] = input_consumers_storage_df
        ['storage_power_kW']
    cons_storage_df['storage_energy_kWh'] = input_consumers_storage_df
        ['storage_energy_kWh']
    cons_storage_df['storage_dod'] = input_consumers_storage_df['storage_dod']

    input_rec_storage_df = pd.read_excel(inputrec_xlsx, 'Storage_facilities',
        skiprows = 1)
    rec_storage_df['member_id'] = input_rec_storage_df['member_id']
    rec_storage_df['storage_power_kW'] = input_rec_storage_df['storage_power_kW']
    rec_storage_df['storage_energy_kWh'] = input_rec_storage_df
        ['storage_energy_kWh']
    rec_storage_df['storage_dod'] = input_rec_storage_df['storage_dod']

    #%% RETURN
    return()
```

A.2 Productivity calculation

This second function has the goal to produce a dataframe that is going to contain the quarter-hour production values from the photovoltaic panels mounted, as explained in Section

3.3

A.2.1 Import libraries, files and folders

The first part, as explained in the previous section, has the objective to introduce all the libraries that are going to be used in the code.

```
#%% IMPORT

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import datetime
import os
import io
import requests
import json
import csv
import pickle

#%% FUNCTION

def pvgis(settings, folders, rec_members, pros_prod_df, plant_prod_df):

    #%% FOLDERS AND FILES
    inputrec_xlsx = settings['input_file']
```

A.2.2 Create dataframe for prosumers

This part reads the prosumers' database and import it to the code, then it is used to compose the link as explained in 3.3.

```
    #%% READ DATABASE AND IMPORT FILES FOR PROSUMERS
    input_pv_prosumers = pd.read_excel(inputrec_xlsx, 'Prosumers', skiprows = 1)
    input_pv_prosumers = input_pv_prosumers.drop(columns = ['family_members',
        'non_working_inhabitants', 'working_inhabitants', 'yearly_consumption_kWh',
    'storage_power_kW', 'storage_energy_kWh', 'storage_dod'])
    input_pv_prosumers['new_id'] = range(len(input_pv_prosumers))
    input_pv_prosumers.set_index('new_id', inplace = True)

    # link
    base_link = 'https://re.jrc.ec.europa.eu/api/'
    # tool name
    tool_name = 'seriescalc'
    # year timestamp
    year = '&startyear=%d&endyear=%d' % (settings['reference_year'],
        settings['reference_year'])

```

```
    # creating the link for every user involved
    input_pv_prosumers['lat&lon_str'] = '?lat=' + (input_pv_prosumers['lat'].
        astype(str)) + '&lon=' + (input_pv_prosumers['lon'].astype(str))
    input_pv_prosumers['power&loss_str'] = '&pvcalculation=1&peakpower=' +
        (input_pv_prosumers['peak_power_kW'].astype(str)) + '&loss=%d' %
        (settings['system_loss'])
    input_pv_prosumers['type&year_str'] = '&mountingtype=' + (input_pv_prosumers
        ['mounting_type']) + (year)
    input_pv_prosumers['tracking_num'] = np.where(input_pv_prosumers['tracking']
        == 'none', 0, (np.where(input_pv_prosumers['tracking'] ==
        'horizontal axis n/s',
        1, (np.where(input_pv_prosumers['tracking'] == 'two axis', 2,
        ((np.where(input_pv_prosumers['tracking'] == 'vertical axis', 3,5)))))))))
    input_pv_prosumers['trackingtype'] = '&trackingtype=' + (input_pv_prosumers
        ['tracking_num']).astype(str)
    input_pv_prosumers['optimization'] = np.where(input_pv_prosumers['tracking']
        == 'two axis', '', np.where(input_pv_prosumers['tracking'] == 'none',
        '&optimalangles=1', '&optimalinclination=1'))
    input_pv_prosumers['tilt_angle'] = np.where(input_pv_prosumers['tracking'] ==
        'two axis', 'nan', input_pv_prosumers['tilt_angle'])
    input_pv_prosumers['angle_str'] = ('&angle=' + input_pv_prosumers
        ['tilt_angle'].astype(str))*(input_pv_prosumers['tilt_angle'].
        astype(str) != 'nan')
    input_pv_prosumers['azimut'] = np.where(input_pv_prosumers['tracking'] !=
        'none', 'nan', input_pv_prosumers['azimut'])
    input_pv_prosumers['azimut_str'] = ('&azimut=' + input_pv_prosumers
        ['azimut'].astype(str))*(input_pv_prosumers['azimut'].astype(str)
        != 'nan')
    input_pv_prosumers['optimization'] = np.where(input_pv_prosumers
        ['angle_str'] == '', input_pv_prosumers['optimization'], '')
    input_pv_prosumers['optimization'] = np.where(input_pv_prosumers['tracking']
        == 'none' , np.where(input_pv_prosumers['azimut'] != 'nan' ,
        np.where(input_pv_prosumers['tilt_angle'] == 'nan' , '&optimalinclination=
        1' , '' ) , np.where(input_pv_prosumers['tilt_angle'] == 'nan' ,
        '&optimalangles=1' , '' ) ) , input_pv_prosumers['optimization'] )
    input_pv_prosumers['link'] = (base_link) + (tool_name) +
        input_pv_prosumers['lat&lon_str'] + input_pv_prosumers['power&loss_str'] +
        input_pv_prosumers['trackingtype'] + input_pv_prosumers['angle_str'] +
        input_pv_prosumers['azimut_str'] + input_pv_prosumers['optimization'] +
        input_pv_prosumers['type&year_str']
    input_pv_prosumers['status_code'] = 0

    production_dict_pros = {}
```

A.2.3 Manipulate prosumers' dataframe

This part of the code, once that with the previous are obtained the raw data, these are post processed and the final dataframe with the quarter-hour consumption is obtained for the prosumers.

```
    #%% LOOP FOR CREATE AND MANIPULATE EACH USER'S DF
    for row in range(len(input_pv_prosumers['member_id'])):
        userid = input_pv_prosumers.loc[row, 'member_id']
        print ('\nprosumer number ',userid, ':')
        sourcefilename = (os.path.join(folders['PV_production'],
            'production_csv_id%d_raw_data.csv' % (input_pv_prosumers.loc
            [row, 'member_id'])))
        response = requests.get(input_pv_prosumers.iloc[row]['link'])
        input_pv_prosumers.loc[row, 'status_code'] = response.status_code
        if response.status_code == 200:
            outputformat=csv
            browser=1
            with open(sourcefilename, 'w') as prod_csv:
                prod_csv.write(response.text)
            production_df_pros = pd.read_csv(sourcefilename, skiprows=range(17))
                production_df_pros.drop(production_df_pros.index[8761:8767],
                inplace = True, axis = 0)
            production_df_pros.rename(columns = {'time':'timestamp'},
                inplace = True)
            production_df_pros['timestamp'] = pd.date_range('01/01/'+str(settings
                ['reference_year'])+' 00:00','1/1/'+str(settings['reference_year']
                +1)+' 00:00',freq=str(60)+'T')
            production_df_pros = production_df_pros.drop(columns=['G(i)', 'H_sun',
                'T2m', 'WS10m', 'Int'])
            production_df_pros.set_index('timestamp', inplace = True)
            production_df_pros['P'] = production_df_pros['P'].astype(float)
            production_df_pros = production_df_pros.resample('15T').interpolate()
            production_df_pros.drop(production_df_pros.index[35040], inplace =
                True, axis = 0)
            production_df_pros['Production_kWh'] = production_df_pros['P']/4000
            production_df_pros['member_id'] = userid
            production_df_pros.rename(columns = {'P':'prod_kW'}, inplace = True)
            production_df_pros['prod_kW'] = production_df_pros['prod_kW']/1000
            production_dict_pros['PV_production_id{0}_df'.format(input_pv_prosumers.
                loc[row, 'member_id'])] = production_df_pros

            print ('completed\n')
        # in case I want to save each df in a csv file
            savefilename = (os.path.join(folders['PV_production'],
                'PV_production_id%d_csv.csv' % (input_pv_prosumers.loc
                [row, 'member_id'])))
            production_df_pros.to_csv(savefilename, sep=';')
        else:
            error_user = input_pv_prosumers.loc[row, 'member_id'].astype(str)
            outputformat=csv
            browser=1
            with open(sourcefilename, 'w') as prod_csv:
                prod_csv.write(response.text)
            print('error for prosumer ' + error_user + ', check the file
                <<production_csv_id' + error_user + '_raw_data.csv>> for details')
    #%% CREATION OF A NEW DF WITH ALL THE DF
    prod_df_all_pros = pd.concat(production_dict_pros, axis = 0)
```

```
    pros_prod_df['member_id'] = prod_df_all_pros['member_id']
    pros_prod_df['prod_kW'] = prod_df_all_pros['prod_kW']
```

A.2.4 Create dataframe for plants

This part reads the plants' database and import it to the code, then it is used to compose the link as explained in 3.3.

```
    #%% READ DATABASE AND IMPORT FILES FOR PLANTS

    # input_pv_df = pd.read_excel(input_pv_xlsx)
    # input_pv_prosumers = pd.read_excel(inputrec_xlsx, 'Prosumers', skiprows = 1)
    input_pv_plants = pd.read_excel(inputrec_xlsx, 'Production_plants',
    skiprows = 1)

    input_pv_plants['new_id'] = range(len(input_pv_plants))
    input_pv_plants.set_index('new_id', inplace = True)

    # link
    base_link = 'https://re.jrc.ec.europa.eu/api/'
    # tool name
    tool_name = 'seriescalc'
    # year timestamp
    year = '&startyear=\%d&endyear=\%d' \% (settings
        ['reference_year'], settings['reference_year'])

    # creating the link for every user involved
    input_pv_plants['lat&lon_str'] = '?lat=' + (input_pv_plants['lat'].astype(str))
        + '&lon=' + (input_pv_plants['lon'].astype(str))
    input_pv_plants['power&loss_str'] = '&pvcalculation=1&peakpower=' +
        (input_pv_plants['peak_power_kW'].astype(str)) + '&loss=\%d' \%
        (settings['system_loss'])
    input_pv_plants['type&year_str'] = '&mountingtype=' + (input_pv_plants
        ['mounting_type']) + (year)
    input_pv_plants['tracking_num'] = np.where(input_pv_plants['tracking'] ==
        'none', 0, (np.where(input_pv_plants['tracking'] == 'horizontal axis n/s',
        1, (np.where(input_pv_plants['tracking'] == 'two axis', 2,
        ((np.where(input_pv_plants['tracking'] == 'vertical axis', 3,5)))))))))
    input_pv_plants['trackingtype'] = '&trackingtype=' + (input_pv_plants
        ['tracking_num']).astype(str)
    input_pv_plants['optimization'] = np.where(input_pv_plants['tracking'] ==
        'two axis', '', np.where(input_pv_plants['tracking'] == 'none',
        '&optimalangles=1', '&optimalinclination=1'))
    input_pv_plants['tilt_angle'] = np.where(input_pv_plants['tracking'] ==
        'two axis', 'nan', input_pv_plants['tilt_angle'])
    input_pv_plants['angle_str'] = ('&angle=' + input_pv_plants['tilt_angle'].
        astype(str))*(input_pv_plants['tilt_angle'].astype(str) != 'nan')
    input_pv_plants['azimut'] = np.where(input_pv_plants['tracking'] != 'none',
        'nan', input_pv_plants['azimut'])
    input_pv_plants['azimut_str'] = ('&azimut=' + input_pv_plants['azimut'].
```

```
        astype(str))*(input_pv_plants['azimut'].astype(str) != 'nan')
    input_pv_plants['optimization'] = np.where(input_pv_plants['angle_str'] == '',
        input_pv_plants['optimization'], '')
    input_pv_plants['optimization'] = np.where(input_pv_plants['tracking'] ==
        'none' , np.where(input_pv_plants['azimut'] != 'nan' , np.where
        (input_pv_plants['tilt_angle'] == 'nan' , '&optimalinclination=1' , '' )
        , np.where(input_pv_plants['tilt_angle'] == 'nan' , '&optimalangles=1' ,
        '' ) ) , input_pv_plants['optimization'] )
    input_pv_plants['link'] = (base_link) + (tool_name) +
        input_pv_plants['lat&lon_str'] + input_pv_plants['power&loss_str'] +
        input_pv_plants['trackingtype'] + input_pv_plants['angle_str'] +
        input_pv_plants['azimut_str'] + input_pv_plants['optimization'] +
        input_pv_plants['type&year_str']
    input_pv_plants['status_code'] = 0
    production_dict_plants = {}
```

A.2.5 Manipulate plants' dataframe

This part of the code, once that with the previous are obtained the raw data, these are post processed and the final dataframe with the quarter-hour consumption is obtained for the prosumers.

```
    #%% LOOP FOR CREATE AND MANIPULATE EACH USER'S DF
    for row in range(len(input_pv_plants['member_id'])):
        userid = input_pv_plants.loc[row, 'member_id']
        print ('plant number ',userid, ':')
        sourcefilename = (os.path.join(folders['PV_production'],
            'production_csv_id\%d_raw_data.csv'\% (input_pv_plants.loc
            [row, 'member_id'])))
        response = requests.get(input_pv_plants.iloc[row]['link'])
        input_pv_plants.loc[row, 'status_code'] = response.status_code
        if response.status_code == 200:
            outputformat=csv
            browser=1
            with open(sourcefilename, 'w') as prod_csv:
                prod_csv.write(response.text)
            production_df_plants = pd.read_csv(sourcefilename, skiprows=range(17))
            production_df_plants.drop(production_df_plants.index[8761:8767],
                inplace = True, axis = 0)
            production_df_plants.rename(columns = {'time':'timestamp'}, inplace =
                True)
            production_df_plants['timestamp'] = pd.date_range('01/01/'+str
                (settings['reference_year'])+' 00:00','1/1/'+str(settings
                ['reference_year']+1)+' 00:00',freq=str(60)+'T')
            production_df_plants = production_df_plants.drop(columns=
                ['G(i)', 'H_sun', 'T2m', 'WS10m', 'Int'])
            production_df_plants.set_index('timestamp', inplace = True)
            production_df_plants['P'] = production_df_plants['P'].astype(float)
            production_df_plants = production_df_plants.resample('15T').
```

```
                interpolate()
            production_df_plants.drop(production_df_plants.index[35040], inplace =
                True, axis = 0)
            production_df_plants['Production_kWh'] = production_df_plants['P']/4000
            production_df_plants['member_id'] = userid
            production_df_plants.rename(columns = {'P':'prod_kW'}, inplace = True)
            production_df_plants['prod_kW'] = production_df_plants['prod_kW']/1000
            production_dict_plants['PV_production_id{0}_df'.format(input_pv_plants
                .loc[row, 'member_id'])] = production_df_plants

            print ('completed\n')
        # save each df in a csv file
            savefilename = (os.path.join(folders['PV_production'],
                'PV_production_id\%d_csv.csv' \% (input_pv_plants.loc
                [row, 'member_id'])))
            production_df_plants.to_csv(savefilename, sep=';')
        else:
            error_user = input_pv_plants.loc[row, 'member_id'].astype(str)
            outputformat=csv
            browser=1
            with open(sourcefilename, 'w') as prod_csv:
                prod_csv.write(response.text)
            print('error for plant ' + error_user + ', check the file
                <<production_csv_id' + error_user + '_raw_data.csv>> for details')

    #%% CREATION OF A NEW DF WITH ALL THE DF
    prod_df_all_plants = pd.concat(production_dict_plants, axis = 0)
    plant_prod_df['member_id'] = prod_df_all_plants['member_id']
    plant_prod_df['prod_kW'] = prod_df_all_plants['prod_kW']
```

Lastly, the function returns the values with

```
    #%% RETURN
    return ()
```

At this point, the production dataframes for the plants and the prosumers are created and can be used in the main code.

A.3 Main code

This final section explains in detail the main code, following the information from Section 3.5.

A.3.1 Import functions, settings, folders and files

This first part has the purpose, as the others, to initialize the various functions that are going to be used in the code, as well as files, folders and settings.

```
1  import pandas as pd
2  import os
3  import numpy as np
4  import sys
5  import pickle
6  import math
7  sys.path.insert(1,'Functions')
8
9  from households_consumption_randomized import hh_cons_rand
10 from pvgis_api import pvgis
11 from datetime import datetime
12
13 if __name__ == '__main__':
14
15
16     #%% SETTINGS
17
18     settings = {}
19     settings['sampling_minutes'] = 15
20     settings['reference_year'] = 2015
21     settings['base_variation'] = 10
22     settings['individual_variation'] = 10
23     settings['system_loss'] = 14
24
25     #%% FILES AND FOLDERS
26
27     # Folders
28     folders = {}
29     folders['path_parent'] = os.path.dirname(os.getcwd())
30     folders['data_folder'] = os.path.join(folders['path_parent'],'Data')
31     folders['database_folder'] = os.path.join(folders['data_folder'],'Database')
32     folders['households_consumption'] = os.path.join(folders['database_folder'],
33         'Households_consumption')
34     folders['input_folder'] = os.path.join(folders['data_folder'],'Input')
35     folders['output_folder'] = os.path.join(folders['data_folder'],'Output')
36     folders['PV_production'] = os.path.join(folders['output_folder'],
37         'PV_production')
38     folders['final_data'] = os.path.join(folders['output_folder'], 'final_data')
39
40     # Files
41     rec_members_xlsx = os.path.join(folders['input_folder'],'REC_members_id.xlsx')
42     consumers_data_csv = os.path.join(folders['final_data'], 'consumers_data.csv')
43     prosumers_data_csv = os.path.join(folders['final_data'], 'prosumers_data.csv')
44     plants_data_csv = os.path.join(folders['final_data'], 'plants_data.csv')
45     storage_data_csv = os.path.join(folders['final_data'], 'storage_data.csv')
46     rec_data_csv = os.path.join(folders['final_data'], 'rec_data.csv')
47     settings['input_file'] = rec_members_xlsx
```

It is worth mentioning that in line 41, the file name "REC_members_id.xlsx" it is the input file.

A.3.2 Reading the input files and dataframe creation

This part's purpose is to read the databases created with the two aforementioned functions and to use the information. Then, it creates a dedicated dataframe for consumers, prosumers, plants and the overall REC.

```
#%% READING INPUT FILE

# Excel File
rec_members_list_sheets = pd.ExcelFile(rec_members_xlsx)

# New dictionary
rec_members = {}

# Saving every dataframe
for sheet_name in rec_members_list_sheets.sheet_names:
    rec_members[sheet_name] = rec_members_list_sheets.parse(sheet_name,header=1)

#%% ENERGY CONSUMPTION OF PROSUMERS & CONSUMERS

# Creating the preliminary dataframes for consumers and prosumers
cons_load_df = pd.DataFrame(columns=['member_id', 'timestamp', 'load_kW'])
pros_load_df = pd.DataFrame(columns=['member_id', 'timestamp', 'load_kW'])
cons_storage_df = pd.DataFrame(columns=['member_id', 'storage_power_kW',
    'storage_energy_kWh', 'storage_dod'])
pros_storage_df = pd.DataFrame(columns=['member_id', 'storage_power_kW',
    'storage_energy_kWh', 'storage_dod'])
rec_storage_df = pd.DataFrame(columns=['member_id', 'timestamp',
    'storage_power_kW', 'storage_energy_kWh', 'storage_dod'])

# Calling the function
hh_cons_rand(settings, folders, rec_members, cons_load_df, pros_load_df,
    pros_storage_df, cons_storage_df, rec_storage_df)

#%% ENERGY PRODUCTION OF PROSUMERS & POWER PLANTS

# Create the preliminary dataframes for prosumers and plants
pros_prod_df = pd.DataFrame(columns=['member_id', 'prod_kW'])
plant_prod_df = pd.DataFrame(columns=['member_id', 'prod_kW'])

# Calling the function
pvgis(settings, folders, rec_members, pros_prod_df, plant_prod_df)
pros_prod_df.reset_index(inplace=True)
pros_prod_df = pros_prod_df.drop(columns=('level_0'))
plant_prod_df.reset_index(inplace=True)
plant_prod_df = plant_prod_df.drop(columns=('level_0'))

#%% PROSUMERS DF CREATION

# creates a df for the prosumers with the data obtained from the consumption
    # and the production functions
```

```
    prosumers_df = pd.DataFrame(columns=['member_id', 'timestamp', 'load_kWh',
        'prod_kWh'])
    prosumers_df['member_id'] = pros_load_df['member_id'].astype(int)
    prosumers_df['timestamp'] = pros_prod_df['timestamp']
    prosumers_df['load_kWh'] = pros_load_df['load_kW']*
        (settings['sampling_minutes']/60)
    prosumers_df.reset_index(inplace=True, drop=True)
    prosumers_df['prod_kWh'] = pros_prod_df['prod_kW']*
        (settings['sampling_minutes']/60)

    # creation of a column that represents the raw balance between the production
        # and the consumption without the battery's influence
    prosumers_df['raw_user_balance_t_kWh'] = prosumers_df['prod_kWh']-
        prosumers_df['load_kWh']

    #%% CONSUMERS DF CREATION

    # creates a df for the consumers with the data obtained from the
        # consumption function
    consumers_df = pd.DataFrame(columns=['member_id', 'timestamp', 'load_kWh',
        'prod_kWh'])
    consumers_df['member_id'] = cons_load_df['member_id'].astype(int)
    consumers_df['timestamp'] = cons_load_df['timestamp']
    consumers_df['load_kWh'] = cons_load_df['load_kW']*
        (settings['sampling_minutes']/60)
    consumers_df['prod_kWh'] = 0 # production = 0 because are only consumers
    # creation of a column that represents the raw balance between the production
        # and the consumption without the battery's influence
    consumers_df['raw_user_balance_t_kWh'] = consumers_df['prod_kWh']-
        consumers_df['load_kWh']

    #%% PLANTS DF CREATION

    # creates a df for the consumers with the data obtained from the production
        # function.
    plant_df = pd.DataFrame(columns=['member_id', 'timestamp', 'load_kWh',
        'prod_kWh'])
    plant_df['member_id'] = plant_prod_df['member_id']
    plant_df['timestamp'] = plant_prod_df['timestamp']
    plant_df['load_kWh'] = 0 # assumed that a production plant does not have
        # consumptions
    plant_df['prod_kWh'] = plant_prod_df['prod_kW']*(settings
        ['sampling_minutes']/60)
    # creation of a column that represents the raw balance between the production
        # and the consumption without the battery's influence
    plant_df['raw_user_balance_t_kWh'] = plant_df['prod_kWh'] - plant_df['load_kWh']

    #%% OVERALL REC

    # create a df that will comprehend producers, consumers and plants all together
    rec_df = pd.concat([prosumers_df, consumers_df, plant_df])
    rec_df['raw_user_balance_t_kWh'] = rec_df['prod_kWh'] - rec_df['load_kWh']
```

```
    # sort the values by timestamp and by member id to have them in order, then
        # reset the index values to allow iterations
    rec_df.sort_values(['timestamp', 'member_id'], ascending = [True, True],
        inplace = True)
    rec_df.reset_index(drop = True, inplace = True)
```

After this part, a dataframe for each category is created and these are going to be used in the following sections to calculate the balances.

A.3.3 Preparate the dataframes before the iteration

This part of the code prepares the dataframes to be inserted in the loop that will perform the calculation. Here, some values that will have constant values overtime are inserted to avoid an extra calculation each iteration, saving time.

```
    #%% CALCULATION OF THE BALANCES

    # creation of an appendix database with the added information needed to
        # calculate every timestamp
    appendix_df = pd.DataFrame(columns=['member_id',
        'previous_energy_in_battery_t_kWh', 'previous_dod',
        'energy_exchangeable_kWh', 'flag', 'energy_in_battery_t_kWh',
        'max_energy_dischargeable_t_kWh', 'previous_space_in_battery_t_kWh',
        'max_energy_chargeable_t_kWh', 'battery_balance_t_kWh',
        'user_and_battery_balance_t_kWh', 'space_in_battery_t_kWh',
        'energy_shared_out_t_kWh', 'energy_sold_to_grid_t_kWh',
        'energy_shared_in_t_kWh', 'energy_bought_from_grid_t_kWh'])

    # calculate the total number of users, it will be needed later to know by how
        # many rows it is composed
    number_of_timestamps = 35040 # 4 timestamp each hour, in 24 hours, in 365 days
    number_of_consumers = len(cons_storage_df)
    number_of_prosumers = len(pros_storage_df)
    number_of_plants = (len(plant_df)//number_of_timestamps)
    number_of_public_storages = len(rec_storage_df)

    # CONSUMERS
    # df creation, merging the consumers_df with the storage file and with the
        # appendix_df
    consumers_calc_df = pd.merge(consumers_df, cons_storage_df, on='member_id')
    consumers_calc_df = pd.merge(consumers_calc_df, appendix_df, how='outer')

    # calculation of the maximum energy that can be exchanged between the prosumers
        # and the storages
    consumers_calc_df['energy_exchangeable_kWh'] = consumers_calc_df
        ['storage_power_kW']*(settings['sampling_minutes']/60)
    consumers_len = len(consumers_calc_df) # lenght of then consumers dataframe, it
        was used to maniplulate the index, it might be needed later

```

```
    # indexing the df
    # sort the values by timestamp and by member id to have them in order, then
        # reset the index values to allow iterations
    consumers_calc_df.sort_values(['timestamp','member_id'], ascending=[True, True]
        ,inplace= True)
    consumers_calc_df.reset_index(drop = True, inplace = True)

    # since it is required that the starting dod is on the "previous_dod" column,
        # the values are moved into that column, while the other one is cleaned,
        # as well as the "previous dod" values for timestamps after the first one
    consumers_calc_df['previous_dod'] = consumers_calc_df['storage_dod'].copy()
    consumers_calc_df['storage_dod'] = np.nan
    consumers_calc_df.loc[range(number_of_consumers, number_of_timestamps),
        'previous_dod'] = np.nan

    # PROSUMERS
    # df creation, merging the prosumers_df with the storage file and with the
        # appendix_df
    prosumers_calc_df = pd.merge(prosumers_df, pros_storage_df, on='member_id')
    prosumers_calc_df = pd.merge(prosumers_calc_df, appendix_df, how='outer')

    # calculation of the maximum energy that can be exchanged between the prosumers
        # and the storages
    prosumers_calc_df['energy_exchangeable_kWh'] = prosumers_calc_df
        ['storage_power_kW']*(settings['sampling_minutes']/60)
    prosumers_len = len(prosumers_calc_df) # lenght of then prosumers dataframe,
        # it was used to maniplulate the index, it might be needed later

    # indexing the df
    # sort the values by timestamp and by member id to have them in order, then
        # reset the index values to allow iterations
    prosumers_calc_df.sort_values(['timestamp','member_id'], ascending=[True, True]
        ,inplace= True)
    prosumers_calc_df.reset_index(drop = True, inplace = True)

    # since it is required that the starting dod is on the "previous_dod" column,
        # the values are moved into that column, while the other one is cleaned,
        # as well as the "previous dod" values for timestamps after the first one
    prosumers_calc_df['previous_dod'] = prosumers_calc_df['storage_dod'].copy()
    prosumers_calc_df['storage_dod'] = np.nan
    prosumers_calc_df.loc[range(number_of_prosumers, number_of_timestamps),
        'previous_dod'] = np.nan

    # PLANTS
    # df creation, merging the plant_df with the the appendix_df because there
        # is no storage
    plant_calc_df = pd.merge(plant_df, appendix_df, how = 'outer')
    plant_len = len(plant_calc_df)

    # indexing the df
    # sort the values by timestamp and by member id to have them in order, then
        # reset the index values to allow iterations
```

```
    plant_calc_df.sort_values(['timestamp','member_id'], ascending=[True, True]
        ,inplace= True)
    plant_calc_df.reset_index(drop=True, inplace=True)

    # setting as zero the values of the battery for completeness
    plant_calc_df['energy_exchangeable_kWh'] = 0
    plant_calc_df['previous_dod'] = 0
    plant_calc_df['previous_energy_in_battery_t_kWh'] = 0
    plant_calc_df['energy_in_battery_t_kWh'] = 0
    plant_calc_df['previous_space_in_battery_t_kWh'] = 0
    plant_calc_df['max_energy_chargeable_t_kWh'] = 0
    plant_calc_df['max_energy_dischargeable_t_kWh'] = 0
    plant_calc_df['battery_balance_t_kWh'] = 0
    plant_calc_df['space_in_battery_t_kWh'] = 0

    # impose the storage energy equal to nan to avoid calculations and divisions
        # by zero during the iteration phase, it can be improved but this works
    consumers_calc_df['storage_energy_kWh'] = np.where
        (consumers_calc_df['storage_energy_kWh'] == 0, np.nan ,
        consumers_calc_df['storage_energy_kWh'])
    prosumers_calc_df['storage_energy_kWh'] = np.where
        (prosumers_calc_df['storage_energy_kWh'] == 0, np.nan ,
        prosumers_calc_df['storage_energy_kWh'])
    consumers_calc_df['battery_balance_t_kWh'] = np.where
        (consumers_calc_df['storage_energy_kWh'] == 0, 0, 0)
    prosumers_calc_df['battery_balance_t_kWh'] = np.where
        (prosumers_calc_df['storage_energy_kWh'] == 0, 0, 0)
    consumers_calc_df['max_energy_dischargeable_t_kWh'] = np.where
        (consumers_calc_df['storage_energy_kWh'] == 0, 0, 0)
    prosumers_calc_df['max_energy_dischargeable_t_kWh'] = np.where
        (prosumers_calc_df['storage_energy_kWh'] == 0, 0, 0)

    # BATTERIES
    # df creation, merging the storage_df with the the appendix_df because
        # there is production
    storages_calc_df = pd.DataFrame()
    storages_calc_df = pd.merge(rec_storage_df, appendix_df, how = 'outer')
    storages_calc_df['previous_dod'] = storages_calc_df['storage_dod'].copy()
    storages_calc_df['storage_dod'] = np.nan
    storages_calc_df['energy_exchangeable_kWh'] =
        storages_calc_df['storage_power_kW']*(settings['sampling_minutes']/60)

    # create a df to store the results for the rec and for the storages
    rec_calc_df = pd.DataFrame(columns=['timestamp','rec_raw_balance_t_kWh',
        'users_and_batteries_energy_surplus_t_kWh',
        'users_and_batteries_energy_needed_t_kWh', 'users_and_batteries_balances'])

    # # Dataframes temporanei
    cons_data_temp_df = pd.DataFrame()
    pros_data_temp_df = pd.DataFrame()
```

At this time, the management of the dataframe is completed and it is possible to proceed

with the iteration process, also explained in details by its Activity Diagram in Section 3.5.

A.4 Calculations and iteration

This final section aims to present the code in the way it is presented in Section 3.5, using the same four subdivision for a better clarity.

Part 1: Self-consumption calculation

```
#%% STARTING WITH THE ITERATION
    # create a variable that identifies the progressive number of the timestamp

    t_i = 0 # it starts from the first timestamp at 00:00
    for t_i in range(number_of_timestamps):
    # for t_i in range(96):

        first_row_cons = t_i * number_of_consumers
        last_row_cons = (t_i + 1) * number_of_consumers
        first_row_pros = t_i * number_of_prosumers
        last_row_pros = (t_i + 1) * number_of_prosumers
        first_row_plants = t_i * number_of_plants
        last_row_plants = (t_i + 1) * number_of_plants

        # the first calculation to do is to obtain the energy stored in the battery
            # in the previous timestamp, from which there will be the calculations
        if t_i == 0:
            consumers_calc_df.loc[range(first_row_cons, last_row_cons),
                'previous_energy_in_battery_t_kWh'] = consumers_calc_df.loc
                [range(first_row_cons,last_row_cons),'storage_energy_kWh']*
                consumers_calc_df.loc[range(first_row_cons,last_row_cons),
                'previous_dod']
            prosumers_calc_df.loc[range(first_row_pros, last_row_pros),
                'previous_energy_in_battery_t_kWh'] = prosumers_calc_df.loc
                [range(first_row_pros, last_row_pros),'storage_energy_kWh']*
                prosumers_calc_df.loc[range(first_row_pros, last_row_pros),
                'previous_dod']
            storages_calc_df.loc[t_i, 'previous_energy_in_battery_t_kWh'] =
                storages_calc_df.loc[t_i, 'storage_energy_kWh']*storages_calc_df.
                loc[t_i, 'previous_dod']

        # copying in the "energy_in_battery_t_kWh because it is needed for the
            #iterations and the calculations
        consumers_calc_df.loc[range(first_row_cons, last_row_cons),
            'energy_in_battery_t_kWh'] = consumers_calc_df.loc[range(first_row_cons,
            last_row_cons),'previous_energy_in_battery_t_kWh']
        prosumers_calc_df.loc[range(first_row_pros, last_row_pros),
            'energy_in_battery_t_kWh'] = prosumers_calc_df.loc[range
            (first_row_pros,last_row_pros),'previous_energy_in_battery_t_kWh']

```

```
        # calculate the space available in the battery
        consumers_calc_df.loc[range(first_row_cons, last_row_cons),
            'previous_space_in_battery_t_kWh'] = consumers_calc_df.loc
            [range(first_row_cons, last_row_cons),'storage_energy_kWh'] -
            consumers_calc_df.loc[range(first_row_cons, last_row_cons),
            'energy_in_battery_t_kWh']
        prosumers_calc_df.loc[range(first_row_pros, last_row_pros),
            'previous_space_in_battery_t_kWh'] = prosumers_calc_df.loc
            [range(first_row_pros, last_row_pros),'storage_energy_kWh']-
            prosumers_calc_df.loc[range(first_row_pros, last_row_pros),
            'energy_in_battery_t_kWh']

        # calculation of the energy chargeable comparing the exchangeable
            # energy with the space
        consumers_calc_df.loc[range(first_row_cons, last_row_cons),
            'max_energy_chargeable_t_kWh'] = np.where(consumers_calc_df.loc
            [range(first_row_cons, last_row_cons),'storage_power_kW'] == 0, 0,
            np.where(consumers_calc_df.loc[range(first_row_cons, last_row_cons),
            'previous_space_in_battery_t_kWh'] >= consumers_calc_df.loc
            [range(first_row_cons, last_row_cons), 'energy_exchangeable_kWh'],
            consumers_calc_df.loc[range(first_row_cons, last_row_cons),
            'energy_exchangeable_kWh'], consumers_calc_df.loc[range(first_row_cons,
            last_row_cons),'previous_space_in_battery_t_kWh']))
        prosumers_calc_df.loc[range(first_row_pros, last_row_pros),
            'max_energy_chargeable_t_kWh'] = np.where(prosumers_calc_df.loc
            [range(first_row_pros, last_row_pros),'storage_power_kW'] == 0, 0,
            np.where(prosumers_calc_df.loc[range(first_row_pros, last_row_pros),
            'previous_space_in_battery_t_kWh'] >= prosumers_calc_df.loc
            [range(first_row_pros, last_row_pros), 'energy_exchangeable_kWh'],
            prosumers_calc_df.loc[range(first_row_pros, last_row_pros),
            'energy_exchangeable_kWh'], prosumers_calc_df.loc[range(first_row_pros,
            last_row_pros),'previous_space_in_battery_t_kWh']))

        # calculation of the energy dischargeable comparing the exchangeable
            # energy with the energy in battery
        consumers_calc_df.loc[range(first_row_cons, last_row_cons),
            'max_energy_dischargeable_t_kWh'] = np.where(consumers_calc_df.loc
            [range(first_row_cons, last_row_cons), 'storage_power_kW'] == 0, 0,
            np.where(consumers_calc_df.loc[range(first_row_cons, last_row_cons),
            'energy_in_battery_t_kWh'] >= consumers_calc_df.loc
            [range(first_row_cons, last_row_cons), 'energy_exchangeable_kWh'],
            consumers_calc_df.loc[range(first_row_cons, last_row_cons),
            'energy_exchangeable_kWh'], consumers_calc_df.loc[range(first_row_cons,
            last_row_cons), 'energy_in_battery_t_kWh']))
        prosumers_calc_df.loc[range(first_row_pros, last_row_pros),
            'max_energy_dischargeable_t_kWh'] = np.where(prosumers_calc_df.loc
            [range(first_row_pros, last_row_pros), 'storage_power_kW'] == 0, 0,
            np.where(prosumers_calc_df.loc[range(first_row_pros, last_row_pros),
            'energy_in_battery_t_kWh'] >= prosumers_calc_df.loc[range
            (first_row_pros, last_row_pros), 'energy_exchangeable_kWh'],
            prosumers_calc_df.loc[range(first_row_pros, last_row_pros),
            'energy_exchangeable_kWh'], prosumers_calc_df.loc[range(first_row_pros,
```

```
            last_row_pros), 'energy_in_battery_t_kWh']))

        # now is used a flag to have a quicker parametrization, just with a
            #number from 1 to 4, of the case in which is the user in that timestamp
            # 1 raw_user_balance > 0 ; surplus can be all stored in the battery
            # 2 raw_user_balance > 0 ; surplus bigger than the space available
                # in the battery
            # 3 raw_user_balance < 0 ; deficit can be fulfilled by the battery
            # 4 raw_user_balance < 0 ; deficit must be fulfilled by external energy

        consumers_calc_df.loc[range(first_row_cons, last_row_cons), 'flag'] =
            np.where(consumers_calc_df.loc[range(first_row_cons, last_row_cons),
            'max_energy_dischargeable_t_kWh'] >= -(consumers_calc_df.loc
            [range(first_row_cons, last_row_cons), 'raw_user_balance_t_kWh']),3 ,4)
        prosumers_calc_df.loc[range(first_row_pros, last_row_pros), 'flag'] =
            np.where(((prosumers_calc_df.loc[range(first_row_pros, last_row_pros),
            'raw_user_balance_t_kWh'] >= 0) & (prosumers_calc_df.loc
            [range(first_row_pros, last_row_pros), 'max_energy_chargeable_t_kWh']
            >= prosumers_calc_df.loc[range(first_row_pros, last_row_pros),
            'raw_user_balance_t_kWh'])), 1, np.where(((prosumers_calc_df.loc
            [range(first_row_pros, last_row_pros), 'raw_user_balance_t_kWh'] >= 0)
            & (prosumers_calc_df.loc[range(first_row_pros, last_row_pros),
            'max_energy_chargeable_t_kWh'] < prosumers_calc_df.loc[range
            (first_row_pros, last_row_pros), 'raw_user_balance_t_kWh'])), 2,
            np.where(((prosumers_calc_df.loc[range(first_row_pros, last_row_pros),
            'raw_user_balance_t_kWh'] < 0) & (prosumers_calc_df.loc[range
            (first_row_pros, last_row_pros), 'max_energy_dischargeable_t_kWh'] >=
            -(prosumers_calc_df.loc[range(first_row_pros, last_row_pros),
            'raw_user_balance_t_kWh']))), 3, 4)))
        plant_calc_df.loc[range(first_row_plants, last_row_plants), 'flag'] =
            np.where(plant_calc_df.loc[range(first_row_plants,last_row_plants),
            'prod_kWh'] > 0, 1, 3)

        # calculate the new battery balance that depends on the flag
        consumers_calc_df.loc[range(first_row_cons, last_row_cons),
            'battery_balance_t_kWh'] = np.where(consumers_calc_df.loc
            [range(first_row_cons, last_row_cons), 'storage_power_kW'] == 0, 0,
            np.where((consumers_calc_df.loc[range(first_row_cons, last_row_cons),
            'flag'] == 3), consumers_calc_df.loc[range(first_row_cons,
            last_row_cons), 'raw_user_balance_t_kWh'], -consumers_calc_df.loc
            [range(first_row_cons, last_row_cons),
            'max_energy_dischargeable_t_kWh']))
        prosumers_calc_df.loc[range(first_row_pros, last_row_pros),
            'battery_balance_t_kWh'] = np.where(prosumers_calc_df.loc
            [range(first_row_pros, last_row_pros), 'storage_power_kW'] == 0, 0,
            np.where(prosumers_calc_df.loc[range(first_row_pros, last_row_pros),
            'flag'] == 1, prosumers_calc_df.loc[range(first_row_pros,
            last_row_pros), 'raw_user_balance_t_kWh'], np.where
            (prosumers_calc_df.loc[range(first_row_pros, last_row_pros), 'flag'] ==
            2, prosumers_calc_df.loc[range(first_row_pros, last_row_pros),
            'max_energy_chargeable_t_kWh'], np.where(prosumers_calc_df.loc
            [range(first_row_pros, last_row_pros), 'flag'] == 3,
```

```
            prosumers_calc_df.loc[range(first_row_pros, last_row_pros),
            'raw_user_balance_t_kWh'], - prosumers_calc_df.loc[range
            (first_row_pros, last_row_pros), 'max_energy_dischargeable_t_kWh'])))

        # calculate the balance of the user and the battery together
        consumers_calc_df.loc[range(first_row_cons, last_row_cons),
            'user_and_battery_balance_t_kWh'] = np.where((consumers_calc_df.loc
            [range(first_row_cons, last_row_cons), 'flag'] == 3), 0,
            consumers_calc_df.loc[range(first_row_cons, last_row_cons),
            'max_energy_dischargeable_t_kWh'] + consumers_calc_df.loc
            [range(first_row_cons, last_row_cons), 'raw_user_balance_t_kWh'])
        prosumers_calc_df.loc[range(first_row_pros, last_row_pros),
            'user_and_battery_balance_t_kWh'] = np.where((prosumers_calc_df.loc
            [range(first_row_pros, last_row_pros), 'flag'] == 1), 0,
            np.where((prosumers_calc_df.loc[range(first_row_pros, last_row_pros),
            'flag'] == 2), prosumers_calc_df.loc[range(first_row_pros,
            last_row_pros), 'raw_user_balance_t_kWh'] - prosumers_calc_df.loc
            [range(first_row_pros, last_row_pros), 'max_energy_chargeable_t_kWh'],
            np.where((prosumers_calc_df.loc[range(first_row_pros, last_row_pros),
            'flag'] == 3), 0, prosumers_calc_df.loc[range(first_row_pros,
            last_row_pros), 'max_energy_dischargeable_t_kWh'] +
            prosumers_calc_df.loc[range(first_row_pros, last_row_pros),
            'raw_user_balance_t_kWh'])))
        plant_calc_df.loc[range(first_row_plants, last_row_plants),
            'user_and_battery_balance_t_kWh'] = plant_calc_df.loc
            [range(first_row_plants, last_row_plants), 'raw_user_balance_t_kWh']

        # calculate the space left in the battery after the user's use
        consumers_calc_df.loc[range(first_row_cons, last_row_cons),
            'space_in_battery_t_kWh'] = np.where(consumers_calc_df.loc
            [range(first_row_cons, last_row_cons), 'storage_energy_kWh'] == 0, 0,
            consumers_calc_df.loc[range(first_row_cons, last_row_cons),
            'previous_space_in_battery_t_kWh'] - consumers_calc_df.loc
            [range(first_row_cons, last_row_cons), 'battery_balance_t_kWh'])
        prosumers_calc_df.loc[range(first_row_pros, last_row_pros),
            'space_in_battery_t_kWh'] = np.where(prosumers_calc_df.loc
            [range(first_row_pros, last_row_pros), 'storage_energy_kWh'] == 0, 0,
            prosumers_calc_df.loc[range(first_row_pros, last_row_pros),
            'previous_space_in_battery_t_kWh'] - prosumers_calc_df.loc
            [range(first_row_pros, last_row_pros), 'battery_balance_t_kWh'])

        # final recalculation of the energy left in the battery
        consumers_calc_df.loc[range(first_row_cons, last_row_cons),
            'energy_in_battery_t_kWh'] = np.where(consumers_calc_df.loc
            [range(first_row_cons, last_row_cons), 'storage_energy_kWh'] == 0, 0,
            consumers_calc_df.loc[range(first_row_cons, last_row_cons),
            'previous_energy_in_battery_t_kWh'] + consumers_calc_df.loc
            [range(first_row_cons, last_row_cons), 'battery_balance_t_kWh'])
        prosumers_calc_df.loc[range(first_row_pros, last_row_pros),
            'energy_in_battery_t_kWh'] = np.where(prosumers_calc_df.loc
            [range(first_row_pros, last_row_pros), 'storage_energy_kWh'] == 0, 0,
            prosumers_calc_df.loc[range(first_row_pros, last_row_pros),
```

```
            'previous_energy_in_battery_t_kWh'] + prosumers_calc_df.loc
            [range(first_row_pros, last_row_pros), 'battery_balance_t_kWh'])
```

Part 2: Introduction of the community's battery

```
# update the rec dataframe, these will be partial before the battery's use
        # take the timestamp from one of the other df
        rec_calc_df.loc[t_i, 'timestamp'] = consumers_calc_df.loc[first_row_cons,
            'timestamp']
        # calculate the overall rec raw balance summing the raw user balance
            # timestamp by timestamp for the three df
        rec_calc_df.loc[t_i, 'rec_raw_balance_t_kWh'] = consumers_calc_df.loc
            [range(first_row_cons, last_row_cons), 'raw_user_balance_t_kWh'].sum()
            + prosumers_calc_df.loc[range(first_row_pros, last_row_pros),
            'raw_user_balance_t_kWh'].sum() + plant_calc_df.loc[range
            (first_row_plants, last_row_plants), 'raw_user_balance_t_kWh'].sum()
        # calculate the energy surplus by summing the positive values for
            # the user_and_battery_balance
        rec_calc_df.loc[t_i, 'users_and_batteries_energy_surplus_t_kWh'] =
            (np.where(consumers_calc_df.loc[range(first_row_cons, last_row_cons),
            'user_and_battery_balance_t_kWh'] >= 0, consumers_calc_df.loc
            [range(first_row_cons,last_row_cons),'user_and_battery_balance_t_kWh']
            ,0)).sum() + (np.where(prosumers_calc_df.loc[range(first_row_pros,
            last_row_pros), 'user_and_battery_balance_t_kWh'] >= 0,
            prosumers_calc_df.loc[range(first_row_pros, last_row_pros),
            'user_and_battery_balance_t_kWh'],0)).sum() + (np.where
            (plant_calc_df.loc[range(first_row_plants, last_row_plants),
            'user_and_battery_balance_t_kWh'] >= 0, plant_calc_df.loc
            [range(first_row_plants, last_row_plants),
            'user_and_battery_balance_t_kWh'],0)).sum()
        rec_calc_df.loc[t_i, 'users_and_batteries_energy_surplus_t_kWh'] =
            np.where(rec_calc_df.loc[t_i,
            'users_and_batteries_energy_surplus_t_kWh'] > 0, rec_calc_df.loc[t_i,
            'users_and_batteries_energy_surplus_t_kWh'], np.nan)
        # calculate the energy surplus by summing the negative values for
            # the user_and_battery_balance
        rec_calc_df.loc[t_i, 'users_and_batteries_energy_needed_t_kWh'] =
            (np.where(consumers_calc_df.loc[range(first_row_cons, last_row_cons),
            'user_and_battery_balance_t_kWh'] < 0, consumers_calc_df.loc
            [range(first_row_cons, last_row_cons),
            'user_and_battery_balance_t_kWh'],0)).sum() + (np.where
            (prosumers_calc_df.loc[range(first_row_pros, last_row_pros),
            'user_and_battery_balance_t_kWh'] < 0, prosumers_calc_df.loc
            [range(first_row_pros, last_row_pros),
            'user_and_battery_balance_t_kWh'],0)).sum() + (np.where
            (plant_calc_df.loc[range(first_row_plants, last_row_plants),
            'user_and_battery_balance_t_kWh'] < 0, plant_calc_df.
            loc[range(first_row_plants, last_row_plants),
            'user_and_battery_balance_t_kWh'],0)).sum()
        rec_calc_df.loc[t_i, 'users_and_batteries_energy_needed_t_kWh'] =
```

```
            np.where(rec_calc_df.loc[t_i, 'users_and_batteries_energy_needed_t_kWh']
            < 0, rec_calc_df.loc[t_i, 'users_and_batteries_energy_needed_t_kWh'],
            np.nan)
        # calculate the energy balance by summing all the user_and_battery_balance
        rec_calc_df.loc[t_i, 'users_and_batteries_balances'] =
            consumers_calc_df.loc[range(first_row_cons, last_row_cons),
            'user_and_battery_balance_t_kWh'].sum() +
            prosumers_calc_df.loc[range(first_row_pros, last_row_pros),
            'user_and_battery_balance_t_kWh'].sum() +
            plant_calc_df.loc[range(first_row_plants, last_row_plants),
            'user_and_battery_balance_t_kWh'].sum()
        # calculate the shared_energy in the rec that is the minimum between
            # the deficit and the surplus (only if the surplus > 0)
        rec_calc_df.loc[t_i, 'total_shared_energy_user2user'] =
            np.where(rec_calc_df.loc[t_i,
            'users_and_batteries_energy_surplus_t_kWh'] == 0, 0,
            np.where(rec_calc_df.loc[t_i,
            'users_and_batteries_energy_surplus_t_kWh'] > -rec_calc_df.loc[t_i,
            'users_and_batteries_energy_needed_t_kWh'], - rec_calc_df.loc[t_i,
            'users_and_batteries_energy_needed_t_kWh'], rec_calc_df.loc[t_i,
            'users_and_batteries_energy_surplus_t_kWh']))

        # update the batteries df with the timestamp number and the energy
            # available in the battery
        storages_calc_df.loc[t_i, 'timestamp'] =
            consumers_calc_df.loc[first_row_cons, 'timestamp']
        storages_calc_df.loc[t_i, 'energy_in_battery_t_kWh'] =
            storages_calc_df.loc[t_i, 'previous_energy_in_battery_t_kWh']
        storages_calc_df.loc[t_i, 'previous_space_in_battery_t_kWh'] =
            storages_calc_df.loc[t_i, 'storage_energy_kWh'] -
            storages_calc_df.loc[t_i, 'previous_energy_in_battery_t_kWh']

        # calculate the max chargeable and dischargeable energy for the rec battery
        storages_calc_df.loc[t_i, 'max_energy_dischargeable_t_kWh'] =
            min(storages_calc_df.loc[t_i, 'energy_exchangeable_kWh'],
            storages_calc_df.loc[t_i, 'previous_energy_in_battery_t_kWh'])
        storages_calc_df.loc[t_i, 'max_energy_chargeable_t_kWh'] =
            np.where(np.isnan(rec_calc_df.loc[t_i,
            'users_and_batteries_energy_surplus_t_kWh']),
            0, ( np.where( storages_calc_df.loc[t_i,
            'previous_space_in_battery_t_kWh'] >= storages_calc_df.loc[t_i,
            'energy_exchangeable_kWh'],storages_calc_df.loc
            [t_i, 'energy_exchangeable_kWh'] , storages_calc_df
            .loc[t_i, 'previous_space_in_battery_t_kWh'] ) ) )

        # create a flag to see if there is surplus or need:
            # 5 = u&b > 0, surplus < chargeable -> all the surplus goes
                # into the battery
            # 6 = u&b > 0, surplus > chargeable -> part goes in the battery,
                # part goes into the grid
            # 7 = u&b < 0, -deficit < dischargeable -> all the deficit
                # is compensated by battery
```

```
            # 8 = u&b < 0, -deficit > dischargeable -> part is compensated by
                # battery, part from the grid

        # flag calculation
        storages_calc_df.loc[t_i, 'flag'] = np.where(((rec_calc_df.loc[t_i,
            'users_and_batteries_balances'] >= 0) & (rec_calc_df.loc[t_i,
            'users_and_batteries_balances'] <= storages_calc_df.loc[t_i,
            'max_energy_chargeable_t_kWh'])), 5 , np.where( ((rec_calc_df.loc[t_i,
            'users_and_batteries_balances'] >= 0) & (rec_calc_df.loc[t_i,
            'users_and_batteries_balances'] > storages_calc_df.loc[t_i,
            'max_energy_chargeable_t_kWh'])), 6 , np.where(((rec_calc_df.loc[t_i,
            'users_and_batteries_balances'] < 0) & ( -rec_calc_df.loc[t_i,
            'users_and_batteries_balances'] < storages_calc_df.loc[t_i,
            'max_energy_dischargeable_t_kWh'] )), 7 , 8 )))

        # update the users & batteries balance to the energy from the battery
            # if present
        storages_calc_df.loc[t_i, 'battery_balance_t_kWh'] = np.where(
            storages_calc_df.loc[t_i, 'flag'] == 5 , rec_calc_df.loc[t_i,
            'users_and_batteries_balances'] , np.where( storages_calc_df.loc[t_i,
            'flag'] == 6, (storages_calc_df.loc[t_i,'max_energy_chargeable_t_kWh']),
            np.where( storages_calc_df.loc[t_i, 'flag'] == 7 , rec_calc_df.loc[t_i,
            'users_and_batteries_balances'] , ( - storages_calc_df.loc[t_i,
            'max_energy_dischargeable_t_kWh'] ) ) ) )

        # now in the rec df it can be seen how much energy it is rebalanced
            # by the batteries
        rec_calc_df.loc[t_i, 'total_grid_exchange'] = rec_calc_df.loc[t_i,
            'users_and_batteries_balances'] - storages_calc_df.loc[t_i,
            'battery_balance_t_kWh']
        rec_calc_df.loc[t_i, 'storage_balance'] = storages_calc_df.loc[t_i,
            'battery_balance_t_kWh']


        # that value is also in the storage df, and it is used to calculate
            # the discharged energy
        # storages_calc_df.loc[t_i, 'battery_balance_t_kWh'] = rec_calc_df.loc
            [t_i, 'rec_battery_balance_t_kWh']
        storages_calc_df.loc[t_i, 'user_and_battery_balance_t_kWh'] =
            storages_calc_df.loc[t_i, 'battery_balance_t_kWh']
        storages_calc_df.loc[t_i, 'energy_in_battery_t_kWh'] =
            storages_calc_df.loc[t_i, 'energy_in_battery_t_kWh'] +
            storages_calc_df.loc[t_i, 'battery_balance_t_kWh']
        storages_calc_df.loc[t_i, 'space_in_battery_t_kWh'] =
            storages_calc_df.loc[t_i, 'storage_energy_kWh'] -
            storages_calc_df.loc[t_i, 'energy_in_battery_t_kWh']

        # calculate the dod paying attention to the division by zero that
            # happens when the capacity is 0
        consumers_calc_df.loc[range(first_row_cons,last_row_cons),'storage_dod'] =
            np.divide(consumers_calc_df.loc[range(first_row_cons, last_row_cons),
            'energy_in_battery_t_kWh'], consumers_calc_df.loc[range(first_row_cons,
```

```
            last_row_cons), 'storage_energy_kWh'])
        prosumers_calc_df.loc[range(first_row_pros,last_row_pros),'storage_dod']=
            np.divide(prosumers_calc_df.loc[range(first_row_pros, last_row_pros),
            'energy_in_battery_t_kWh'], prosumers_calc_df.loc[range(first_row_pros,
            last_row_pros), 'storage_energy_kWh'])
        storages_calc_df.loc[t_i, 'storage_dod'] = np.divide
            (storages_calc_df.loc[t_i, 'energy_in_battery_t_kWh'],
            storages_calc_df.loc[t_i, 'storage_energy_kWh'])

```

Part 3: Sharing and redistributing

```
    # it is needed to check the rec_battery storage status in the rec db,
        # then update the energy shared to the storage
        rec_calc_df.loc[t_i, 'storage_dod'] = storages_calc_df.loc[t_i,
            'storage_dod'] # just for check
        rec_calc_df.loc[t_i, 'total_shared_energy_user2storage'] = np.where(
            rec_calc_df.loc[t_i, 'storage_balance'] > 0, rec_calc_df.loc[t_i,
            'storage_balance'] , 0 )
        rec_calc_df.loc[t_i, 'total_shared_energy'] = rec_calc_df.loc[t_i,
            'total_shared_energy_user2storage'] + rec_calc_df.loc[t_i,
            'total_shared_energy_user2user']


        # REDISTRIBUTE ENERGY IN THE EMPTY SPACES IN BATTERIES
        # redistribution will sure work here, but it might cause unnecessary
            recalculation and it in that case might need optimization
        # calculate the maximum amount of energy chargeable in a timestamp
        consumers_calc_df.loc[range(first_row_cons, last_row_cons),
            'potential_redistribution_t_kWh'] = consumers_calc_df.loc
            [range(first_row_cons, last_row_cons), 'max_energy_chargeable_t_kWh'] -
            consumers_calc_df.loc[range(first_row_cons, last_row_cons),
            'battery_balance_t_kWh']
        prosumers_calc_df.loc[range(first_row_pros, last_row_pros),
            'potential_redistribution_t_kWh'] = prosumers_calc_df.loc
            [range(first_row_pros, last_row_pros), 'max_energy_chargeable_t_kWh'] -
            prosumers_calc_df.loc[range(first_row_pros, last_row_pros),
            'battery_balance_t_kWh']
        # then, the sum is inserted in a column in the rec df
        rec_calc_df.loc[t_i, 'max_redistribution'] =  consumers_calc_df.loc
        [range(first_row_cons, last_row_cons), 'potential_redistribution_t_kWh']
        .sum() + prosumers_calc_df.loc[range(first_row_pros, last_row_pros),
        'potential_redistribution_t_kWh'].sum()
        # then, calculate the actual redistributed energy, if the energy to
            # grid > max redistribution -> redistributed = max, energy to grid =
            # energy to grid - max. Vice versa, all is redistributed and 0
            #to the grid
        rec_calc_df.loc[t_i, 'redistributed_energy'] = np.where(rec_calc_df.loc
            [t_i, 'total_grid_exchange'] > 0  , np.where(rec_calc_df.loc[t_i,
            'total_grid_exchange'] < rec_calc_df.loc[t_i, 'max_redistribution'],
```

```
            rec_calc_df.loc[t_i, 'total_grid_exchange'], rec_calc_df.loc[t_i,
            'max_redistribution']), 0)
        rec_calc_df.loc[t_i, 'final_grid_exchange'] = np.where(rec_calc_df.loc
            [t_i, 'total_grid_exchange'] > 0 , np.where(rec_calc_df.loc[t_i,
            'total_grid_exchange'] < rec_calc_df.loc[t_i, 'max_redistribution'], 0,
            rec_calc_df.loc[t_i, 'total_grid_exchange'] - rec_calc_df.loc[t_i,
            'max_redistribution']), rec_calc_df.loc[t_i, 'total_grid_exchange'])
        # calculate how much free space each user has proportionally with the
            total potential free space
        consumers_calc_df.loc[range(first_row_cons, last_row_cons),
            'fraction_of_free_space'] = consumers_calc_df.loc[range(first_row_cons,
            last_row_cons), 'potential_redistribution_t_kWh'] / rec_calc_df.loc
            [t_i, 'max_redistribution']
        prosumers_calc_df.loc[range(first_row_pros, last_row_pros),
            'fraction_of_free_space'] = prosumers_calc_df.loc[range(first_row_pros,
            last_row_pros), 'potential_redistribution_t_kWh'] / rec_calc_df.loc
            [t_i, 'max_redistribution']
        # now can be seen how much energy is actually redistributed
        consumers_calc_df.loc[range(first_row_cons, last_row_cons),
            'redistributed_energy_t_kWh'] = consumers_calc_df.loc
            [range(first_row_cons, last_row_cons), 'fraction_of_free_space'] *
            rec_calc_df.loc[t_i, 'redistributed_energy']
        prosumers_calc_df.loc[range(first_row_pros, last_row_pros),
            'redistributed_energy_t_kWh'] = prosumers_calc_df.loc[range
            (first_row_pros, last_row_pros), 'fraction_of_free_space'] *
            rec_calc_df.loc[t_i, 'redistributed_energy']
```

Part 4: individual contribution

```
    # all the previous values must be updated to calculate the final values
        # of every parameter
        consumers_calc_df.loc[range(first_row_cons, last_row_cons),
            'final_battery_balance_t_kWh'] = consumers_calc_df.loc
            [range(first_row_cons, last_row_cons), 'battery_balance_t_kWh'] +
            consumers_calc_df.loc[range(first_row_cons, last_row_cons),
            'redistributed_energy_t_kWh']
        prosumers_calc_df.loc[range(first_row_pros, last_row_pros),
            'final_battery_balance_t_kWh'] = prosumers_calc_df.loc
            [range(first_row_pros, last_row_pros), 'battery_balance_t_kWh'] +
            prosumers_calc_df.loc[range(first_row_pros, last_row_pros),
            'redistributed_energy_t_kWh']
        consumers_calc_df.loc[range(first_row_cons, last_row_cons),
            'final_user_and_battery_balance_t_kWh'] = consumers_calc_df.loc
            [range(first_row_cons,last_row_cons),'user_and_battery_balance_t_kWh']+
            consumers_calc_df.loc[range(first_row_cons, last_row_cons),
            'redistributed_energy_t_kWh']
        prosumers_calc_df.loc[range(first_row_pros, last_row_pros),
            'final_user_and_battery_balance_t_kWh'] = prosumers_calc_df.loc
            [range(first_row_pros,last_row_pros),'user_and_battery_balance_t_kWh']+
            prosumers_calc_df.loc[range(first_row_pros, last_row_pros),
```

```
            'redistributed_energy_t_kWh']
        consumers_calc_df.loc[range(first_row_cons, last_row_cons),
            'final_energy_in_battery_t_kWh'] = consumers_calc_df.loc
            [range(first_row_cons, last_row_cons),
            'previous_energy_in_battery_t_kWh'] +
            consumers_calc_df.loc[range(first_row_cons, last_row_cons),
            'final_battery_balance_t_kWh']
        prosumers_calc_df.loc[range(first_row_pros, last_row_pros),
            'final_energy_in_battery_t_kWh'] = prosumers_calc_df.loc
            [range(first_row_pros, last_row_pros),
            'previous_energy_in_battery_t_kWh'] +
            prosumers_calc_df.loc[range(first_row_pros, last_row_pros),
            'final_battery_balance_t_kWh']
        consumers_calc_df.loc[range(first_row_cons, last_row_cons),
            'final_storage_dod'] = np.divide(consumers_calc_df.loc
            [range(first_row_cons,last_row_cons),'final_energy_in_battery_t_kWh'],
            consumers_calc_df.loc[range(first_row_cons, last_row_cons),
            'storage_energy_kWh'])
        prosumers_calc_df.loc[range(first_row_pros, last_row_pros),
            'final_storage_dod'] = np.divide(prosumers_calc_df.loc
            [range(first_row_pros,last_row_pros),'final_energy_in_battery_t_kWh'],
            prosumers_calc_df.loc[range(first_row_pros, last_row_pros),
            'storage_energy_kWh'])
        consumers_calc_df.loc[range(first_row_cons, last_row_cons),
            'final_space_in_battery_t_kWh'] = consumers_calc_df.loc
            [range(first_row_cons, last_row_cons), 'storage_energy_kWh'] -
            consumers_calc_df.loc[range(first_row_cons, last_row_cons),
            'final_energy_in_battery_t_kWh']
        prosumers_calc_df.loc[range(first_row_pros, last_row_pros),
        'final_space_in_battery_t_kWh'] = prosumers_calc_df.loc
        [range(first_row_pros, last_row_pros), 'storage_energy_kWh'] -
        prosumers_calc_df.loc[range(first_row_pros, last_row_pros),
        'final_energy_in_battery_t_kWh']

        # update also the amount of shared out and redistributed energy
        rec_calc_df.loc[t_i, 'total_shared_out_and_redistributed_energy'] =
            rec_calc_df.loc[t_i, 'redistributed_energy'] + rec_calc_df.loc
            [t_i, 'total_shared_energy']


        # after the sharing calculation, it can be calculated the fraction of
            # the shared energy, to see each user's contribution
        # firstly it starts by calculating the \% of shared and the \% of sold
            # energy to the grid on the rec perspective. Then, that \% is applied
            # to all the individual surplus to see how much each user contributed
            # to the shared and sold energy.
        rec_calc_df.loc[t_i, 'fraction_of_energy_to_grid_%'] =
            np.where(rec_calc_df.loc[t_i, 'final_grid_exchange'] > 0 ,
            rec_calc_df.loc[t_i, 'final_grid_exchange'] / rec_calc_df.loc
            [t_i, 'users_and_batteries_energy_surplus_t_kWh'] , 0) #old
        rec_calc_df.loc[t_i, 'fraction_of_shared_out_energy_%'] =
            rec_calc_df.loc[t_i,'total_shared_out_and_redistributed_energy']/
```

```
            rec_calc_df.loc[t_i, 'users_and_batteries_energy_surplus_t_kWh']
        rec_calc_df.loc[t_i, 'fraction_of_shared_out_energy_%'] = np.where
            (np.isnan(rec_calc_df.loc[t_i, 'fraction_of_shared_out_energy_%']) ,
            0, rec_calc_df.loc[t_i, 'fraction_of_shared_out_energy_%'])

        # the same with the energy acquired, there is no shared in energy when
            # the storage gives energy because it is already counted. it can
            # be evaluated by using the difference, but has not incentives
        rec_calc_df.loc[t_i, 'fraction_of_energy_from_grid_%'] = np.where(
            rec_calc_df.loc[t_i, 'final_grid_exchange'] < 0 , rec_calc_df.loc
            [t_i, 'final_grid_exchange'] / rec_calc_df.loc[t_i,
            'users_and_batteries_energy_needed_t_kWh'] , 0)
        rec_calc_df.loc[t_i, 'fraction_of_shared_in_energy_%'] = 1 -
            rec_calc_df.loc[t_i, 'fraction_of_energy_from_grid_%']

```

Part 5: iteration

```
        # firstly the energy shared out to other members
        consumers_calc_df.loc[range(first_row_cons, last_row_cons),
            'energy_shared_out_t_kWh'] = np.where( consumers_calc_df.loc
            [range(first_row_cons, last_row_cons),'user_and_battery_balance_t_kWh']
            > 0 , (consumers_calc_df.loc[range(first_row_cons, last_row_cons),
            'user_and_battery_balance_t_kWh'] * rec_calc_df.loc[t_i,
            'fraction_of_shared_out_energy_%']) , 0)
        prosumers_calc_df.loc[range(first_row_pros, last_row_pros),
            'energy_shared_out_t_kWh'] = np.where( prosumers_calc_df.loc
            [range(first_row_pros, last_row_pros),'user_and_battery_balance_t_kWh']
            > 0 , (prosumers_calc_df.loc[range(first_row_pros, last_row_pros),
            'user_and_battery_balance_t_kWh'] * rec_calc_df.loc[t_i,
            'fraction_of_shared_out_energy_%']) , 0)
        plant_calc_df.loc[range(first_row_plants, last_row_plants),
            'energy_shared_out_t_kWh'] = np.where( plant_calc_df.loc
            [range(first_row_plants, last_row_plants),
            'user_and_battery_balance_t_kWh'] > 0 ,
            (plant_calc_df.loc[range(first_row_plants, last_row_plants),
            'user_and_battery_balance_t_kWh'] * rec_calc_df.loc[t_i,
            'fraction_of_shared_out_energy_%']) , 0)
        # then, the energy sold to the grid
        consumers_calc_df.loc[range(first_row_cons, last_row_cons),
            'energy_sold_to_grid_t_kWh'] = np.where( consumers_calc_df.loc
            [range(first_row_cons, last_row_cons),'user_and_battery_balance_t_kWh']
            > 0 , (consumers_calc_df.loc[range(first_row_cons, last_row_cons),
            'user_and_battery_balance_t_kWh'] * rec_calc_df.loc[t_i,
            'fraction_of_energy_to_grid_%']) , 0)
        prosumers_calc_df.loc[range(first_row_pros, last_row_pros),
            'energy_sold_to_grid_t_kWh'] = np.where( prosumers_calc_df.loc
            [range(first_row_pros, last_row_pros),'user_and_battery_balance_t_kWh']
            >0,(prosumers_calc_df.loc[range(first_row_pros, last_row_pros),
            'user_and_battery_balance_t_kWh'] * rec_calc_df.loc[t_i,
```

```
            'fraction_of_energy_to_grid_%']) , 0)
        plant_calc_df.loc[range(first_row_plants, last_row_plants),
            'energy_sold_to_grid_t_kWh'] = np.where(plant_calc_df.loc
            [range(first_row_plants, last_row_plants),
            'user_and_battery_balance_t_kWh'] > 0 ,
            (plant_calc_df.loc[range(first_row_plants, last_row_plants),
            'user_and_battery_balance_t_kWh'] * rec_calc_df.loc[t_i,
            'fraction_of_energy_to_grid_%']) , 0)
        # next up the energy shared in from other members, for statistical purposes
        consumers_calc_df.loc[range(first_row_cons, last_row_cons),
            'energy_shared_in_t_kWh'] = np.where(consumers_calc_df.loc
            [range(first_row_cons, last_row_cons),'user_and_battery_balance_t_kWh']
            < 0 , (consumers_calc_df.loc[range(first_row_cons, last_row_cons),
            'user_and_battery_balance_t_kWh'] * rec_calc_df.loc[t_i,
            'fraction_of_shared_in_energy_%']) , 0)
        prosumers_calc_df.loc[range(first_row_pros, last_row_pros),
            'energy_shared_in_t_kWh'] = np.where(prosumers_calc_df.loc
            [range(first_row_pros, last_row_pros),'user_and_battery_balance_t_kWh']
            < 0, (prosumers_calc_df.loc[range(first_row_pros, last_row_pros),
            'user_and_battery_balance_t_kWh'] * rec_calc_df.loc[t_i,
            'fraction_of_shared_in_energy_%']) , 0)
        plant_calc_df.loc[range(first_row_plants, last_row_plants),
            'energy_shared_in_t_kWh'] = np.where( plant_calc_df.loc
            [range(first_row_plants, last_row_plants),
            'user_and_battery_balance_t_kWh'] < 0 ,
            (plant_calc_df.loc[range(first_row_plants, last_row_plants),
            'user_and_battery_balance_t_kWh'] * rec_calc_df.loc[t_i,
            'fraction_of_shared_in_energy_%']) , 0)
        # finally, the energy bought from the grid
        consumers_calc_df.loc[range(first_row_cons, last_row_cons),
            'energy_bought_from_grid_t_kWh'] = np.where( consumers_calc_df.loc
            [range(first_row_cons, last_row_cons),'user_and_battery_balance_t_kWh']
            < 0 , (consumers_calc_df.loc[range(first_row_cons, last_row_cons),
            'user_and_battery_balance_t_kWh'] * rec_calc_df.loc[t_i,
            'fraction_of_energy_from_grid_%']) , 0)
        prosumers_calc_df.loc[range(first_row_pros, last_row_pros),
            'energy_bought_from_grid_t_kWh'] = np.where(
            prosumers_calc_df.loc[range(first_row_pros, last_row_pros),
            'user_and_battery_balance_t_kWh'] < 0 ,
            (prosumers_calc_df.loc[range(first_row_pros, last_row_pros),
            'user_and_battery_balance_t_kWh'] * rec_calc_df.loc[t_i,
            'fraction_of_energy_from_grid_%']) , 0)
        plant_calc_df.loc[range(first_row_plants, last_row_plants),
            'energy_bought_from_grid_t_kWh'] = np.where(
            plant_calc_df.loc[range(first_row_plants, last_row_plants),
            'user_and_battery_balance_t_kWh'] < 0 ,
            (plant_calc_df.loc[range(first_row_plants, last_row_plants),
            'user_and_battery_balance_t_kWh'] * rec_calc_df.loc[t_i,
            'fraction_of_energy_from_grid_%']) , 0)

        # the upgrades must be done only if t_i < 35040
        if t_i < (number_of_timestamps - 1):
```

```
            # FINAL: Update the latest values as well as the iterations
            # update them into the next timestamp previous_energy_in_battery_t_kWh
                # for consumers and prosumers
            consumers_calc_df.loc[range(first_row_cons + number_of_consumers,
                last_row_cons + number_of_consumers),
                'previous_energy_in_battery_t_kWh'] =
                list(consumers_calc_df.loc[range(first_row_cons, last_row_cons),
                'final_energy_in_battery_t_kWh'].values)
            prosumers_calc_df.loc[range(first_row_pros + number_of_prosumers,
                last_row_pros + number_of_prosumers),
                'previous_energy_in_battery_t_kWh'] =
                list(prosumers_calc_df.loc[range(first_row_pros, last_row_pros),
                'final_energy_in_battery_t_kWh'].values)

            # update also the dod values
            prosumers_calc_df.loc[range(first_row_pros + number_of_prosumers,
                last_row_pros + number_of_prosumers), 'previous_dod'] =
                list(prosumers_calc_df.loc[range(first_row_pros, last_row_pros),
                'final_storage_dod'].values)
            consumers_calc_df.loc[range(first_row_cons + number_of_consumers,
                last_row_cons + number_of_consumers), 'previous_dod'] =
                list(consumers_calc_df.loc[range(first_row_cons, last_row_cons),
                'final_storage_dod'].values)

            #update for the storages
            storages_calc_df.loc[t_i + number_of_public_storages,
                'storage_power_kW'] = (storages_calc_df.loc[t_i,
                'storage_power_kW']) # storage power doesn't change
            storages_calc_df.loc[t_i + number_of_public_storages, 'member_id'] =
                (storages_calc_df.loc[t_i, 'member_id']) #member_id doesn't change
            storages_calc_df.loc[t_i + number_of_public_storages,
                'storage_energy_kWh'] = (storages_calc_df.loc[t_i,
                'storage_energy_kWh']) # capacity doesn't change
            storages_calc_df.loc[t_i + number_of_public_storages,
                'energy_exchangeable_kWh'] = (storages_calc_df.loc[t_i,
                'energy_exchangeable_kWh']) # exchangeable doesn't change
            storages_calc_df.loc[t_i + number_of_public_storages,
                'previous_energy_in_battery_t_kWh'] = storages_calc_df.loc[t_i,
                'energy_in_battery_t_kWh']
            storages_calc_df.loc[t_i + number_of_public_storages, 'previous_dod'] =
                (storages_calc_df.loc[t_i, 'storage_dod'])


        # feedback to the console
        print('\ntimestamp number ' + str(t_i) + ' completed\n')
        print('calculating day number: ' + str((t_i//96)+1) + '\n')



    # save data as csv file
    consumers_calc_df.to_csv(consumers_data_csv, sep=';')
    prosumers_calc_df.to_csv(prosumers_data_csv, sep=';')
```

```
137     plant_calc_df.to_csv(plants_data_csv, sep=';')
138     storages_calc_df.to_csv(storage_data_csv, sep=';')
139     rec_calc_df.to_csv(rec_data_csv, sep=';')
```

The main code is concluded. All the detailed explanation about each part is contained in the Section 3.5.

Bibliography

[1] United Nations. What are the sustainable development goals?, 2015. (Cited at pages 1 e 5)

[2] Eurostat. Electricity and heat statistics, 2022. (Cited at page 4)

[3] Clean Energy for EU Islands, Energy Center. Agenda per la transizione energetica - isola di pantelleria, 2020. (Cited at pages 4 e 8)

[4] B.K. Sovacool M. Laldjebaev. *Energy security, poverty, and sovereignty: complex interlinkages and compelling implications*. 2015. (Cited at page 5)

[5] United Nations. Accelerating sdg 7 achievement, 2018. (Cited at page 5)

[6] VaasaETT. Electricity prices monthly update, 2022. (Cited at page 6)

[7] ARERA. Andamento del prezzo dell'energia elettrica per il consumatore domestico tipo in maggior tutela, 2022. (Cited at page 5)

[8] Statista. Emissions in the eu - statistics facts, 2022. (Cited at page 6)

[9] European Environmental Agency. Energy and climate change, 2021. (Cited at page 6)

[10] European Commission. Clean energy for all europeans package, 2016. (Cited at page 6)

[11] European Union. A european green deal, striving to be the first climate-neutral continent, 2019. (Cited at page 7)

[12] Camera dei Deputati. Dal protocollo di kyoto all'accordo di parigi: gli impegni per il 2020 e il 2030, 2022. (Cited at page 7)

[13] Agenzia delle Entrate. Superbonus 110, 2020. (Cited at page 7)

[14] Presidenza del Consiglio dei Ministri. Idrogeno, 2022. (Cited at page 8)

[15] Presidenza del Consiglio dei Ministri. Transizione 4.0, 2022. (Cited at page 8)

[16] Presidenza del Consiglio dei Ministri. Promozione rinnovabili per le comunità energetiche e l'auto-consumo, 2022. (Cited at pages 8 e 11)

[17] European Parliament. Directive (eu) 2018/2001, 2015. (Cited at page 9)

[18] Camera dei Deputati. I principali contenuti della direttiva red ii, 2022. (Cited at page 10)

[19] M.A. Egido-Aguilera A. A. Eras-Almeida. Hybrid renewable mini-grids on non-interconnected small islands: Review of case studies, 2019. (Cited at page 10)

[20] Enel X. Le comunità energetiche rinnovabili, 2022. (Cited at pages 10 e 11)

[21] G. Boye Olsen. Energy communities in denmark: Good examples and concerns, 2020. (Cited at page 10)

[22] Jens Lowitzsch. *Energy transition financing consumer co-ownership in renewables*. Palgrave Macmillan, 2018. (Cited at pages 10, 14, 15 e 16)

[23] Presidenza del Consiglio dei Ministri. Decreto-legge 30/12/2019 n.162, 2019. (Cited at page 11)

[24] ARERA. Aggiornamento dei prezzi minimi garantiti per l'anno 2022, 2022. (Cited at page 12)

[25] Legambiente. Comunità rinnovabili 2021. sole, vento, acqua, terra, biomasse. lo scenario della generazione distribuita nel territorio italiano. lo sviluppo dei nuovi modelli energetici nei territori in attesa del completo recepimento della direttiva europea, 2021. (Cited at page 12)

[26] ICA. Co-operative identity, values principles, 1995. (Cited at page 14)

[27] Confartigianato. Come si costituisce una comunità energetica, 2022. (Cited at page 17)

[28] D. Infield I. Richardson, M. Thomson. Richardsonpy tool, 2019. (Cited at pages 20 e 38)

[29] U.S. Energy Information Administration. Solar explained: Solar energy and the environment, 2022. (Cited at page 21)

[30] GSE. Rapporto statistico solare fotovoltaico 2020, 2020. (Cited at page 21)

[31] R. Wang T. Zhang. *Handbook of Energy Efficiency in Buildings*. 2015. (Cited at page 22)

[32] Sinovoltaics. Single axis trackers. (Cited at page 23)

[33] Sinovoltaics. Dual axis trackers. (Cited at page 24)

[34] PVGIS. Global irradiation and solar electricity potential, 2019. (Cited at pages 26, 27, 45 e 47)

[35] D.Moser M. Mazzer. Come l'energia solare può alimentare l'italia senza consumare più suolo, 2021. (Cited at page 24)

[36] Arera. Dati statistici, 2018. (Cited at page 29)

[37] Tesla. Tesla powerwall datasheet, 2018. (Cited at page 29)

[38] LG. Lg chem datasheet, 2018. (Cited at page 29)

[39] Sonnen. Sonnen hybrid 9.53 datasheet, 2021. (Cited at page 29)

[40] ISTAT. Indagine sui consumi energetici delle famiglie, 2013. (Cited at pages 39 e 41)

[41] EnergyUseCalculator.com. (Cited at page 40)

[42] ISTAT. Internet: accesso e tipi di utilizzo, 2020. (Cited at page 40)