Università degli Studi di Pavia

DIPARTIMENTO DI INGEGNERIA INDUSTRIALE E DELL'INFORMAZIONE
Corso di Laurea Magistrale in Ingegneria Elettrica

TESI DI LAUREA MAGISTRALE

Development of a tool for energy analysis of Renewable Energy Communities (RECs) from limited GIS data Case study: primary cabin AC001E01166

Candidato
Kevin Dalla Rosa
Matricola 505879

Relatore
Prof.ssa Norma Anglani

Anno Accademico 2024/2025

Abstract

The study focuses on the development of tools for energy assessment of buildings below a primary cabin, with the task of identifying potential energy communities in rural and urban settings. The model developed uses limited geographical data to approximate real-world scenarios that can build useful data sets for planning projects to meet European climate deals. The scenario examined in the study assesses the potential of a renewable energy community to counter the growing problem of national energy poverty. By analyzing the availability of photovoltaic energy production from homes, farms and municipal facilities, the purpose is to size an energy community capable of promoting distributed and sustainable energy models.

List of Figures

List of Tables

Listings

Contents

Bibliography **133**

Chapter 1

Introduction

1.1 What are RECs'

A REC, a renewable energy community, is a collection of citizens, small and medium-sized enterprises, local authorities, religious bodies and others, who share the renewable energy produced by entities of the community. The main actors in the community are:

- Producers: those who own a renewable energy production plant (photovoltaic, wind, hydroelectric, biomass, et.) and make all or part of the energy produced by the plant available to the community,
- Consumers: entities in the community, which, despite not having an energy production system, can virtually self-consume the energy provided by other members; vulnerable customers and low-income households also fall into this group;
- Referent: who manages relations with the GSE, not necessarily a member of the RECs.

The energy patterns of RECs' members do not change: excess energy is sold to the grid while purchasing through suppliers, but simultaneous consumption and production are remunerated through dedicated tariffs. In addition to the environmental benefits they bring, renewable energy communities lead economic benefits to consumers and producers, and counteract the spread of energy poverty. From a social point of view,

they increase awareness of renewable energy sources by creating a culture of sustainability, and they contribute to the innovation and development of alternative methods of electricity management. RECs', due to their potentiality, have been pushed and promoted by European legislation. They were introduced in 2019 by the 'Clean Energy Package'[2] and later deepened by the RED II[3] and RED III[4] directives, where they are defined as legal entity based on open and voluntary participation, whose purpose is not to generate financial profits but to bring environmental, economic and social benefits to community members. Therefore, renewable energy communities are non-profit entities in which energy companies are not allowed. As for Italian legislation, the Milleproroghe decree[5] is the main one to introduce the concepts of energy community and self-consumption, while the MASE decree promotes the creation of RECs through PNRR incentives and tariff concessions.
There is a fundamental aspect that regulates the RECs', the PODs of the members of the same community, must necessarily fall under the same primary cabin. In fact, in the study, the potential of a REC of the buildings below a primary cabin will be evaluated. [6][7][8][9][10][11][12]

1.2 What are GIS maps

Geographical maps have a very ancient history, used for exploration, strategy or navigation they have been fundamental in the history of humanity. Maps today are very different from a hundred years ago, although the purpose has remained almost unchanged. Starting in the 1960s, geographer Roger Tomlinson coined the term GIS. Known as the 'father of GIS', he first developed a computer system for the cataloguing and analysis of Canada's land resources. His work contributed enormously to the revolution in cartography.
GIS Mapping took about 30 years to spread and take hold, one boost to the diffusion was the development of computers, from their propagation to the growth of their computing power. Although it has divided the world of cartography, it is evident how powerful this tool is in the modern world, from spatial analysis to everyday life. Basically, GIS maps, by georeferencing data sets to the map, open a door to spatial analysis

unthinkable before their invention. The combination of multiple data layers allows urban planning, infrastructure and environmental management decisions to be supported easily and quickly. [13] [14]
In the case of this study, starting from vector maps available on the geographical portal of Emilia-Romagna, it was possible to analyse a vast territory in a short time. Often one of the problems that GIS maps bring with them is that of having unbiased data. The constant changes in the territory, the large amount of information that exists, the high costs for updating, the legal restrictions and the purposes of using GIS information, inevitably lead to not having the information useful for the research in question. Despite this minor flaw, thanks to the way GIS maps are designed, this obstacle can be overcome and the goals set can be achieved.

This thesis is structured into two main chapters. Chapter 2 establishes the theoretical framework underpinning the development of the proposed method for identifying energy communities. It critically examines the challenges inherent in mapping and managing a large number of buildings, details the process of creating customised profiles for each building based on their unique characteristics and consumption patterns, and discusses the obstacles encountered during development alongside the strategies implemented to address them.
Building on the robust models developed in Chapter 2, Chapter 3 is devoted to the territorial study in which these models are applied. This chapter focuses on an in-depth analysis of the results obtained and a comprehensive evaluation of the energy exchanges within the study area. Finally, the Conclusions section synthesises the insights gained and offers reflections on potential future developments and enhancements to the proposed method.

Chapter 2

RECs Analysis tool

This chapter will describe the decisions and motivations that led us to develop this analysis model. A detailed description of the functioning of the tool developed will be carried out, without specifying the number of buildings examined, showing the versatility that an analysis of this type can have.

2.1 Objective of the method and similar tools available.

Starting with a simple request, *'Can the construction of an energy community including public administration buildings, public housing, the curia, some housing and farms be considered?'* opens up an important set of considerations. One of the first is to check whether systems that already perform this type of analysis exist. Recon[15], for example, is one such system; by inputting users, producer and consumers, it returns a reasonably accurate analysis with several useful perspectives to assess feasibility. Unfortunately, it has a significant disadvantage: scalability. For an analysis of 10 buildings, it is very effective, but with 50 buildings, the process becomes tedious, repetitive, and mechanical. It is evident that scalability is a fundamental parameters to consider. Unfortunately, our research did not reveal any truly scalable tool capable of assessing the feasibility of a REC.

To conduct the study independently, we need to evaluate two fundamental aspects of Renewable Energy Communities: the production of energy for self-consumption and the community's energy requirements.

According to REC regulations, production can be diversified, with the only constraint being that it must come from renewable sources. Another significant limitation concerns installation size, which must not exceed 1MW[6] of installed power. For simplicity of analysis, construction, and to avoid excessive morphological specificity, this study does not consider renewable sources other than photovoltaics. In conclusion, regarding production, we choose to use photovoltaics as a renewable generator due to its installation simplicity (disregarding the legislative aspects governing its installation). As is well known, photovoltaics entails a considerable challenge: non-programmability. The data required to generate the energy production curves from photovoltaic installations consists of the set of plant parameters and their production potential. To obtain plant data, it is necessary to evaluate the buildings on which installation is planned, while for the production curve, we can use systems widely adopted by planners, such as PVGis[16] or the Italian Solar Radiation Atlas[17]. Unfortunately, both suffer from the same non-scalability problem. One might consider using tables containing monthly or daily average radiation data, but this would sacrifice the ability to create quarter-

hourly analysis. In practice, it is necessary to find an efficient and time-saving method for constructing production curves.

For construction of consumption profiles, less information is needed: we must understand the building type and associate it with the relevant profile. However, finding complete energy profiles is not straightforward. There are electricity consumption profiles linked to monthly or daily averages, such as those available from the Arera website[18].

In summary, starting from the area under analysis, it is necessary to create production and consumption curves for each building of interest to size the REC and estimate potential energy exchanges. Despite not having extensive data available, from the territorial analysis, it is possible to create a functional model of the renewable energy community.

2.2 Available data

Starting from the need to study a portion of territory, correctly identifying buildings of interest is essential for constructing production and consumption curve. Various GIS maps of the territory under examination are readily available. This study analyzed a portion of the Emilia-Romagna region's territory, which led us to consult the region's geoportal[19]. To identify, catalog and process the data of each building in our work area, we used the following layers:

- V_EDI_GPG: Layer containing the shapes identifying individual buildings, as shown in figure 2.1with data presented in table 33;
- V_FAB_GPG: Layer containing the shapes identifying the building groups, the structure, as shown in figure 2.2 with data presented in table 34;
- V_UVL_GPG: Layer containing the shapes of the volumes that make up each building, where the shapes are as in figure 2.3 and the data as in table 35;
- Uso_Suolo: Layer containing the shapes identifying the use of each piece of area,as shown in figure 2.4 with data presented in table 36.

Figure 2.1: *Figure of the V_EDI_GPG layer.*

Looking at the structure of the data contained in the downloaded layers, and consulting the document[1] prepared by the region for the use of geographical datasets, it is evident that many attributes are incomplete and superfluous for our analysis purpose. The processing and reworking of these data will be demonstrated in subsequent sections. Additional useful data for analysis, such as the population in the area of interest,

Figure 2.2: *Figure of the V_FAB_GPG layer.*

Figure 2.3: *Figure of the V_UVL_GPG layer.*

the average area of dwellings, and the technological status of homes, were obtained from the ISTAT portal[20].

Databases containing household consumption are of two types. The first is the average monthly consumption by time bands, showing the monthly average of energy consumed by dwelling category and province. The second is the series of databases divided by supply size, showing the monthly hourly average by dwelling type and province. Their structure can be seen in table 37 and 38 respectively.

The available data, while limited, is sufficient to determine the information needed to conduct a feasibility analysis of Renewable Energy Communities (RECs).

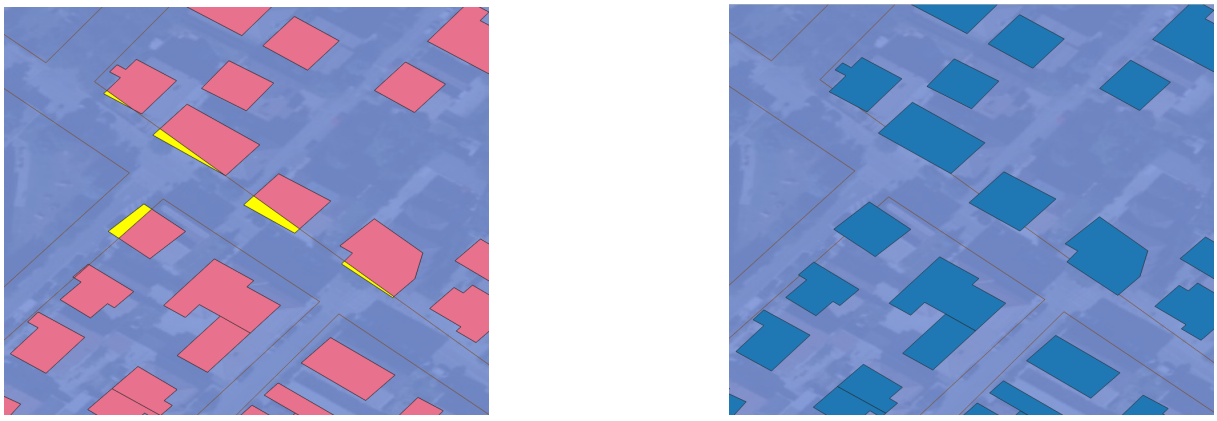
Figure 2.4: *Figure of the Uso_Suolo layer.*

2.3 Manipulation of geographical data

The key to making the methodology efficient is to derive the necessary parameters for the study, i.e. size of the building's PV system, southern orientation, inclination and building type, without having to exanimate the building through satellite maps. As a first step, we can filter the layers to remove buildings outside our territory of interest, and then eliminate unnecessary attributes from the remaining date. It is essential to retain the tags representing the building category and the unique code for each building.

2.3.1 Building categorization method

One of the most crucial operations of this method is the categorization of various buildings. Although seemingly trivial, this process is not straightforward. For security reasons, we do not have detailed descriptions of each building, but rather general categories. As shown in the tables of the EDI33 and FAB34 layer, there are attributes TY_EDI and D_TY_EDI which contain short description as presented in table 2.1. However, in these two layers of interest, general terminology predominates, compelling us to find an alternative approach to catalog the buildings. The solution involves the Uso_Suolo layer. As shown in table 36, it contains only a shape 2.4 database with land descriptions. The legend [21] for this layer is not included because it contains 90 different land use types. Through the use of software specifically built for GIS maps, in our case QGIS, various functions can be employed to manipulate geometric figures



(a) *Intersection mistake.* **(b)** *Correct intersection.*

Figure 2.5: *Displaying the overlay of EDI and USO_Suolo layers.*

and their attributes. To generate an attribute that identifies each building, we used the *intersection* function, which, when passed two layers, returns the first layer with the attributes of the second added. Specifically, by selecting the EDI_layer as the first layer and Uso_Suolo_layer as the second, we obtain a layer with the geometric shapes of the first but with the intersection shape categories of the latter. There is, however, a complication. The two layers do not always overlap perfectly. As illustrated in figure 2.5a, the yellow sections are the artifact, essentially the building shape overlaps with both the street shape and the urban residential fabric. One risk when the two layers do not intersect perfectly is that, if you have previously calculated the available area of each building, the intersection function generates two different shapes. The attributes are identical except for the ground description, creating the risk of counting the same building as two separate entities later in the analysis. One solution is to check for each shape in the EDI_layer how many intersections are generated between the two layers, and then create the new one by reporting all intersection attributes in the land use field. To make this process efficient,we developed a processing tool 1 in Python within QGIS that handles this task. The result of the operation is shown in figure 2.5b. With two attributes for each building that identify its type, we can easily categorize buildings according to these two attributes.

Table 2.1: *Description of TY_EDI and D_TY_EDI values. [1, 109]*

TY_EDI	D_TY_EDI
1	generica
2	anfiteatro
3	battistero
4	campanile
5	castello
6	chiesa/basilica
7	edificio industriale
799	altro edificio industriale
702	hangar
701	capannone
8	edificio monumentale
9	edificio rurale
10	faro
1099	altro faro
1002	fanale
1001	radiofaro
11	minareto, moschea
12	mulino
13	osservatorio
14	palazzo a torri/grattacielo
15	palazzo dello sport
16	rifugio montano
17	tempio
18	tribuna di stadio
19	villa
20	villetta a schiera
97	non conosciuto
98	non assegnato
99	altro

2.3.2 Creation of PV database

In our analysis, we assume that we are evaluating the maximum photovoltaic power that can be installed on each building to assess the maximum potential of the Renewable Energy Community (REC). As already mentioned, the key parameters for assessing installable power are the available surface area and the roof pitch. The ground area, represented by the attribute Sup_Suolo, indicates the area occupied by the building on the ground (i.e. the horizontal projection). This is obtained on QGIS through the *$area* function, which calculates the area of each shape representing the buildings. For the roof inclination, ex examination of the V_UVL layer data, tab. 35, should reveal information describing the type of roof (Open Gable, Dormer, Cross Hipper, etc. 2.6), as well as the ridge and eaves of the roof, which would allow for calculation of the real roof inclination. However, such information are not given in the dataset. Given this lack of data, and to avoid the time-consuming process of examining each building individually via satellite figurery, we have chosen to assume all building roofs are Open Gable type with a standard pitch of 35°.

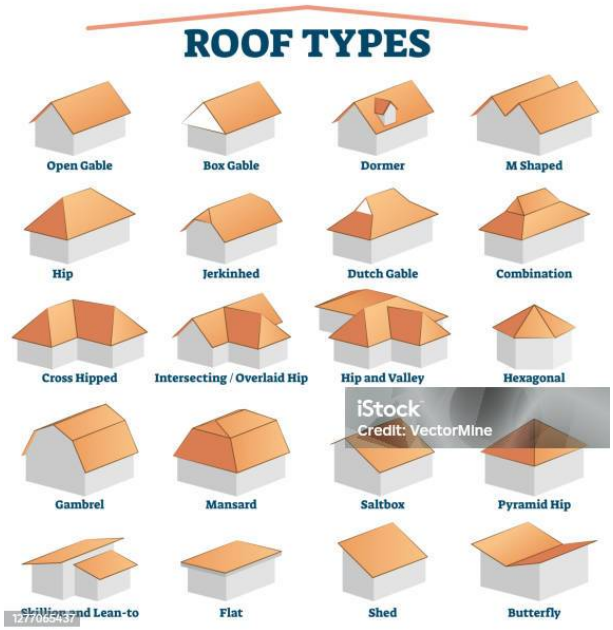


Figure 2.6: *Picture of the various types of roofs that can be found in cities.*

Another assumption made to determine the useful surface area for installing photovoltaic systems is to consider only the best south-facing side of each building, and that

a maximum of 80% of this surface area can be utilized. This limitation accounts for the possible presence of chimneys, skylights and other architectural constraints. This calculation is easy to implement in QGIS, require only a multiplication of the attribute Sup_Suolo by 0.4 (representing half of the roof area for the south-facing portion, multiplied by the 80% utilization factor). The next step, after selecting one of the many types of PV panels on the market, is to estimate the number of panels and the installable power by following the equations 2.1 and 2.2.

$$N_{pfv} = \frac{AU}{A_{pfv} * cos(35)} \tag{2.1}$$

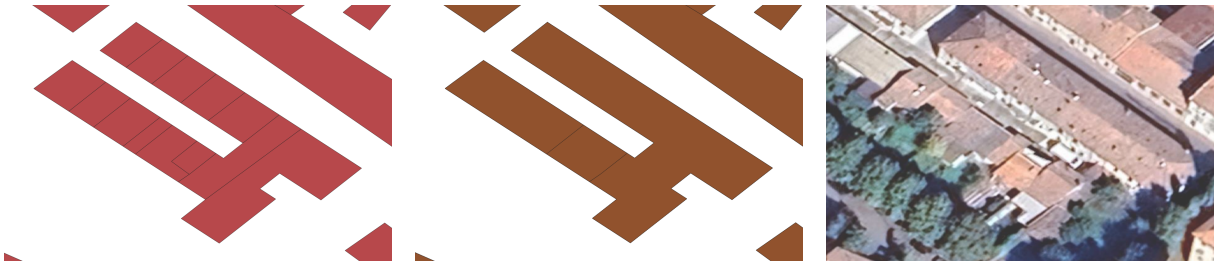
$$P = N_{pfv} * P_{pfv} \tag{2.2}$$

Where N_{pfv} is the number of PV panels, AU is the useful area, A_{pfv} is the single panel area and P is the size of power plant. Having dimensioned the PV system, it is now necessary to find the azimuth angle of each plant. It is the fundamental angle that determines the production of electricity during the day.

As previously mentioned, the purpose of this tool is to be scalable and to eliminate the need for data verification with satellite images, which would otherwise require human intervention or an image recognition algorithm. Due to the lack of roof description, we treat each buildings as a black box described by a shape. The power of GIS maps allows these shapes and their vertices to be geo-referenced, allowing us to find angles, such as the slope of a segment.
For simplicity, let us begin by considering buildings with four sides. As we can see in figure 2.7a, these buildings have four sides, and examining the satellite image 2.7b, confirms that our assumption of a Open Gable roof with 80% of the most south-facing side covered with PV are reasonable. As a general rule for this category of buildings, we derive the azimuth angle using the long side that we consider to be the eaves line. Unfortunately, due to the construction of the EDI_layers and how we want to derive the angle, other building types could be erroneously classified as four-sided. As demonstrated in Figure 2.8, some buildings fall into this category, but applying the same criteria would lead to an assumption of common eaves side, suggesting an M

(a) *EDI_Layer* **(b)** *SAT_Image*

Figure 2.7: *Visualization of buildings formed by four side on QGis.*



(a) *EDI_Layer* **(b)** *FAB_Layer* **(c)** *SAT_Image*

Figure 2.8: *Visualization of buildings on QGis.*

Shaped roof. However, looking at the satellite image2.8c, this does not reflect reality. To address this problem, a simple solution is to calculate the azimuth angles of the buildings using the FAB_layer. As shown in figure 2.8b, each shape is the set of several buildings if they are part of a complex, otherwise it is identical to the shape of the EDI_layer if isolated structures as in figure 2.7a.

The logic behind the calculation of the azimuth angle for buildings with four sides is as follows:

- extraction of vertices with their coordinates;
- elimination of the last vertex which is equal to the first;
- major side search, where $p1.x$ $p1.y$ are the x and y coordinates of points,

$$dx = p2.x - p1.x \tag{2.3}$$

$$dy = p2.y - p1.y \tag{2.4}$$

$$l = \sqrt{dx^2 + dy^2} \tag{2.5}$$

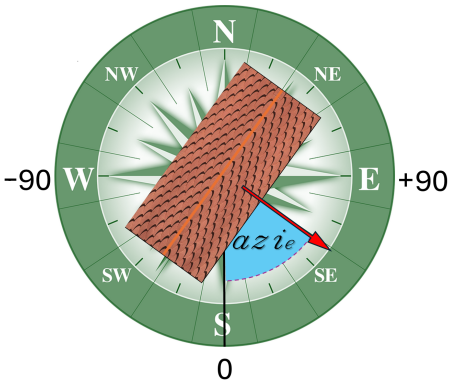


Figure 2.9: *Azimuth reference system.*

- angle calculation with formula

$$\gamma_e = \arctan \frac{dy}{dx} \tag{2.6}$$

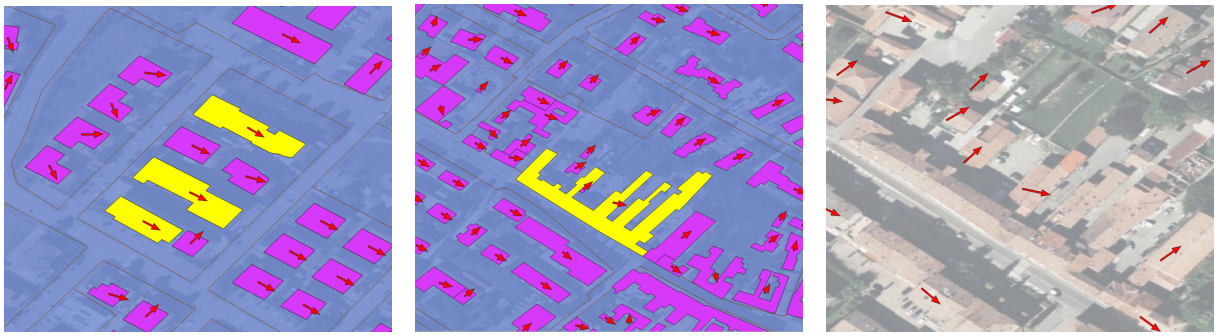
- bring the angle between -90 and +90;
- assign the calculated angle to the building.

These operations are implemented in the function on line 108 of the 2 script. It should be noted that since the angle calculated with the arc tangent uses the Cartesian reference system, but we want to reference it to the south, we must add 90° to rotate the reference system. However, the calculated angle does not indicate the normal direction of the surface but rather the direction of the eaves line. To determine the normal direction, we must subtract 90°. Effectively, the angle calculated with the arc tangent returns the direction of the surface referring to the south. For simplicity, since we are carrying out the analysis with only the better exposed half of the roof, we can keep the azimuth angle between +90 and -90, as in figure 2.9. To summarize, we now have a tool to derive the south orientation of FABs with 4 sides.

Next, we need to find the azimuth of all buildings with more than four sides. We consider complexes of buildings and then associate the angle of the complex to each building. Since we are analyzing RECs, a structure consisting of several buildings can be considered as a single plant oriented as the structure. However, for detailed

(a) *FAB with few vertices.* **(b)** *FAB with enough vertices.* **(c)** *FAB with many vertices.*

Figure 2.10: *Visualization of different FAB on QGis.*



(a) *Simple FAB orientation.* **(b)** *Complex FAB orientation.* **(c)** *Satellite complex FAB.*

Figure 2.11: *Visualization of the orientation of selected building complex on QGis.*

analysis, we decompose it into many small plants with the same angle.

The significant challenge is that it becomes difficult to create additional subgroups for buildings not belonging to four side category. Treating FABs as black boxes, we know the number of vertices and can distinguish them into categories of more than four vertices and less than four vertices. The category of structures with more than four vertices include a wide variety of building typologies, as in figures 2.10 ranging from those with six vertices to those with dozens of vertices.

One promising approach is to use PCA Principal Component Analysis [22]. This is a mathematical method widely used in machine learning to change the dimension space of data. PCA determines the principal directions of variation inside a set of points in a space. These directions are called *principal axes* and correspond to the eigenvalues of the system's covariance matrix. Starting with understanding of PCA, we will explore how it can be applied to determining the orientation of a complex buildings.

Given a data matrix $\mathbf{X} \in \mathbb{R}^{n \times p}$ containing n sample of p variables, the PCA tries to find a linear transformation that maps the original data into a new co-ordinate space, where:

1. the components are orthogonal to each other;
2. The components are ordered according to the amount of variance they capture.

Mathematically, the PCA is obtained by calculating the eigenvectors and eigenvalues of the data covariance matrix:

$$C = \frac{1}{n-1} X^T X \tag{2.7}$$

where X are the data centered by subtracting the average. Eigenvectors v_i of the matrix C, represent the directions of the principal components, while the corresponding eigenvalues λ_i indicate the amount of variance explained along each direction.

In the specific case of analyzing the orientation of a building, PCA can be used to determine the principal axis of the polygonal shape describing the FAB. The eigenvector associated with the largest eigenvalue indicates the direction of maximum variance, which corresponds to the principal axis of the structure. The implemented algorithm that determines the azimuth angle proceeds for each FAB with more than 4 sides as follows:

- extraction from the shape the vertices and their coordinates, the vertices are represented as points in a two-dimensional space (x, y);
- elimination of the last vertex which is equal to the first;
- centroid calculation and vertex centreing by subtracting the centroid from the coordinates

$$\mathbf{c} = \frac{1}{n} \sum_{i=1}^{n} \mathbf{p}_i \tag{2.8}$$

$$\mathbf{p}'_i = \mathbf{p}_i - \mathbf{c} \tag{2.9}$$

  where p_i represents the coordinates of the i-th vertex of the building, c the centroid and p'_i the centered coordinate;

- calculation of the covariance matrix with centered points

$$C = \frac{1}{n-1} \sum_{i=1}^{n} p_i' p_i'^T = \begin{bmatrix} \sigma_x^2 & \sigma_{xy} \\ \sigma_{xy} & \sigma_y^2 \end{bmatrix} \tag{2.10}$$

  where σ_x^2 is the variance of coordinates x, σ_y^2 is the variance of coordinates y, and σ_{xy} is the covariance between x and y.

- calculation of eigenvalues $\lambda_1 \geq \lambda_2$ and corresponding eigenvectors v_1 e v_2 of the covariance matrix.

- the eigenvector v_1 associated with the largest eigenvalue λ_1 indicates the direction of maximum variance, i.e. the principal axis of the complex. The orientation angle γ_e, azimuth of building, can be calculated as:

$$\gamma_e = \arctan \frac{v_{1y}}{v_{1x}} \tag{2.11}$$

  where v_{1x} and v_{1y} are the x and y components of the principal eigenvector.

These procedures are described in the 2 script from line 138. Examining figures 2.11 reveals the system's overall effectiveness. For elongated geometries, as illustrated in figure 2.11a, the method performs exceptionally well, as the principal components efficiently identify the primary axes of the structure. In contrast, for more complex, square-like buildings—such as those shown in figure 2.11b the method is less effective. This diminished performance is particularly evident for square-shaped structures, where PCA yields principal axes with similar variances, making it more challenging to determine the correct orientation. Figure 2.11c provides a magnified satellite image of the yellow building highlighted in figure 2.11b, further underscoring the difficulties encountered with such geometries.

To recapitulate, for calculating the azimuth angles of each building, we use the process tool implemented in python which employs the two previously described methodologies. In addition to the angle calculation, a function has been implemented that assigns each angle an arrow oriented according to the calculated angle to better visualize the results.

2.3.3 Creation of housing database

In order to associate the correct consumption curves with buildings as accurately as possible, it is necessary to identify the building types. This has already been done in the section on building classification. We have successfully identified residential buildings, buildings related to agricultural activities, municipally-owned properties, service establishments, and church-owned buildings. For non-residential buildings, we simply need to associate the appropriate consumption curves with each building type to create consumption profiles. For residential buildings, however, another difficulty arises. Dwellings, when viewed solely as two-dimensional shapes occupying ground space, can vary significantly. They could be single-family houses, apartment blocks, multi-family buildings, or even sheds. To differentiate between these various residential types, we can use the V_UVL_layer data 35. This layer provides each building with a rough shape describing its massing, showing one or more areas with different heights. Certainly, starting from the provincial average value of the area occupied by dwellings is a good start. However, the fact remains that some dwellings may be larger or smaller than this average area. The ideal solution would be to access the cadastral geoportal where each building's property number is precisely indicated. This would allow for the identification, with almost absolute precision, of the real distribution of dwellings, apartments, and houses throughout the territory. In the absence of these data, it was assumed to reason as follows.
Specifying, what we are actually interested in is associating, or rather estimating, the number of PODs (Point of Delivery) with each building. Some dwellings that we know for sure have a unique POD, are villas and terraced houses. Sometimes these characteristics appear in the data, so by finding one of these tags we associate a unique POD with the building. For all other dwellings, we assume parameters that discriminate between buildings. We distinguish two cases, when the building has a single volume or several volumes:

1. Multiple Volume Blocks Analysis (when a building has multiple sections):

    - If the building's ground area is less than $350m^2$, it's classified as a single residence;

- Otherwise, each volume block is analyzed separately:
  - Floor count is calculated by dividing the height by $3.5m$ (average height between floors);
  - A block with height under $3.5m$ is considered a warehouse/storage (0 POD);
  - A block with area less than $80m^2$ and only one floor is considered a warehouse (0 POD);
  - A block with area less than $200m^2$ is considered to have one apartment per floor;
  - A block with area over $200m^2$ is calculated as multiple apartments: ((area-12)/110) $\times$ floor count.

2. Single Volume Block Analysis:

   - Floor count is calculated by dividing the height by $3.5m$;
   - A block with height under $3.5m$ is considered a warehouse (0 POD);
   - A block with area less than $60m^2$ is considered a warehouse (0 POD);
   - A block with area less than $200m^2$ and fewer than 3 floors is considered a single-family house (1 POD);
   - A block with area less than $250m^2$ is considered to have one large apartment per floor;
   - A block with area of $250m^2$ or more is calculated as multiple apartments: ((area-12)/110) $\times$ floor count.

This formula accounts for common areas like stairwells ($12m^2$) and assumes an average apartment size of $110m^2$. The procedure described is shown in the 3 code implemented in QGIS, this code also calculates and returns the volume of each building. Certainly, from the perspective of counting PODs for residential buildings, this approach could be improved. However, in the absence of precise data, this method can provide a count that nevertheless allows for apriori hypotheses regarding a potential energy community in a region.

2.4 Development of production profiles

One goal is to conduct an energy analysis every 15 minutes throughout the day. Tools such as PVGis, although valuable, are not scalable and do not offer this capability. An alternative would be to build an algorithm that can calculate the solar radiation incident on each photovoltaic plant and derive the power generated. Such a tool would also allow the analysis to be carried out for any day of the year and immediately calculate all the power generated by the plants in the energy community for that specific day. The parameters of each PV system have already been determined; now we need to calculate the position of the sun and the components of the radiation affecting the surface.

2.4.1 Sun position

As a first step, we must identify the geographical location where the incident radiation to the surface is to be calculated. In the case of an energy community, we use as coordinates the barycenter of all the buildings on which we are going to perform the analysis. We will then obtain a latitude Lat and a longitude Lon. To calculate the position of the sun, we follow the solar equations developed by Spencer (1971) and later refined by Duffie and Beckman in their seminal text 'Solar Engineering of Thermal Processes'[23]. These equations allow us to accurately determine the solar angles for any hour of a day chosen apriori. To calculate the position of the sun relative to an earth surface at a given time, we follow the next logical steps.

First of all, we define the day of the year (day number) n which varies from 1 to 365 (or 366 for leap years).

The solar declination δ represents the angle between the sun's rays and the earth's equatorial plane, and varies throughout the year due to the tilt of the earth's axis. It is calculated using the Cooper equation:

$$\delta = 23.45 \cdot \sin\left(\frac{360 \cdot (284 + n)}{365}\right) \tag{2.12}$$

where δ is expressed in degrees.

To account for the variation in the Earth's orbital velocity, we calculate the equation of

time (E) that corrects for the difference between mean solar time and true solar time:

$$B = \frac{360 \cdot (n-1)}{365}$$

$$E = 229.25 \cdot (0.000075 + 0.001868\cos B - 0.032077\mathrm{B} - 0.014615\cos 2B - 0.04089\sin 2B) \tag{2.13}$$

where E is in minutes.

We convert local time to true solar time:

$$Solar\_time = t + \frac{(4 \cdot (15 - Lon) + E)}{60} \tag{2.14}$$

Where t is the local time in which we want to calculate the solar heights, it ranges from $00:00$ to $23:59$ with a desired time step, in our case $15minutes$. Lon is the longitude of the place of analysis, 15 is the longitude of the meridian of reference of Italian local time.

The hour angle ω measure the displacement of the sun from east to west:

$$\omega = 15 \cdot (Sola\_time - 12) \tag{2.15}$$

ω is in degree, positive in the afternoon and negative in the morning.

Let us now calculate the fundamental solar angles:

- solar altitude α_s

$$\sin\alpha_s = \sin Lat \cdot \sin\delta + \cos Lat \cdot \cos\delta \cdot \cos\omega \tag{2.16}$$

$$\alpha_s = \arcsin\left(\sin Lat \cdot \sin\delta + \cos Lat \cdot \cos\delta \cdot \cos\omega\right) \tag{2.17}$$

- solar azimuth γ_s

$$\gamma_s = \arcsin\left(\frac{\cos\delta \cdot \sin\omega}{\cos\alpha_s}\right) \tag{2.18}$$

With these equations we have calculated the angles describing the position of the sun at any hour during a day. To determine the irradiation that strikes an inclined surface with azimuth γ and inclination β, we need the angle of incidence θ, which is obtained with:

$$\cos\theta = \cos\alpha_s \cdot \cos\left(\gamma_s - \gamma\right) \cdot \sin\beta + \sin\alpha_s \cdot \cos\beta \tag{2.19}$$

It can be seen that the cosine of θ, $\cos\theta$, varies throughout the day and depends on the inclined surface representing the photovoltaic system.

2.4.2 Solar radiation

After determining the position of the sun and the angle of incidence on the inclined surface, we can proceed with the calculation of solar radiation using the atmospheric transmissivity model developed by Bird and Hulstrom and refined by Duffie and Beckman[23], in agreements with [24, 25, 26]. We know that radiation incident on a surface consists of three parts:

- Direct radiation R;
- Diffuse radiation I_d;
- Reflected radiation I_r.

consequently we will proceed with their calculation. We begin by calculating extraterrestrial solar radiation, which represents the solar energy available outside the Earth's atmosphere:

$$I_0 = S_0 \cdot \left(1 + 0.0344 \cdot \cos\left(\frac{2\pi \cdot d}{\tau_a}\right)\right) \tag{2.20}$$

where:

- S_0 is the solar constant (1367 W/m^2);
- d is the day of the year;
- τ_a is the period of the year (365 or 366 days);
- $fcirc$ is the factor 2π that converts to radians.

This equation takes into account the variation of the Earth-Sun distance during the year.
As a next step, we calculate the relative air mass (Air Mass Ratio), which represents the relative thickness of the atmosphere that solar radiation must pass through:

$$M = \sqrt{1229 + (614 \cdot \sin\alpha_s)^2} - 614 \cdot \sin\alpha_s \tag{2.21}$$

where α_s is the solar height.

Next, we calculate the transmissivity coefficients representing the fraction of radiation that passes through the atmosphere:

$$\tau_b = 0.56 \cdot (e^{-0.65 \cdot M} + e^{-0.095 \cdot M}) \tag{2.22}$$

where τ_b is the direct radiation transmissivity coefficient. For diffuse radiation we compute:

$$\tau_d = 0.271 - 0.294 \cdot \tau_b \tag{2.23}$$

For the reflected component, we use:

$$\tau_r = 0.271 + 0.706 \cdot \tau_b \tag{2.24}$$

We now calculate the individual components of the radiation:

$$I_s = I_0 \cdot \tau_b \tag{2.25}$$

where I_0 is the extraterrestrial radiation and I_s is the potential direct radiation on a surface normal to the sun's rays. Direct radiation on an inclined surface is given by:

$$R = I_s \cdot \cos \theta_i \tag{2.26}$$

where θ_i is the angle of incidence on the inclined surface. We evaluate $R = 0$ when the result is negative. Diffuse radiation is calculated as:

$$I_d = I_0 \cdot \tau_d \cdot \frac{\cos^2(\beta)}{2} \cdot \sin \alpha_s \tag{2.27}$$

where β is the inclination of the surface and $\cos^2(\beta)$ represents the portion of the sky visible from the inclined surface.

The radiation reflected from the ground is:

$$I_r = I_0 \cdot r \cdot \tau_r \cdot \frac{\sin^2(\beta)}{2} \cdot \sin \alpha_s \tag{2.28}$$

where r is the albedo of the terrain and $\sin^2(\beta)$ represents the portion of the terrain visible from the sloping surface.

Finally, we add up the three components to obtain the total radiation on the inclined surface:

$$R_{tot} = R + I_d + I_r \tag{2.29}$$

This value is calculated for each time interval (every 15 minutes) for the inclined surface under consideration, allowing to obtain a detailed solar irradiation profile.

2.4.3 Creation of production profiles

The equations described in the chapters above, allow us to obtaining the irradiance for a specific time of day on an inclined surface at a given location. What we want to do is to create, for each building in a region, its own production curve related to the angle of incidence $\cos\theta_i$. Wanting to obtain a value of the power produced by each building every 15 minutes, and considering that it makes sense to do this for at least one day per month to visualize the seasonal diversity of production, we can calculate the number of parameters for a building:

$$n_h \cdot \frac{60}{p_a} \cdot n_d = 24 \cdot 4 \cdot 12 = 1152 \tag{2.30}$$

where n_h is the number of hours in a day, p_a is the analysis time step in minutes, and n_d the number of days to be analyses. So for n buildings we will have $n \cdot 1152$ parameters. These are all independent because they depend on the position of the sun on that specific day and time, as well as on the inclination of each photovoltaic system.
To reduce the computational operations required to obtain the production curves, we can proceed as follows. By considering a barycenter point of al buildings, we obtain a single latitude and longitude. Since the analysis area is relatively small compared to the earth's circumference, the sun angles do not vary significantly between the buildings at the extremes of the region. Using a point common to all buildings, the solar positions α_s and γ_s became common to all n buildings. What varies are the components R (eq.2.26), I_d (eq.2.27) and I_r (eq.2.28) that depend on the characteristic angle of incidence $\cos\theta_i$ (eq.2.19) of each building.
By modifying the generation and management of data, it is possible to calculate once for all buildings the components related to the sun angles, and then compute the three radiations components for each time step to be analyzed.
For each time step we calculate:

- $t_1 = \gamma_s$ equation 2.18;
- $t_2 = sin(\alpha_s)$ equation 2.16;
- $t_3 = \cos\alpha_s$;

- $t_4 = I_s$ equation 2.25;
- $t_5 = I_0 \cdot r \cdot \tau_r \cdot \frac{\sin \alpha_s}{2}$ modified equation 2.27;
- $t_6 = I_0 \cdot r \cdot \tau_r \cdot \frac{\sin \alpha_s}{2}$ modified equation 2.28.

For each day of the analysis, we calculate $96 \cdot 6$ values that we store and then use for each building. The next step is for each building to reconstruct the daily production curve. It is necessary to calculate from the 6 terms of daily irradiation, the point radiation on each plant in each building.

For each building, and doing this for all buildings, we must, for each time step of the analysis, reconstruct the cosine of the angle of incidence using the eq. 2.19:

$$\cos \theta_i = t_3 \cdot \cos (t_1 - \gamma_e) \cdot \sin (35) + t2 \cdot \cos (35) \tag{2.31}$$

where for each moment t of the day, θ_i is the angle of incidence between the solar rays and the surface, γ_e is the orientation of the building, and 35 is the roof inclination hypothesis made previously.
After that, we use the other terms to calculate the radiation reaching the surface in $[w/m^2]$:

$$R_i = t_4 \cdot \cos \theta_i + t_5 \cdot \cos (35)^2 + t_6 \cdot r \cdot \sin (35)^2 \tag{2.32}$$

By retrieving information about the number of panels installed on that system, the area of each panel, and their efficiency, we calculate the power in $[kW]$ produced in that time step.

$$P_i = R_i \cdot \frac{(N_{pfv} \cdot A_{pfv} \cdot \rho_{pfv})}{1000} \tag{2.33}$$

As can be seen in scripts 4, 6 and 5, the first is the main script, the second one calculates the block of 6 unique terms for each building, while the third script shows how the data contained in table 39 are loaded, processed, and transformed into the data in table 41. To summarize, in this way we have successfully obtained a general overview of the photovoltaic production from a number n of buildings throughout a year, starting from the information calculated for each building and from a geographical position, considering that we analyze the 15th day of each month. Obviously, this provides a clear-sky curve; by implementing a correct creation of coefficients that simulate cloud

cover, we could also obtain production profiles for non-clear days. Regarding the creation of monthly energy profiles, we take the clear-sky curve of the month for each building and multiply it by the number of days in the month, by the time step of the analysis, and by a factor of 0.7 to simulate that, obviously, not all days are clear.

2.5 Development of consumption profiles

Once the hypothetical number of PODs, or the actual number obtained through alternative methodologies, is determined, it becomes possible to calculate the consumption of each building at every 15-minute interval throughout the month. By employing script 7 and utilizing the province-level hourly data for the analyses region provided by ARERA, as shown in table 38, the consumption curve for each building can be reconstructed. The outcome of this self-developed script is a comprehensive database in which each building's consumption trend is recorded at 15-minute intervals for every month. Additionally, for each month, there is a summary table presenting the annual consumption for each category (i.e. the aggregation of consumption from all buildings and across all hours), accompanied by a series of tables that display the hourly consumption for every month by category. When combined with the production data, this approach enables the execution of the necessary analyses to achieve our stated objectives.

2.6 Summary

In this chapter, we have thoroughly examined the main issues arising from the use of the QGIS layer. We highlighted the scarcity of data available in the maps for the identification of buildings, and introduced tools designed to address this limitation and resolve overlapping errors. In order to simplify the analysis process, we developed methods to determine key parameters, such as the azimuth and the installable power for each building, eliminating the need to examine every structure individually. Finally, we presented the functions required to calculate the power output of the systems at fifteen-minute intervals, and explained the process used to assign the correct

consumption curve to each building. This comprehensive approach lays the foundation for an in-depth territorial analysis and paves the way for the optimal development of renewable energy communities.

Chapter 3

Case study primary cabin AC001E01166

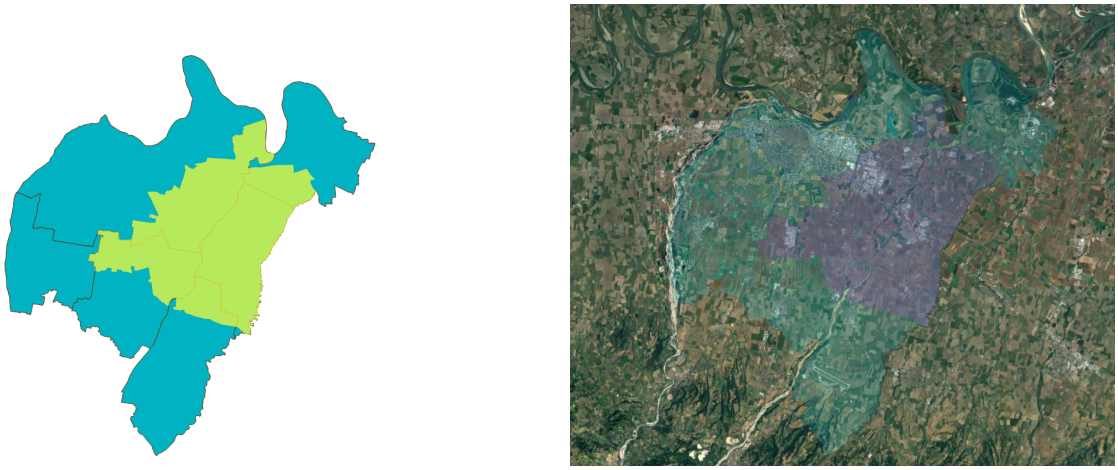
In this chapter, we analyze primary cabin AC001E01166 with the aim of evaluating the feasibility of a single large energy community or several separate communities (one for each municipality). Using the methodology described in Chapter 2, we will assess the maximum installable photovoltaic power capacity and estimate the corresponding energy consumption patterns. This study will develop various analytical tools to support future management of renewable energy communities, while also establishing a sustainable framework for redesigning regional energy distribution, specifically for the area served by substation AC001E01166.

3.1 Primary station territory analysis

In order to begin studying the territory we want to analyze, it is necessary to properly frame the area. The primary cabin AC001E01166 is located in the province of Piacenza, which belongs to the Italian region of Emilia-Romagna. According to the GSE portal[27] (Gestore dei Servizi Energetici) for the determination of primary cabin, developed for creating energy communities, the municipalities associated whit this primary cabin are:

- Caorso;
- Caselle Landi;
- Gossolengo;
- Piacenza;
- Podenzano;
- Pontenure;
- San Giorgio Piacentino.

Through further research, we found that Caselle Landi belongs to the province of Lodi in Lombardy, not Piacenza. Upon closer examination, it appears that no buildings in Caselle Landi are actually served by this primary cabin.
By overlaying the primary cabin layer with municipalities boundaries, we observed that only Pontenure falls entirely within the cabin's coverage area, while the other municipalities are only partially served by this primary cabin, figure 3.1. For Gossolengo, we see that a very small percentage of the municipality is affected by the primary cabin. Looking at the satellite imagery 3.2, we can observe that the territory covered by the primary cabin AC001E01166 is predominantly rural and agricultural in nature. The landscape is characterized by a mosaic of cultivated fields typical of the Po Valley. Within this area, we find scattered rural farmhouses and agricultural buildings , small rural villages and fractions with compact residential clusters, and some industrial/-commercial buildings primarily located along the main roads. The eastern periphery



(a) *Figure of the layers.* **(b)** *Satellite image.*

Figure 3.1: *Figures showing the portions of the municipalities below the primary cabin AC001E01166, in yellow the shape of the primary cabin, in blue the municipal competences.*

of Piacenza is partially included in the substation's service territory and presents a low-density residential development. The building typology is predominantly rural, consisting mainly of agricultural service structures, farmhouses, and small village residential buildings, with limited urban development.

Using the municipal area layers in QGIS, we can calculate the total area in hectares for each municipality. By also computing the portions of each municipality that fall below the primary cabin's service area, we can determine the percentages of municipally territory included in our study. As shown in table 3.1, Pontenure has 98.92% of its total municipal area covered by the cabin, while the other municipalities have an average of 21.4%. Additionally by incorporating building layers in QGis, we can analyze what percentage of the total buildings area is occupied by buildings connected to the primary cabin.

3.2 Building analysis

By importing building shapes and attribute data into QGis, we can conduct initial analyses on the buildings within each municipality. Using the methodologies developed in

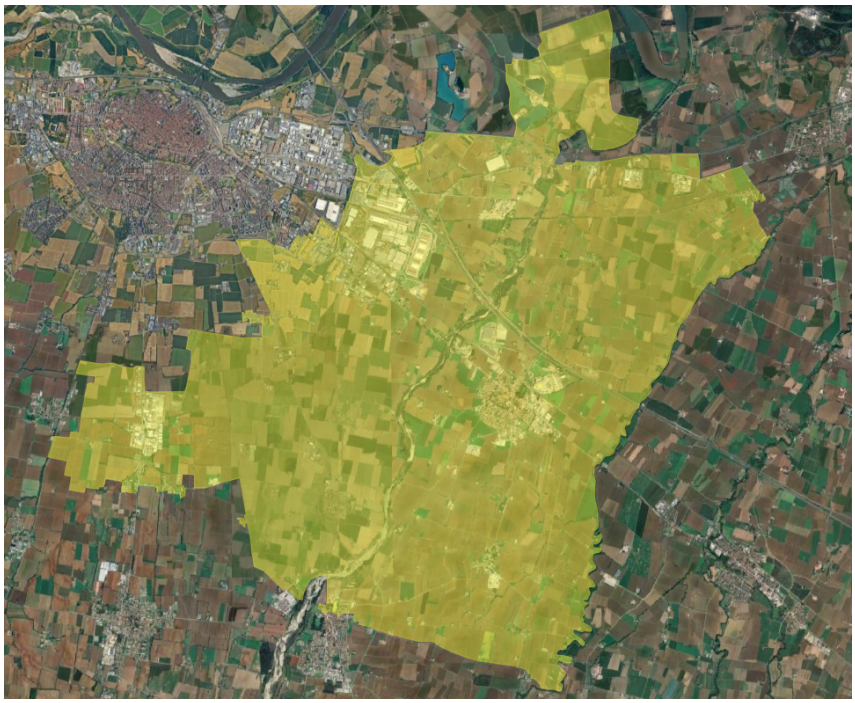


Figure 3.2: *Detail showing the type of land below the primary cabin AC001E01166.*

Table 3.1: *Municipal Coverage Analysis.*

Municipality	Municipal area [ha]	Area Under the Cabin [ha]	% Area U.C	% A_EDI_CAB / A_EDI_COM
Pontenure	3405,037	3368,350	98,92	98,01
Piacenza	11854,801	3325,009	28,05	20,14
Podenzano	4454,148	1944,299	43,65	48,43
Caorso	4097,637	914,369	22,31	27,49
Gossolengo	3144,223	1,599	0,05	0
San Giorgio Piacentino	4879,669	723,714	14,83	10,30

Table 3.2: *Municipal building Analysis.*

Municipality	% A_EDI_INT / A_EDI_COM	% A_EDI_INT / A_EDI_CAB	Surface EDI_INT [ha]	Useful S. EDI_INT [ha]
Pontenure	54,49	55,59	40,758961	16,3025
Piacenza	4,51	22,37	29,532949	11,81270
Podenzano	20,85	43,05	24,057065	9,62300
Caorso	10,19	37,05	7,571567	3,0290
Gossolengo	0	0	0	0
San Giorgio Piacentino	10,30	100	6,875391	2,7500

section 2.3.1, we determine the typology of each building, filtering them to hold only those of interest: dwellings, agricultural structures, church buildings, and municipally-owned properties. This analysis allows us to calculate the percentage of buildings of interest relative to the total buildings in each municipality, as well as their proportion within the primary substation service area.
The table 3.2 shows the following values from left to right:

- the area occupied by the buildings of interest compared to the area occupied by all buildings in the municipality;
- the area occupied by the buildings of interest compared to the area occupied by all buildings under the cabin;
- the area in hectares occupied by the buildings of interest;
- the useful area of the buildings of interest, also in hectares.

The ratio between Useful S. EDI_INT and Surface EDI_INT is 0.4 because it was imposed by us as an initial condition. Tables 3.1 and 3.2 provide an initial analysis of the building stock that could potentially be utilized for the creation of one or more energy communities. Although only a small fraction of each municipality falls within our study area, the available roof surface for PV installation is considerable. Considering the total available area, we are looking at approximately 109 hectares of roof surface, of which 43.5 hectares are considered useful under our specific conditions. Below we will

look specifically at each municipality, except for Gossolengo, which has no buildings of interest under the primary cabin under analysis.

3.2.1 Pontenure

The first municipality we analyze in detail is Pontenure, the only one entirely covered by electricity supply from the primary cabin AC001E01166. As shown in table 3.1, 98.92% of its area is served by the cabin in question, while 98.01% of buildings belong to it, meaning almost all of them. Regarding the percentage of buildings of interest, table 3.2 shows that 55.59% of buildings are of interest, with a total ground surface area of 40.76 hectares potentially available, excluding existing power plants. Going back to the categories concerned, we constructed table 3.3, where we show for each building type the number of buildings belonging to that category and the relative percentage. This municipality comprises 1935 useful buildings with a usable area of 16.3 hectares. It can be seen that the majority of the buildings are for residential use. Looking at the orientation distribution table 3.4 and graph 3.4 shows that most buildings are oriented between -30° and -20° and between 40° and 70°.

Table 3.3: *Buildings type Pontenure.*

Type	**Count**	**%**
Residential	1596	82.48%
Agricultural	203	10.49%
Nurseries	5	0.26%
Sports Areas	5	0.26%
Municipal	4	0.21%
Churches	13	0.67%
Services	14	0.72%
Other	95	4.91%
TOTAL	1935	100.00%

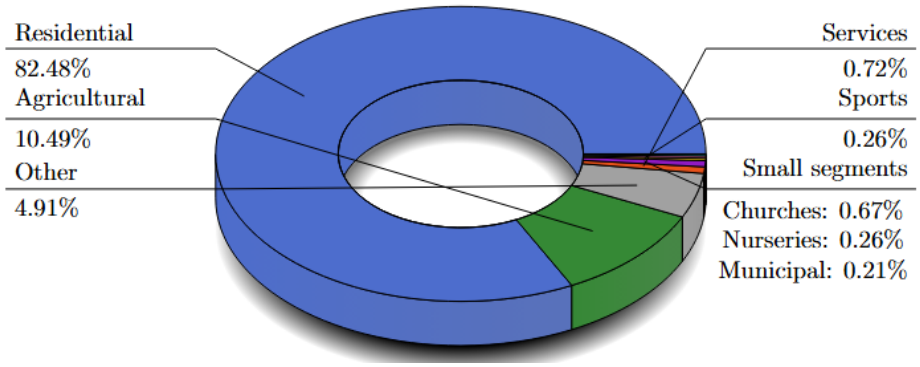


Figure 3.3: *Distribution of building types in Pontenure.*

Table 3.4: *Building Distribution by Orientation Angle of Pontenure.*

Angle Range	Count	% of Buildings	Useful Area [m^2]	% of Area
-90° to -80°	31	1.60%	2575	1.58%
-80° to -70°	37	1.91%	3192	1.96%
-70° to -60°	49	2.53%	3362	2.06%
-60° to -50°	56	2.89%	4473	2.74%
-50° to -40°	175	9.04%	12362	7.58%
-40° to -30°	165	8.53%	12883	7.90%
-30° to -20°	212	10.96%	18987	11.65%
-20° to -10°	168	8.68%	13466	8.26%
-10° to 0°	84	4.34%	6154	3.77%
0° to 10°	40	2.07%	4227	2.59%
10° to 20°	55	2.84%	4646	2.85%
20° to 30°	36	1.86%	3349	2.05%
30° to 40°	48	2.48%	4235	2.60%
40° to 50°	166	8.58%	14712	9.02%
50° to 60°	184	9.51%	17387	10.67%
60° to 70°	184	9.51%	16503	10.12%
70° to 80°	141	7.29%	12966	7.95%
80° to 90°	104	5.37%	7546	4.63%
TOTAL	1,935	100.00%	163025	100.00%

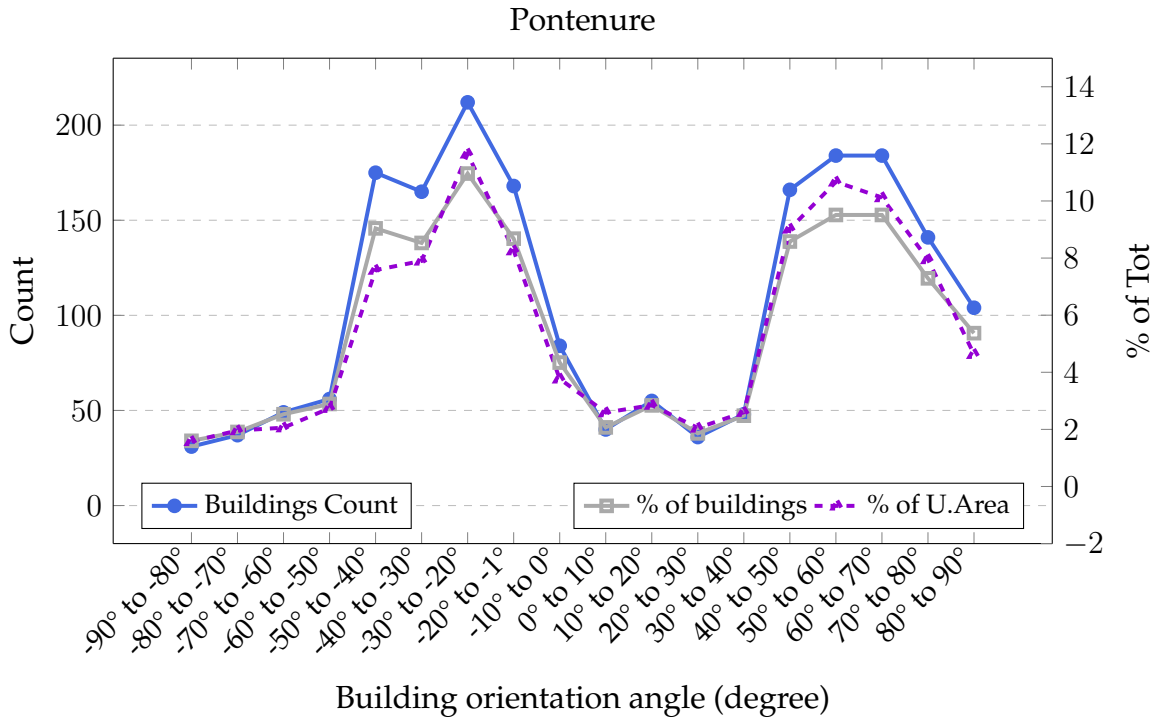


Figure 3.4: *Graph values of table 3.4, distribution of buildings and useful area by orientation of Pontenure.*

3.2.2 Piacenza

This municipality has the second highest number of buildings of interest with a total of 1,470. The useful surface area is 11.46 hectares, of which 77.63% is made up of residential units and 15.51% agricultural activities, as shown in the table 3.5. In this municipality, unlike Pontenure, there are not many municipal buildings, but the five buildings that appear, they belong to a building school. From the 3.6 graph, we can see that the orientation between -20° and -10° and between 70° and 80° predominates.

Table 3.5: *Buildings type Piacenza.*

Type	Count	%
Residential	1141	77.63%
Agricultural	228	15.51%
Nurseries	8	0.54%
Sports Areas	6	0.41%
Municipal	5	0.34%
Churches	3	0.20%
Services	49	3.33%
Other	30	2.04%
TOTAL	1470	100.00%

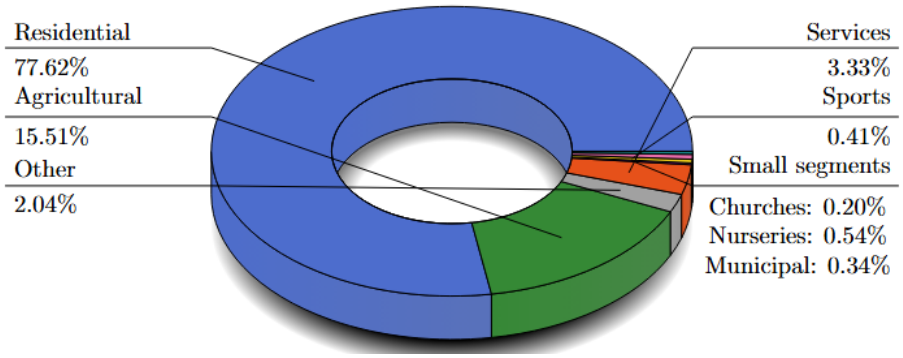


Figure 3.5: *Distribution of building types in Piacenza.*

Table 3.6: *Building Distribution by Orientation Angle of Piacenza.*

Angle Range	Count	% of Buildings	Useful Area [m^2]	% of Area
-90° to -80°	53	3,61%	4748	4,14%
-80° to -70°	38	2,59%	3445	3,01%
-70° to -60°	54	3,67%	5217	4,55%
-60° to -50°	17	1,16%	3302	2,88%
-50° to -40°	106	7,21%	7569	6,60%
-40° to -30°	98	6,67%	8787	7,67%
-30° to -20°	139	9,46%	8000	6,98%
-20° to -10°	155	10,54%	12328	10,76%
-10° to 0°	90	6,12%	6279	5,48%
0° to 10°	66	4,49%	4044	3,53%
10° to 20°	35	2,38%	2270	1,98%
20° to 30°	41	2,79%	3673	3,20%
30° to 40°	26	1,77%	2621	2,29%
40° to 50°	83	5,65%	6756	5,89%
50° to 60°	95	6,46%	6974	6,08%
60° to 70°	135	9,18%	9361	8,17%
70° to 80°	151	10,27%	12476	10,89%
80° to 90°	88	5,99%	6763	5,90%
TOTAL	1,470	100.00%	114613	100.00%

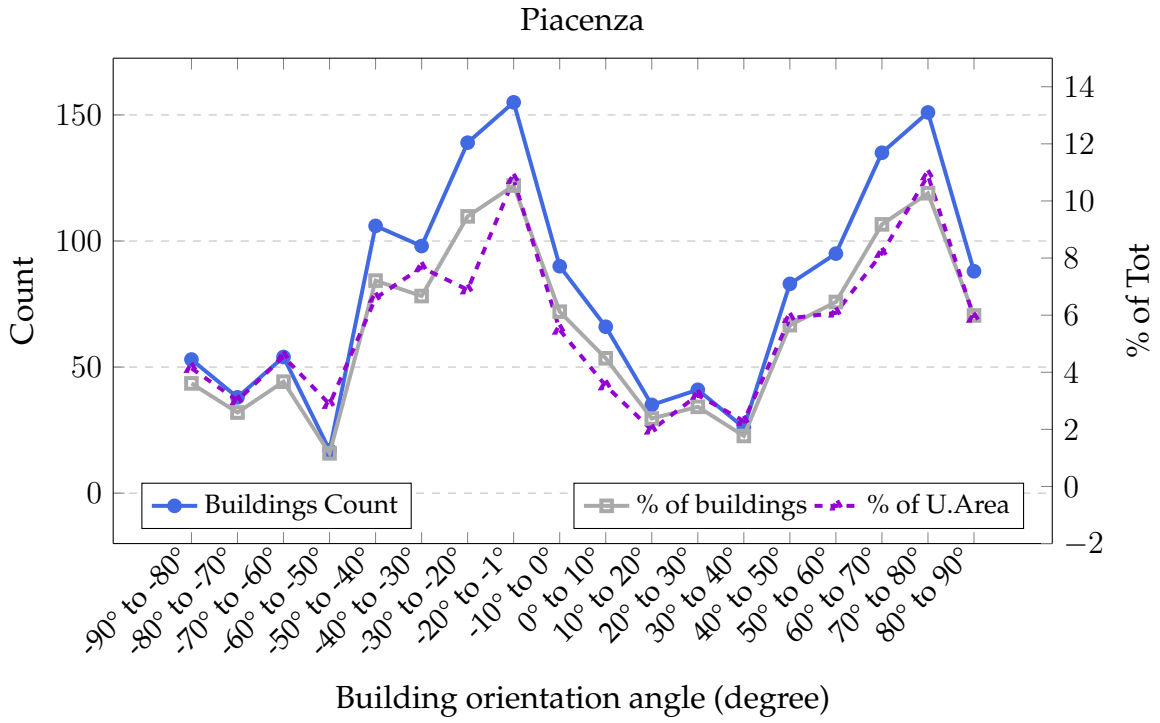


Figure 3.6: *Graph values of table 3.6, distribution of buildings and useful area by orientation of Piacenza.*

3.2.3 Podenzano

Podenzano is the third largest municipality in terms of size and number of buildings. Almost all the structures are dedicated to residential and agricultural use, with a few buildings for sports and church properties, while 5% are related to services and other structures such as landfills, scrapyards, and other buildings, as shown in table 3.7. With 985 buildings, more than 300 buildings are oriented between south and east, graph 3.8, it has a total area of 9.62 hectares.

Table 3.7: *Buildings type Podenzano.*

Type	Count	%
Residential	736	74.72%
Agricultural	182	18.48%
Nurseries	0	0.00%
Sports Areas	2	0.20%
Municipal	0	0.00%
Churches	4	0.41%
Services	43	4.37%
Other	18	1.83%
TOTAL	985	100.00%

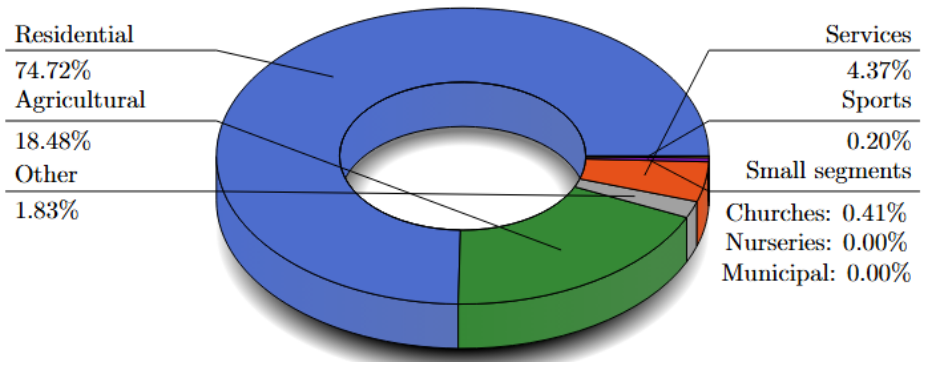


Figure 3.7: *Distribution of building types in Podenzano.*

Table 3.8: *Building Distribution by Orientation Angle of Podenzano.*

Angle Range	Count	% of Buildings	Useful Area [m^2]	% of Area
-90° to -80°	46	4,67%	3696	3,84%
-80° to -70°	35	3,55%	3703	3,85%
-70° to -60°	27	2,74%	4213	4,38%
-60° to -50°	14	1,42%	1059	1,10%
-50° to -40°	30	3,05%	3351	3,48%
-40° to -30°	30	3,05%	3619	3,76%
-30° to -20°	45	4,57%	3629	3,77%
-20° to -10°	109	11,07%	8796	9,14%
-10° to 0°	150	15,23%	11696	12,15%
0° to 10°	41	4,16%	3447	3,58%
10° to 20°	39	3,96%	3128	3,25%
20° to 30°	16	1,62%	1413	1,47%
30° to 40°	22	2,23%	2568	2,67%
40° to 50°	21	2,13%	2380	2,47%
50° to 60°	23	2,34%	2865	2,98%
60° to 70°	43	4,37%	6306	6,55%
70° to 80°	133	13,50%	15802	16,42%
80° to 90°	161	16,35%	14559	15,13%
TOTAL	985	100.00%	96230	100.00%

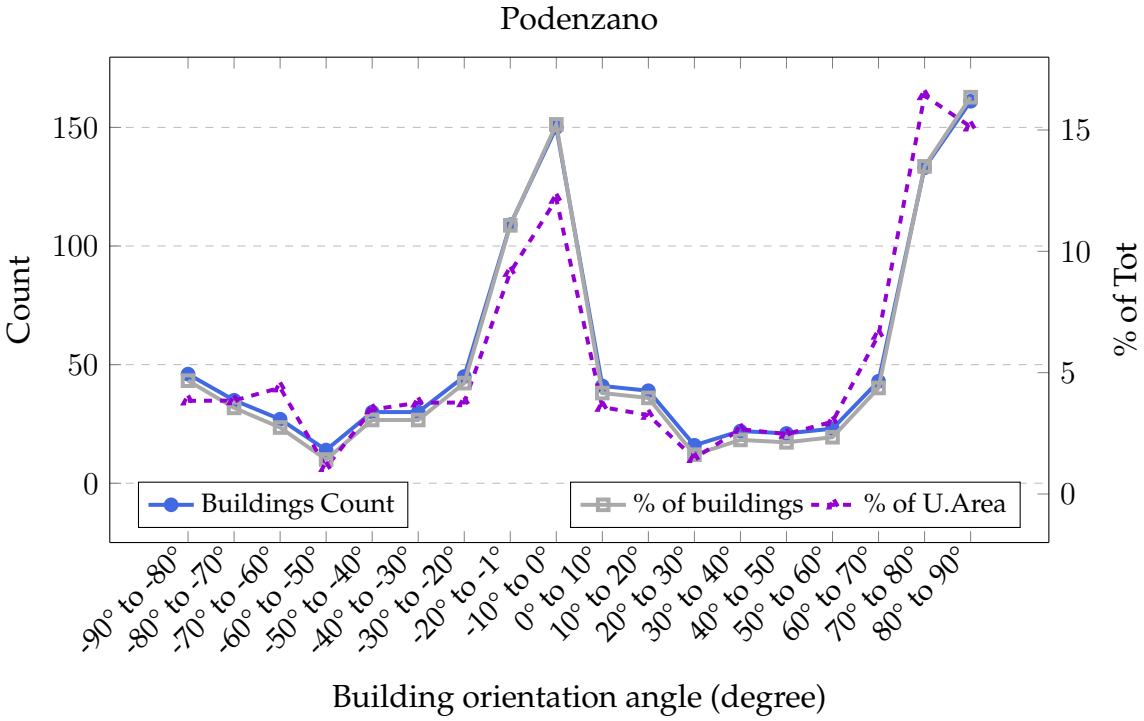


Figure 3.8: *Graph values of table 3.8, distribution of buildings and useful area by orientation of Podenzano.*

3.2.4 Caorso

Caorso, with 369 buildings, has 78.05% residential and 15.99% agricultural buildings (see table 3.9). 6% belong to other categories, and no municipal buildings or schools are located under the primary substation. As we can see from graph 3.10, there is no real predominance of orientation.

Table 3.9: *Buildings type Caorso.*

Type	Count	%
Residential	288	78.05%
Agricultural	59	15.99%
Nurseries	2	0.54%
Sports Areas	1	0.27%
Municipal	0	0.00%
Churches	2	0.54%
Services	1	0.27%
Other	16	4.34%
TOTAL	369	100.00%

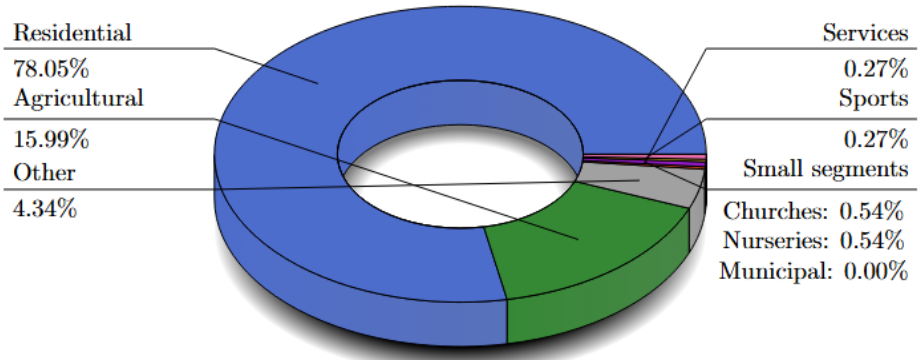


Figure 3.9: *Distribution of building types in Caorso.*

Table 3.10: *Building Distribution by Orientation Angle of Caorso.*

Angle Range	Count	% of Buildings	Useful Area [m^2]	% of Area
-90° to -80°	11	2.98%	939	3.15%
-80° to -70°	22	5.96%	1623	5.44%
-70° to -60°	25	6.78%	1935	6.48%
-60° to -50°	22	5.96%	2029	6.80%
-50° to -40°	11	2.98%	1064	3.56%
-40° to -30°	21	5.69%	1644	5.51%
-30° to -20°	22	5.96%	1687	5.65%
-20° to -10°	31	8.40%	2262	7.58%
-10° to 0°	34	9.21%	2521	8.45%
0° to 10°	30	8.13%	2269	7.60%
10° to 20°	18	4.88%	1292	4.33%
20° to 30°	9	2.44%	677	2.27%
30° to 40°	25	6.78%	1838	6.16%
40° to 50°	11	2.98%	1232	4.13%
50° to 60°	14	3.79%	1827	6.12%
60° to 70°	16	4.34%	1353	4.53%
70° to 80°	28	7.59%	2498	8.37%
80° to 90°	19	5.15%	1159	3.88%
TOTAL	369	100.00%	29849	100.00%

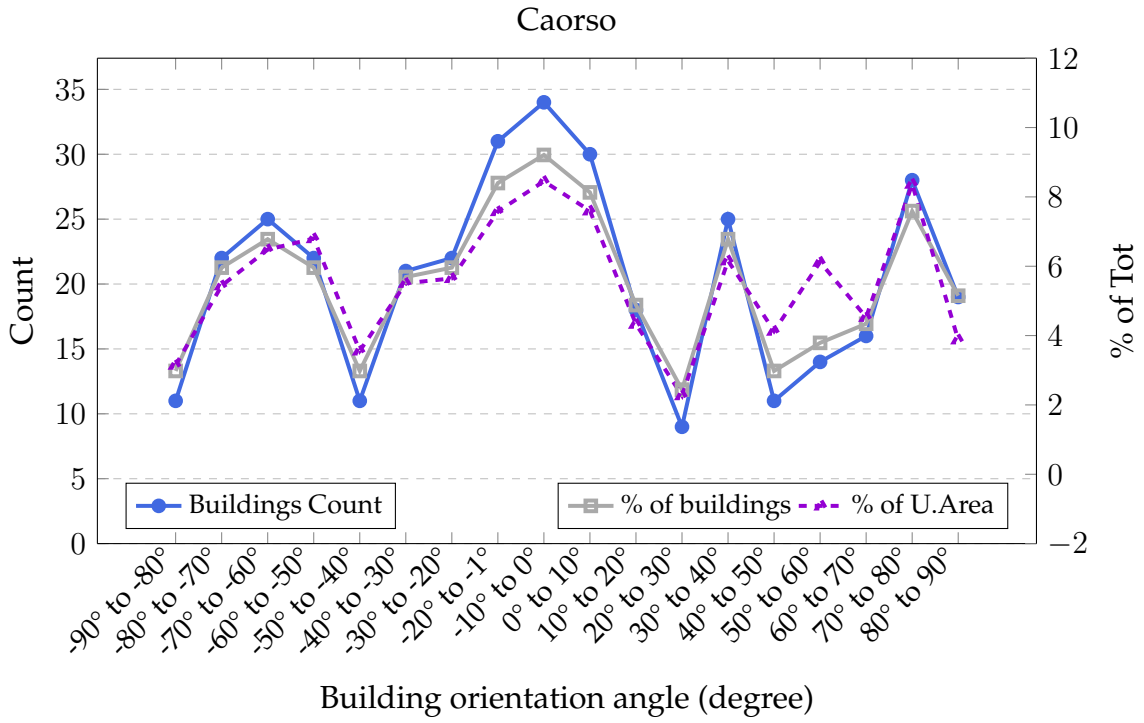


Figure 3.10: *Graph values of table 3.10, distribution of buildings and useful area by orientation of Caorso.*

3.2.5 San Giorgio Piacentino

San Giorgio Piacentino is the municipality with the lowest number of buildings of interest, only 242, and the smallest surface area of 2.75 hectares. 96% is divided between housing and agricultural activities, table 3.11 and graph 3.11. As clearly visible in relation to other municipalities, in figure 3.12, the solid gray line representing the percentage of buildings and the purple dotted line showing the percentage of useful area (U.Area) diverge significantly at several orientation angles. For example, at the 0°-10° orientation, there's a notable peak in building percentage that isn't matched by an equivalent peak in useful area. Conversely, at 70°-80°, the percentage of useful area exceeds the percentage of buildings. These divergences indicate that buildings with certain orientations have disproportionate sizes. When the percentage of useful area exceeds the building percentage (as seen at 10°-20° and 70°-80°), it suggests fewer but larger buildings are oriented in those directions. When building percentage exceeds useful area percentage (as at 0°-10°), it indicates numerous smaller buildings with that

Table 3.11: *Buildings type San Giorgio Piacentino.*

Type	Count	%
Residential	170	70.25%
Agricultural	62	25.62%
Nurseries	0	0.00%
Sports Areas	0	0.00%
Municipal	0	0.00%
Churches	2	0.83%
Services	6	2.48%
Other	2	0.83%
TOTAL	242	100.00%

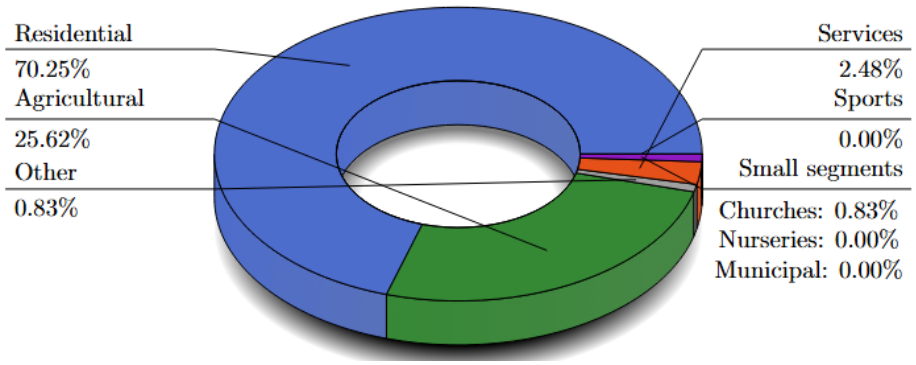


Figure 3.11: *Distribution of building types in San Giorgio Piacentino.*

orientation. This relationship between the two curves provides information into the spatial distribution and dimensional characteristics of buildings across different orientations in San Giorgio Piacentino.

Table 3.12: *Building Distribution by Orientation Angle of San Giorgio Piacentino.*

Angle Range	Count	% of Buildings	Useful Area [m^2]	% of Area
-90° to -80°	11	4.55%	750	2.73%
-80° to -70°	5	2.07%	505	1.84%
-70° to -60°	1	0.41%	105	0.38%
-60° to -50°	5	2.07%	446	1.62%
-50° to -40°	5	2.07%	346	1.26%
-40° to -30°	24	9.92%	2124	7.72%
-30° to -20°	19	7.85%	2008	7.30%
-20° to -10°	35	14.46%	2486	9.04%
-10° to 0°	28	11.57%	2958	10.76%
0° to 10°	15	6.20%	3652	13.28%
10° to 20°	6	2.48%	1134	4.12%
20° to 30°	6	2.48%	425	1.55%
30° to 40°	6	2.48%	573	2.08%
40° to 50°	6	2.48%	254	0.92%
50° to 60°	8	3.31%	898	3.27%
60° to 70°	18	7.44%	3512	12.77%
70° to 80°	31	12.81%	3706	13.48%
80° to 90°	13	5.37%	1618	5.88%
TOTAL	242	100.00%	27500	100.00%

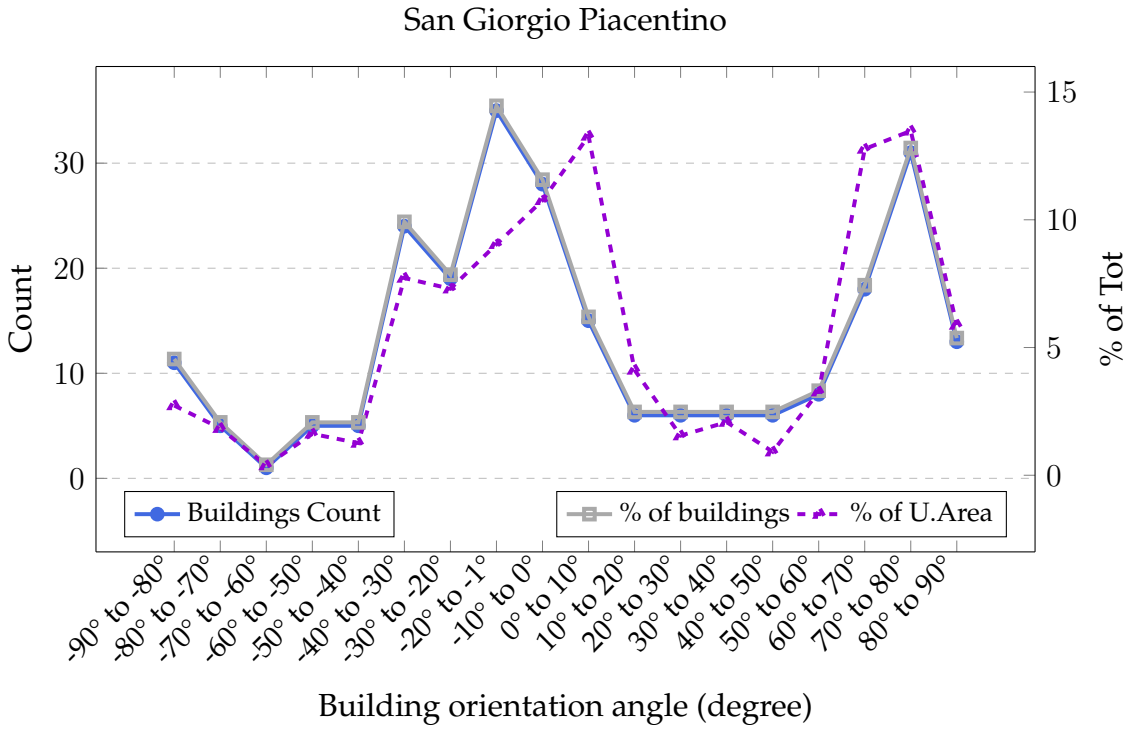


Figure 3.12: *Graph values of table 3.12, distribution of buildings and useful area by orientation of San Giorgio Piacentino.*

3.3 Power plant estimate analysis by municipality

Using the algorithms constructed in the previous chapter, we analyzed the territory assuming three different types of photovoltaic systems. By changing the different parameters of the PV system, we can see how the installable power and the number of PV panels that can be placed on roofs varies. To make it easier to understand, we designate the analyses for each type of panel with the label TY1 and so on, where:

- TY1: indicates panel model SS-410-54MDH of brand Sunova Solar as shown in figure 61;
- TY2: indicates panel model MS510MB-50H of brand Maysun Solar as shown in figure 62;
- TY3: indicates panel model 82S355 of brand Aleo Solar as in figure 63;

By eliminating installations with a power output less than $3.5kW$, since they would be

Table 3.13: *Photovoltaic power by ty of Pontenure by typology.*

Type	TY1 [kW]	TY2 [kW]	T3 [kW]
Residential	17434,84	17652,12	16506,435
Agricultural	4725,25	4786,35	4475,13
Nurseries	254,2	257,04	240,69
Sports Areas	11,89	11,73	11,36
Municipal	350,55	354,96	331,925
Churches	361,62	368,73	342,575
Services	474,78	480,93	448,72
Other	3499,35	3546,03	3316,765
TOTAL	27112,48	27457,89	25673,6

Table 3.14: *Photovoltaic power by ty of Piacenza by typology.*

Type	TY1 [kW]	TY2 [kW]	T3 [kW]
Residential	10990,46	11154,21	10403,275
Agricultural	4868,34	4936,29	4605,06
Nurseries	219,76	221,85	207,675
Sports Areas	383,35	388,62	363,165
Municipal	42,64	43,35	40,47
Churches	120,54	129,03	114,665
Services	1425,57	1442,28	1349
Other	1052,06	1061,31	997,905
TOTAL	19102,72	19376,94	18081,215

installed on small areas such as garages, small houses or warehouses, we obtain the power plant for each municipality from table 3.13 to 3.17.

Looking at the tables, particularly table 3.18 which displays the totals for each municipality across the three panel types and building types, it can be seen that an average of 71MW of photovoltaic systems could potentially be installed throughout the entire primary cabina area. Under current regulations, this would correspond to 71 energy communities. Furthermore, examining table 3.19 reveals the percentages of each building type for each municipality, providing a detailed breakdown of the distribution. The next step is to visualize how much energy these installations could generate annually, and then determine the appropriate number of energy communities needed to meet

Table 3.15: *Photovoltaic power by ty of Podenzano by typology.*

Type	TY1 [kW]	TY2 [kW]	T3 [kW]
Residential	8110,21	8203,35	7678,29500000001
Agricultural	5309,5	5378,97	5025,38
Nurseries	0	0	0
Sports Areas	11,48	11,73	11,005
Municipal	0	0	0
Churches	181,22	182,58	171,465
Services	1256,65	1272,96	1189,96
Other	1227,95	1244,91	1161,56
TOTAL	16097,01	16294,5	15237,665

Table 3.16: *Photovoltaic power by ty of Caorso by typology.*

Type	TY1 [kW]	TY2 [kW]	T3 [kW]
Residential	3058,19	3119,16	2896,445
Agricultural	999,17	1014,39	946,075
Nurseries	74,62	75,48	70,645
Sports Areas	0	0	0
Municipal	0	0	0
Churches	91,84	95,88	86,62
Services	16,81	16,83	15,62
Other	721,6	726,75	683,375
TOTAL	4962,23	5048,49	4698,78

Table 3.17: *Photovoltaic power by ty of San Giorgio Piacentino by typology.*

Type	TY1 [kW]	TY2 [kW]	T3 [kW]
Residential	1947,09	1972,68	1844,225
Agricultural	2083,21	2105,28	1969,895
Nurseries	0	0	0
Sports Areas	0	0	0
Municipal	0	0	0
Churches	64,37	64,77	60,705
Services	79,13	80,58	75,26
Other	433,37	438,6	411,09
TOTAL	4607,17	4661,91	4361,175

Table 3.18: *Photovoltaic power by ty of the Primary Cabin by typology.*

Type	TY1 [kW]	TY2 [kW]	T3 [kW]
Residential	41540,79	42101,52	39328,675
Agricultural	17985,47	18221,28	17021,54
Nurseries	548,58	554,37	519,01
Sports Areas	406,72	412,08	385,53
Municipal	393,19	398,31	372,395
Churches	819,59	840,99	776,03
Services	3252,94	3293,58	3078,56
Other	6934,33	7017,6	6570,695
TOTAL	71881,61	72839,73	68052,435

Table 3.19: *Percentage by type of each municipality.*

Type	Pontenure	Piacenza	Podenzano	Caorso	San G.P.
Residential	0,4197	0,264	0,195	0,073	0,046
Agricultural	0,262	0,270	0,295	0,055	0,115
Nurseries	0,463	0,400	0	0,136	0
Sports Areas	0,029	0,942	0,028	0	0
Municipal	0,891	0,108	0	0	0
Churches	0,441	0,147	0,221	0,112	0,078
Services	0,145	0,438	0,386	0,005	0,024
Other	0,504	0,1518	0,177	0,104	0,062
TOTAL	0,377	0,265	0,223	0,069	0,064

the energy requirements of the buildings under consideration.

3.4 Maximum energy production of the primary cabin

Using the tools developed previously, we calculate the maximum theoretical electricity production that the cabin can generate. By examining the annual graphs for each municipality and building type, we obtain that:

- Pontenure: Total annual production for Ty1=$34,50GWh$ (graph 3.13), Ty2=$34,78GWh$ (graph 3.14), Ty3=$32,59GWh$ (graph 3.15), with production variations shown in

table 3.20;

- Piacenza: Total annual production for Ty1=$28,59GWh$ (graph 3.16), Ty2=$28,60GWh$, Ty3=$26,96GWh$, with production variations shown in table 3.21;
- Podenzano: Total annual production for Ty1=$23,55GWh$ (graph 3.17), Ty2=$23,64GWh$, Ty3=$22,21GWh$, with production variations shown in table 3.22;
- Caorso: Total annual production for Ty1=$7,52GWh$ (graph 3.18), Ty2=$7,51GWh$, Ty3=$7,09GWh$, with production variations shown in table 3.23;
- San Giorgio P.: Total annual production for Ty1=$7,01GWh$ (graph 3.19), Ty2=$7,04GWh$, Ty3=$6,62GWh$, with production variations shown in table 3.24;

For municipalities other than Pontenure, we have included the annual graph only for PV panel type 1, as we specify the total variation for each type in the respective tables. Adding up the annual production estimates of each municipality,the primary cabin could potentially produce a total of $101,17GWh$ for type 1, $101,57GWh$ for type 2 and $95,47GWh$ for the third type of PV panels.

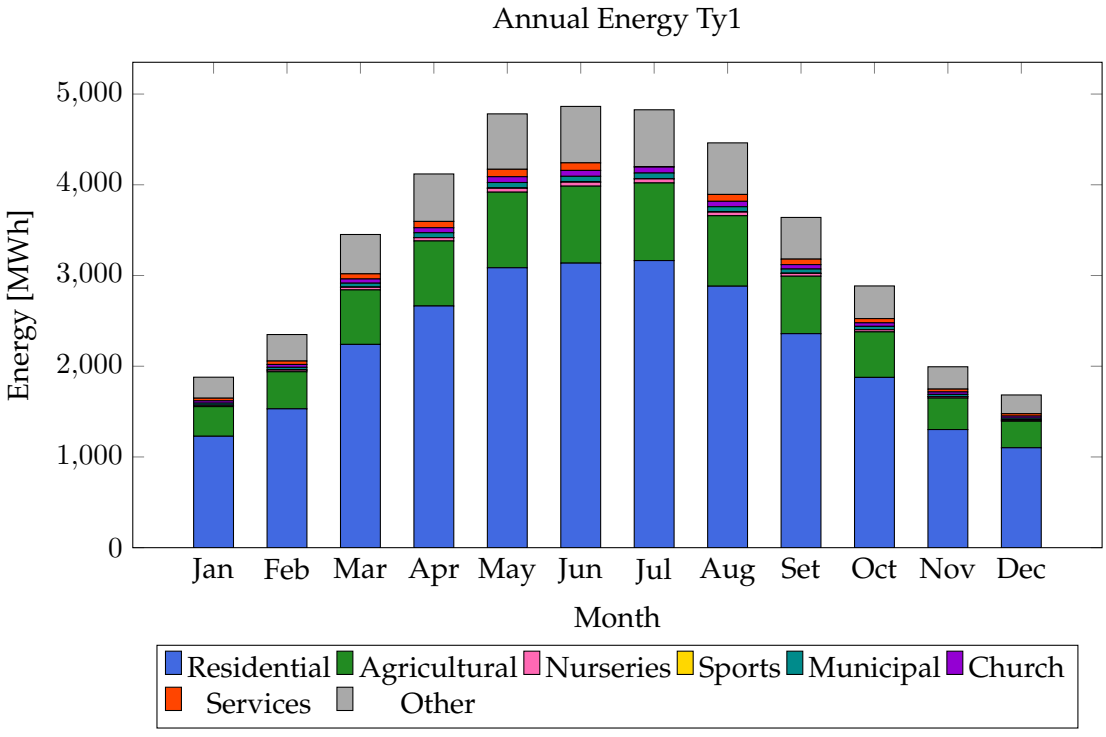


Figure 3.13: *Global monthly production for the ty1 panel category of Pontenure.*

Table 3.20: *Variation in MWh between types of PV panels for the municipality of Pontenure, taking type 1 as reference.*

Ty	Jan	Feb	Mar	Apr	May	Jun	Jul	Aug	Set	Oct	Nov	Dec
Ty1	0	0	0	0	0	0	0	0	0	0	0	0
Ty2	16,07	20,13	29,63	35,43	41,18	41,9	42,26	38,4	31,28	24,73	17,06	-264,39
Ty3	-122,02	-152,55	-224,15	-267,47	-310,48	-315,83	-318,61	-289,72	-236,36	-187,33	-129,48	169,47

Table 3.21: *Variation in MWh between types of PV panels for the municipality of Piacenza, taking type 1 as reference.*

Ty	Jan	Feb	Mar	Apr	May	Jun	Jul	Aug	Set	Oct	Nov	Dec
Ty1	0	0	0	0	0	0	0	0	0	0	0	0
Ty2	13,44	16,83	24,78	29,63	34,46	35,06	35,37	32,12	26,15	20,69	14,27	-266,78
Ty3	-86,66	-108,86	-160,76	-192,77	-224,59	-228,66	-230,59	-209,16	-169,86	-133,93	-92,09	201,29

Table 3.22: *Variation in MWh between types of PV panels for the municipality of Podenzano, taking type 1 as reference.*

Ty	Jan	Feb	Mar	Apr	May	Jun	Jul	Aug	Set	Oct	Nov	Dec
Ty1	0	0	0	0	0	0	0	0	0	0	0	0
Ty2	8,25	10,52	15,8	19,26	22,68	23,16	23,34	21,01	16,81	13,02	8,8	-95,12
Ty3	-67,03	-85,08	-127,08	-154,16	-180,91	-184,66	-186,08	-167,91	-134,95	-105,13	-71,44	42,69

Table 3.23: *Variation in MWh between types of PV panels for the municipality of Caorso, taking type 1 as reference.*

Ty	Jan	Feb	Mar	Apr	May	Jun	Jul	Aug	Set	Oct	Nov	Dec
Ty1	0	0	0	0	0	0	0	0	0	0	0	0
Ty2	4,68	5,82	8,5	10,08	11,67	11,87	11,98	10,9	8,94	7,13	4,96	-104,09
Ty3	-24,18	-30,18	-44,23	-52,64	-61,1	-62,15	-62,69	-57	-46,58	-37,03	-25,65	86,62

Table 3.24: *Variation in MWh between types of PV panels for the municipality of San Giorgio Piacentino, taking type 1 as reference.*

Ty	Jan	Feb	Mar	Apr	May	Jun	Jul	Aug	Set	Oct	Nov	Dec
Ty1	0	0	0	0	0	0	0	0	0	0	0	0
Ty2	2,4	3,02	4,47	5,37	6,27	6,39	6,44	5,83	4,73	3,72	2,55	-19,68
Ty3	-20,75	-25,89	-37,93	-45,12	-52,43	-53,39	-53,84	-48,88	-39,93	-31,76	-22,01	3,23

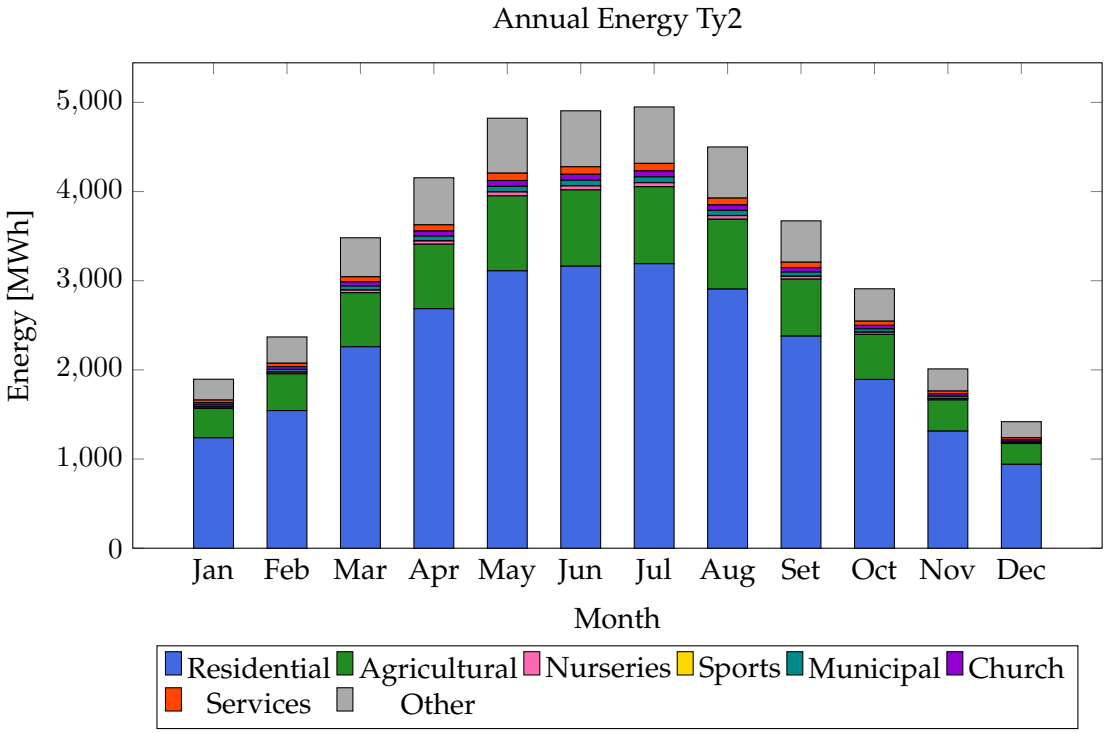


Figure 3.14: *Global monthly production for the ty2 panel category of Pontenure.*

Below we examine the energy production curves of the municipality of Podenzano on the 15th day of each month, as it is the only municipality entirely under the primary cabin. For all other municipalities, we will analyze only the data from July 15th, which represents the month with the longest daylight hours. This approach allows us to observe how the distribution of photovoltaic installations affects the production curve under clear sky conditions. We have chosen to present the complete twelve-month analysis for Podenzano exclusively, as extending this detailed temporal analysis to all municipalities would result in an excessive number of graphs, given the substantial volume of data and information that this methodology can generate. In figure 3.20 through 3.31 (which may appear at different locations throughout the document due to LaTeX's floating figure placement), we can observe how the production curve varies during clear days and consequently how it changes throughout the year.

As regards the variation of production as a function of the distribution of buildings, for Pontenure's production curve, figure 3.32, we note a smooth profile with a sharp morning rise beginning around 6-7 AM, reaching a well-defined peak around 13-14 hours, followed by a gradual afternoon decline until approximately 20 hours. This pattern

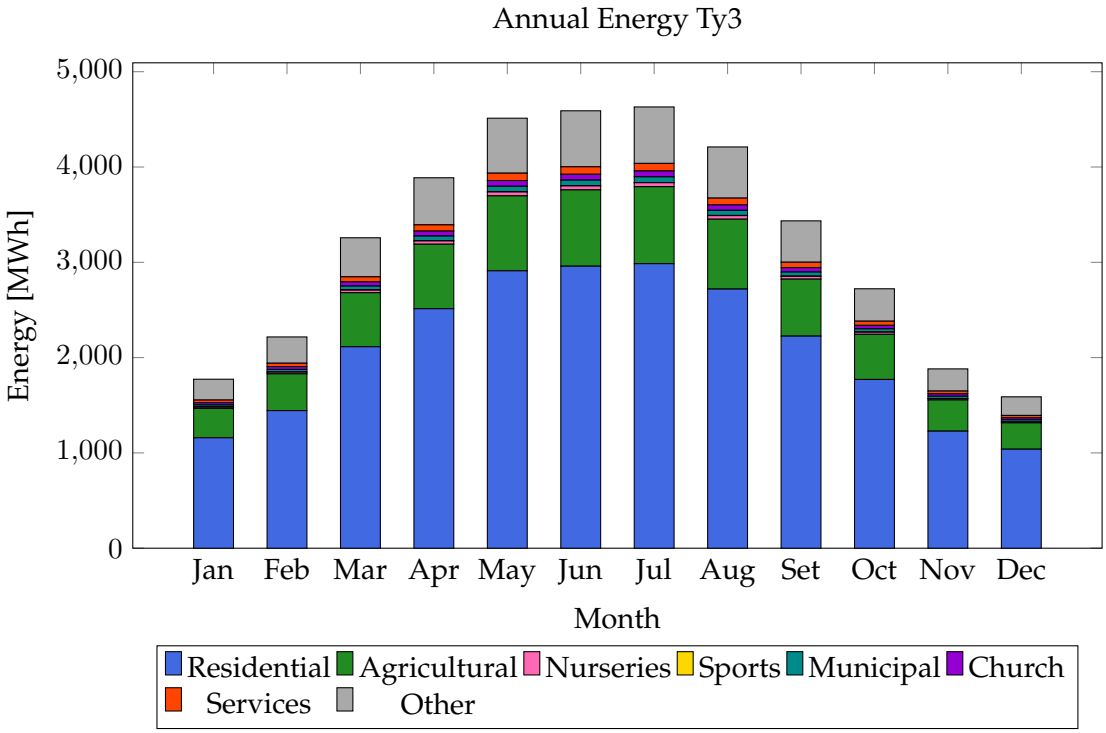


Figure 3.15: *Global monthly production for the ty3 panel category of Pontenure.*

corresponds directly with its building orientation distribution, figure 3.4, which shows two major concentrations: one at approximately -20° (southwest-facing) and another significant cluster at around 50° (southeast). The substantial number of south-facing installations explains the pronounced midday peak production, while the southwest-oriented buildings contribute to sustained afternoon generation. Caorso presents a more balanced production curve, figure 3.33 with a smoother progression throughout daylight hours. This aligns with its more distributed building orientation pattern 3.10, showing multiple peaks across the angular spectrum from -90° to 90°. The notable clusters at approximately -45° (northwest), 0° (north/south), and 45° (northeast) create a diverse array of photovoltaic exposures that collectively generate a more uniform daily production profile. Podenzano exhibits the highest overall production (figure 3.34) with a characteristic plateau in the morning hours (8-10 AM) before rising to its peak around 13-14 hours. This distinctive shape correlates directly with its building orientation distribution, (figure 3.8), which shows two pronounced peaks: one around 70° (east) and another very significant concentration at approximately 0° (south). The high percentage of east-facing installations contributes to the strong morning perfor-

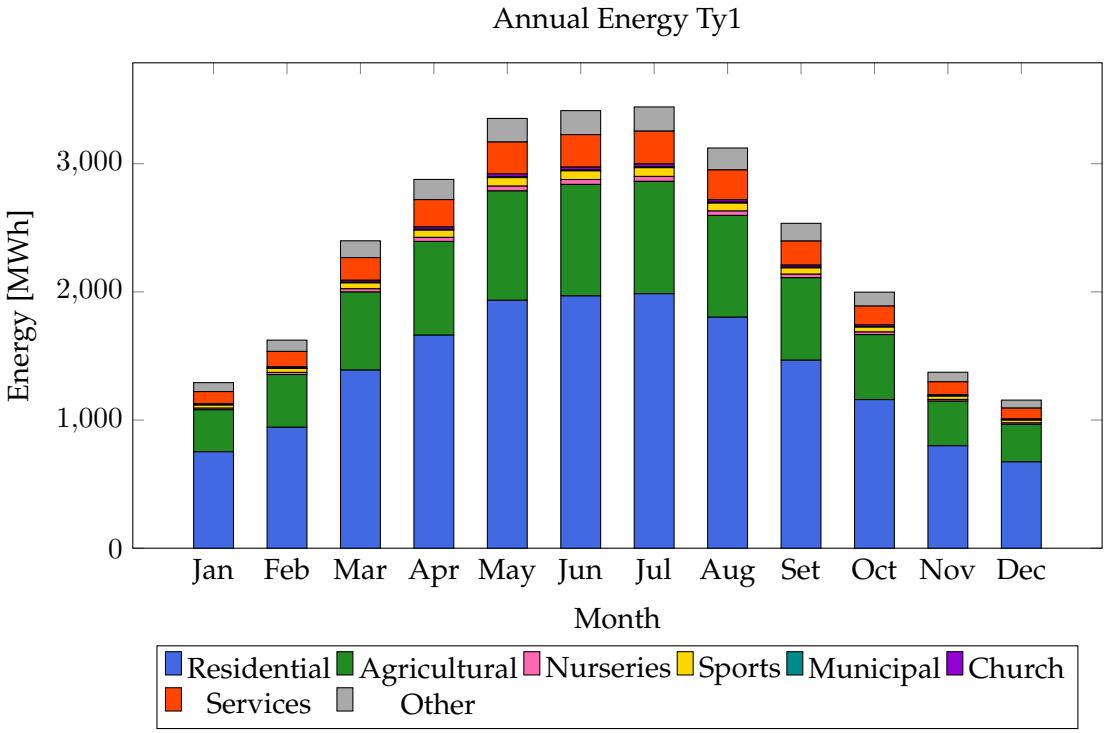


Figure 3.16: *Global monthly production for the ty1 panel category of Piacenza.*

mance, while the south-facing buildings optimize midday production, creating the observed plateau effect and high overall generation capacity. These relationships between building orientation angles and production curve characteristics demonstrate how the spatial distribution and angular positioning of photovoltaic installations significantly influence the temporal generation patterns at the municipal level.

A detailed report of the results obtained is shown in table 3.25, where for each municipality, and for each orientation, the usable area, the installed capacity of the systems, and the annual energy production are listed. This demonstrates the versatility of the method for creating charts and tables according to specific requirements.

3.5 Consumption

In this section we analyze the estimated electricity consumption patterns of municipalities served by primary cabin AC001E01166. Using the algorithm 3 detailed in section 2.3.3, we estimated the number of dwellings and PODs (Points of Delivery) for each residential building type, while for non-residential, we derive consumption based

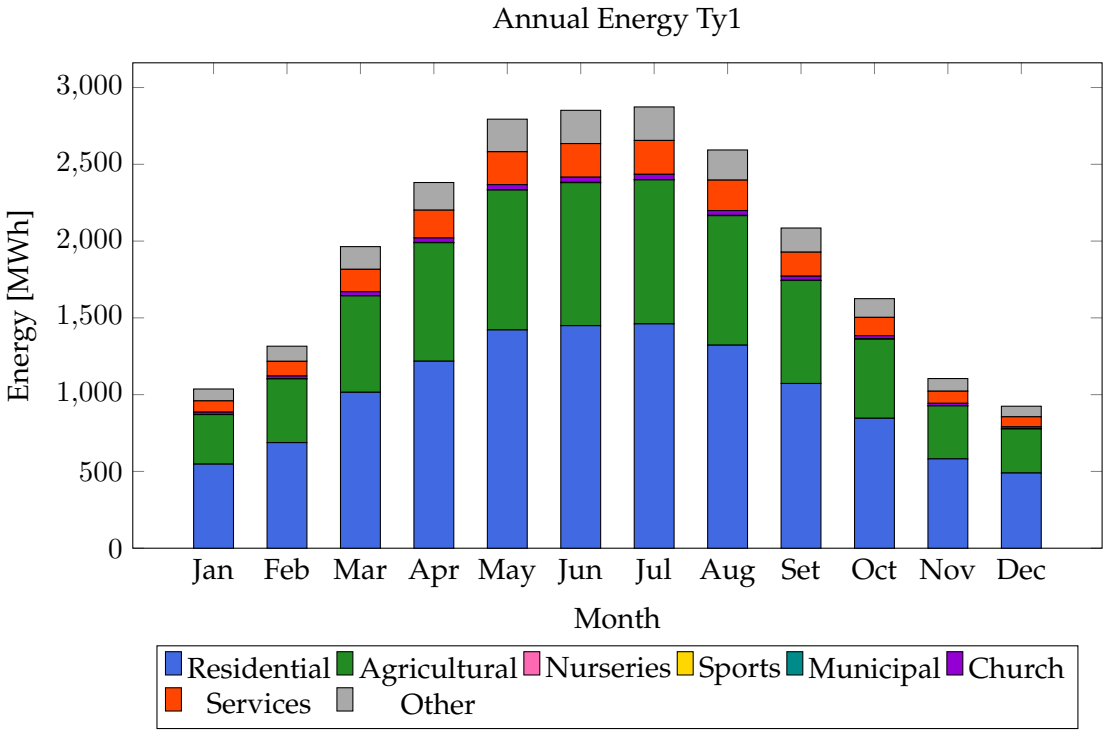


Figure 3.17: *Global monthly production for the ty1 panel category of Podenzano.*

on average building area surface. We begin by examining the monthly consumption profiles of each municipality. It should be noted that the annual consumption graph across municipalities exhibits similar patterns due to limited availability of consumption data. This similarity originated from our reliance on the Arera database[18], which provides average monthly provincial consumption profiles. Our comprehensive analysis reveals the following annual consumption estimates for each municipality:

- Pontenure, which has the largest number of cabin buildings, shows the highest

Table 3.25: *Summary of useful area, peak power, and annual energy production by municipality and orientation (South, East, West) for TY1 photovoltaic installations.*

	Useful Area $[m^2]$			**Peck Power**$[kW_p]$			**Energy** $[MWh/y]$		
Municipally	**South**	**East**	**West**	**South**	**East**	**West**	**South**	**East**	**West**
Pontenure	49951	71846	38023	8476,74	12193,39	6442,32	14043,39	17333,38	9639,77
Piacenza	36020	44160	32618	6089,31	7480,03	5533,35	10123,39	10407,6	8056,59
Podenzano	31276	44114	19331	5306,21	7504,23	3286,56	8870,24	10028,32	4651,69
Caorso	10434	9727	9118	1767,92	1648,61	1545,7	2949,78	2350,37	2219,74
San Giorgio	12462	10444	4176	2120,52	1778,17	708,48	3543,21	2423,26	1048,91
TOTAL	**140143**	**180291**	**103266**	**23760,72**	**30604,44**	**17516,42**	**39530,03**	**42542,96**	**25616,71**

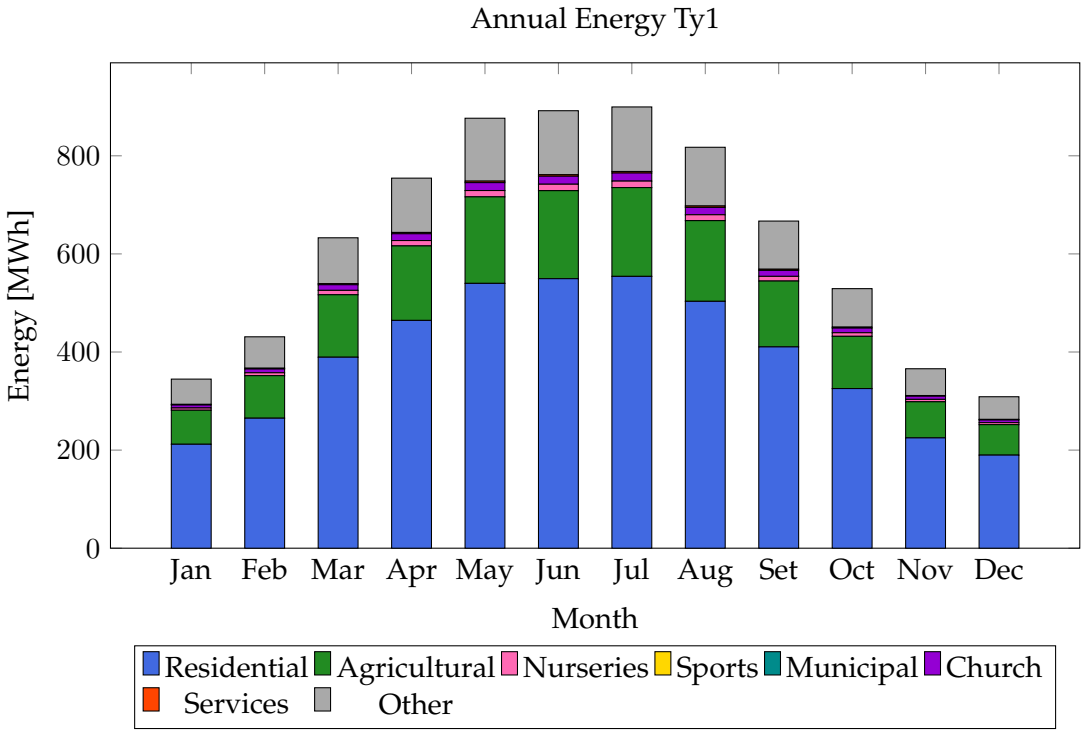


Figure 3.18: *Global monthly production for the ty1 panel category of Caorso.*

consumption at $3,04GWh$ annually (figure 3.35);

- Piacenza, the second largest municipality in our study area, consumed approximately $2,30GWh$ annually (figure 3.36);
- Podenzano, account for $1,74GWh$ of annual consumption (figure 3.37);
- Caorso demonstrates a significantly lower consumption profile at $520MWh$ annually (figure 3.38);
- San Giorgio P, the smallest municipality in our analysis, draws approximately $482MWh$ from the network annually (figure 3.39).

When comparing these consumption data with our maximum production estimates, it become immediately apparent that potential production capacity significantly exceeds current consumption levels across all municipalities. To illustrate temporal consumption patterns in greater detail, we present Pontenure's daily consumption profiles on the 15th day of each month in Figures 3.40 through 3.51. These profiles clearly demon-

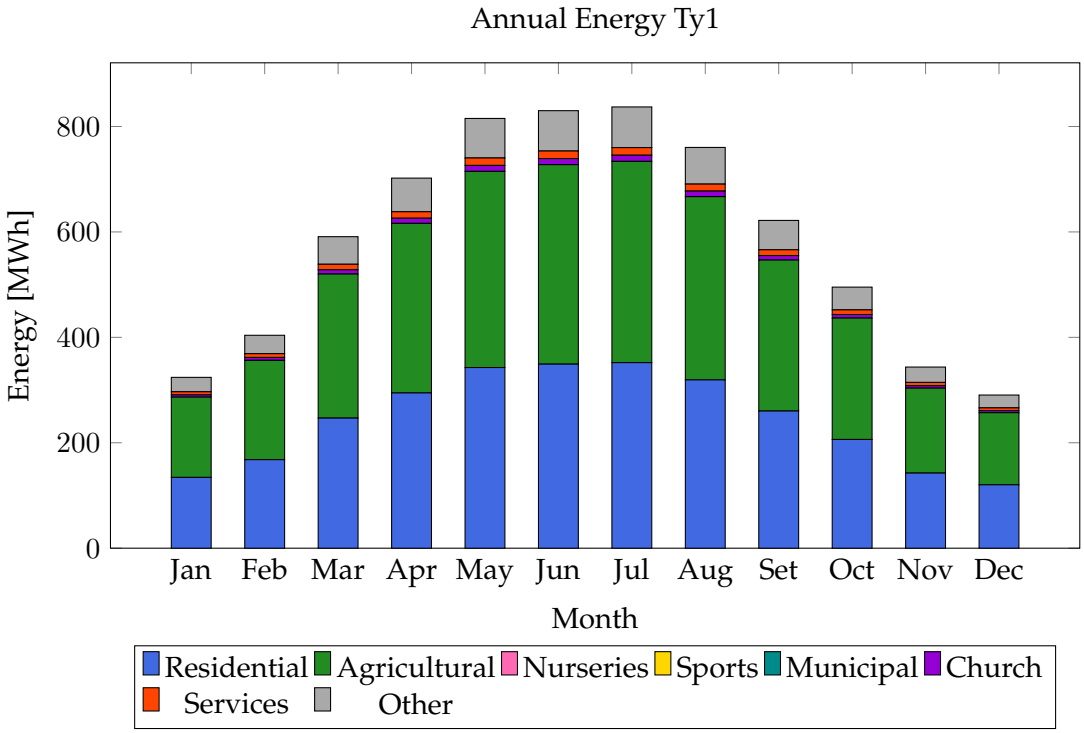


Figure 3.19: *Global monthly production for the ty1 panel category of San Giorgio Piacentino.*

strate seasonal variations in energy demand throughout the year, with notable differences between winter and summer months.

3.6 Estimate of renewable energy communities

After constructing the general production and consumption profiles for each municipality or area to be analyzed with this methodology, we can now draw conclusions and estimate the number or size of the energy community needed to meet the needs of the analyzed territory. Starting from the consumption profile, we will determine the photovoltaic installations necessary to satisfy either a specific category of buildings or all categories combined. Various strategies can be employed, depending on several factors. We must consider whether energy storage systems are incorporated into the design, as these would significantly affect the community's ability to utilize produced energy outside peak production hours. Another important consideration is whether the energy community should entirely self-consume all energy produced during a clear

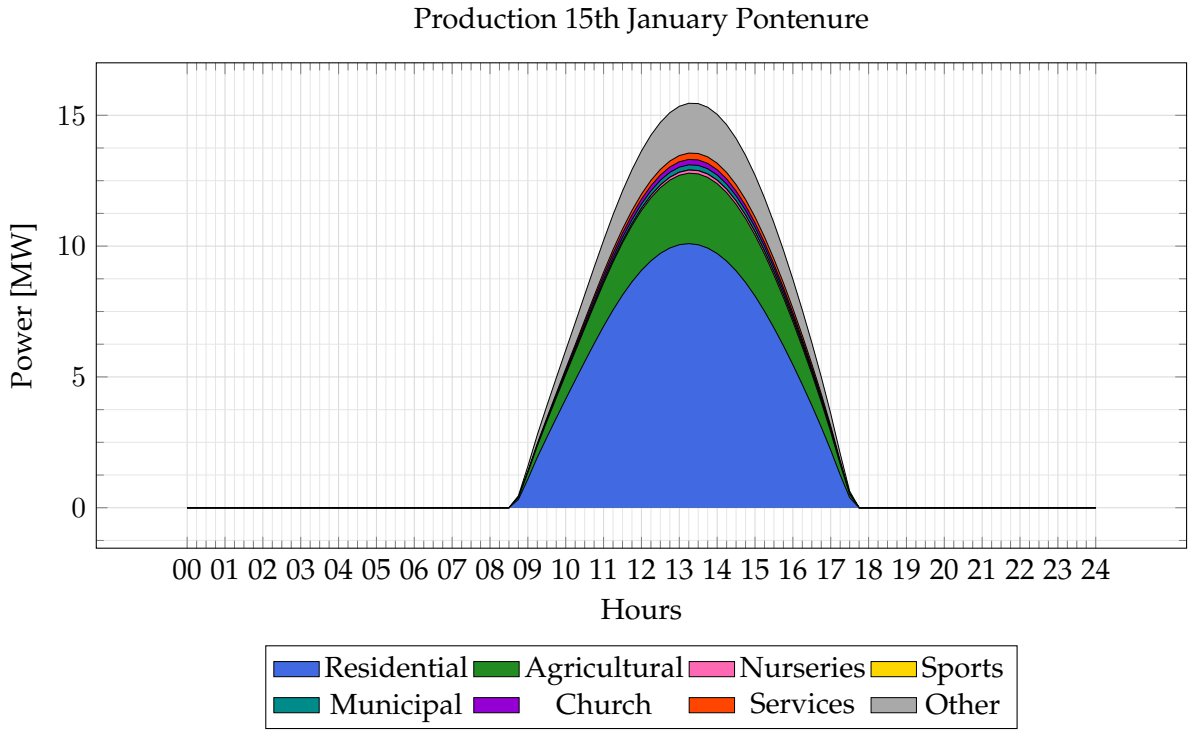


Figure 3.20: *Energy production of the municipality of Pontenure calculated for January 15th using the true solar hour.*

day, which would require careful sizing of the installation to match daily consumption patterns. We also need to determine if the goal is to maximize self-consumption during equinoxes, even if this approach results in production exceeding demand during certain periods, potentially leading to selling excess energy to third parties at disadvantageous prices. Additionally, there are other analytical criteria that may influence the strategy selection based on specific local conditions, regulatory frameworks, and community preferences. Having a comprehensive overview of consumption patterns and the maximum theoretical power production capacity allows us to size the energy community according to different criteria, using the valid analytical tools we have developed. For simplicity, and in line with the primary objective of energy communities, we will focus on sizing one or more communities in the territory to maximize self-consumption while avoiding selling energy to third parties.

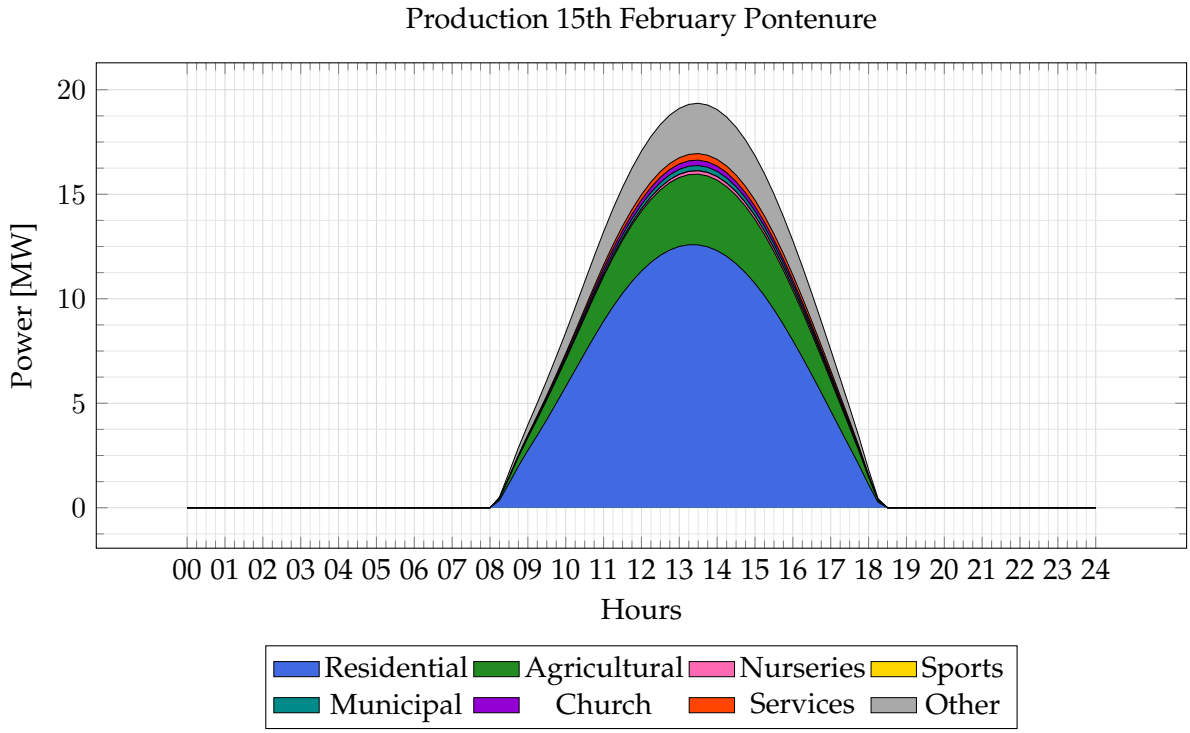


Figure 3.21: *Energy production of the municipality of Pontenure calculated for February 15th using the true solar hour.*

3.6.1 Self-consumption without storage

To determine the necessary installed power capacity that would allow the energy community to completely self-consume the produced energy, it is essential to observe the consumption curves and identify the power demand during peak production hours in the month of maximum production. By analyzing the production graph, we need to find the power demand between 1 PM to 2 PM during the month of July. Based on this analysis, it appears that by globally self-consuming energy during the peak solar power extraction period, a single energy community would be sufficient for the entire primary cabin to meet the energy needs of the buildings under consideration, assuming this hypothetical consumption curve. The annual energy balance is visually represented in graphs 3.52 through 3.56, which illustrate each municipality's energy profile, while graph 3.57 shows the comprehensive data for the primary cabin. In these visualizations, yellow portions indicate energy absorbed from the grid, while green portions represent self-consumed energy generated by the photovoltaic installations. Additionally, in graphs 3.58, 3.59, and 3.60, we can observe the daily power curves

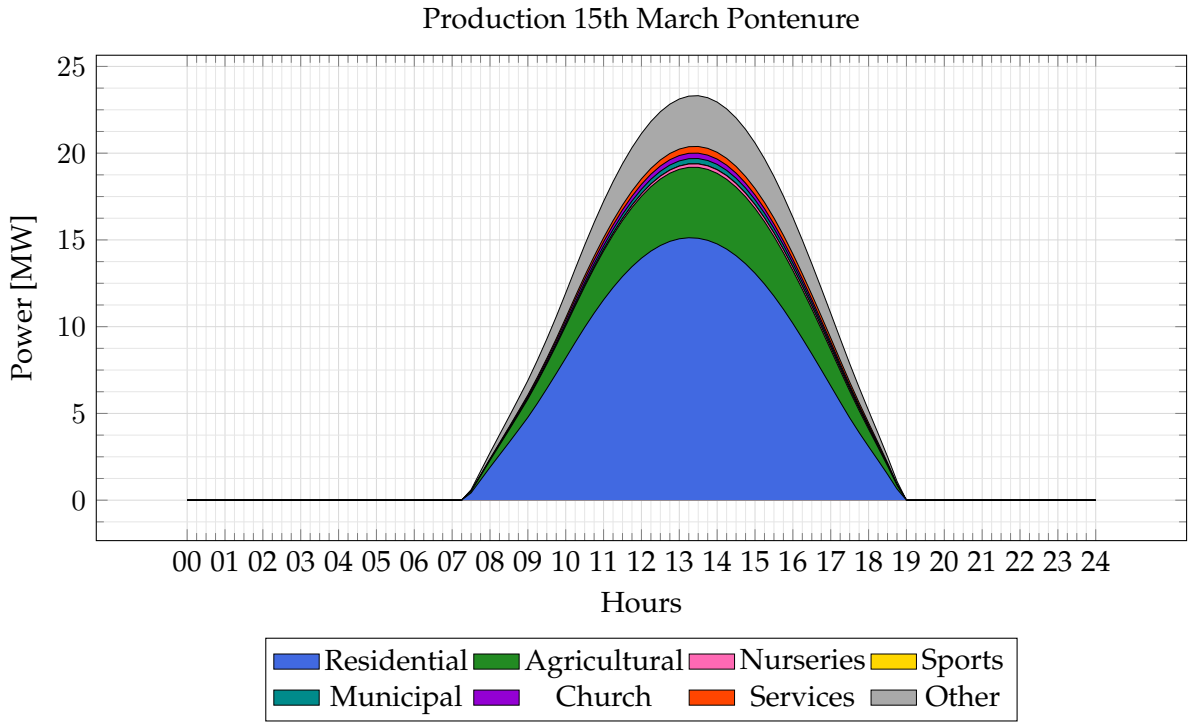


Figure 3.22: *Energy production of the municipality of Pontenure calculated for March 15th using the true solar hour.*

for the municipality of Pontenure on the 15th day of January, April, and July respectively, providing a detailed view of seasonal variations in energy demand patterns. These graphical representations provide a clear picture of the potential impact of the proposed energy community across different seasons.

3.7 Results RECs Primary Cabin AC001E01166

As a summary and conclusion of the case study, the analysis of a community of selected buildings served by the primary cabin AC001E01166 shows that by installing photovoltaic systems with power capacity for each municipality as shown in table 3.26, total self-consumption of energy produced by a potential community is achieved with an annual overall self-consumption of the primary cabin of $1,675GWh$, reducing energy absorbed from the grid by 20% from $8,083GWh$ to $6,408GWh$, with a small portion sold to the grid of $6,464MWh$, as shown in figure 3.57. Considering the obtained consumption curves to be correct, table 3.32 shows the hypothetical percentages of self-

Table 3.26: *Powers requested by municipalities during 1 PM on July 15th.*

Municipally	Power [kW]
Pontenure	423,75
Piacenza	315,90
Podenzano	239,73
Caorso	72,13
San Giorgio P.	67,25
TOTAL	1118,76

Table 3.27: *Three scenarios to satisfy Pontenure's self-consumption, as a percentage of the maximum power available on July 15th at 1 PM.*

Edi Ty	Scenario 1	Scenario 2	Scenario 3
Residential	0,413%	2,427%	0%
Agricultural	0%	0%	9,095%
Municipal	100%	0%	0%

Table 3.28: *Three scenarios to satisfy Piacenza's self-consumption, as a percentage of the maximum power available on July 15th at 1 PM.*

Edi Ty	Scenario 1	Scenario 2	Scenario 3
Residential	2,519%	2,924%	0%
Agricultural	0%	0%	6,639%
Municipal	100%	0%	0%

Table 3.29: *Three scenarios to satisfy Podenzano's self-consumption, as a percentage of the maximum power available on July 15th at 1 PM.*

Edi Ty	Scenario 1	Scenario 2	Scenario 3
Residential	0,486%	2,993%	0%
Agricultural	0%	0%	4,641%
Sports	100%	0%	0%
Church	100%	0%	0%

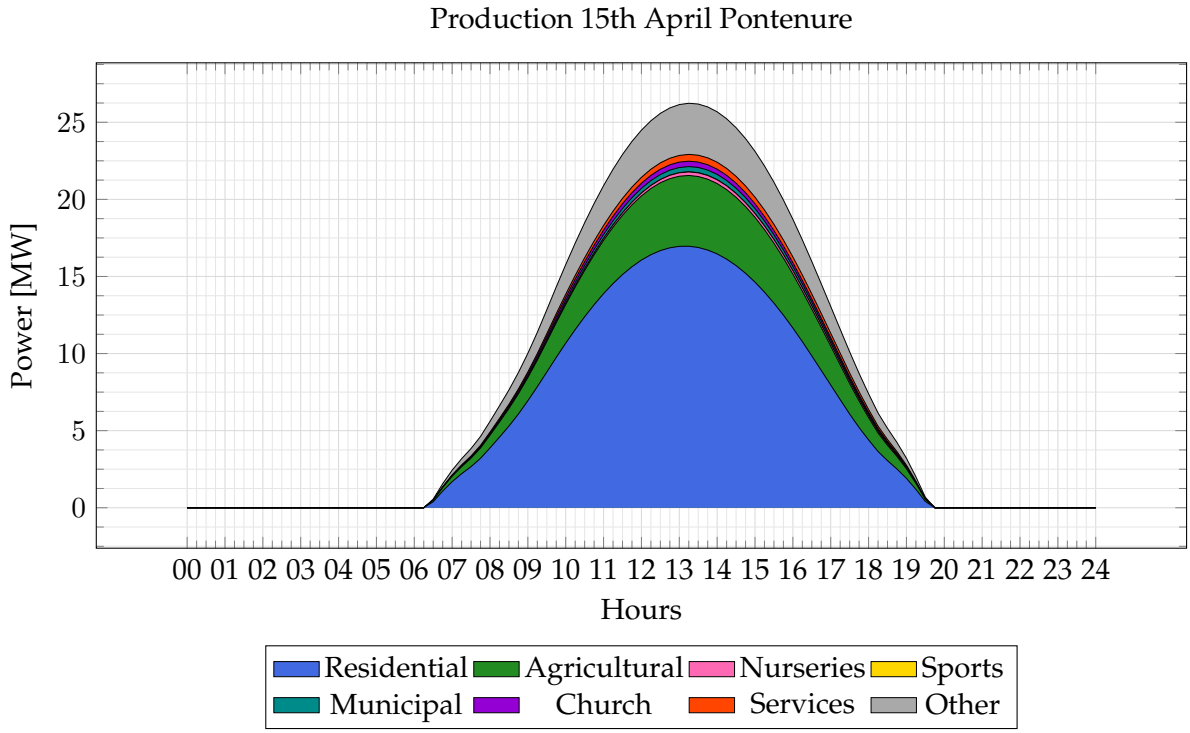


Figure 3.23: *Energy production of the municipality of Pontenure calculated for April 15th using the true solar hour.*

consumption, when sizing the energy community according to previous estimates, and it results that in this way each municipality saves approximately 20% of annual energy, reaching savings of up to 30% in some months. As can be seen in figure 3.60, we have correctly sized the Pontenure energy community so that the maximum peak covers the maximum demand for the month of July, however, as can be seen in figure 3.59, in some months it is possible that energy will be provided to the grid.
The methodology we've developed has successfully identified the power requirements for the energy community by employing one of several possible sizing strategies. Specif-

Table 3.30: *Three scenarios to satisfy Caorso's self-consumption, as a percentage of the maximum power available on July 15th at 1 PM.*

Edi Ty	Scenario 1	Scenario 2	Scenario 3
Residential	0,151%	2,389%	0%
Agricultural	0%	0%	7,141%
Nursery	100%	0%	0%

Table 3.31: *Three scenarios to satisfy San Giorgio P.'s self-consumption, as a percentage of the maximum power available on July 15th at 1 PM.*

Edi Ty	Scenario 1	Scenario 2
Residential	3,475%	0%
Agricultural	0%	3,163%

Table 3.32: *Monthly percentages per municipality of self-consumed energy compared to the energy required.*

Month	Pontenure	Piacenza	Podenzano	Caorso	San Giorgio P.
January	8.75%	8.749%	8.828%	9.084%	9.046%
February	14.049%	14.061%	14.225%	14.538%	14.557%
March	20.522%	20.483%	20.777%	21.101%	21.206%
April	28.841%	28.732%	29.059%	29.308%	29.382%
May	33.996%	33.96%	34.181%	34.373%	34.265%
June	29.828%	29.842%	30.119%	30.475%	30.324%
July	24.122%	24.148%	24.414%	24.654%	24.645%
August	26.026%	25.851%	25.699%	26.355%	25.718%
September	27.623%	27.612%	27.432%	28.107%	27.655%
October	22.515%	22.754%	22.817%	23.3%	23.355%
November	12.421%	12.517%	12.6%	12.918%	12.946%
December	8.507%	8.501%	8.554%	8.832%	8.76%
TOTAL	20.657%	20.605%	19.606%	21.128%	21.048%

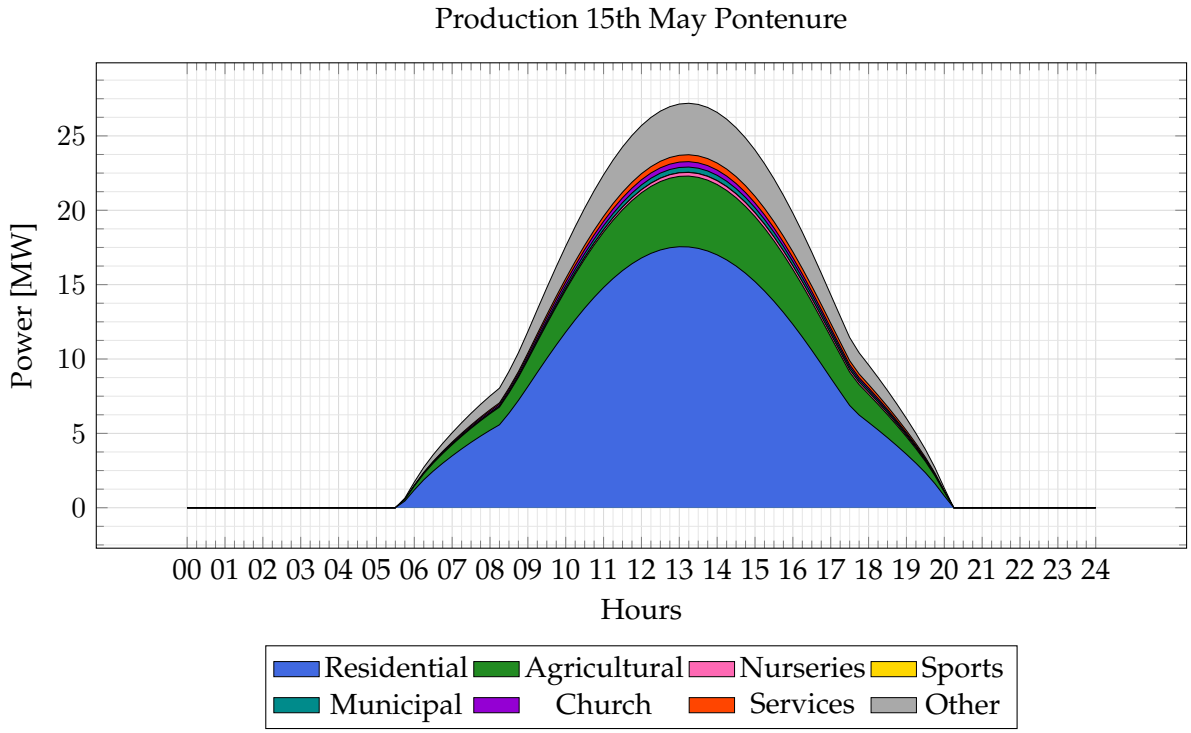


Figure 3.24: *Energy production of the municipality of Pontenure calculated for May 15th using the true solar hour.*

ically, we chose to minimize grid injection during peak solar production periods. By analyzing both the characteristic consumption curves and the territory's solar production potential, we were able to calculate with precision the self-consumed energy and grid-exported energy at 15-minute intervals throughout the day. This granular approach, based on matching detailed production and consumption curves, provides an accurate picture of energy flows and enables optimal sizing of renewable installations to maximize local benefits.

3.8 Summary

In this chapter, we have presented a wealth of data derived from the analysis methodology described in Chapter Two. The primary focus here is on the vast amount of information generated for a specific territorial area, which can serve as valuable tools in optimally sizing renewable energy communities. By analyzing energy production and consumption at fifteen-minute intervals, the method led to the reduction of energy

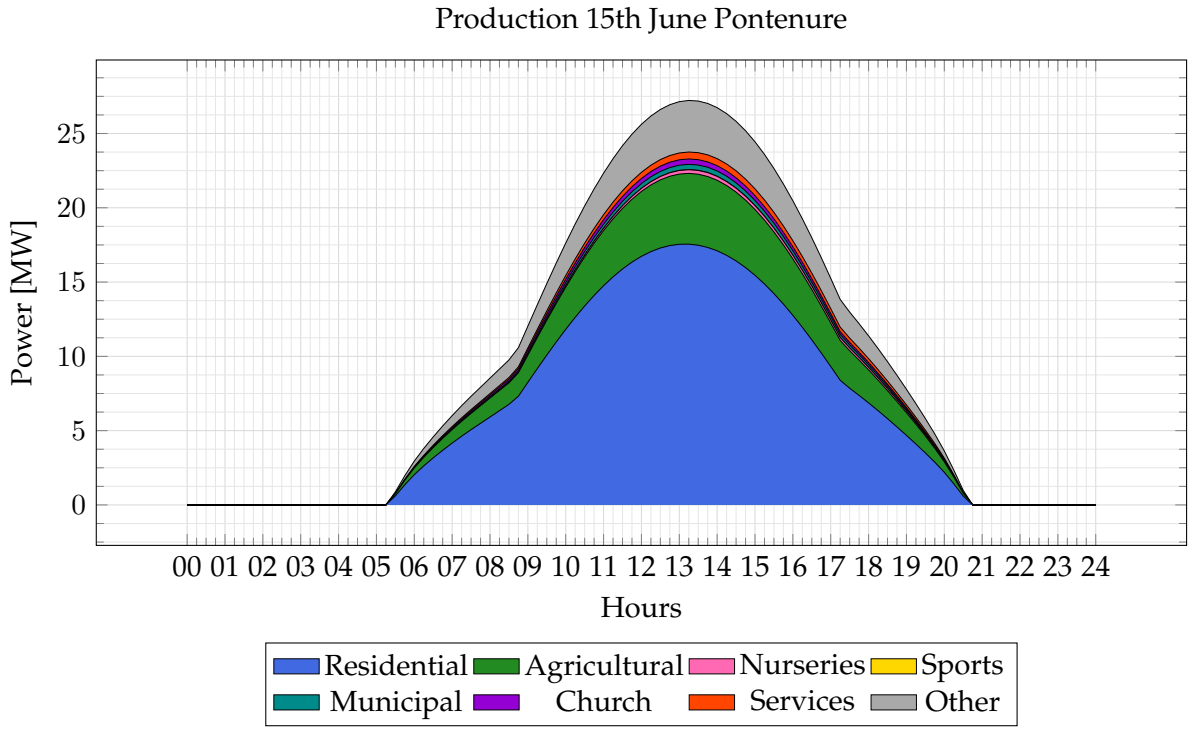


Figure 3.25: *Energy production of the municipality of Pontenure calculated for June 15th using the true solar hour.*

fed into the grid by promoting self-consumption within the community. This approach not only enhances energy efficiency but also opens up numerous avenues and methodologies for creating communities that are as efficient as possible.

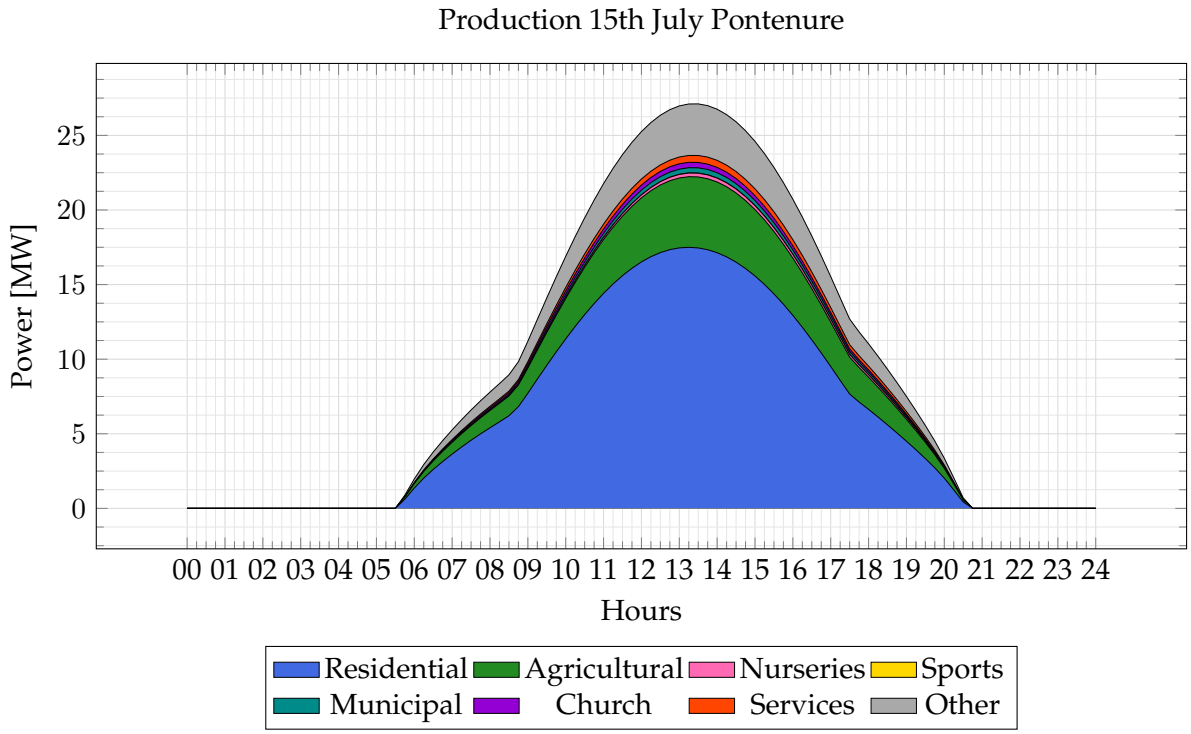


Figure 3.26: *Energy production of the municipality of Pontenure calculated for July 15th using the true solar hour.*

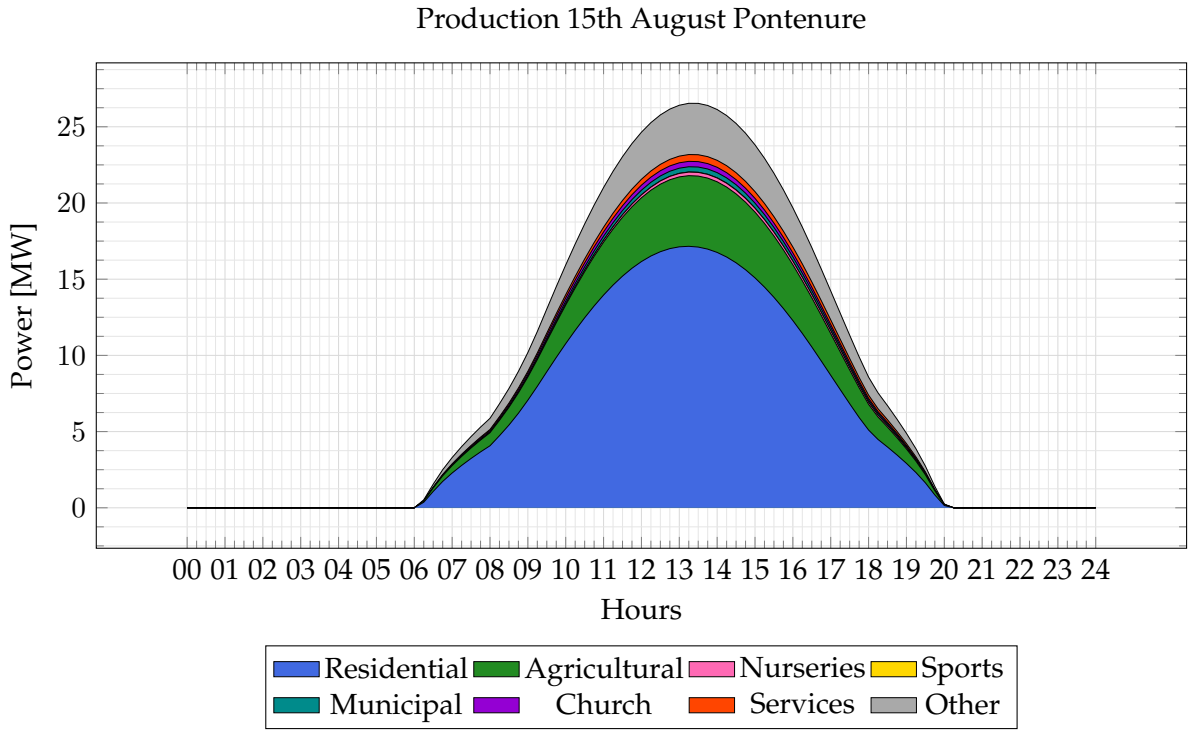


Figure 3.27: *Energy production of the municipality of Pontenure calculated for August 15th using the true solar hour.*

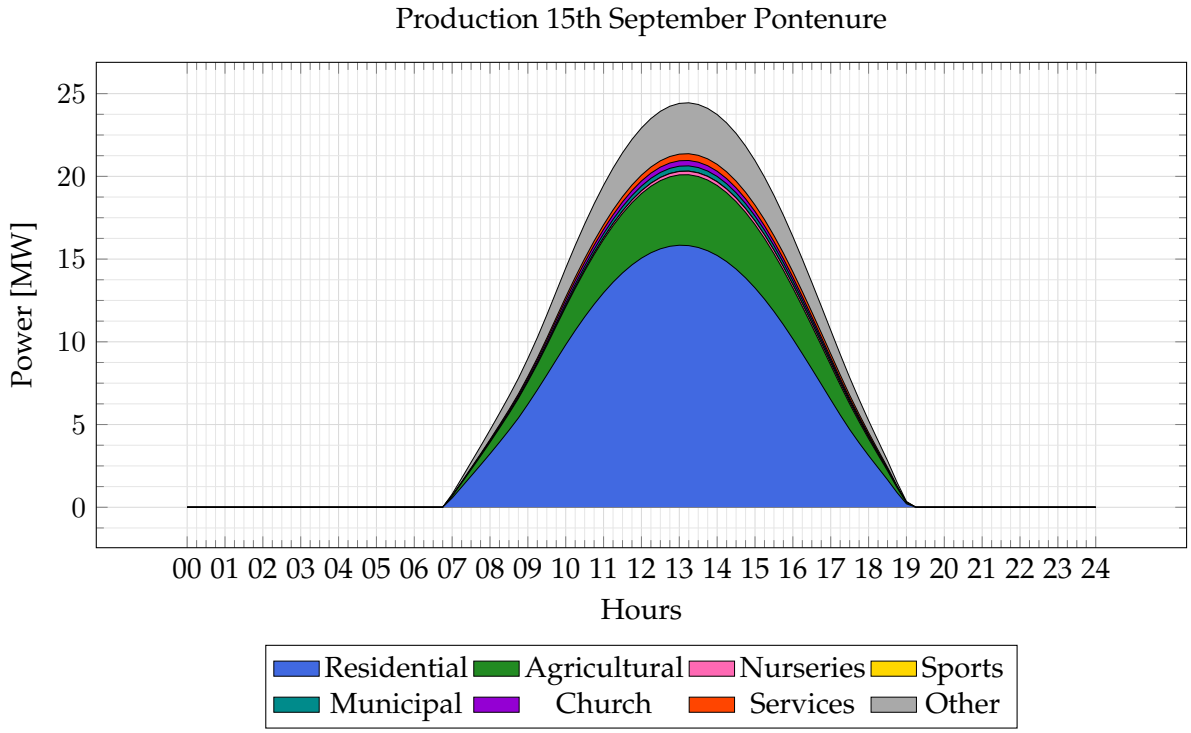


Figure 3.28: *Energy production of the municipality of Pontenure calculated for September 15th using the true solar hour.*

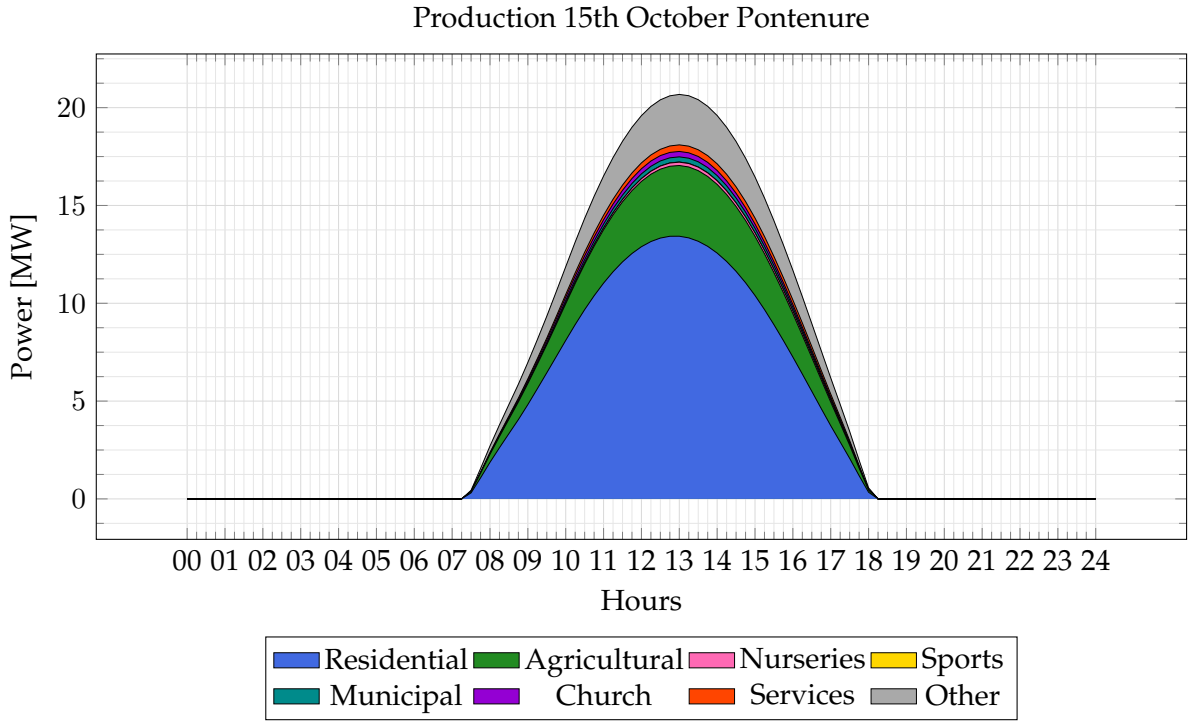


Figure 3.29: *Energy production of the municipality of Pontenure calculated for October 15th using the true solar hour.*

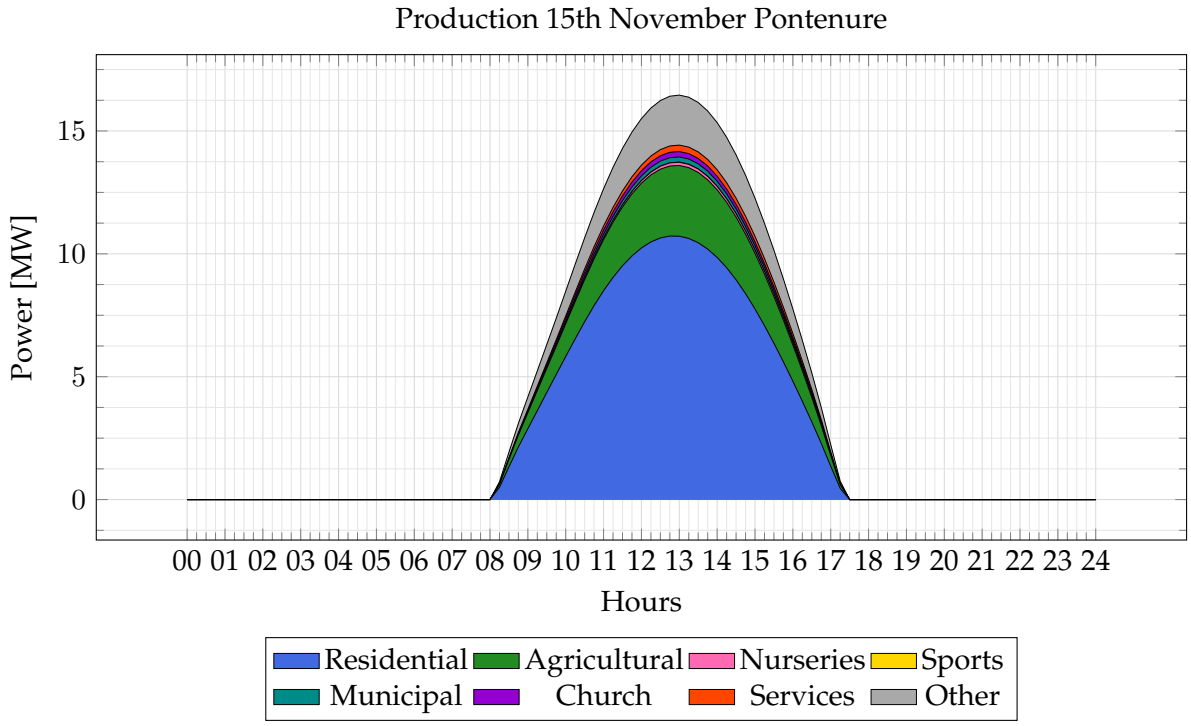


Figure 3.30: *Energy production of the municipality of Pontenure calculated for November 15th using the true solar hour.*

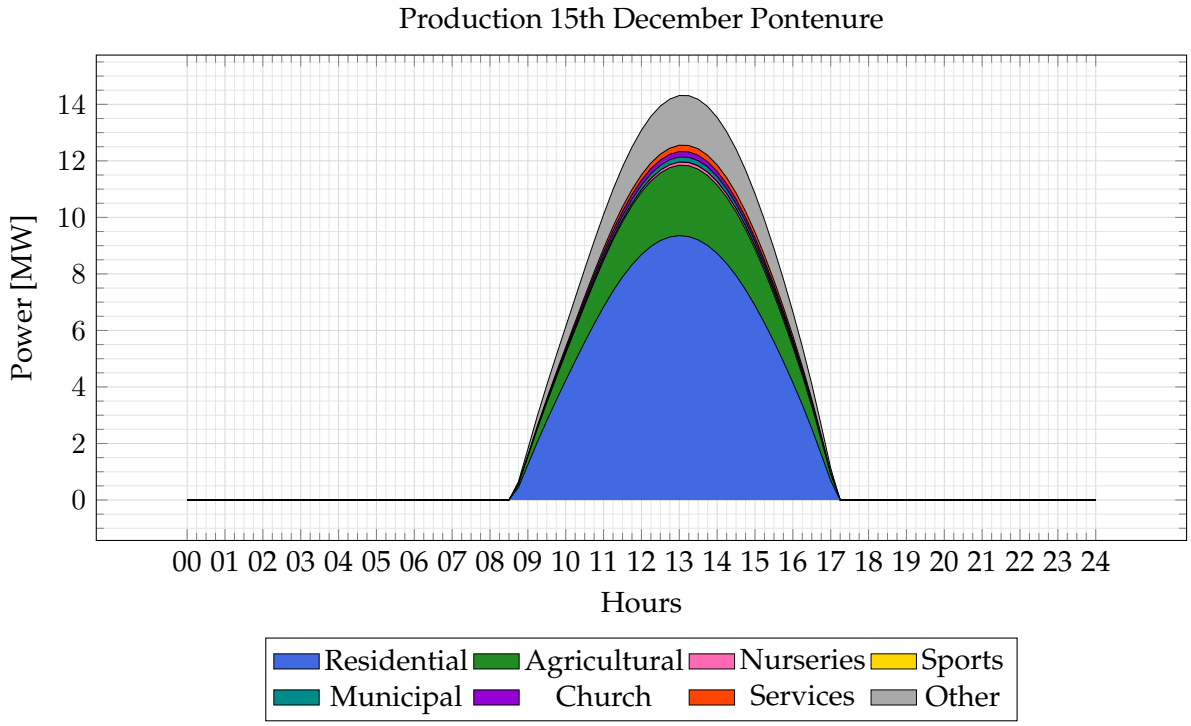


Figure 3.31: *Energy production of the municipality of Pontenure calculated for December 15th using the true solar hour.*

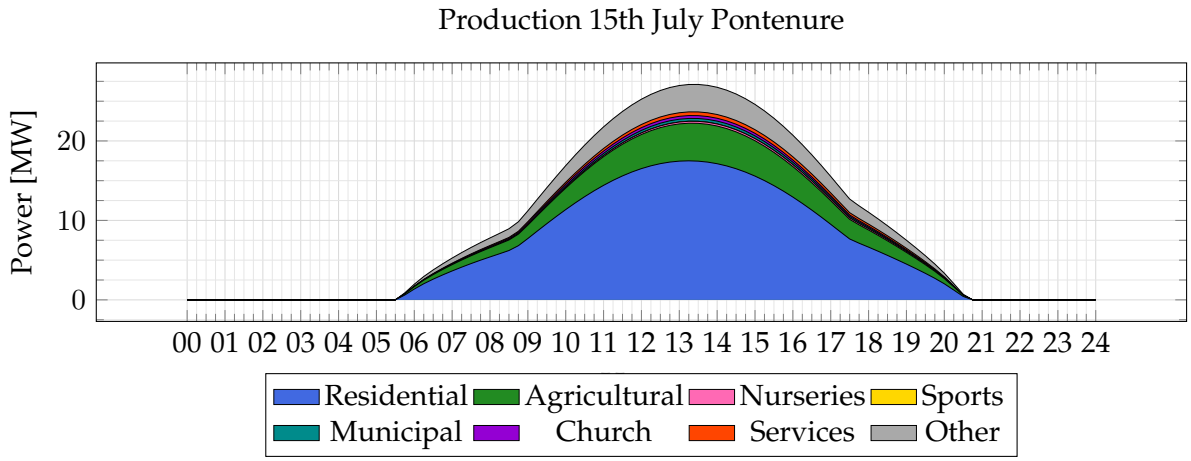


Figure 3.32: *Pontenure 15th July curve used for analyze orientation patterns.*

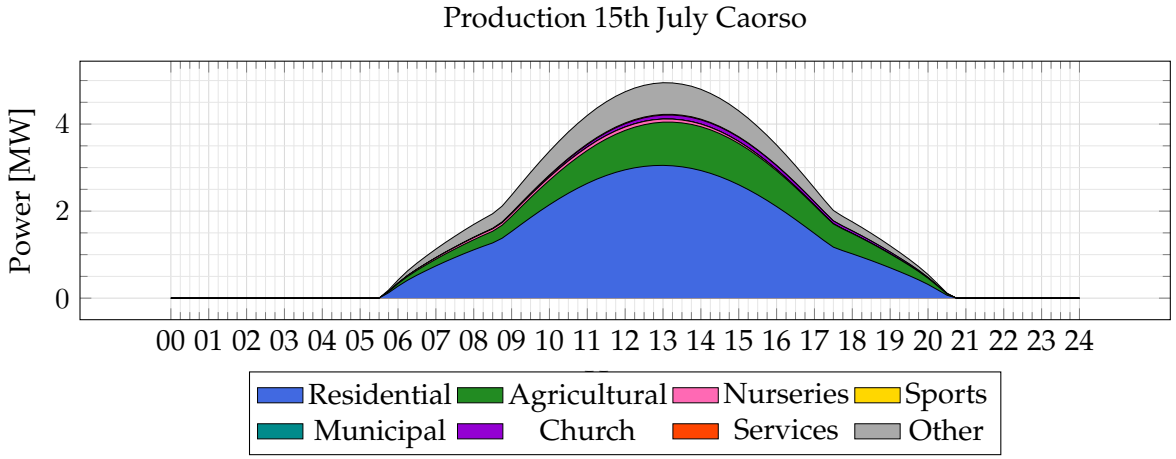


Figure 3.33: *Caorso 15th July curve used for analyze orientation patterns.*

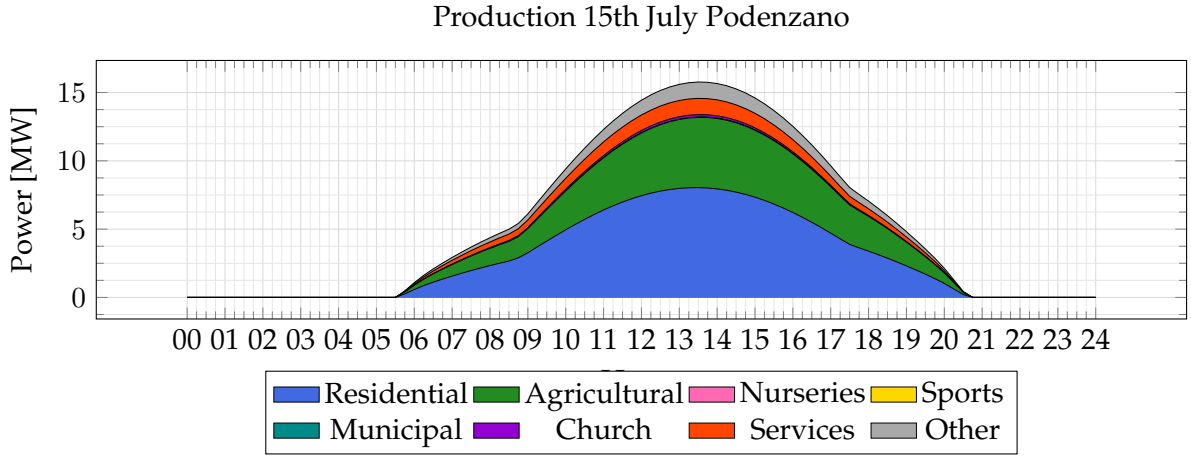


Figure 3.34: *Podenzano 15th July curve used for analyze orientation patterns.*

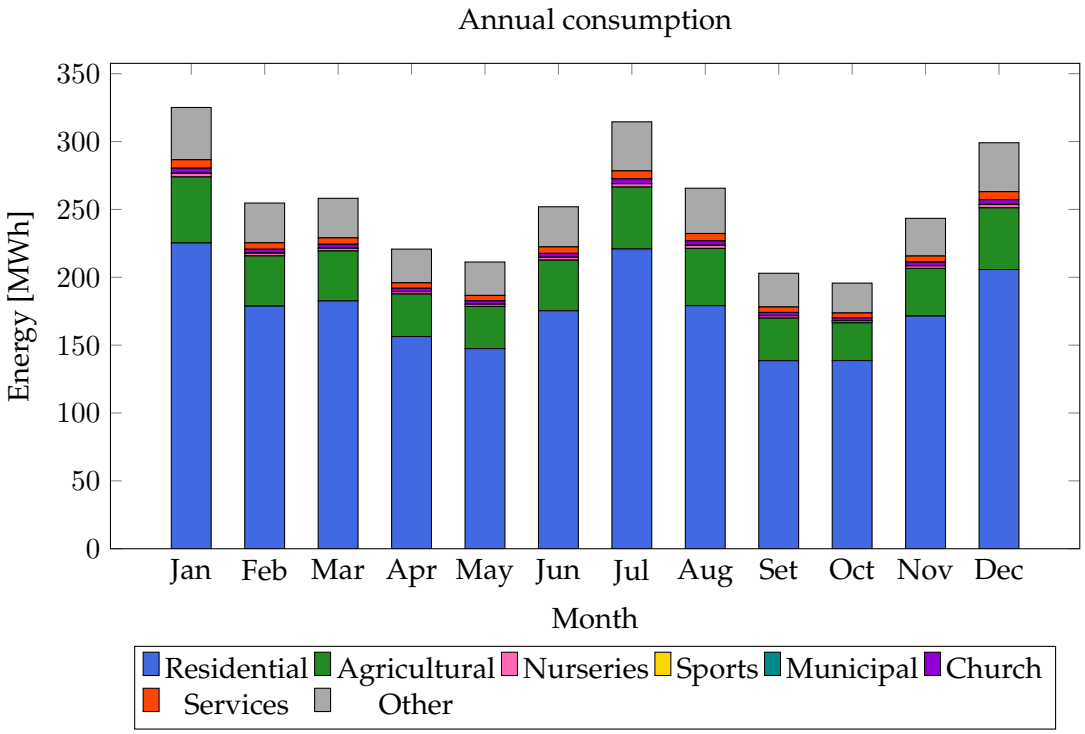


Figure 3.35: *Annual consumption of Pontenure.*

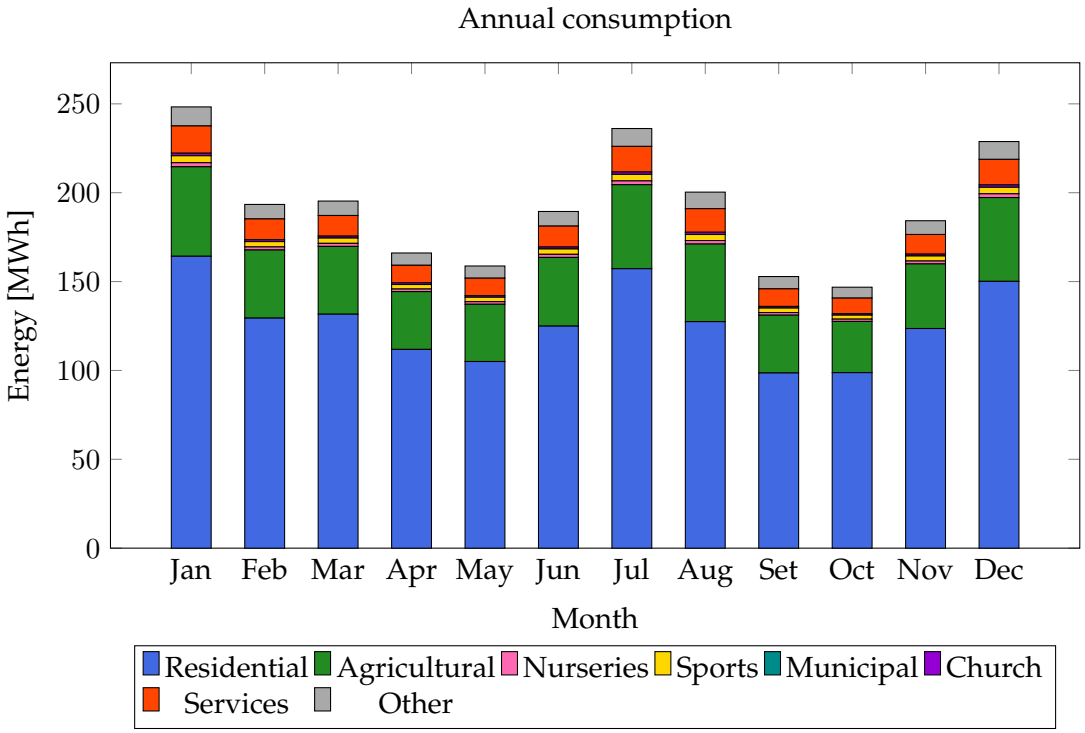


Figure 3.36: *Annual consumption of Piacenza.*

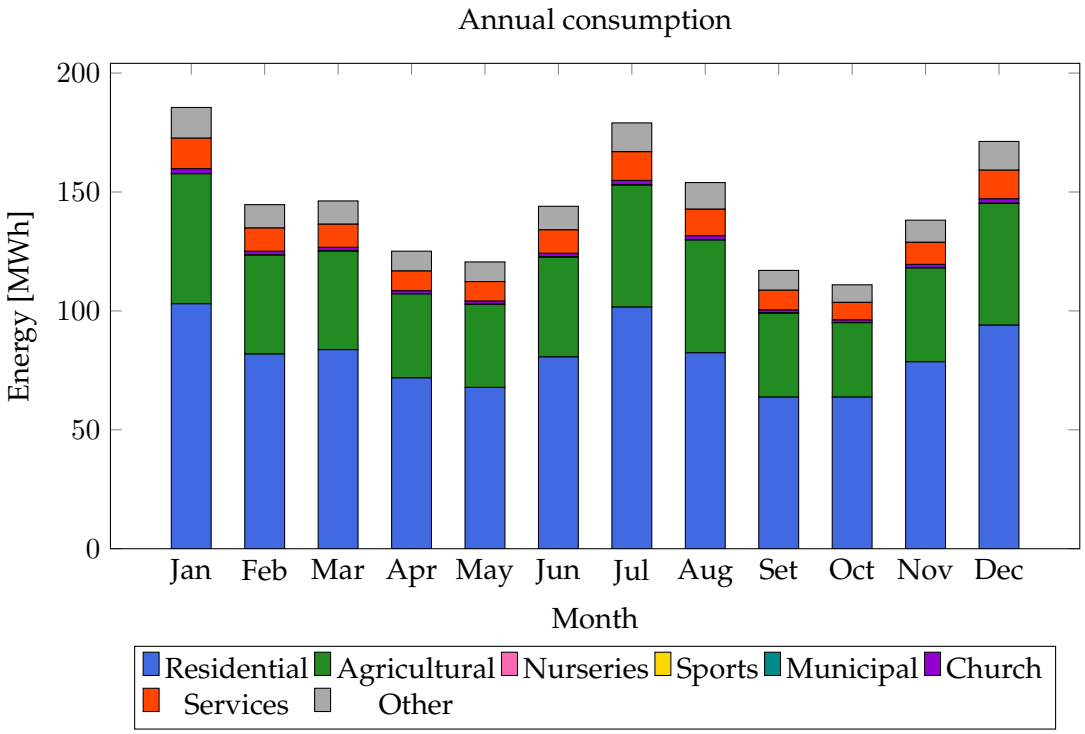


Figure 3.37: *Annual consumption of Podenzano.*

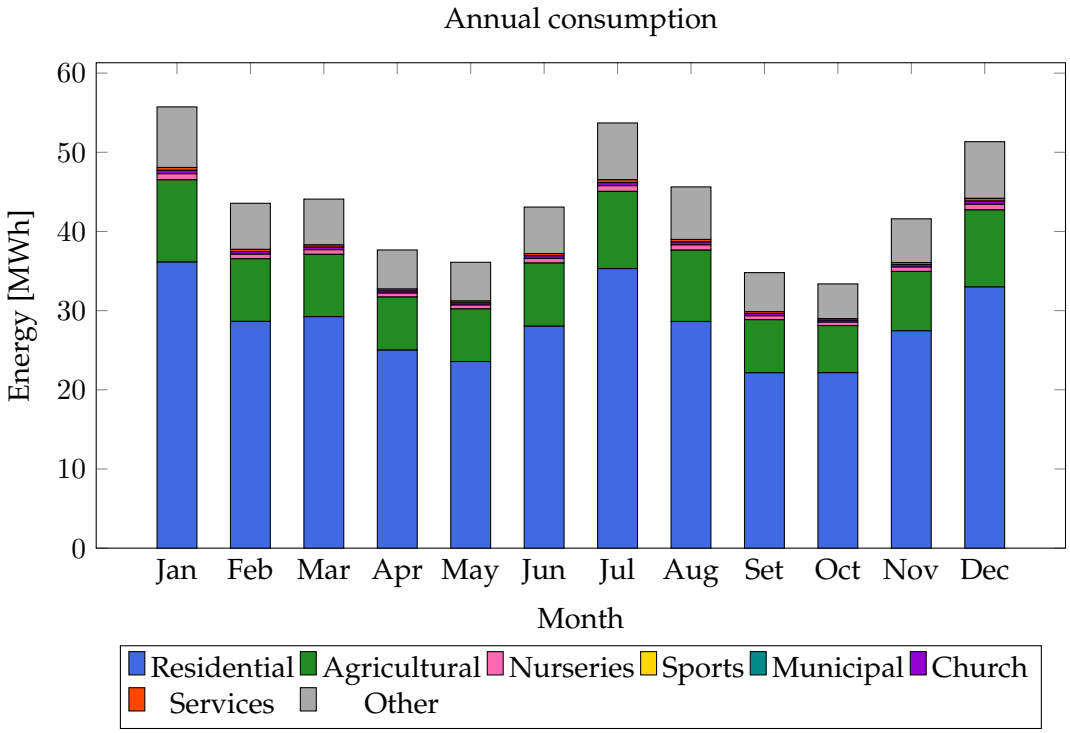


Figure 3.38: *Annual consumption of Caorso.*

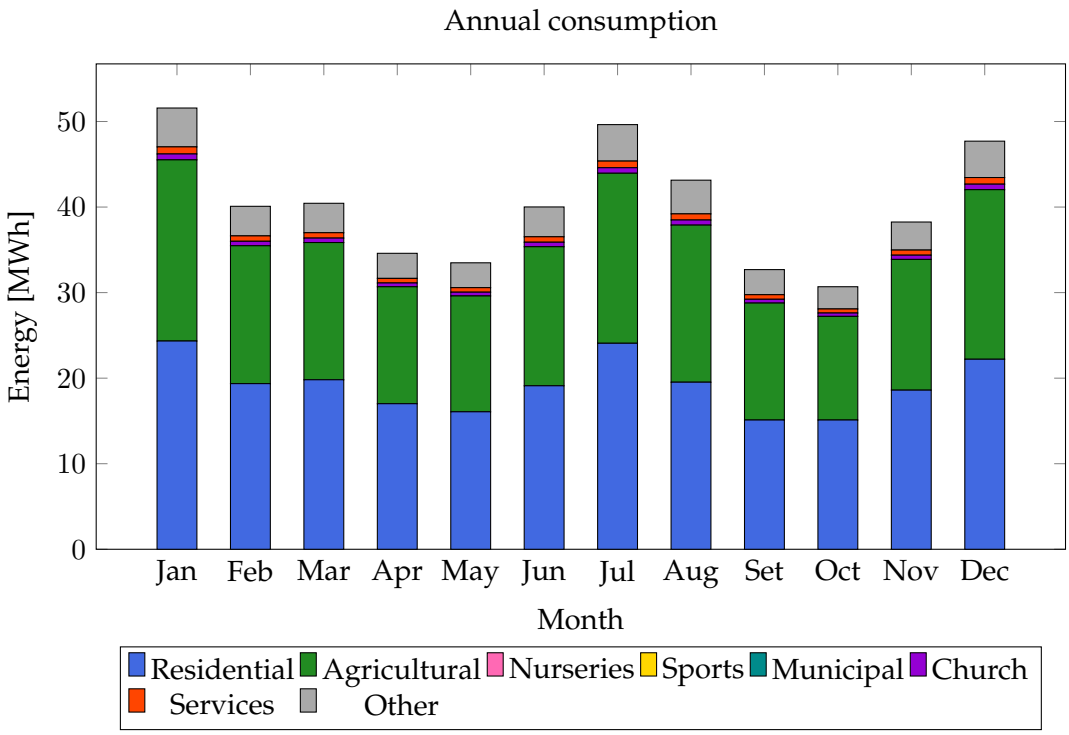


Figure 3.39: *Annual consumption of San Giorgio P.*

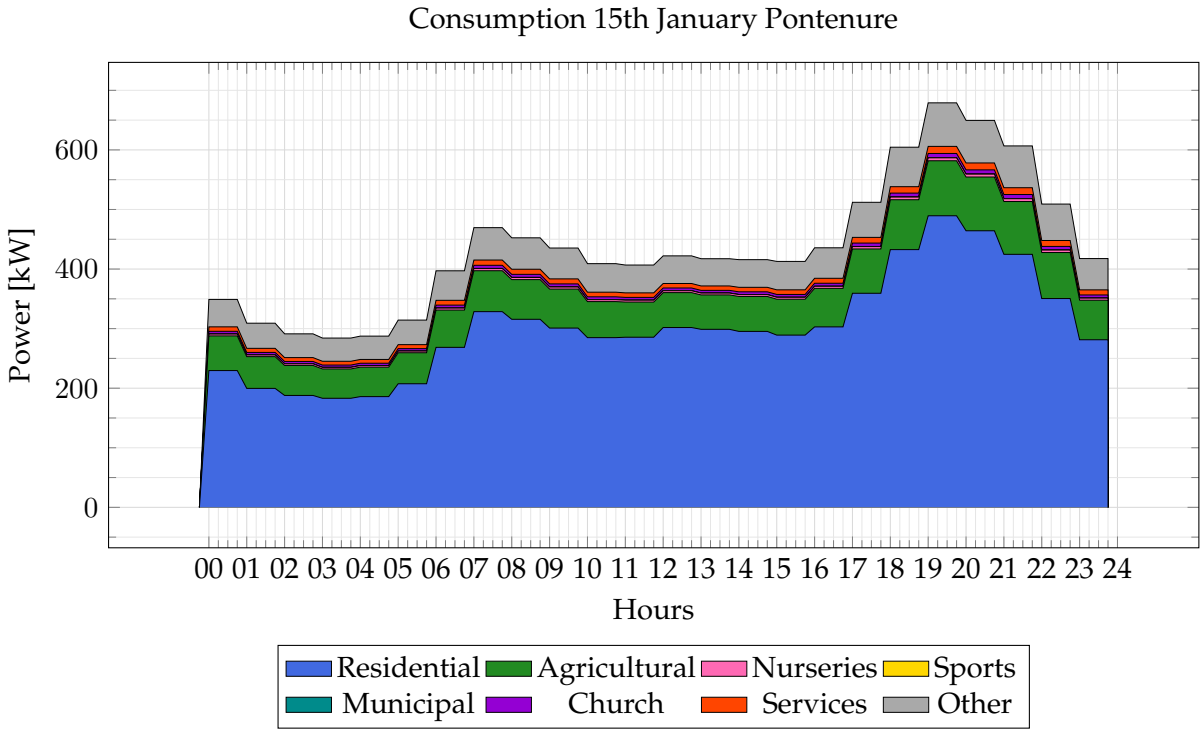


Figure 3.40: *Energy consumed by Pontenure according to our estimates on January 15th.*

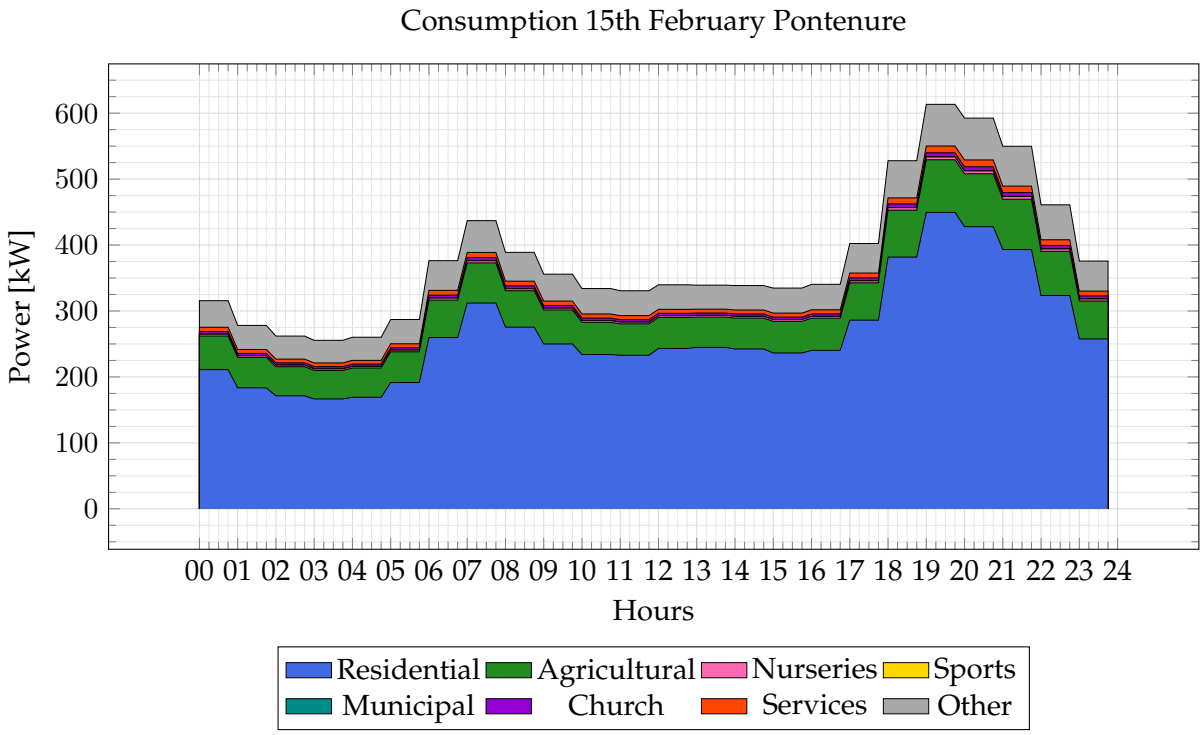


Figure 3.41: *Energy consumed by Pontenure according to our estimates on February 15th.*

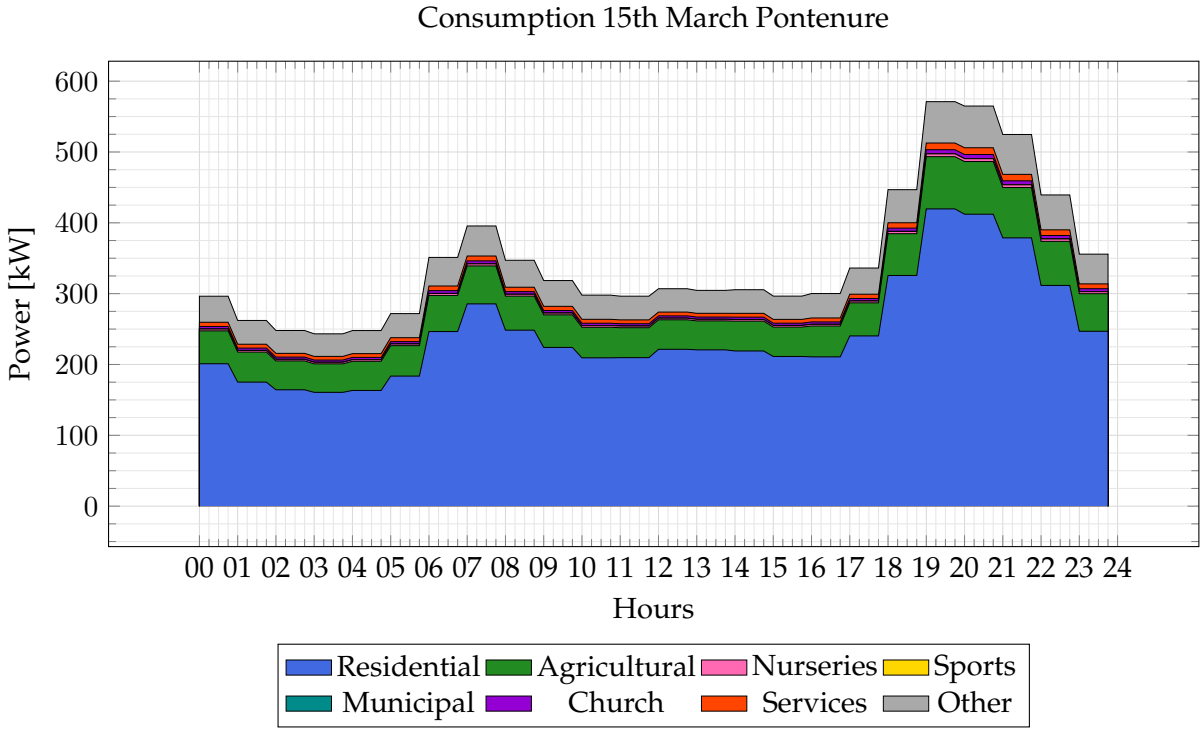


Figure 3.42: *Energy consumed by Pontenure according to our estimates on March 15th.*

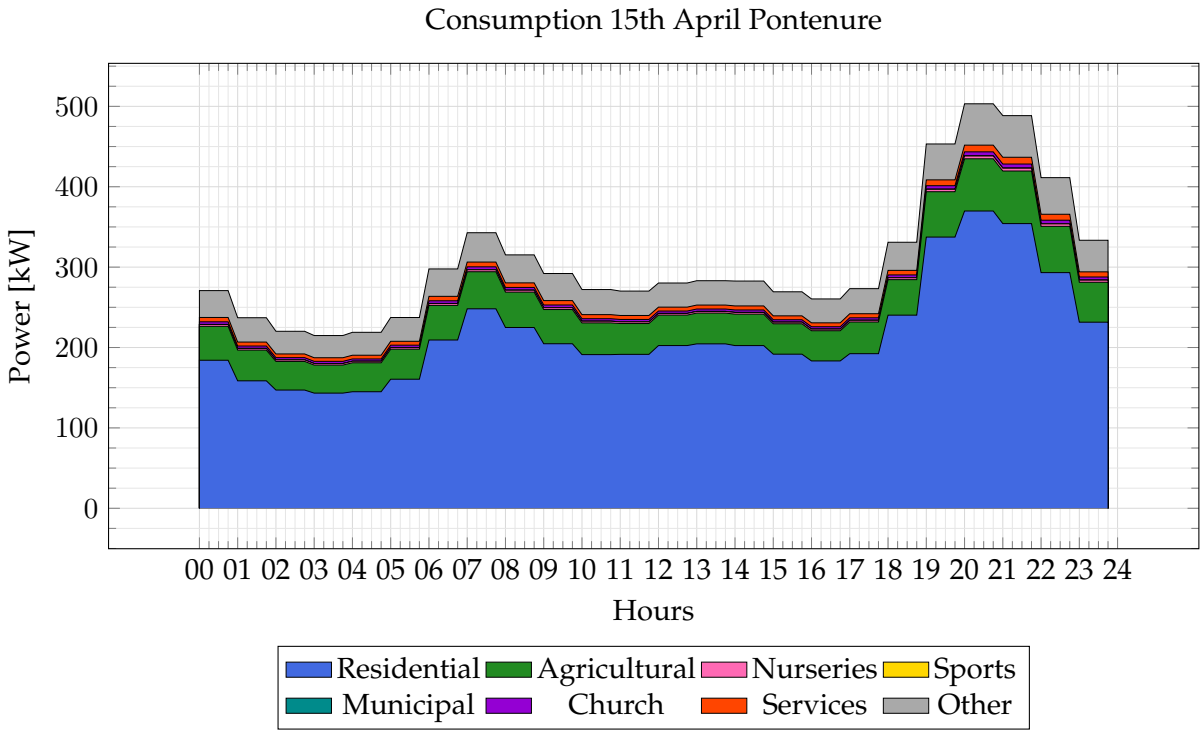


Figure 3.43: *Energy consumed by Pontenure according to our estimates on April 15th.*

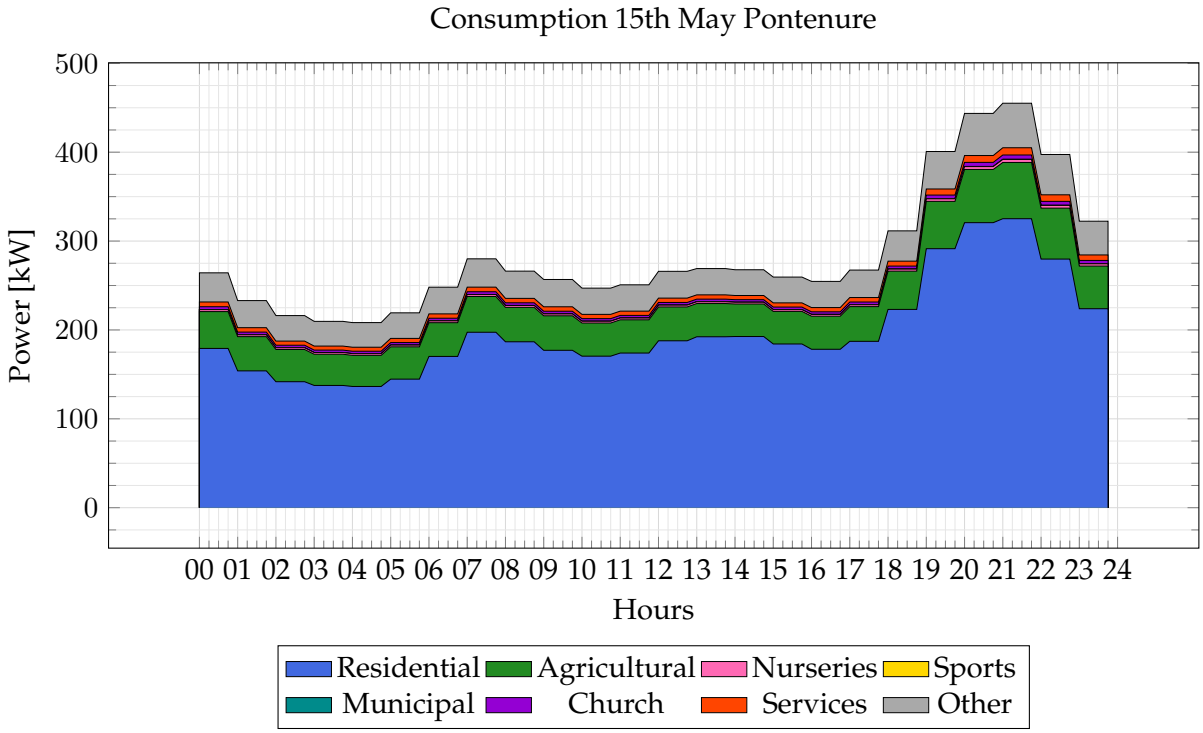


Figure 3.44: *Energy consumed by Pontenure according to our estimates on May 15th.*

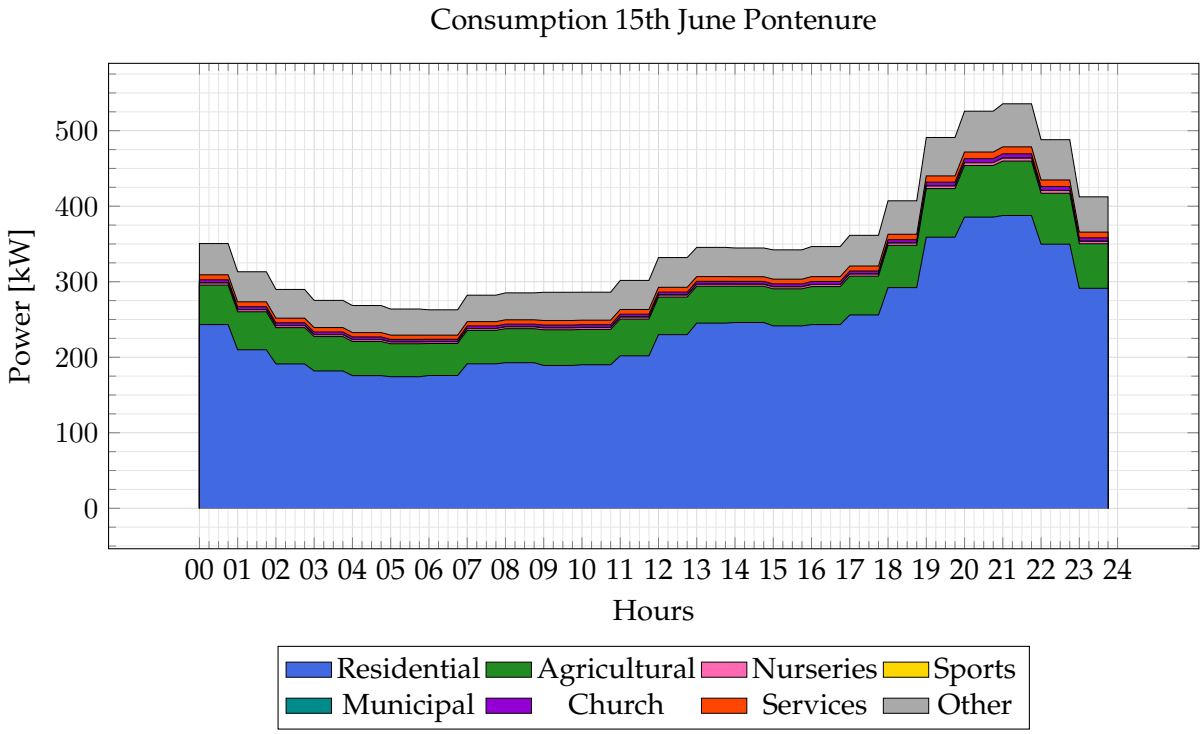


Figure 3.45: *Energy consumed by Pontenure according to our estimates on June 15th.*

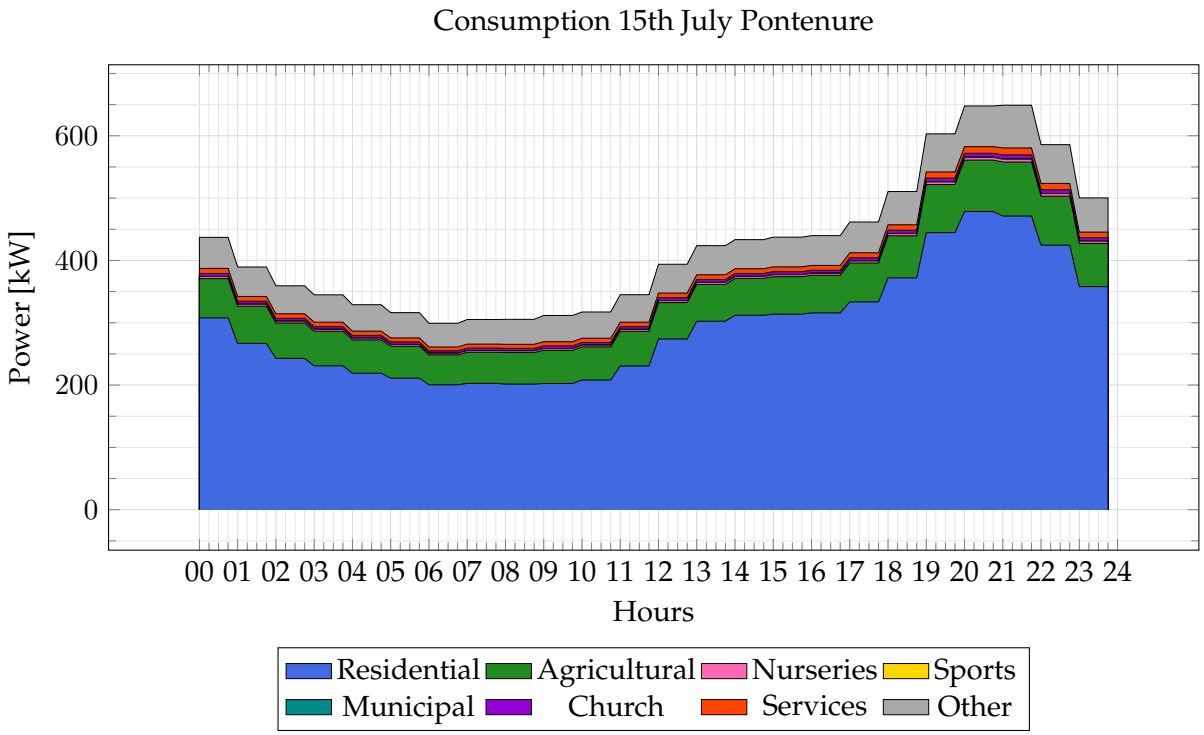


Figure 3.46: *Energy consumed by Pontenure according to our estimates on July 15th.*

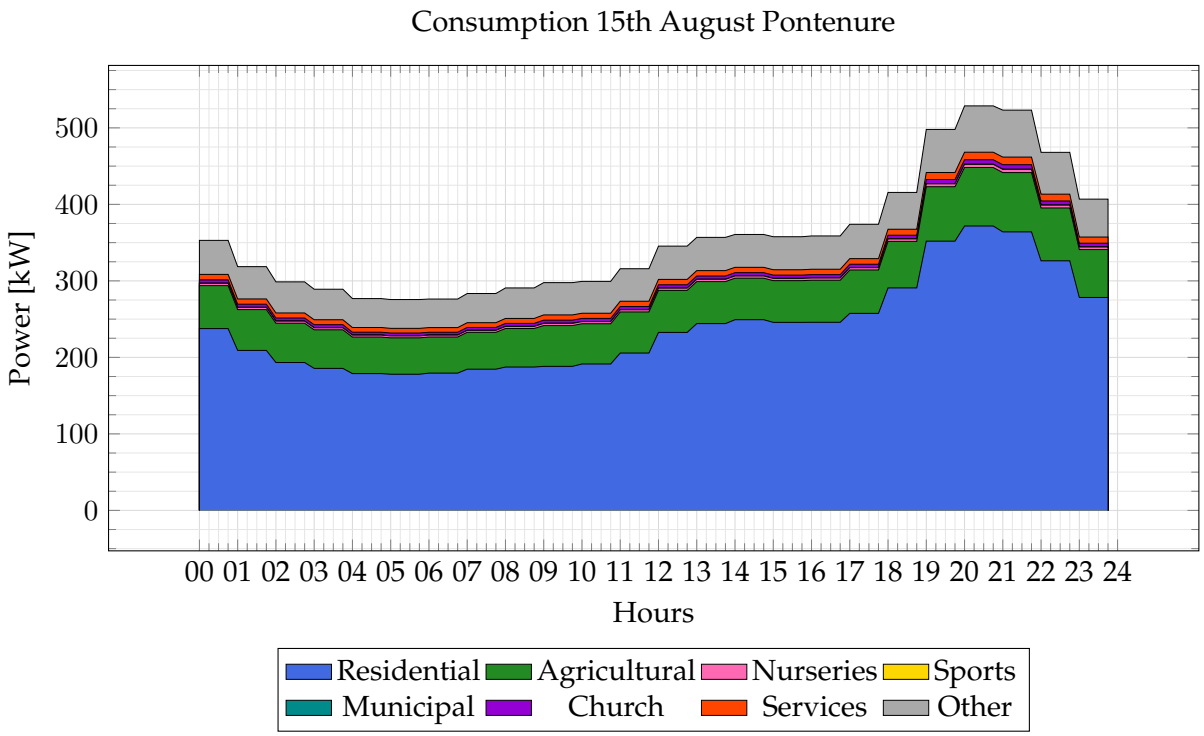


Figure 3.47: *Energy consumed by Pontenure according to our estimates on August 15th.*

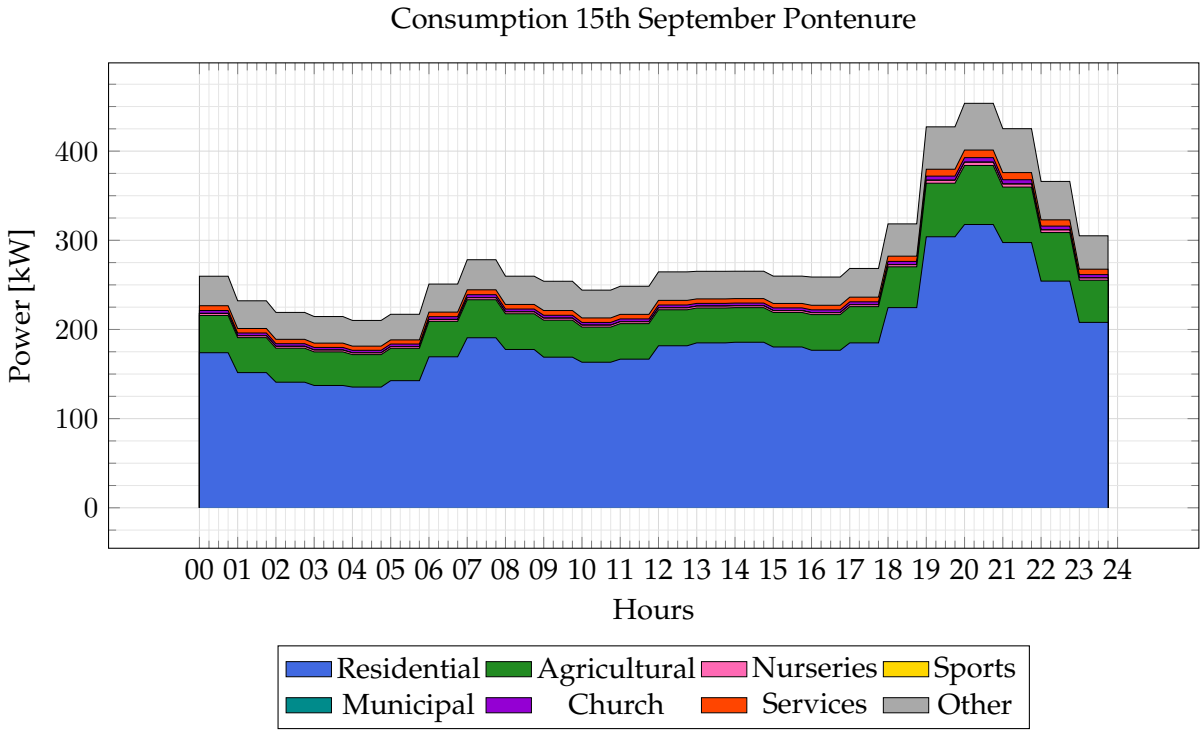


Figure 3.48: *Energy consumed by Pontenure according to our estimates on September 15th.*

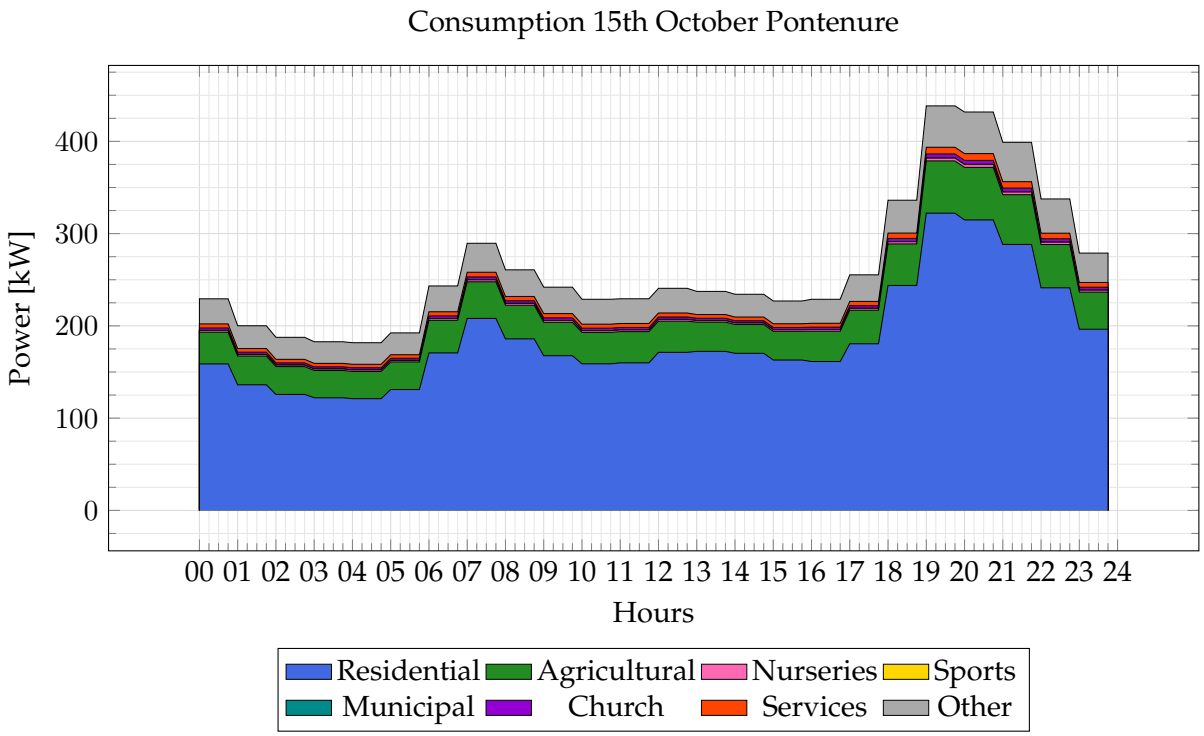


Figure 3.49: *Energy consumed by Pontenure according to our estimates on October 15th.*

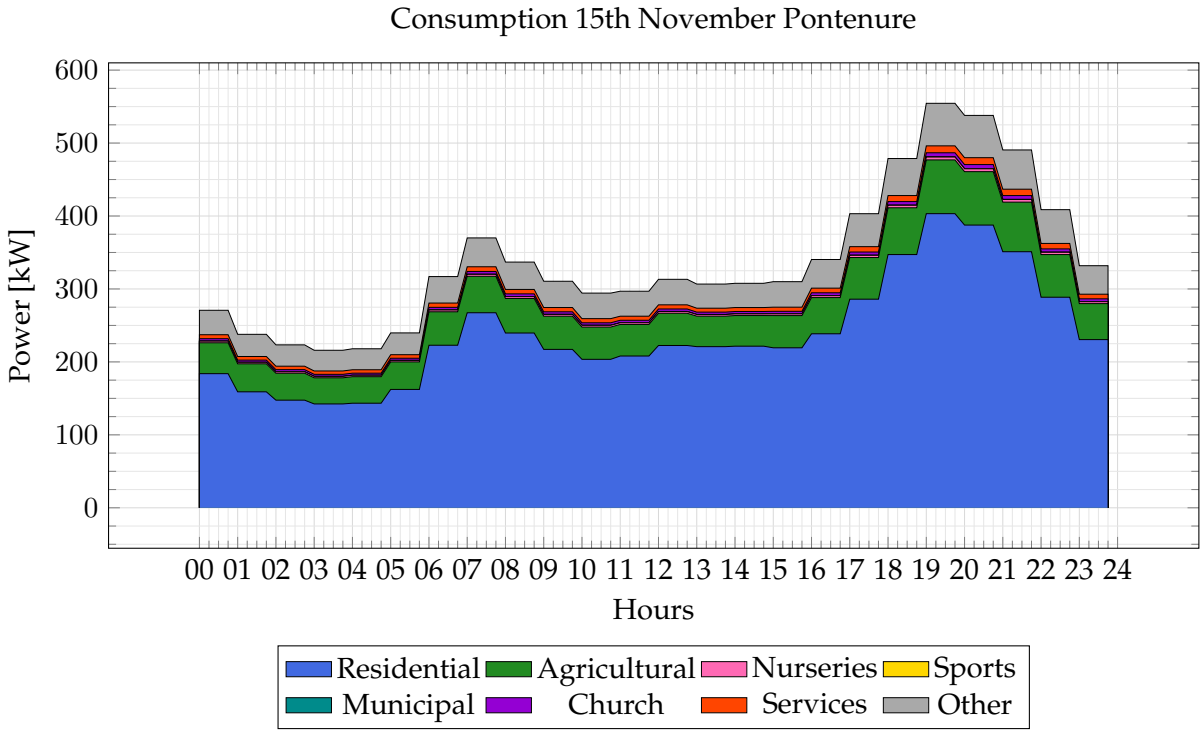


Figure 3.50: *Energy consumed by Pontenure according to our estimates on November 15th.*

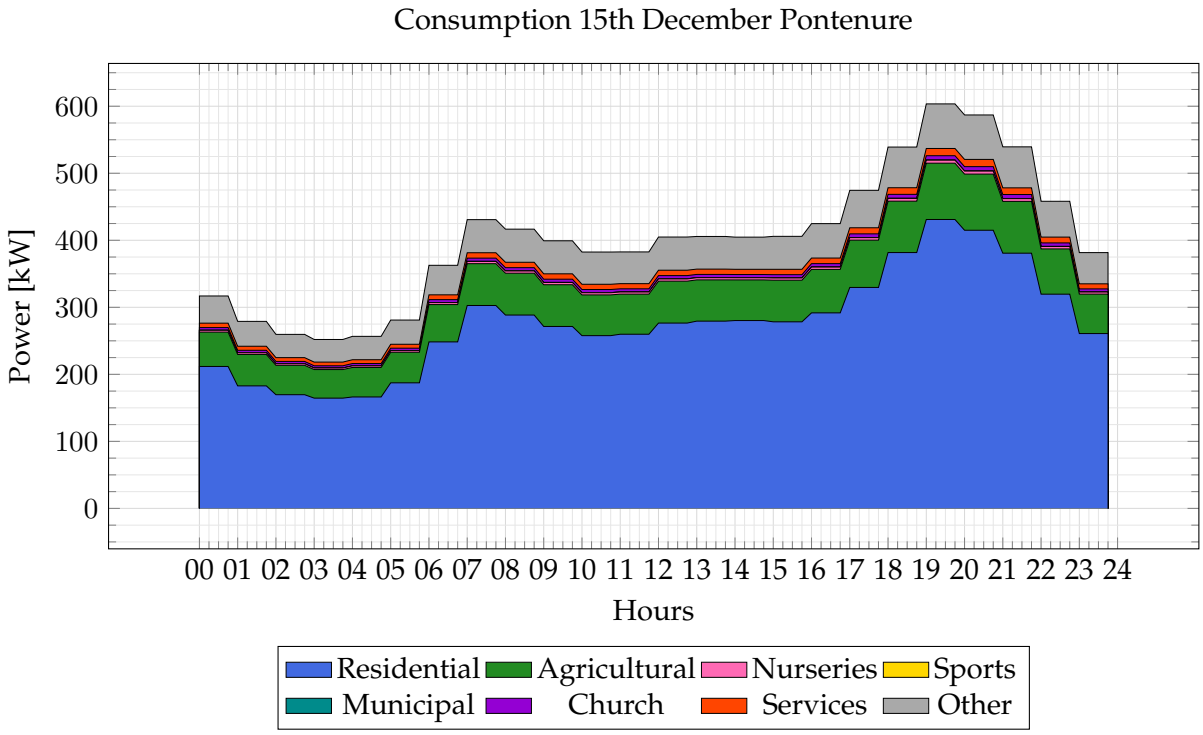


Figure 3.51: *Energy consumed by Pontenure according to our estimates on December 15th.*

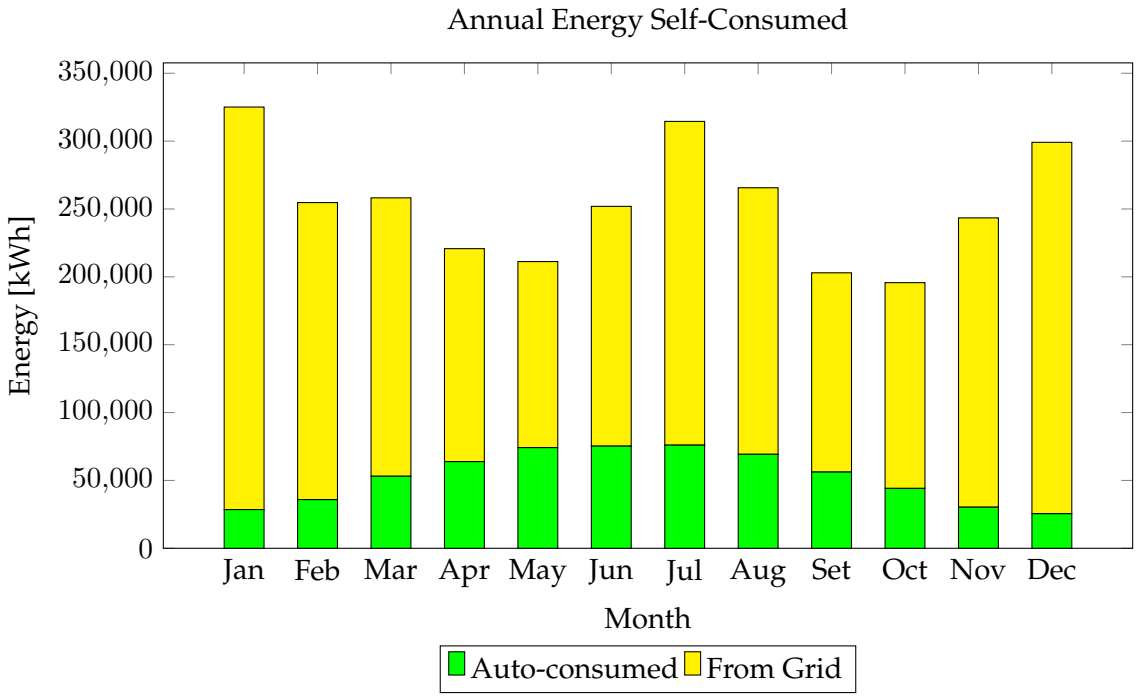


Figure 3.52: *Energy balance between energy absorbed from the grid and self-consumed considering scenario 1 (table 3.27) for Pontenure.*

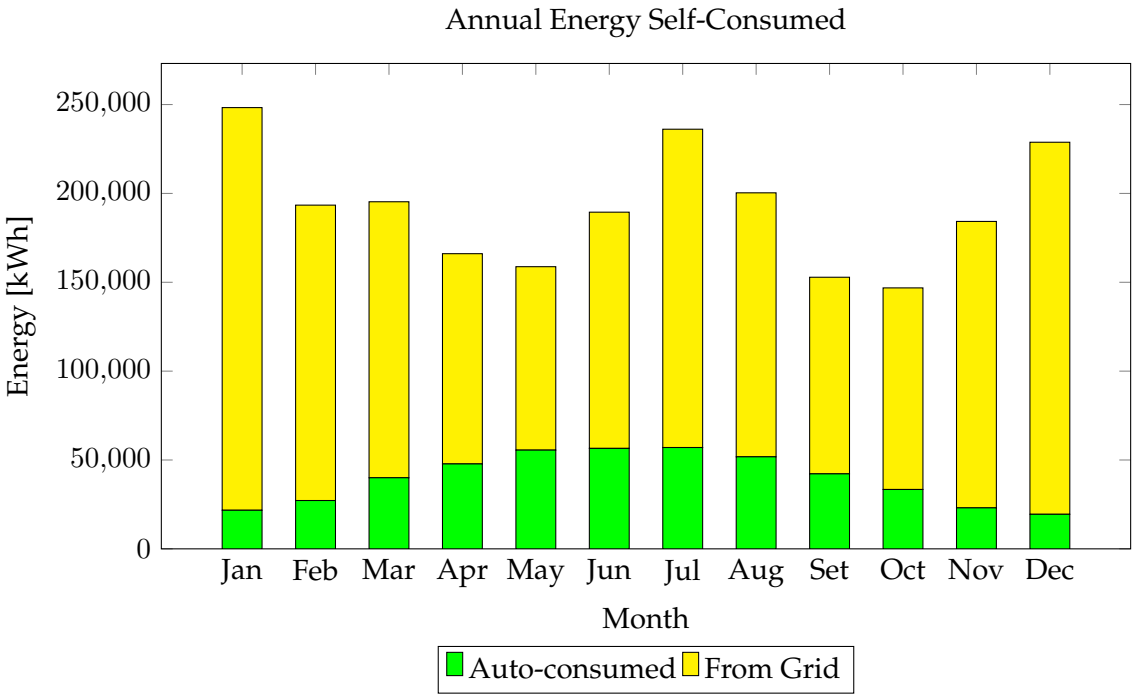


Figure 3.53: *Energy balance between energy absorbed from the grid and self-consumed considering scenario 1 (table 3.28) for Piacenza.*

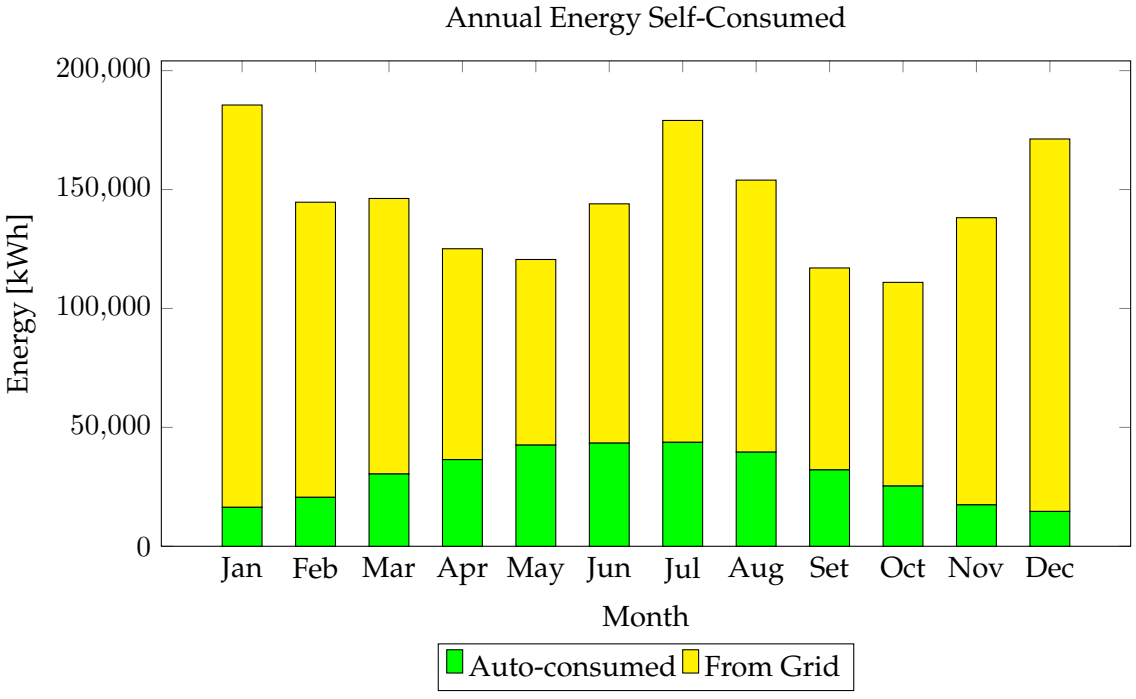


Figure 3.54: *Energy balance between energy absorbed from the grid and self-consumed considering scenario 2 (table 3.29) for Podenzano.*

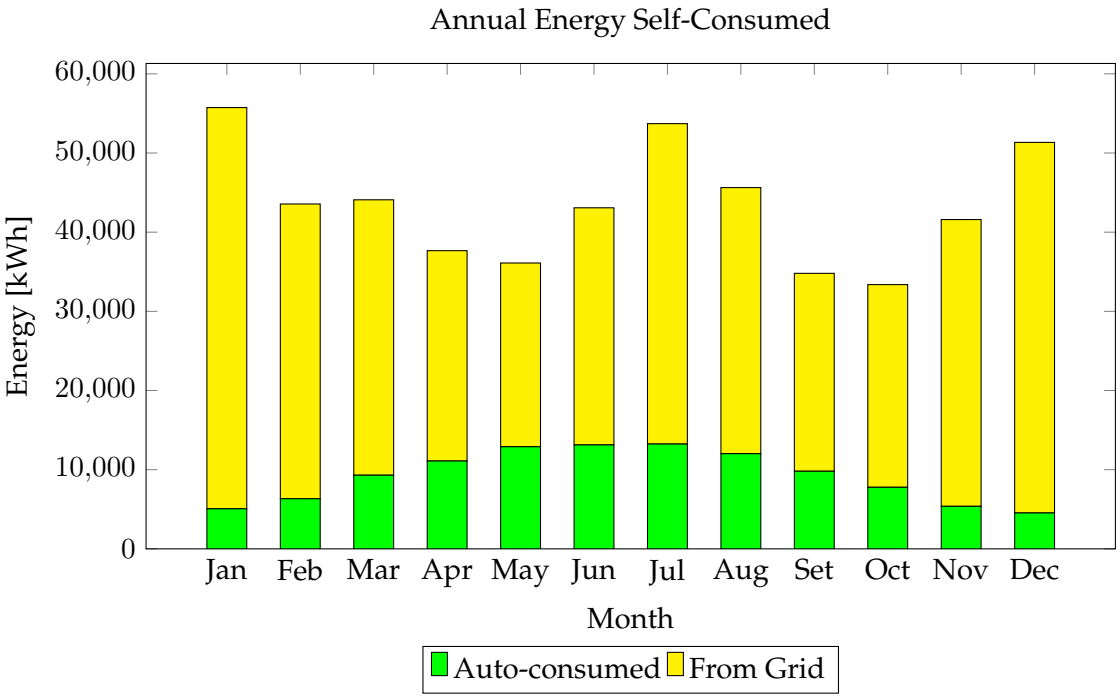


Figure 3.55: *Energy balance between energy absorbed from the grid and self-consumed considering scenario 2 (table 3.30) for Caorso.*

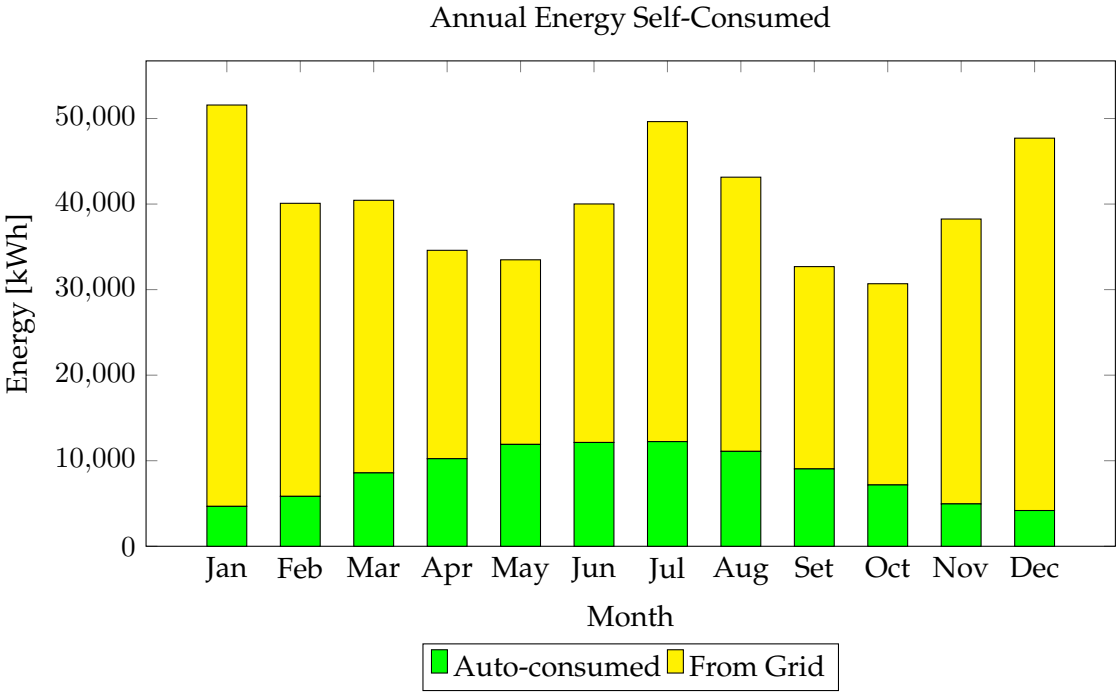


Figure 3.56: *Energy balance between energy absorbed from the grid and self-consumed considering scenario 2 (table 3.31) for San Giorgio P.*

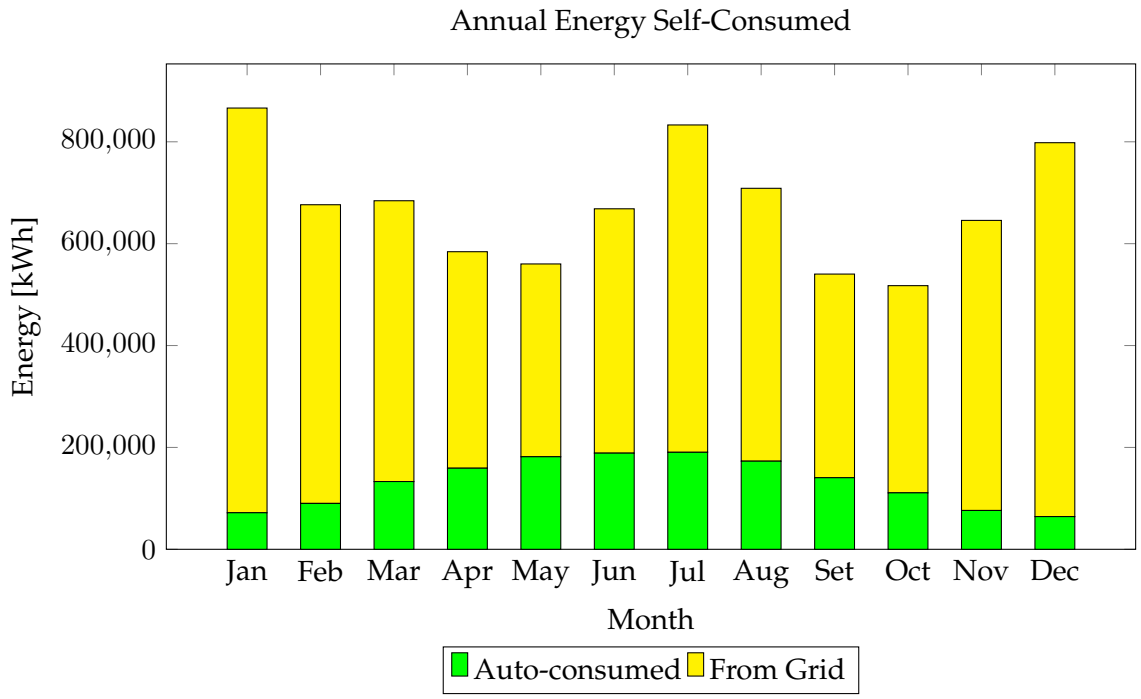


Figure 3.57: *Energy balance between energy absorbed from the grid and self-consumed considering the all primary cabin.*

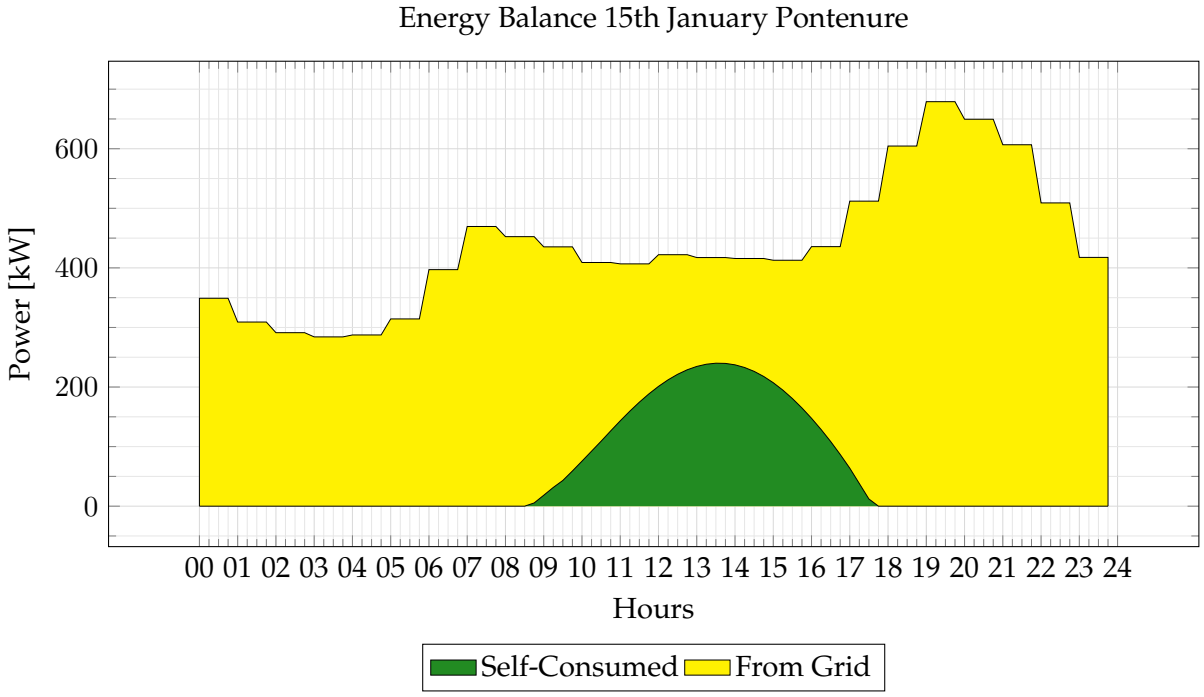


Figure 3.58: *Energy balance of Pontenure according to our estimates on January 15th.*

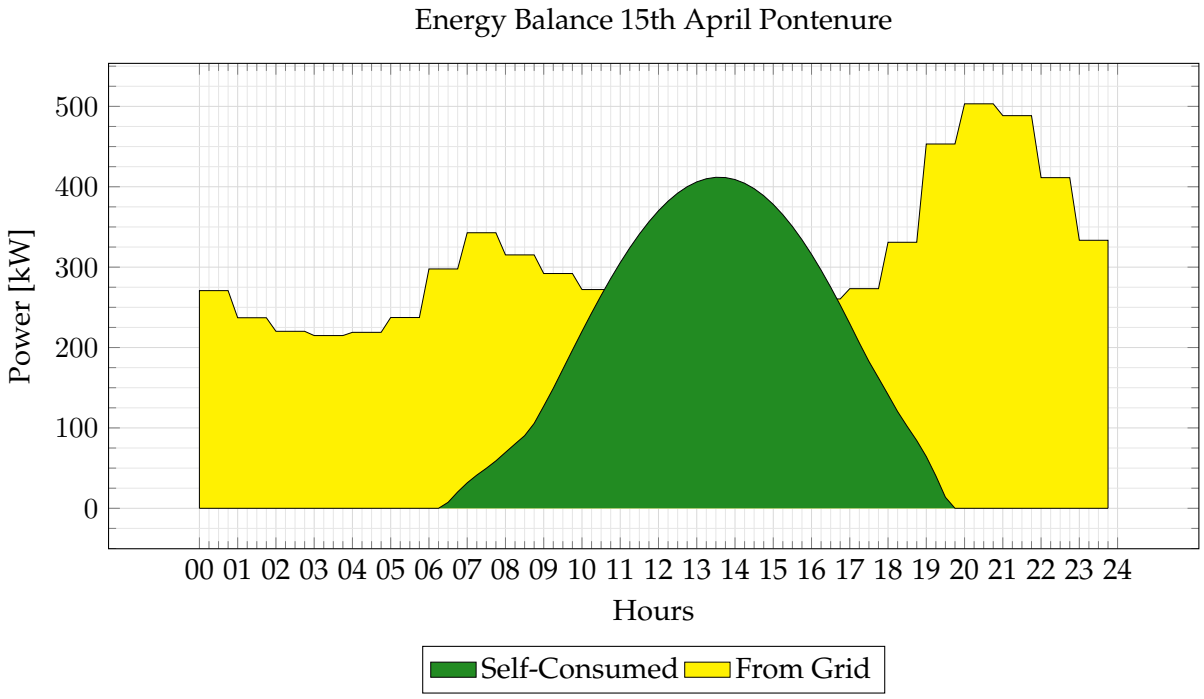


Figure 3.59: *Energy balance of Pontenure according to our estimates on April 15th.*

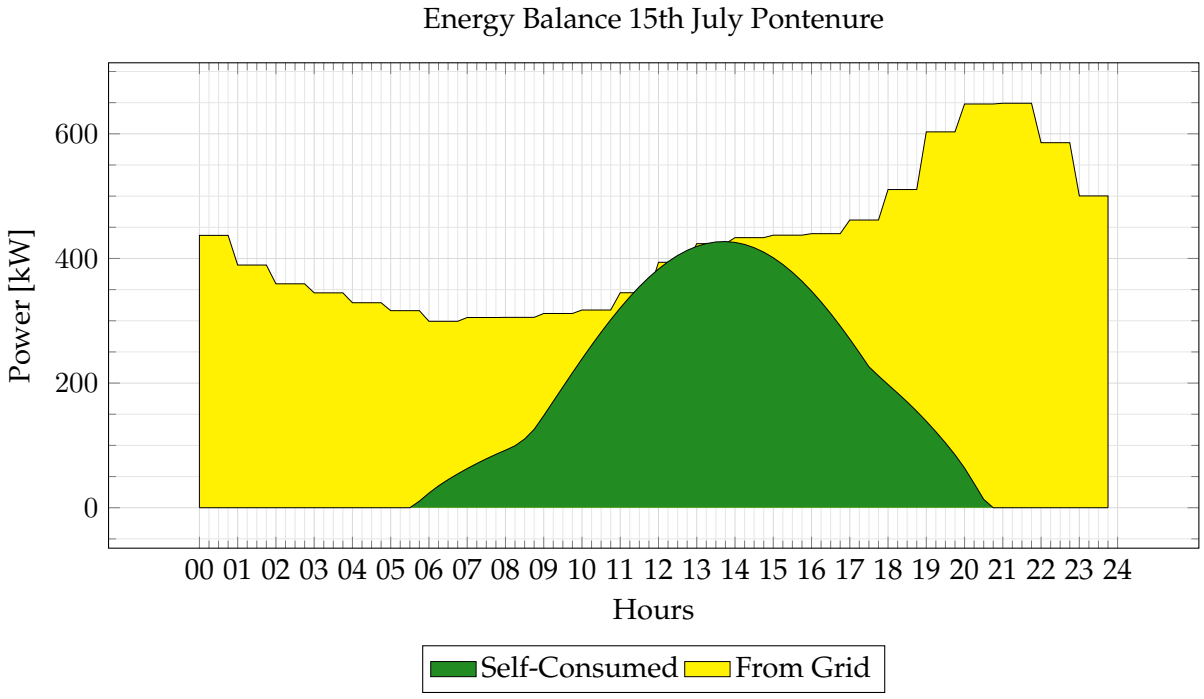


Figure 3.60: *Energy balance of Pontenure according to our estimates on July 15th.*

Conclusion

The methodology developed in this study harnesses scalability, an essential asset in territorial planning, by allowing adjustable analysis across various land segments. Constructing annual production and consumption databases with 15-minute intervals for each building not only enables precise verification of exchanged and self-consumed energy but also provides customized estimates when needed. This approach delivers indispensable tools and data for strategic territorial decision-making, whether in establishing renewable energy communities with self-consumption goals, designing storage systems, or setting photovoltaic installation limits for substations. Our research demonstrates that, even when starting from limited data, it is possible to obtain valuable information on potential energy production and specific consumption needs. Considering emerging trends, residential energy demand, particularly in rural areas, is expected to rise with the growing adoption of electric vehicles, heat pumps, and induction hobs. Similarly, agricultural practices are on the verge of transformation as Agriculture 4.0 and 5.0 drive the shift toward electrification and the reduction of dependence on fossil fuels. Refined iterations of this tool could therefore play a key role in future energy planning and environmental sustainability, establishing a robust foundation for further research and innovative practical applications. However, before such software can be made available to on the market, it is imperative to rigorously validate the proposed methodology to ensure that users can truly rely on this type of analysis.

Future developments

By harnessing these sophisticated algorithms and territorial analysis methods we have developed, it is possible to further refine the creation of production profiles. This could involve incorporating specific losses for each installation, accounting for the shading effects between buildings and the surrounding landscape, and adjusting the efficiency of solar panels in response to seasonal temperature variations. Additionally, integrating historical meteorological data would enable the derivation of realistic production curves for each building, capturing not only clear-sky conditions but also the variability due to cloud cover. Furthermore, as noted during the analysis of cadastral data, there is a diverse range of roof types present across the territory. By obtaining additional information or by identifying the percentage distribution of these roof shapes within the areas under analysis, the accuracy of photovoltaic production estimates can be further enhanced. This comprehensive approach ensures that all relevant factors are considered for the optimal design and operation of renewable energy communities.

A notable strength of the developed approach is the adaptability of the underlying code, which facilitates the relatively rapid implementation of new functionalities. In terms of consumption profiles, future advances should focus on their development. Modern energy efficient homes, often equipped with photovoltaic systems, tend to report net consumption profiles that already reflect on-site production and self consumption. Crafting consumption profiles based on demographic parameters such as age, lifestyle, and habits would undoubtedly enhance the analysis and optimization of both renewable energy community development and territorial planning.

Moreover, simplifying data recovery from existing installations and accurately associating it with corresponding buildings would contribute to a more comprehensive and precise dataset. These improvements could significantly bolster the method's role as an indispensable tool in future energy planning and territorial management.

Ringraziamenti

Un sincero ringraziamento va alla Professoressa Anglani Norma per avermi seguito e per avermi dato l'opportunità di sperimentare un nuovo approccio all'analisi territoriale, imparare a utilizzare QGis, software ormai fondamentale nell'ingegneria elettrica, e per avermi permesso di redigere documenti in maniera particolarmente gradevole con Overleaf. Le sono inoltre grato per avermi riacceso la fiamma per l'ingegneria, per la risoluzione dei problemi e per la continua ricerca di nuove strade volte a superare gli ostacoli, passione che, nel corso degli anni universitari e a causa degli esami, ha rischiato di spegnersi.

Beh, fondamentali i finanziatori di questo percorso, i miei Genitori, Mamma Paola e Papa Giancarlo, che oltre ad avermi generato hanno permesso di coltivare tutte le mie passioni in questi anni, fra qui questa magistrale anche se sudata ma conclusa.

Ringrazio la mia adorata Elena, che mi ha supportato e sopportato con pazienza nei momenti frustranti degli esami, dimostrando pazienza e dedizione rimanendo al mio fianco.

Desidero ricordare con affetto il Collegio Spallanzani, non quella cosa fracita del Don Bosco, la mia vecchia casa che ha reso il percorso universitario meno gravoso, offrendomi avventure (Il Dialogo Perduto) e conoscenze che hanno contribuito in maniera decisiva a plasmare la persona che sono oggi.

Infine, un ringraziamento speciale va a tutti gli amici incontrati durante questo percorso, che portero con me lungo il mio cammino. Un particolare apprezzamento va anche alla caserma dei vigili del fuoco di Gazzaniga, dove ho trascorso innumerevoli ore nel Mio Ufficio per raggiungere questo traguardo, vivendo esperienze inconsuete con la crew del giorno (Ettore, Michele, Cinzia, Luca, Il Comunista e quelli nuovi), che hanno reso le lunghe sessioni universitarie più leggere e piacevoli.

Bravo Me

Collection of vector data tables

In this appendix we would like to report the attributes contained in the layers from the Emilia-Romagna geoportal used during the study, and report the databases created to formulate the results obtained. A description of their use is given in the second chapter2. In the tables we will not list all the items contained in the databases since there would be thousands of them, but we will list only three items per type to show how they appear.

Vectorial layer attribute tables downloaded from the geo-portal of Emilia-Romagna.

The tables of this group are: 33, 34,35.36.

ARERA database average household consumption.

This class include table 37 and table 38.

Databases created with the developed method.

The tables show the results obtained by processing the source data, table 39 pertains to power plant, table 40 enumerates PODs, and the table 41 present production data used in energy community analysis, which is identical to consumption data.

Table 33: *Table containing data from the V_EDI_GPG layer.*

STAT_E	D_STAT_E	TY_EDI	D_TY_EDI	NOME	COD_COM	SEZ	FOGLIO	NUMERO	ALLEGATO	SVILUPPO	...
1	in esercizio	7	edificio industriale	NULL	NULL	NULL	0	NULL	NULL	NULL	...
1	in esercizio	6	chiesa/basilica	NULL	H887	_	6	20	A	0	...
1	in esercizio	1	generica	NULL	NULL	NULL	NULL	NULL	NULL	NULL	...

...	TY_E	ID_E	DT_INI_VAL	DATA_AGG	D_TIPO_AGG	DT_PRES	ST_VALID	ST_CERTIF	...
...	EDI	b94ce9df-6aeb-4660-9193-10faed0d5fe9	2003-07-31	2022-05-19	Puntuale	2022-05-19	0	0	...
...	EDI	e9afd01a-eae4-44fb-9549-754ebbff98ee	1979-01-01	2022-05-19	Puntuale	2022-05-19	0	0	...
...	EDI	2d6429f8-3d13-40df-accc-a61ac330effd	2008-07-31	2011-10-21	Massivo	2008-07-31	0	0	...

...	QUALITA	METODO	D_METODO	COMP_FONTI	SEZ_ID_E	TIPO_FONTE	DATA_DA	DATA_A
...	00	2	Aggiornamento geometria	00	4737ac40-6ee7-4b63-9efc-628a3705d5a0	Quick Bird 2002-2003	2002-01-01	2003-12-31
...	00	2	Aggiornamento geometria	00	4737ac40-6ee7-4b63-9efc-628a3705d5a0	CTR5	1977-01-01	1998-12-31
...	03	1	Primo caricamento	02	NULL	AGEA 2008	2008-01-01	2008-12-31

Table 34: *Table containing data from the V_FAB_GPG layer.*

TY_E	ID_E	DT_INI_VAL	DATA_AGG	D_TIPO_AGG	DT_PRESS	ST_VALID	ST_CERTIF	...
FAB	02fab564-caad-4851-8d87-88b189d98f29	18/01/2012	2011-10-21	NULL	Massivo	0	0	...
FAB	2b66f0c8-a49d-417b-9ea1-42318724945f	2012-01-18	2022-05-19	Puntuale	2022-05-19	0	0	...
FAB	16d457d6-5e48-42c1-8992-1ec441c3ad0f	2012-01-18	2022-05-19	Puntuale	2022-05-19	0	0	...

...	QUALITA	METODO	D_METODO	COMP_FONTI	SEZ_ID_E	TIPO_FONTE	DATA_DA	DATA_A
...	01	1	Primo caricamento	03	NULL	CTR5	1977-01-01	1998-12-31
...	00	2	Aggiornamento geometria	00	4737ac40-6ee7-4b63-9efc-628a3705d5a0	AGEA 2008	2008-01-01	2008-12-31
...	00	2	Aggiornamento geometria	00	4737ac40-6ee7-4b63-9efc-628a3705d5a0	Quick Bird 2002-2003	2002-01-01	2003-12-31

Table 35: *Table containing data from the V_UVL_GPG layer.*

TY_PORZ	D_TY_PORZ	TY_INTR	D_TY_INTR	H_INTR	BASE	D_BASE	TETTO	D_TETTO	B_UVL	H_UVL	V_UVL	...
1	generica	98	Non assegnato	0	98	Non assegnato	98	Non assegnato	144.37	4	0	...
1	generica	98	Non assegnato	NULL	98	Non assegnato	98	Non assegnato	98.62	5	NULL	...
1	generica	98	Non assegnato	0	98	Non assegnato	98	Non assegnato	134.03	3 0	...	

...	EDI_ID_E	TY_E	ID_E	DT_INI_VAL	DATA_AGG	D_TIPO_AGG	DT_PRES	ST_VALID	...
...L	0d66c787-0e3c-49a2-b53c-dc672d426ae4	UVL	753f50fc-34ae-4c1e-956b-6f25ee3e8c7a	2012-01-18	2011-10-21	Massivo	NULL	NULL	...
...	7223354f-5b79-4cca-9e30-3bc9fad4fd1e	UVL	dc86e850-b667-49dc-9c5a-a654126e8c6a	2012-01-18	2011-10-21	Massivo	NULL	NULL	...
...	82e37547-0779-413a-a853-9dc106fc84bd	UVL	581fa2c9-c620-45c2-bc23-bd6d387283fa	2012-01-18	2022-05-19	Puntuale	2022-05-19	0	...

...	ST_CERTIF	QUALITA	METODO	D_METODO	COMP_FONTI	SEZ_ID_E	TIPO_FONTE	DATA_DA	DATA_A	H_GRONDA	H_COLMO
...	NULL	20	1	Primo caricamento	00	NULL	DTM DSM 2008	2008-01-01	2008-12-31	0	0
...	NULL	20	1	Primo caricamento	00	NULL	DTM DSM 2008	2008-01-01	2008-12-31	NULL	NULL
...	00	00	1	Primo caricamento	00	NULL	DTM DSM 2008	2008-01-01	2008-12-31	0	0

Table 36: *Table containing data from the Use_Suolo layer.*

SIGLA	COD_1	COD_2	COD_3	COD_4	COD_TOT	DESCR	HECTARES
Aa	5	1	2	4	5124	Acquacolture in ambiente continentale	1,01637
Cv	2	2	1	0	2210	Vigneti	4,92775
Aa	1	1	2	2	1122	Strutture residenziali isolate	1,75716

Table 37: *Table of withdrawal data for domestic customers Provincial year 2022.*

Anno Mese	Tipo Mercato	Classe potenza	Residenza	Provincia	F1(%)	F2(%)	F3(%)	Medio consumo mensile
gen-22	Maggior Tutela	0<potenza_impegnata<=1.5	Tutti	Piacenza	30,56%	23,68%	45,75%	18
gen-22	Maggior Tutela	0<potenza_impegnata<=1.5	Residente	Piacenza	29,42%	24,56%	46,02%	44
gen-22	Maggior Tutela	0<potenza_impegnata<=1.5	Non Residente	Piacenza	31,98%	22,58%	45,43%	10
gen-22	Maggior Tutela	1.5<potenza_impegnata<=3	Tutti	Piacenza	30,48%	27,64%	41,88%	146
gen-22	Maggior Tutela	1.5<potenza_impegnata<=3	Residente	Piacenza	30,58%	27,78%	41,63%	173
gen-22	Maggior Tutela	1.5<potenza_impegnata<=3	Non Residente	Piacenza	29,47%	26,28%	44,25%	57
gen-22	Maggior Tutela	3<potenza_impegnata<=4.5	Tutti	Piacenza	29,76%	28,00%	42,24%	297
gen-22	Maggior Tutela	3<potenza_impegnata<=4.5	Residente	Piacenza	29,75%	28,16%	42,09%	318

Table 38: *Table of hourly withdrawal data by province, size 0-1,5.*

Anno Mese	Provincia	Tipo Mercato	Classe potenza	Residenza	Working Day	Orario	Prelievo medio Orario Provinciale (kWh)
mag-22	Piacenza	Mercato Libero	1.5<potenza_impegnata<=3	Residente	Giorno_feriale	Ora1	0,162
mag-22	Piacenza	Mercato Libero	1.5<potenza_impegnata<=3	Residente	Giorno_feriale	Ora2	0,139
mag-22	Piacenza	Mercato Libero	1.5<potenza_impegnata<=3	Residente	Giorno_feriale	Ora3	0,1274
mag-22	Piacenza	Mercato Libero	1.5<potenza_impegnata<=3	Residente	Giorno_feriale	Ora4	0,1225
mag-22	Piacenza	Mercato Libero	1.5<potenza_impegnata<=3	Residente	Giorno_feriale	Ora5	0,1217
mag-22	Piacenza	Mercato Libero	1.5<potenza_impegnata<=3	Residente	Giorno_feriale	Ora6	0,1299
mag-22	Piacenza	Mercato Libero	1.5<potenza_impegnata<=3	Residente	Giorno_feriale	Ora7	0,1583
mag-22	Piacenza	Mercato Libero	1.5<potenza_impegnata<=3	Residente	Giorno_feriale	Ora8	0,1895

Table 39: *Table containing data to classify buildings and those characterising installable photovoltaic systems.*

D_TY_EDI	ID_E	Sup_Suolo	SIGLA	DESCR	ID_Fab	...
generica	13bd6683-ef5c-4f5b-b50a-72015633684d	237,62	Ed	Tessuto residenziale urbano	4d86e5e7-d5bf-4cc3-8c8e-c0b1790d3be2	...
generica	1350edf2-a5c8-4600-bc9f-682a15caf644	738,19	Ed	Tessuto residenziale urbano	4bec9a49-82ec-4b0a-bacb-fabe9a57e5d8	...
generica	21acdecb-c783-42dc-bc81-b4b0a818da96	195,32	Ed	Tessuto residenziale urbano	606ad5df-08ef-4543-baf0-67ef434de4d4	...

...	orientamento	A_disp	N_pFV	P_peack [kW]
...	8,41	95	48	19,68
...	-72,32	295	150	61,50
...	20,62	78	39	15,99

Table 40: *Table containing data to classify buildings and create consumption profiles.*

D_TY_EDI	ID_E	Sup_Suolo	SIGLA	DESCR	ID_Fab	...
generica	ad3898d6-5d0d-4724-a1e2-035001c44ab1	233,714514069958	Es, Iz	Insediamenti agro-zootecnici, Strutture residenziali isolate	c1a55f36-a075-4a5c-b683-1beae88e3e56	...
generica	57520b23-81dd-46be-bb75-178f0539302e	107,414396014065	Iz	Insediamenti agro-zootecnici	5c6224dd-dbfd-4e68-9462-cbc6c1a298bb	...
generica	e623292f-782b-42c3-a5cd-6219d4b7e911	192,318265801121	Er	Tessuto residenziale rado	f4e07176-9665-44a8-a24d-96031b5feb54	...

...	orientamento	A_disp	N_Abi	Volume [m^3]	cat
...	21,537	93	1	1402,29	8
...	-3,854	43	0	322,242	2
.. .	-18,237	77	1	1923,18	1

Table 41: *Table added to Table 39 where there are the hourly powers for each building.*

...	Month	00:00	00:15	00:30	00:45	01:00	01:15	01:30	01:45	02:00	02:15	02:30	02:45	03:00	03:15	03:30	03:45	04:00	04:15	04:30	04:45	...
...	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	...
...	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	...
...	3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	...

...	05:00	05:15	05:30	05:45	06:00	06:15	06:30	06:45	07:00	07:15	07:30	07:45	08:00	08:15	08:30	08:45	...
...	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0,4918056	...
...	0	0	0	0	0	0	0	0	0	0	0	0	0	0,500428296	1,304590313	2,139560591	...
...	0	0	0	0	0	0	0	0	0	0	0,375504742	0,985328019	1,684299016	2,433133004	3,223919018	4,045944222	...

...	09:00	09:15	09:30	09:45	10:00	10:15	10:30	10:45	11:00	11:15	11:30	11:45	...
...	1,35241294	2,238344145	3,061358831	3,835918195	4,573781651	5,276027028	5,938393424	6,554920349	7,119601432	7,627017776	8,072530422	8,452300306	...
...	2,958208013	3,770615489	4,578031629	5,373672864	6,148011763	6,891430159	7,595189912	8,251678407	8,854387148	9,39781108	9,877339007	10,28915821	...
...	4,885609828	5,729088829	6,563639352	7,377992671	8,162342138	8,90819913	9,608226295	10,25608752	10,84632502	11,37426293	11,83593358	12,22802188	...

Table 42: *Continue of table 41.*

...	12:00	12:15	12:30	12:45	13:00	13:15	13:30	13:45	14:00	14:15	14:30	14:45	...
...	8,76324983	9,003011997	9,16988425	9,262792435	9,28126585	9,225422932	9,095966992	8,894191701	8,621996537	8,281912817	7,87714109	7,411600114	...
...	10,63017839	10,89797378	11,09074053	11,20726635	11,24691022	11,20959013	11,09577807	10,9065015	10,64335136	10,30849746	9,904711944	9,435403089	...
...	12,54782381	12,7932159	12,96263304	13,05505293	13,06998569	13,00746784	12,86806005	12,65284856	12,36345027	12,00202205	11,57127488	11,07449411	...

...	15:00	15:15	15:30	15:45	16:00	16:15	16:30	16:45	17:00	17:15	17:30	17:45	...
...	6,88998539	6,31782893	5,701536631	5,048343897	4,366051009	3,662239868	2,942438087	2,206912653	1,450365895	0,695981475	0,132478392	0	...
...	8,904661325	8,317320625	7,679037998	6,996392782	6,277002897	5,529642764	4,764316696	3,992170118	3,224972598	2,473688679	1,745924488	1,04700186	...
...	10,51556738	9,899022258	9,230076387	8,514703514	7,759719193	6,972890133	6,163069654	5,340356982	4,516265864	3,703860863	2,917769395	2,173923587	...

...	18:00	10:15	18:30	18:45	19:00	19:15	19:30	19:45	20:00	20:15	20:30	20:45	21:00	21:15	21:30	21:45	...
...	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	...
...	0,416996291	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	...
...	1,489048864	0,881527601	0,38257163	0,068672559	0	0	0	0	0	0	0	0	0	0	0	0	...

...	22:00	22:15	22:30	22:45	23:00	23:15	23:30	23:45
...	0	0	0	0	0	0	0	0
...	0	0	0	0	0	0	0	0
...	0	0	0	0	0	0	0	0

Collection of scripts used in the project

In this section are reported all the pieces of code used in QGis and matlab to automate all the necessary operations. The programming languages used are Python for QGis and C for matlab. Excel VBA code snippets were also used to organize the data and construct charts for Overleaf, although these are not included in the document. To understand where and how they have been used, look at the chapter 2.

Python codes implemented on QGIS.

Algorithm for the intersection of the EDI_layer with USO_Suolo_layer without duplicating objects in the first layer.

Listing 1: *Code overlapping layers*

```
from qgis.core import *
from qgis.utils import iface
from PyQt5.QtCore import QVariant
import processing
# =========== PARAMETRI DA MODIFICARE ==========
# Specifica i nomi dei layer
input_layer1_name = 'V_edi_Podenzano'  # Nome del primo layer (target)
input_layer2_name = 'Uso_Suolo_Cabina_AC001E01166'  # Nome del secondo
    layer (join)
output_layer_name = 'Podenzano_EDI_TAG'  # Nome del layer di output
# Specifica i nomi degli attributi che vuoi estrarre dal layer 2
attributo1 = 'SIGLA'
attributo2 = 'DESCR'
# ============================================
# Ottieni i riferimenti ai layer dal progetto corrente
```

```
project = QgsProject.instance()
layer1 = project.mapLayersByName(input_layer1_name)[0]
layer2 = project.mapLayersByName(input_layer2_name)[0]
# Verifica che gli attributi esistano nel layer2
if layer2.fields().indexFromName(attributo1) == -1:
    print(f"ERRORE: L'attributo '{attributo1}' non esiste nel layer {
        input_layer2_name}")
    exit()
if layer2.fields().indexFromName(attributo2) == -1:
    print(f"ERRORE: L'attributo '{attributo2}' non esiste nel layer {
        input_layer2_name}")
    exit()
# Tieni traccia di tutti i layer temporanei creati
temp_layers = []
# Step 1: Esegui un Join by Location direttamente usando memory output
print("Esecuzione del join by location...")
result = processing.run("qgis:joinattributesbylocation", {
    'INPUT': layer1,
    'JOIN': layer2,
    'PREDICATE': [0],  # 0 = intersects
    'JOIN_FIELDS': [attributo1, attributo2],
    'METHOD': 0,  # 0 = creare una feature separata per ogni feature
        coincidente (one-to-many)
    'DISCARD_NONMATCHING': False,  # Mantieni tutte le feature
    'PREFIX': '',
    'OUTPUT': 'memory:'  # Usa memory output per evitare di creare layer
        temporanei visibili
})
join_layer = result['OUTPUT']
temp_layers.append(join_layer)
print(f"Join completato: {join_layer.featureCount()} feature trovate")
# Step 2: Crea un nuovo layer che    una copia esatta del layer1 ma con i
    campi aggiuntivi
print("Creazione del layer di output finale...")
crs = layer1.crs().authid()
geom_type = QgsWkbTypes.displayString(layer1.wkbType())
```

```
output_layer = QgsVectorLayer(f"{geom_type}?crs={crs}", output_layer_name, "memory")
# Aggiungi al nuovo layer tutti i campi del layer1
output_layer_data_provider = output_layer.dataProvider()
attributes = layer1.fields()
output_layer_data_provider.addAttributes(attributes)
# Aggiungi i due nuovi campi per gli attributi che vogliamo estrarre
fields_to_add = [
    QgsField(attributo1, QVariant.String),
    QgsField(attributo2, QVariant.String)
]
output_layer_data_provider.addAttributes(fields_to_add)
output_layer.updateFields()
# Copia tutte le feature dal layer1
features = []
for feature1 in layer1.getFeatures():
    new_feature = QgsFeature(output_layer.fields())
    new_feature.setGeometry(feature1.geometry())
    # Copia tutti gli attributi dal layer1
    for i in range(len(layer1.fields())):
        new_feature[i] = feature1[i]
    features.append(new_feature)
# Aggiungi tutte le feature al layer di output
output_layer_data_provider.addFeatures(features)
print(f"Layer di output creato con {output_layer.featureCount()} feature")
# Step 3: Prepara un dizionario per memorizzare gli attributi da join per ogni feature del layer1
print("Preparazione dell'aggregazione degli attributi...")
# Trova un campo identificativo univoco nel layer1 (preferibilmente l'ID originale)
id_field_name = 'id'  # Predefinito, ma cerchiamo un campo pi  adatto
for field in layer1.fields():
    if 'id' in field.name().lower() or 'fid' in field.name().lower():
        id_field_name = field.name()
        break
print(f"Utilizzo del campo '{id_field_name}' come identificatore")
# Crea il dizionario per memorizzare i valori aggregati
```

```
aggregated_values = {}
# Indice del campo ID nel layer di join
id_idx = join_layer.fields().indexFromName(id_field_name)
attr1_idx = join_layer.fields().indexFromName(attributo1)
attr2_idx = join_layer.fields().indexFromName(attributo2)
# Se gli indici non sono validi, cerchiamo campi alternativi
if id_idx == -1:
    print("Campo ID non trovato nel layer di join, utilizzo dell'ID interno")
    # Usa l'ID interno di QGIS
    for feature in join_layer.getFeatures():
        fid = feature.id()
        if fid not in aggregated_values:
            aggregated_values[fid] = {attributo1: set(), attributo2: set()}
        # Aggiungi i valori agli insiemi (set) per evitare duplicati
        if attr1_idx != -1 and feature[attr1_idx]:
            aggregated_values[fid][attributo1].add(str(feature[attr1_idx]))
        if attr2_idx != -1 and feature[attr2_idx]:
            aggregated_values[fid][attributo2].add(str(feature[attr2_idx]))
else:
    # Usa il campo ID trovato
    for feature in join_layer.getFeatures():
        feature_id = feature[id_idx]
        if feature_id not in aggregated_values:
            aggregated_values[feature_id] = {attributo1: set(), attributo2:
                set()}

        # Aggiungi i valori agli insiemi (set) per evitare duplicati
        if attr1_idx != -1 and feature[attr1_idx]:
            aggregated_values[feature_id][attributo1].add(str(feature[
                attr1_idx]))
        if attr2_idx != -1 and feature[attr2_idx]:
            aggregated_values[feature_id][attributo2].add(str(feature[
                attr2_idx]))
print(f"Aggregati attributi per {len(aggregated_values)} feature")
# Step 4: Aggiorna il layer di output con i dati aggregati
print("Aggiornamento del layer di output con i valori aggregati...")
```

```
output_attr1_idx = output_layer.fields().indexFromName(attributo1)
output_attr2_idx = output_layer.fields().indexFromName(attributo2)
output_layer.startEditing()
for feature in output_layer.getFeatures():
    # Ottieni l'ID della feature
    if id_idx != -1:
        feature_id = feature[id_field_name]
    else:
        feature_id = feature.id()
    # Se abbiamo dati aggregati per questa feature
    if feature_id in aggregated_values:
        # Converti i set in liste e unisci con virgole
        values1 = aggregated_values[feature_id][attributo1]
        values2 = aggregated_values[feature_id][attributo2]
        if values1:
            output_layer.changeAttributeValue(feature.id(),
                output_attr1_idx, ", ".join(values1))
        if values2:
            output_layer.changeAttributeValue(feature.id(),
                output_attr2_idx, ", ".join(values2))
output_layer.commitChanges()
# Step 5: Pulisci TUTTI i layer temporanei e aggiungi il layer di output al
    progetto
print("Pulizia dei layer temporanei...")
# Rimuovi i layer temporanei che abbiamo creato esplicitamente
for temp_layer in temp_layers:
    QgsProject.instance().removeMapLayer(temp_layer.id())
# Verifica e rimuovi anche eventuali altri layer temporanei creati
    indirettamente
# Cerca tutti i layer che hanno nomi temporanei generati da Processing
all_layers = QgsProject.instance().mapLayers().values()
for layer in all_layers:
    layer_name = layer.name()
    if (layer_name.startswith('Joined layer') or
        layer_name.startswith('OUTPUT') or
        'memory' in layer_name or
        'temp' in layer_name.lower() or
```

```
146         'output' in layer_name.lower()):
147         if layer.id() != output_layer.id():  # Non rimuovere il nostro
                layer di output
148             print(f"Rimozione layer temporaneo: {layer_name}")
149             QgsProject.instance().removeMapLayer(layer.id())
150 # Aggiungi il layer di output al progetto
151 QgsProject.instance().addMapLayer(output_layer)
152 print(f"Processo completato. Creato un nuovo layer: {output_layer_name} con
      {output_layer.featureCount()} geometrie.")
153 print(f"Sono stati aggiunti due nuovi attributi: '{attributo1}' e '{
      attributo2}'")
154 print("Gli attributi contengono tutti i valori aggregati delle feature del
      layer2 che intersecano le feature del layer1.")
155 print("Tutti i layer temporanei sono stati eliminati.")
```

Algorithm for calculating the azimuth angles of each shape of each structure contained in the FAB_layer. The algorithm is structured as follow:

- creation of dialogue interface and request of input layer (FAB_layer) and two output layer;
- load items from input layer;
- azimuth angle computation;
- creation of the layer containing angles and the layer containing arrows.

Listing 2: *Code computing azimuth angle of buildings*

```
1 from qgis.PyQt.QtCore import QCoreApplication, QVariant
2 from qgis.core import (QgsProcessing,
3                        QgsProcessingAlgorithm,
4                        QgsProcessingException,
5                        QgsProcessingParameterFeatureSource,
6                        QgsProcessingParameterFeatureSink,
7                        QgsProcessingParameterNumber,
8                        QgsField,
9                        QgsFeature,
10                       QgsMapLayer,
```

```
                       QgsGeometry,
                       QgsPointXY,
                       QgsProject,
                       QgsVectorLayer,
                       QgsSingleSymbolRenderer,
                       QgsLineSymbol,
                       QgsSymbolLayer,
                       QgsProcessingUtils,
                       QgsMarkerLineSymbolLayer,
                       QgsFeatureSink,
                       QgsSymbol,
                       QgsWkbTypes)
from PyQt5.QtCore import QVariant
from PyQt5.QtGui import QColor
import math
import numpy as np

class AnalisiEdificiConArrays(QgsProcessingAlgorithm):
    # Definizione dei parametri
    INPUT_LAYER = 'INPUT_LAYER'
    OUTPUT_LAYER = 'OUTPUT_LAYER'
    ARROWS_LAYER = 'ARROWS_LAYER'
    ARROW_LENGTH = 'ARROW_LENGTH'
    def tr(self, string):
        return QCoreApplication.translate('Processing', string)
    def createInstance(self):
        return AnalisiEdificiConArrays()
    def name(self):
        return 'analisi_edifici_con_arrays'
    def displayName(self):
        return self.tr('Analisi Edifici con Arrays')
    def group(self):
        return self.tr('Analisi Edifici')
    def groupId(self):
        return 'analisi_edifici'
    def shortHelpString(self):
        return self.tr('Script che utilizza arrays per manipolare i dati
```

```
            degli edifici.')
    def initAlgorithm(self, config=None):
        # Parametri di input
        self.addParameter(
            QgsProcessingParameterFeatureSource(
                self.INPUT_LAYER,
                self.tr('Layer principale (EI)'),
                [QgsProcessing.TypeVectorPolygon]
            )
        )
        # Parametro per la lunghezza delle frecce
        self.addParameter(
            QgsProcessingParameterNumber(
                self.ARROW_LENGTH,
                self.tr('Lunghezza delle frecce'),
                QgsProcessingParameterNumber.Double,
                10.0,  # valore di default
                False,  # non obbligatorio
                0.1     # valore minimo
            )
        )
        # Output per gli edifici con orientamento
        self.addParameter(
            QgsProcessingParameterFeatureSink(
                self.OUTPUT_LAYER,
                self.tr('Layer edifici con orientamento')
            )
        )
        # Output per le frecce
        self.addParameter(
            QgsProcessingParameterFeatureSink(
                self.ARROWS_LAYER,
                self.tr('Layer frecce di orientamento')
            )
        )
    def normalizza_angolo_90(self, angolo):
        """
```

```
84          Normalizza un angolo nell'intervallo -90 a +90 gradi
85          Args:
86              angolo (float): Angolo in gradi
87          Returns:
88              float: Angolo normalizzato tra -90 e +90 gradi
89          """
90          # Prima normalizza tra 0 e 180
91          angolo = angolo % 180
92          # Poi, se l'angolo    maggiore di 90, sottrarre 180 per ottenere il
                valore nel range -90 a +90
93          if angolo > 90:
94              angolo = angolo - 180
95          return angolo
96
97      def calcola_angolo_lato_pi_lungo(self, vertici):
98          """
99          Calcola l'angolo del lato p i  lungo di un poligono
100         Args:
101             vertici (list): Lista di QgsPointXY rappresentanti i vertici
                   del poligono
102         Returns:
103             float: L'angolo in gradi (0-180)
104         """
105         # Rimuovi l'ultimo punto (duplicato del primo)
106         vertici = vertici[:-1]
107         max_lunghezza = 0
108         angolo_max = 0
109         # Calcola la lunghezza di ogni lato e trova il p i  lungo
110         for i in range(len(vertici)):
111             p1 = vertici[i]
112             p2 = vertici[(i + 1) % len(vertici)]
113             dx = p2.x() - p1.x()
114             dy = p2.y() - p1.y()
115             lunghezza = math.sqrt(dx*dx + dy*dy)
116             if lunghezza > max_lunghezza:
117                 max_lunghezza = lunghezza
118                 # Calcola l'angolo rispetto all'asse x (in radianti)
```

```
119                 angolo = math.atan2(dy, dx)
120                 # Converti in gradi
121                 angolo_gradi = math.degrees(angolo)
122 
123                 # Normalizza tra -90 e +90
124                 angolo_max = self.normalizza_angolo_90(angolo_gradi)
125         return angolo_max
126 
127     def calcola_angolo_autovettore(self, vertici):
128         """
129         Calcola l'angolo dell'autovettore principale della matrice di
                covarianza
130         Args:
131             vertici (list): Lista di QgsPointXY rappresentanti i vertici
                    del poligono
132         Returns:
133             float: L'angolo in gradi (0-180)
134         """
135         # Converti i vertici in array numpy
136         punti = np.array([[v.x(), v.y()] for v in vertici[:-1]])  # Escludi
                l'ultimo punto (duplicato)
137         # Calcola il centroide
138         centroide = np.mean(punti, axis=0)
139         # Centra i punti
140         punti_centrati = punti - centroide
141         # Calcola la matrice di covarianza
142         cov_matrix = np.cov(punti_centrati.T)
143         # Calcola gli autovalori e gli autovettori
144         autovalori, autovettori = np.linalg.eigh(cov_matrix)
145         # L'autovettore principale    quello associato all'autovalore
                massimo
146         idx_max = np.argmax(autovalori)
147         autovettore_principale = autovettori[:, idx_max]
148         # Calcola l'angolo dell'autovettore principale (in radianti)
149         angolo = math.atan2(autovettore_principale[1],
                autovettore_principale[0])
150         # Converti in gradi e normalizza tra -90 e +90
```

```
151         angolo_gradi = math.degrees(angolo)
152         return self.normalizza_angolo_90(angolo_gradi)
153 
154     def calcola_angolo_orientamento(self, geometria):
155         """
156         Calcola l'angolo di orientamento di una geometria
157         Args:
158             geometria (QgsGeometry): La geometria dell'edificio
159         Returns:
160             float: L'angolo di orientamento in gradi (0-180)
161         """
162         # Estrai i vertici della geometria
163         if geometria.isMultipart():
164             # Per geometrie multipart, prendi la prima parte
165             vertici = geometria.asMultiPolygon()[0][0]
166         else:
167             vertici = geometria.asPolygon()[0]
168         num_vertici = len(vertici) - 1  # -1 perch  l'ultimo punto
                uguale al primo
169         if num_vertici == 4:  # Edificio con 4 vertici (quadrato,
                rettangolo, ecc.)
170             return self.calcola_angolo_lato_pi_lungo(vertici)
171         else:  # Edificio con pi  di 4 vertici
172             return self.calcola_angolo_autovettore(vertici)
173 
174     def crea_freccia(self, punto_centrale, angolo, lunghezza):
175         # Converti l'angolo in radianti
176         angolo_rad = math.radians(angolo)
177         # Calcola il punto finale della freccia
178         x_fine = punto_centrale.x() + lunghezza * math.cos(angolo_rad)
179         y_fine = punto_centrale.y() + lunghezza * math.sin(angolo_rad)
180         punto_finale = QgsPointXY(x_fine, y_fine)
181         # Crea la geometria della linea
182         return QgsGeometry.fromPolylineXY([punto_centrale, punto_finale])
183 
184     def processAlgorithm(self, parameters, context, feedback):
185             # Caricamento del layer di input
```

```
186         source = self.parameterAsSource(parameters, self.INPUT_LAYER,
                context)
187         if source is None:
188             raise QgsProcessingException(self.tr('Layer di input non valido
                ''))
189         # Ottieni la lunghezza delle frecce
190         lunghezza_freccia = self.parameterAsDouble(parameters, self.
                ARROW_LENGTH, context)
191         # Prepara i campi del layer di output (stessi campi dell'input +
                orientamento)
192         fields = source.fields()
193         fields.append(QgsField("orientamento", QVariant.Double))
194         # Prepara il layer di output per gli edifici
195         (edifici_sink, edifici_dest_id) = self.parameterAsSink(
196             parameters,
197             self.OUTPUT_LAYER,
198             context,
199             fields,
200             source.wkbType(),
201             source.sourceCrs()
202             #QgsFeatureSink.RegeneratePrimaryKey,
203             #output_edifici_name
204         )
205         # Crea un layer temporaneo per ottenere i fields
206         temp_layer = QgsVectorLayer("LineString?crs=" + source.sourceCrs().
                authid(), "temp", "memory")
207         temp_provider = temp_layer.dataProvider()
208         temp_provider.addAttributes([
209             QgsField("edificio_id", QVariant.Int),
210             QgsField("orientamento", QVariant.Double)
211         ])
212         temp_layer.updateFields()
213         # Usa i campi del layer temporaneo
214         (frecce_sink, frecce_dest_id) = self.parameterAsSink(
215             parameters,
216             self.ARROWS_LAYER,
217             context,
```

```
218             temp_layer.fields(),
219             QgsWkbTypes.LineString,
220             source.sourceCrs()
221             #QgsFeatureSink.RegeneratePrimaryKey,
222             #output_frecce_name
223         )
224         if edifici_sink is None or frecce_sink is None:
225             raise QgsProcessingException(self.tr('Impossibile creare i
                layer di output'))
226         # Contatore di progresso
227         total = 100.0 / source.featureCount() if source.featureCount() else
                0
228         # Raccogli le features prima di aggiungerle (processo batch)
229         edifici_features = []
230         frecce_features = []
231         # Processa ogni feature
232         for current, feature in enumerate(source.getFeatures()):
233             # Verifica se l'elaborazione    stata annullata
234             if feedback.isCanceled():
235                 break
236             # Calcola l'orientamento
237             orientamento = self.calcola_angolo_orientamento(feature.
                geometry())
238             # Crea una nuova feature per il layer degli edifici
239             edificio_feature = QgsFeature(fields)
240             edificio_feature.setGeometry(feature.geometry())
241             # Copia gli attributi esistenti
242             for i in range(len(feature.attributes())):
243                 edificio_feature[i] = feature[i]
244             # Aggiungi l'attributo orientamento
245             edificio_feature["orientamento"] = orientamento
246             # Aggiungi alla lista di edifici
247             edifici_features.append(edificio_feature)
248             # Crea una feature per la freccia
249             freccia_feature = QgsFeature(temp_provider.fields())
250             centroide = feature.geometry().centroid().asPoint()
251             freccia_geometry = self.crea_freccia(centroide, orientamento,
```

```
                lunghezza_freccia)
252             freccia_feature.setGeometry(freccia_geometry)
253             freccia_feature.setAttribute("edificio_id", current)
254             freccia_feature.setAttribute("orientamento", orientamento)
255             # Aggiungi alla lista di frecce
256             frecce_features.append(freccia_feature)
257             # Aggiorna il progresso
258             feedback.setProgress(int(current * total))
259         # Aggiungi tutte le features in una volta sola (pi  efficiente)
260         edifici_sink.addFeatures(edifici_features, QgsFeatureSink.
                FastInsert)
261         frecce_sink.addFeatures(frecce_features, QgsFeatureSink.FastInsert)
262         # Alla fine del tuo metodo processAlgorithm
263         frecce_layer = QgsProcessingUtils.mapLayerFromString(frecce_dest_id
                , context)
264         if frecce_layer:
265             self.imposta_stile_frecce(frecce_layer)
266         # Restituisci i riferimenti ai layer creati
267         return {
268             self.OUTPUT_LAYER: edifici_dest_id,
269             self.ARROWS_LAYER: frecce_dest_id
270         }
```

Algorithm that calculates the number of PODs for residential buildings starting from the EDI_layer and using the UVL_layer.

Listing 3: *Calculation of PODs*

```
1  from qgis.PyQt.QtCore import QCoreApplication
2  from qgis.core import (QgsProcessing,
3                         QgsFeatureSink,
4                         QgsProcessingException,
5                         QgsProcessingAlgorithm,
6                         QgsProcessingParameterFeatureSource,
7                         QgsProcessingParameterFeatureSink,
8                         QgsProcessingParameterField,
9                         QgsField,
10                        QgsFeature)
11 from PyQt5.QtCore import QVariant
```

```
12 class AnalisiEdificiConArrays(QgsProcessingAlgorithm):
13     # Definizione dei parametri
14     INPUT_LAYER_1 = 'INPUT_LAYER_1'
15     INPUT_LAYER_2 = 'INPUT_LAYER_2'
16     ID_E = 'id_e'
17     EDI_ID_E = 'EDI_ID_E'
18     TY_EDI = 'D_TY_EDI'
19     SIGLA = 'SIGLA'
20     AREA_P='Area_Pz'
21     H_UVL='H_UVL'
22     OUTPUT = 'OUTPUT'
23     def tr(self, string):
24         return QCoreApplication.translate('Processing', string)
25     def createInstance(self):
26         return AnalisiEdificiConArrays()
27     def name(self):
28         return 'analisi_edifici_con_arrays'
29     def displayName(self):
30         return self.tr('Analisi Edifici con Arrays')
31     def group(self):
32         return self.tr('Analisi Edifici')
33     def groupId(self):
34         return 'analisi_edifici'
35     def shortHelpString(self):
36         return self.tr('Script che utilizza arrays per manipolare i dati
               degli edifici.')
37     def initAlgorithm(self, config=None):
38         # Parametri di input
39         #blocchi self.parameter(...
40
41     def processAlgorithm(self, parameters, context, feedback):
42         # Caricamento dei layer
43         source1 = self.parameterAsSource(parameters, self.INPUT_LAYER_1,
               context)
44         source2 = self.parameterAsSource(parameters, self.INPUT_LAYER_2,
               context)
45         if source1 is None or source2 is None:
```

```
46              raise QgsProcessingException(self.tr('Layer di input non validi
                    '))
47          fields = source1.fields()
48          fields.append(QgsField('Abitazioni', QVariant.Int))
49          fields.append(QgsField('Volume', QVariant.Double))
50          (sink, dest_id) = self.parameterAsSink(
51              parameters,
52              self.OUTPUT,
53              context,
54              fields,
55              source1.wkbType(),
56              source1.sourceCrs()
57          )
58          if sink is None:
59              raise QgsProcessingException(self.tr('Impossibile creare il
                    layer di output'))
60          # Contatore di progresso
61          total = 100.0 / source1.featureCount() if source1.featureCount()
                else 0
62          # Elaborazione di ogni feature del layer principale
63          for current, feature1 in enumerate(source1.getFeatures()):
64              if feedback.isCanceled():
65                  break
66              # Creazione di un array con tutti i valori dei campi della
                    feature corrente
67              dati_feature1 = {}
68              for field in source1.fields().names():
69                  dati_feature1[field] = feature1[field]
70              # Recupero dei valori chiave
71              id_edi = dati_feature1['ID_E']
72              ty_edi = dati_feature1['D_TY_EDI']
73              sigla = dati_feature1['SIGLA']
74              # Creazione feature di output
75              out_feature = QgsFeature(fields)
76              out_feature.setGeometry(feature1.geometry())
77              # Copia gli attributi esistenti dal layer originale
78              attributes = feature1.attributes()
```

```
79              # Aggiungi un valore iniziale per il nuovo campo 'Abitazioni'
80              attributes.append(0)  # Valore predefinito 0
81              attributes.append(0)  # Valore predefinito 0
82              out_feature.setAttributes(attributes)
83              #out_feature.setAttributes(feature1.attributes())
84              # Verifica delle condizioni
85              if ty_edi == 'Villetta a Schiera' or sigla == 'Vv':
86                  # Prima condizione verificata
87                  out_feature.setAttribute('Abitazioni',1)
88                  # Crea un array di feature del layer secondario che
                        corrispondono al criterio
89                  volumetrie = []
90                  #calcoliamo la volumetria in m3
91                  for feature2 in source2.getFeatures():
92                      if feature2['EDI_ID_E'] == id_edi:
93                          # Crea un dizionario con tutti i valori dei campi
                                della feature
94                          dati_feature2 = {}
95                          for field in source2.fields().names():
96                              dati_feature2[field] = feature2[field]
97                          # Aggiungi il dizionario all'array
98                          volumetrie.append(dati_feature2)
99                  out_feature.setAttribute('Volume',self.volume(volumetrie))
100             elif self.verifica_chiavi(sigla,['Es', 'Er', 'Ed','Ec']):
101                 # Seconda condizione verificata
102                 # Crea un array di feature del layer secondario che
                        corrispondono al criterio
103                 volumetrie = []
104                 for feature2 in source2.getFeatures():
105                     if feature2['EDI_ID_E'] == id_edi:
106                         # Crea un dizionario con tutti i valori dei campi
                                della feature
107                         dati_feature2 = {}
108                         for field in source2.fields().names():
109                             dati_feature2[field] = feature2[field]
110                         # Aggiungi il dizionario all'array
111                         volumetrie.append(dati_feature2)
```

```
112                 risultato = self.calcolo_personalizzato(dati_feature1,
                        volumetrie)
113                 out_feature.setAttribute('Abitazioni',risultato) #risultato
114                 #calcolo volume
115                 out_feature.setAttribute('Volume',self.volume(volumetrie))
116             else:
117                 # Nessuna condizione verificata
118                 out_feature.setAttribute('Abitazioni',0)
119                 # Crea un array di feature del layer secondario che
                        corrispondono al criterio
120                 volumetrie = []
121                 #calcoliamo la volumetria in m3
122                 for feature2 in source2.getFeatures():
123                     if feature2['EDI_ID_E'] == id_edi:
124                         # Crea un dizionario con tutti i valori dei campi
                                della feature
125                         dati_feature2 = {}
126                         for field in source2.fields().names():
127                             dati_feature2[field] = feature2[field]
128                         # Aggiungi il dizionario all'array
129                         volumetrie.append(dati_feature2)
130                 out_feature.setAttribute('Volume',self.volume(volumetrie))
131             # Aggiunta della feature al layer di output
132             sink.addFeature(out_feature, QgsFeatureSink.FastInsert)
133             # Aggiornamento della barra di progresso
134             feedback.setProgress(int(current * total))
135         return {self.OUTPUT: dest_id}
136     #funzione per verificare che almeno una chiave sia contenuta nella
            classificazione dell'edificio'
137     def verifica_chiavi(self,sigla, chiavi):
138         # Verifica con early return (interruzione al primo match)
139         for chiave in chiavi:
140             if chiave in sigla:
141                 return True
142         return False
143     #funzione calcolo volumetria
144     def volume(self,volumetrie):
```

```
145         volu=0
146         #per ogni parte calcolo il volume di ogni parziale e restituisco il
                volume totale
147         for vol in volumetrie:
148             volu+=vol['Area_Pz']*vol['H_UVL']
149         return volu
150     def calcolo_personalizzato(self, edificio, volumetrie):
151         #identifico quanti appartamenti esistono nell'edificio oppure se e'
                una casa unifamiliare o se un "magazzino"
152         #l'edifico e diviso in piu blocchi'
153         if len(volumetrie)>1:
154             risultato=0
155             if edificio['Sup_Suolo']<350:
156                 risultato=1
157             else:
158                 for vol in volumetrie:
159                     ab=0
160                     #guardo quanti piani ha, tengo conto della misura di 3.
                        5m come valore per piano
161                     piani=vol['H_UVL']//3.5
162                     #se e' sotto il valore di 3.5 lo considero come
                        magazzino
163                     if piani== 0:
164                         ab=0
165                     else:
166                         #se minore di 80m2 e' magazzino
167                         if vol['Area_Pz'] < 80 and piani==1:
168                             ab=0
169                         #se superficie contenuta considero un appartamento
                            per piano
170                         elif vol['Area_Pz'] < 200 :
171                             ab = piani
172                         #se sopra i 200m2 lo considero come serie di
                            appartamenti
173                         else:
174                             risultato= (vol['Area_Pz']-12)//110 * piani
175                     risultato+=ab
```

```
176         #l'edificio e' formato da una sola unita volumetrica quindi ottimo'
177         elif len(volumetrie)==1 :
178             for vol in volumetrie:
179                 #guardo quanti piani ha, tengo conto della misura di 3.5m
                        come valore per piano
180                 piani=vol['H_UVL']//3.5
181                 #se e' sotto il valore di 3.5 lo considero come magazzino
182                 if piani== 0:
183                     risultato=0
184                 else:
185                     #se minore di 80m2 e' magazzino
186                     if vol['Area_Pz'] < 60:
187                         risultato=0
188                     #se superficie contenuta e non e' su tanti piani
                            massimo due e' una singola casa
189                     elif vol['Area_Pz'] < 200 and piani<3:
190                         risultato = 1
191                     #se sotto 350m2 lo considero come una casa per piano
                            molto grande
192                     elif vol['Area_Pz'] < 250:
193                         risultato = piani
194                     #se sopra 350m2 lo considero come un condominio tolgo
                            la metratura approssimata di una scala condominiale
                            e divido per il numero di m2 medio di una casa
195                     elif vol['Area_Pz'] >= 250:
196                         risultato= (vol['Area_Pz']-12)//110 * piani
197                     else:
198                         risultato=89
199                         return
200         else:
201             risultato=59
202         return risultato
```

C codes implemented on MatLab.

Algorithm for determining the production curve of buildings, consists of two main functions, the first for calculating the 6 components of radiation and the second for calculating the production of each building. It returns an xlsx file with the data obtained.

Listing 4: *Code for production profiles*

```
%importiamo il file excel
file_excel='E:\01_Electrical Engineering 2022-20xx\Progetto CER Tirocinio Formativo 2024\01_Analisi Tesi Magistrale\Fogli Excel\Produzione Comuni Interesse\San_Giorgio_P_EI.xlsx';
%leggiamo i dati di excel
[data, testo] = xlsread(file_excel);
% inizializzamo il vettore delle potenze
DatiEDImp=zeros(size(data,1),2);
%dati FV;
    %dati prima tipologia di pannelli
DatiFV=[1.724*1.134,0.21,0.41,0,0.41,4];
    %dati Seconda tipologia di pannelli
%DatiFV=[2.185*1.098,0.212,0.51,0,0.51,4];
    %dati Terza tipologia di pannelli
%DatiFV=[1.564*1.144,0.198,0.355,0,0.41,4];

%lista Coordinate comuni
    %San Giorgio
PosizioneComune=[44.96,9.752];
    %Pontenure
%PosizioneComune=[44.99,9.79];
    %Podenzano
%PosizioneComune=[44.98,9.704];
    %Piacenza
%PosizioneComune=[45.027,9.756];
    %Caorso
%PosizioneComune=[45.039,9.829];

% e1 generale per tutti i valori
e1=0.0008;
```

```
%solo per il valore di NubiSparse
e2=0.0005;
DatiAnalisi=[67,15/60,1];
DatiED=[35,2,3,0.3,0,4];
GiornoAnalisi=[15,46,74,105,135,166,196,227,258,288,319,349];

%matrice dA ESPORTARE
Testo_exp=cell(12*(size(testo,1)-1)+1,size(testo,2));
Dati_exp=zeros(12*size(data,1),size(data,2)+1+24/DatiAnalisi(2));
%variabile di scrittura
s=1;
%Copio intestazione
Testo_exp(s,:)=testo(s,:);

% Calcoliamo la potenza prodotte per i 12 mesi
for i = 1:12 %da gennaio a dicembre
    %correggo il giorno di analisi per ogni mese
    DatiAnalisi(1)=GiornoAnalisi(i);
    %calcolo le potenze producibili per i 12 mesi da ogni impianto
    DatiSole=SolarRad_Position(PosizioneComune,DatiAnalisi);
    %per ogni edificio calcolo quanto produce
    for j=1:size(data,1)
        %DatiFV=[ AreaPfv RendimentoPfv kWpPfv IminPfv PmaxPfv nPFv]
        % DatiEd=[Slope Azimuth AreaTetto r=coefRiflessione fcFV]
        DatiED(2)=data(j,9);
        DatiFV(6)=fix(data(j,10)/DatiFV(1)*cos(deg2rad(DatiED(1))));
        Testo_exp(s+1,:)=testo(j+1,:);
        Dati_exp(s,1:10)=data(j,1:10);
        %scrivo il conto di pannelli fv giusti e la potenza globale
        %dell'impianto
        Dati_exp(s,11)=DatiFV(6);
        Dati_exp(s,12)=DatiFV(6)*DatiFV(3);
        Dati_exp(s,13)=i;
        Dati_exp(s,14:end)=Power_EDI_Sereno(DatiED,DatiSole,DatiFV);
        s=s+1;
    end
end
```

```
%esportiamo il file
file_export='E:\01_Electrical Engineering 2022-20xx\Progetto CER Tirocinio
    Formativo 2024\01_Analisi Tesi Magistrale\Fogli Excel\Produzione Comuni
    Interesse\San_Giorgio_P_EI_Pmax_1.xlsx';
%creaimo il file con le tag precedentemente estratte dal file originale e
%di conseguenza aggiungiamo i dati numerici partendo dalla cella C2 che
%la prima cella libera

writecell(Testo_exp,file_export);
writematrix(Dati_exp(:,1:2),file_export,'range','A2');
writematrix(Dati_exp(:,5),file_export,'range','E2');
writematrix(Dati_exp(:,9:end),file_export,'range','I2');
```

Function for calculating radiation.

Listing 5: *Function SolarRad_Position*

```

function srad = SolarRad_Position(position,DatiAnalisi)
%% dati in ingresso
% position[lat,long] del luogo
% DatiAnalisi=[day0,n,nday]
%day0 giorno di partenza dell'analisi
%n ogni quanti minuti viene fatta e nday per quanti giorni
day0=DatiAnalisi(1);
n=DatiAnalisi(2);
nday=DatiAnalisi(3);
%% Data Fixed
tau_a= 365; %giorni dell'anno
dayend=day0+nday; %guardo una settimana quindi finisco il giorno
S0 = 1367;          % solar constant W m^-2   default 1367
dr= 0.0174532925;   % degree to radians conversion factor
%creo il cubo che passer  ad ogni edificio
srad=zeros(nday,24*1/n,5);
%% Convert data in radious
Lat=position(1)*dr;
Lon=position(2)*dr;
fcirc=360*dr;% 360 degrees in radians
%% some setup calculations
```

```
23 sinL=sin(Lat);
24 cosL=cos(Lat);
25 %% loop over day
26 i=1;
27 for d = day0:1:dayend-1
28     % clear sky solar radiation
29     I0 = S0 * (1 + 0.0344*cos(fcirc*d/tau_a)); % extraterr rad per day
30     % sun declination dS
31     dS = 23.45 * dr* sin(fcirc * ( (284+d)/tau_a ) ); %in radians, correct/
           verified
32     %% equazioni del tempo
33     % in rad coef
34     B=deg2rad((360/365)*(d-1));
35     % minuti orari che differiscono l'ra locale dal orario vero del sole
36     E=(0.000075+0.001868*cos(B)-0.032077*sin(B)-0.014615*cos(2*B)-0.04089*
           sin(2*B))*229.2;
37     %%  daily loop
38     h=1;
39     for t=0:n:23 % loop day hours
40         %calcoliamo Ora vera sole
41         Svh=t+(4*(15-Lon)+E)/60;
42         % hourangle of sun hs  in radianti
43         hs=15*(Svh-12)*dr;                % hs(t)
44         %solar angle and azimuth
45         alpha = asin(sinL*sin(dS)+cosL*cos(dS)*cos(hs));% solar altitude
               angle
46         sinAlpha = sinL*sin(dS)+cosL*cos(dS)*cos(hs);
47         if asin(sinAlpha)>0
48             azi_s = asin(cos(dS)*sin(hs)/cos(alpha)); % solar azimuth angle
49             %Angolo di incidenza fra radiazione solare e superficie
                   comunque
50             %inclinata
51             %cos_i = (sin(dS)*term1) + (cos(dS)*cos(hs)*term2) + (cos(dS)*
                   term3*sin(hs));
52             t1=azi_s;
53             t2=sinAlpha;
54             t3=cos(alpha);
```

```
55          else
56              t1=0;
57              t2=0;
58              t3=0;
59          end
60          % correction  using atmospheric transmissivity taub_b
61          M=sqrt(1229+((614*sinAlpha)).^2)-614*sinAlpha; % Air mass ratio
62          tau_b = 0.56 * (exp(-0.65*M) + exp(-0.095*M));
63          tau_d = 0.271-0.294*tau_b; % radiation diffusion coefficient for
                diffuse insolation
64          tau_r = 0.271+0.706*tau_b; % reflectance transmitivity
65          Is = I0 * tau_b; % potential incoming shortwave radiation at
                surface normal (equator)
66          Id = I0* tau_d/ 2*sinAlpha; %diffuse radiation;
67          Ir = I0*  tau_r/ 2* sinAlpha; % reflectance
68          Is(Is<0)=0;
69          Id(Id<0)=0;
70          Ir(Ir<0)=0;
71          %creo matrice contenente le componenti di radiazione Diretta,
72          %Riflessa e Diffusa che varieranno per ogni impianto
73          srad(i,h,1) = t1;
74          srad(i,h,2) =t2;
75          srad(i,h,3) =t3;
76          srad(i,h,4) =Is;
77          srad(i,h,5) =Id;
78          srad(i,h,6) =Ir;
79          h=h+1;
80       end % end of sun hours in day loop
81  %%  add up radiation part melt for every day
82      i=i+1;
83  end
```

Function for calculating the powers of each building.

Listing 6: *Function Power_Ris*

```
1  function Power_Ris= Power_EDI(DatiEd,DatiMeteo,DatiSole,DatiFV)
2      %% DatiFV
3      %Specifiche dati FV per calcolo potenza
```

```
4   %% Elaborazione DatiED in ingresso
5   dr= 0.0174532925;   % degree to radians conversion factor
6   slope_R=DatiEd(1)*dr;
7   azi_R=DatiEd(2)*dr;
8   Area=DatiEd(3);
9   r=DatiEd(4);
10  fcFV=DatiEd(5);
11  cosSlope=cos(slope_R);
12  sinSlope=sin(slope_R);
13  %% Dati Impianto FV
14  AreaPfv=DatiFV(1);
15  RendimentoPfv=DatiFV(2);
16  kWpPfv=DatiFV(3);
17  IminPfv=DatiFV(4);
18  PmaxPfv=DatiFV(5);
19  nPFv=DatiFV(6);
20  %variabili di Processo
21  cosi=zeros(1,size(DatiSole,2));
22  Rad=zeros(2,size(DatiSole,2));
23  Power=null(2,0);
24  %controllo se hanno a disposizione un area minima
25  if nPFv==0
26      Power_Ris=0;
27      return
28  end
29  %% Creazione Vettore Potenze
30  for i=1:1:size(DatiSole,1)
31      cosi=DatiSole(i,:,3).*cos(DatiSole(i,:,1)-azi_R)*sinSlope +
            DatiSole(i,:,2)*cosSlope;
32      cosi(cosi<0)=0;
33      Rad(1,:)=DatiSole(i,:,4).*cosi + DatiSole(i,:,5)*cosSlope^2 +
            DatiSole(i,:,6)*r*sinSlope^2;
34      Rad(2,:)=Rad(1,:).*DatiMeteo(i,:); %[w/m2]
35      Rad(1,:)=Rad(1,:)*(nPFv*AreaPfv*RendimentoPfv)/1000; %[kW]
            Produzione Imp Sereno
36      %calcolo potenza prodotta dall'impianto
37      Rad(2,:)=Rad(2,:)*(nPFv*AreaPfv*RendimentoPfv)/1000;  %[kW]
```

```
            Produzione Imp Meteo
        Rad(Rad(2,:)>nPFv*PmaxPfv)=nPFv*PmaxPfv;
        Power=[Power,Rad];
    end
    Power_Ris=Power;
end
```

Function for creation of consumption database as in table 40 + 41.

Listing 7: *Function Consumption calculation*

```
nome={'SGP','Pontenure','Caorso','Piacenza','Podenzano'};
consumi = readtable('E:\01_Electrical Engineering 2022-20xx\Progetto CER
    Tirocinio Formativo 2024\01_Analisi Tesi Magistrale\Fogli Excel\
    Consumi_stima\Dati_interesse_me.xlsx');
for com=1:5
    clearvars -except com nome consumi;
    nome_comune=nome(com);
% Import data from Excel files
comune = readtable(string(strcat('E:\01_Electrical Engineering 2022-20xx\
    Progetto CER Tirocinio Formativo 2024\01_Analisi Tesi Magistrale\Fogli
    Excel\Consumi_stima\',nome_comune,'.xlsx')));
%consumi = readtable('E:\01_Electrical Engineering 2022-20xx\Progetto CER
    Tirocinio Formativo 2024\01_Analisi Tesi Magistrale\Fogli Excel\
    Consumi_stima\Dati_interesse_me.xlsx');
% Get number of rows in the comune table
numRows = height(comune);
giornimese=[31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31];
% Create new Excel workbook for results
resultsFileName = string(strcat('E:\01_Electrical Engineering 2022-20xx\
    Progetto CER Tirocinio Formativo 2024\01_Analisi Tesi Magistrale\Fogli
    Excel\Consumi_stima\risultati_',nome_comune,'.xlsx'));
% Initialize arrays to store results - 8 categories like in the first
    script
consumiTot = cell(15*13,100); % Categories (8) + header row + months
aConsumi = cell(1+numRows*12, 12+98); % For detailed consumption data
consAnno=zeros(9,12);
consGiorn=zeros(9,96,12);
% Copy header row from comune table
```

```
aConsumi(1,1:14) = [comune.Properties.VariableNames, {'Categoria', 'Mese'}];
for h = 1:96
    headerIndex = length(comune.Properties.VariableNames) + 2 + h;
    if headerIndex <= size(aConsumi, 2)
        aConsumi{1, headerIndex} = sprintf('Ora_%d', h);
    end
end
% Nomi delle categorie (come nel primo script)
nomiCategorie = {'Residential', 'Agricultural', 'Nurseries', 'Sports', 'Municipal', 'Church', 'Services', 'Other','TOT'};
% Nomi dei mesi
nomiMesi = {'Gennaio', 'Febbraio', 'Marzo', 'Aprile', 'Maggio', 'Giugno', ...
            'Luglio', 'Agosto', 'Settembre', 'Ottobre', 'Novembre', 'Dicembre'};
% Process each building
for i = 2:numRows
    for mese=1:12
        indi=i+(numRows-1)*(mese-1);
        cat=getETy(string(comune{i,6}),string(comune{i,3}));
        area=comune{i,5};
        Nab=comune{i,11};
        aConsumi(indi,1:12)=table2cell(comune(i,1:12));
        aConsumi{indi,13}=cat;
        aConsumi{indi,14}=mese;
        totMese=0;
        for ora=1:96
            potOra=0;
            if cat==1 && area< 150
                potOra=Nab*consumi{mese,7+floor((ora-1)/4)}/4;
            elseif cat==1 && area>= 150
                potOra=Nab*consumi{14+mese,7+floor((ora-1)/4)}/4;
            else
                potOra=(area/110)*consumi{28+mese,7+floor((ora-1)/4)}/4;
            end
            aConsumi{indi,ora+14}=potOra;
```

```
53              %totale per ogni mese per ogni categoria
54              consGiorn(cat,ora,mese)=consGiorn(cat,ora,mese)+ potOra;
55              %totale categorie mensili
56              consGiorn(9,ora,mese)=consGiorn(9,ora,mese)+ potOra;
57              totMese=totMese+potOra*0.25;
58          end
59          consAnno(cat,mese)=consAnno(cat,mese)+ totMese*giornimese(mese);
60          consAnno(9,mese)=consAnno(9,mese)+ totMese*giornimese(mese);
61      end
62  end
63  %scrittura cella per esportazione dati
64  for mese=1:12
65      consumiTot{14+(mese-1)*12,1}=nomiMesi(mese);
66      for cat=1:9
67          for ora=1:96
68              consumiTot{14+(mese-1)*12+cat, 1+ora} = consGiorn(cat,ora,mese)
                    ;
69          end
70          consumiTot{1+cat,1+mese}=consAnno(cat,mese);
71          consumiTot{1,1+mese}=nomiMesi(mese);
72          consumiTot{1+cat,1}=nomiCategorie(cat);
73      end
74  end
75  %formattazione dati per tabelle e grafici overleaf
76  % Converti eventuali celle nidificate in una matrice utilizzabile
77  for i = 1:numel(consumiTot)
78      if iscell(consumiTot{i})
79          consumiTot{i} = cell2mat(consumiTot{i}); % Converti celle
                nidificate in array
80      end
81  end
82  % Save results to Excel - using writecell like in the first script
83  writecell(consumiTot, resultsFileName, 'Sheet', 'ConsumiTot', 'WriteMode',
      'overwritesheet');
84  writecell(aConsumi, resultsFileName, 'Sheet', 'AConsumi', 'WriteMode', '
      overwritesheet');
85  fprintf('Analisi dei consumi completata. Risultati salvati in %s\n',
```

```
    resultsFileName);
end
% Function to determine the category based on abbreviation and description
function numero = getETy(sigla, descr)
    % Using the same function from the first script
    if (strcmpi(descr,'municipio') || ...
            strcmpi(descr,'poste') || ...
            strcmpi(descr,'biblioteca') || ...
            strcmpi(descr,'RSA') || ...
            strcmpi(descr,'scuola'))
        numero=5;
   elseif strcmpi(descr,'chiesa') || ....
           strcmpi(descr,'campanile')||....
           contains(descr, 'chiesa')||....
           contains(descr, 'campanile')
        numero=6;
    elseif strcmpi(sigla,'Sv')
        numero=3;
    elseif strcmpi(sigla,'Iz')
        numero=2;
    elseif strcmpi(sigla,'Vs')
        numero=4;
    elseif strcmpi(sigla,'Ed') || ...
       strcmpi(sigla,'Er') || ...
       strcmpi(sigla,'Es') || ...
       strcmpi(sigla,'Vv')
        numero=1;

    elseif strcmpi(sigla,'Is')
        numero=7;
    else
        numero=8;
    end
end
```

Collection of data sheets of PV panels

SS-410-54MDH 108 cells

Caratteristiche Elettriche

Modello	SS-400-54MDH		SS-405-54MDH		SS-410-54MDH	
	STC	NOCT	STC	NOCT	STC	NOCT
Potenza Massima— P_{mp} (W)	400	298	405	302	410	305
Tensione a vuoto— V_{oc} (V)	37.18	34.95	37.33	35.09	37.68	35.42
Corrente di cc— I_{sc} (A)	13.39	10.85	13.44	10.89	13.59	11.01
Tensione a Pot. max — V_{mp} (V)	31.42	29.22	31.55	29.35	31.84	29.61
Corrente a Pot. max — I_{mp} (A)	12.74	10.21	12.84	10.29	12.88	10.31
Efficienza del Modulo	20.5%		20.7%		21.0%	
Tolleranza sulla Potenza (W)	(0,+5)					
Tensione Massima di sistema (V)	1500					
Sovracorrente Massima (A)	25					
Temperatura di esercizio (°C)	-40~+85 °C					

STC: Irraggiamento 1000W/m², Temperatura celle 25C°, Massa d'aria AM1,5 secondo EN60904-3.
NOCT: Irraggiamento 800W/m², Temperatura ambiente 20°C, Velocità vento 1 m/s

Caratteristiche strutturali

Dimensioni Modulo (AxLxP)	1724 x 1134 x 30 mm
Peso	21.5 kg
Numero di Celle	108 Celle
Celle	Celle mono-cristalline PERC 182×91mm
Vetro frontale	Temperato Anti Riflesso/3.2mm; alta trasmittanza
Cornice	Lega di alluminio anodizzato/Colore Nero
Scatola di giunzione	IP68,3Diodi bypass
Cavi (lunghezza/sezione)	1000mm/4mm²
Connettori	MC4 Compatibili
Carico meccanico	Lato frontale: 5400 Pa / Lato posteriore: 2800 Pa
Impatto alla grandine	25 mm di diametro alla velocità di 23m/s

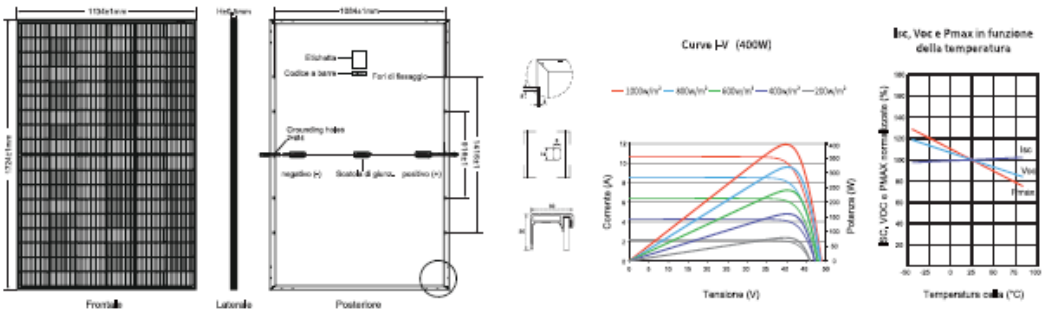
Caratteristiche Termiche

Coefficiente di temperatura(P_{max})	-0.35 %/°C
Coefficiente di temperatura(V_{oc})	-0.27 %/°C
Coefficiente di temperatura(I_{sc})	+0.05 %/°C
Temp. Nominale di esercizio Cella (NOCT)	45±2 °C

Caratteristiche di spedizione

Container	40HQ
Quantità/pallet	36
Pallets/container	26
Quantità/container	936

Dimensioni (mm)



SUNOVA SOLAR

Web: www.sunova-solar.com E-mail: info@sunova-solar.com

* Sunova Solar Technology Co., Ltd reserves the right to make any adjustment to the information described herein without further notice. Please contact our company to use the lastest version for contract.

Powering the Future

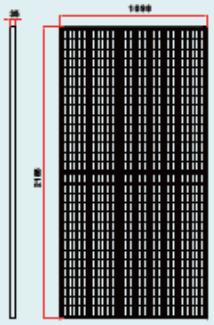
SD20220300IJT

Figure 61: *Data of PV panel ty: SS-410-54MDH*

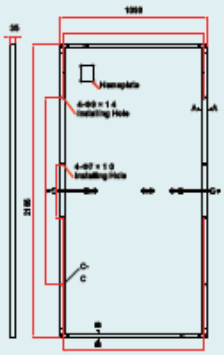
Maysun Solar

MS(485-510)MB-50H Silver Frame

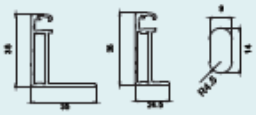
DIMENSIONS OF PV MODULE(mm)



Front View

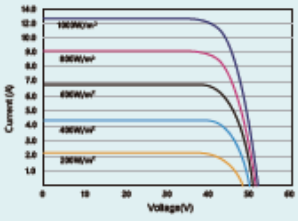


Back View

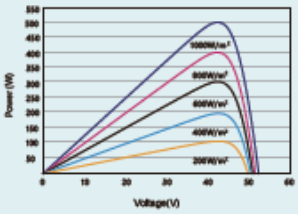


A-A B-B C-C

I-V CURVES OF PV MODULE(500W)



P-V CURVES OF PV MODULE(500W)



ELECTRICAL DATA (STC)

Peak Power Watts-PMAX (Wp)*	485	490	495	500	505	510
Power Tolerance-PMAX (W)	0 ~+5					
Maximum Power Voltage-VMPP (V)	42.2	42.4	42.6	42.8	43.0	43.2
Maximum Power Current-IMPP (A)	11.49	11.56	11.63	11.69	11.75	11.81
Open Circuit Voltage-Voc (V)	51.1	51.3	51.5	51.7	51.9	52.1
Short Circuit Current-Isc (A)	12.07	12.14	12.21	12.28	12.35	12.42
Module Efficiency η m (%)	20.1	20.3	20.5	20.7	21.0	21.2

STC: Irradiance 1000W/m², Cell Temperature 25°C,
Air Mass AM1.5. *Measuring tolerance: ±3%.

ELECTRICAL DATA (NOCT)

Maximum Power-PMAX (Wp)	365	369	373	377	381	385
Maximum Power Voltage-VMPP (V)	39.9	40.0	40.2	40.4	40.6	40.5
Maximum Power Current-IMPP (A)	9.17	9.22	9.28	9.33	9.38	9.50
Open Circuit Voltage-Voc (V)	48.1	48.2	48.4	48.6	48.8	49.0
Short Circuit Current-Isc (A)	9.73	9.78	9.84	9.90	9.95	10.01

NOCT: Irradiance at 800W/m², Ambient Temperature 20°C, Wind Speed 1m/s.

MECHANICAL DATA

Solar Cells	Monocrystalline
Cell Orientation	150 cells (5 x 30)
Module Dimensions	2185X1098X35 mm (86.02X43.22X1.38 inches)
Weight	26.5 kg
Glass	3.2 mm (0.13 inches), High Transmission, AR Coated Heat Strengthened Glass
Encapsulant Material	EVA
Backsheet	White
Frame	35 mm(1.38 inches) Silver, anodized aluminium alloy
J-Box	IP 68 rated (3 bypass diodes)
Cables	Photovoltaic Technology Cable 4.0mm² (0.006 inches²) Portrait: N 300mm/P 300mm(11.8/11.8 inches) Length can be customized
Connector	MC4 Compatible

*Please refer to regional datasheet for specified connector.

TEMPERATURE RATINGS

NOCT(Nominal Operating Cell Temperature)	43°C (±2°C)
Temperature Coefficient of PMAX	- 0.34%/°C
Temperature Coefficient of Voc	- 0.25%/°C
Temperature Coefficient of Isc	0.04%/°C

MAXIMUMRATINGS

Operational Temperature	- 40 ~ +85°C
Maximum System Voltage	1500V DC (IEC)
	1000V DC (IEC)
Max Series Fuse Rating	20A

WARRANTY

15 year Product Workmanship Warranty
25 year Power Warranty
2.5% first year degradation
0.5% Annual Power Attenuation

*Please refer to product warranty for details.

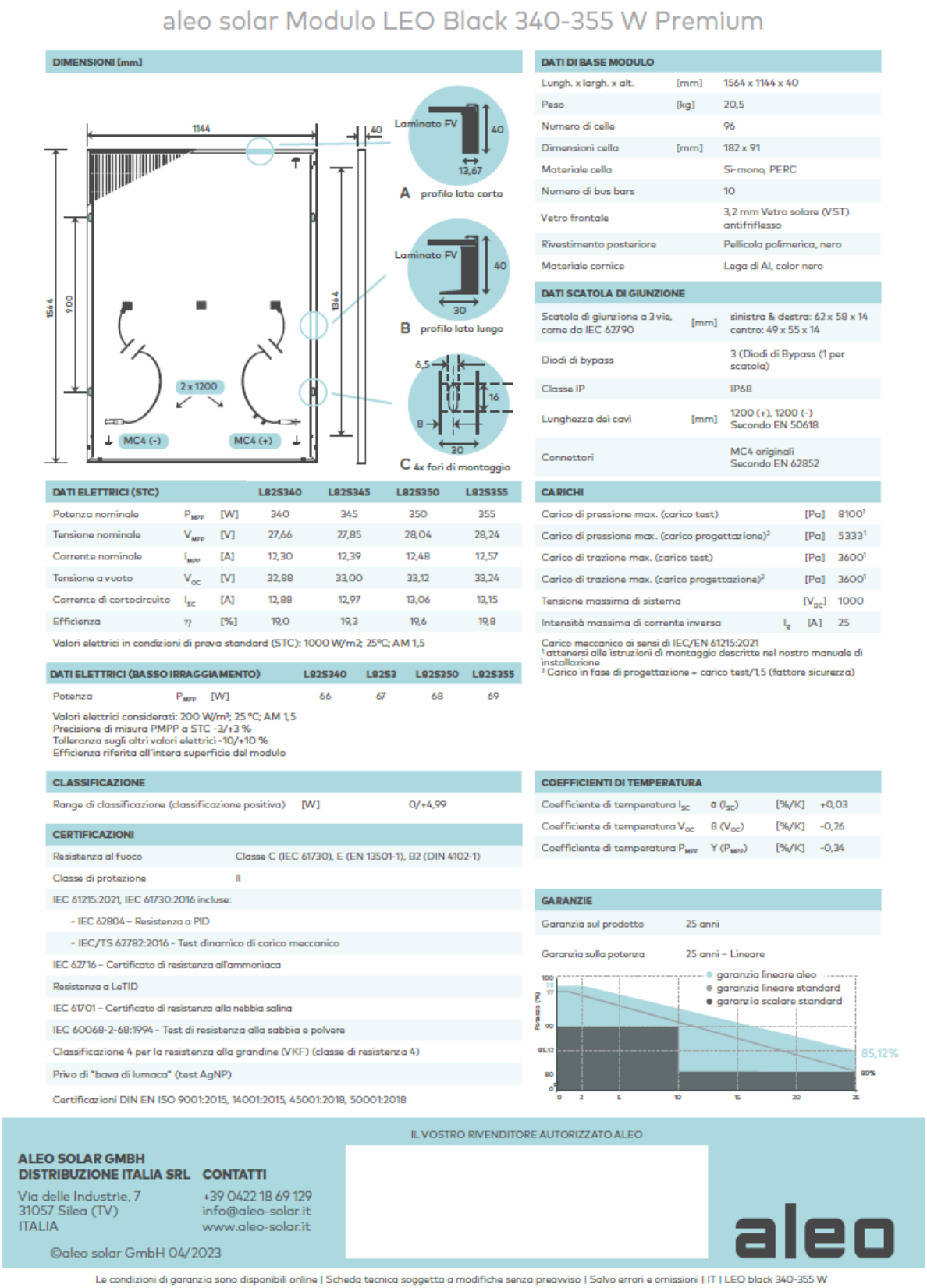
PACKAGING CONFIGUREATION

Modules per pallet: 31pieces
Modules per 40' container: 620 pieces

CAUTION: READ SAFETY AND INSTALLATION INSTRUCTIONS BEFORE USING THE PRODUCT.

Website: www.maysunsolar.com

Figure 62: *Data of PV panel ty: MS510MB-50H*



aleo solar Modulo LEO Black 340-355 W Premium

DIMENSIONI [mm]

1144 · 40 · 1564 · 900 · 1364 · 2 x 1200 · MC4 (-) · MC4 (+)

Laminato FV 40 13,67 — A profilo lato corto

Laminato FV 40 30 — B profilo lato lungo

6,5 · 16 · 8 · 30 — C 4x fori di montaggio

DATI DI BASE MODULO

Lungh. x largh. x alt.	[mm]	1564 x 1144 x 40
Peso	[kg]	20,5
Numero di celle		96
Dimensioni cella	[mm]	182 x 91
Materiale cella		Si-mono, PERC
Numero di bus bars		10
Vetro frontale		3,2 mm Vetro solare (VST) antiriflesso
Rivestimento posteriore		Pellicola polimerica, nero
Materiale cornice		Lega di Al, color nero

DATI SCATOLA DI GIUNZIONE

Scatola di giunzione a 3 vie, come da IEC 62790	[mm]	sinistra & destra: 62 x 58 x 14 centro: 49 x 55 x 14
Diodi di bypass		3 (Diodi di Bypass (1 per scatola)
Classe IP		IP68
Lunghezza dei cavi	[mm]	1200 (+), 1200 (-) Secondo EN 50618
Connettori		MC4 originali Secondo EN 62852

DATI ELETTRICI (STC)

			L82S340	L82S345	L82S350	L82S355
Potenza nominale	P_{MPP}	[W]	340	345	350	355
Tensione nominale	V_{MPP}	[V]	27,66	27,85	28,04	28,24
Corrente nominale	I_{MPP}	[A]	12,30	12,39	12,48	12,57
Tensione a vuoto	V_{OC}	[V]	32,88	33,00	33,12	33,24
Corrente di cortocircuito	I_{SC}	[A]	12,88	12,97	13,06	13,15
Efficienza	η	[%]	19,0	19,3	19,6	19,8

Valori elettrici in condizioni di prova standard (STC): 1000 W/m2; 25°C; AM 1,5

DATI ELETTRICI (BASSO IRRAGGIAMENTO)

			L82S340	L82S3	L82S350	L82S355
Potenza	P_{MPP}	[W]	66	67	68	69

Valori elettrici considerati: 200 W/m²; 25 °C; AM 1,5
Precisione di misura PMPP a STC -3/+3 %
Tolleranza sugli altri valori elettrici -10/+10 %
Efficienza riferita all'intera superficie del modulo

CARICHI

Carico di pressione max. (carico test)	[Pa]	8100[1]
Carico di pressione max. (carico progettazione)[2]	[Pa]	5333[1]
Carico di trazione max. (carico test)	[Pa]	3600[1]
Carico di trazione max. (carico progettazione)[2]	[Pa]	3600[1]
Tensione massima di sistema	[V_{DC}]	1000
Intensità massima di corrente inversa I_R	[A]	25

Carico meccanico ai sensi di IEC/EN 61215:2021
[1] attenersi alle istruzioni di montaggio descritte nel nostro manuale di installazione
[2] Carico in fase di progettazione = carico test/1,5 (fattore sicurezza)

CLASSIFICAZIONE

Range di classificazione (classificazione positiva) [W] 0/+4,99

CERTIFICAZIONI

Resistenza al fuoco	Classe C (IEC 61730), E (EN 13501-1), B2 (DIN 4102-1)
Classe di protezione	II

IEC 61215:2021, IEC 61730:2016 incluse:
- IEC 62804 – Resistenza a PID
- IEC/TS 62782:2016 - Test dinamico di carico meccanico

IEC 62716 – Certificato di resistenza all'ammoniaca
Resistenza a LeTID
IEC 61701 – Certificato di resistenza alla nebbia salina
IEC 60068-2-68:1994 - Test di resistenza alla sabbia e polvere
Classificazione 4 per la resistenza alla grandine (VKF) (classe di resistenza 4)
Privo di "bava di lumaca" (test AgNP)
Certificazioni DIN EN ISO 9001:2015, 14001:2015, 45001:2018, 50001:2018

COEFFICIENTI DI TEMPERATURA

Coefficiente di temperatura I_{SC}	α (I_{SC})	[%/K]	+0,03
Coefficiente di temperatura V_{OC}	β (V_{OC})	[%/K]	-0,26
Coefficiente di temperatura P_{MPP}	γ (P_{MPP})	[%/K]	-0,34

GARANZIE

Garanzia sul prodotto 25 anni
Garanzia sulla potenza 25 anni – Lineare

IL VOSTRO RIVENDITORE AUTORIZZATO ALEO

ALEO SOLAR GMBH
DISTRIBUZIONE ITALIA SRL
Via delle Industrie, 7
31057 Silea (TV)
ITALIA

CONTATTI
+39 0422 18 69 129
info@aleo-solar.it
www.aleo-solar.it

aleo

Le condizioni di garanzia sono disponibili online | Scheda tecnica soggetta a modifiche senza preavviso | Salvo errori e omissioni | IT | LEO black 340-355 W

Figure 63: *Data of PV panel ty: L82S355*

Bibliography

[1] R. Emilia-Romagna, *Specifiche di contenuto del DataBase Topografico della Regione Emilia Romagna*. Regione Emilia-Romagna - Servizio Sviluppo dell'Amministrazione digitale e Sistemi Informativi geografici, 18-12-2011.

[2] "Directive (eu) 2019/944 of the european parliament and of the council of 5 june 2019 on common rules for the internal market for electricity and amending directive 2012/27/eu," *Official Journal of the European Union*, 2019.

[3] "Directive (eu) 2018/2001 of the european parliament and of the council of 11 december 2018 on the promotion of the use of energy from renewable sources," *Official Journal of the European Union*, 2018.

[4] "Regulation (eu) 2023/2413 of the european parliament and of the council of 18 october 2023 amending directive (eu) 2018/2001, regulation (eu) 2018/1999 and directive 98/70/ec as regards the promotion of energy from renewable sources," *Official Journal of the European Union*, 2023.

[5] "Dl. 30 dicembre 2019, n.162," *Gazzetta Ufficiale della Repubblica Italiana*, 2019.

[6] M. dell'Ambiente e della Sicurezza Energetica, "Decreto cer," 2023, m_ante.UDCM.DECRETI MINISTERO.R.0000414.07-12-2023. [Online]. Available: https://www.mase.gov.it/sites/default/files/Decreto%20CER.pdf

[7] ——, "Le comunità energetiche rinnovabili - faq_1," 2023. [Online]. Available: https://www.mase.gov.it/sites/default/files/Le%20Comunita%CC%80%20Energetiche%20Rinnovabili%20-%20FAQ_1.pdf

[8] T. Sara, "La comunità energetica," 2021, vademecum.

[9] GSE. Le comunità energetiche rinnovabili in pillole. [Online]. Available: https://www.gse.it/servizi-per-te/autoconsumo/le-comunita-energetiche-rinnovabili-in-pillole

[10] Edison. Comunità energetiche rinnovabili: cosa sono, come funzionano, vantaggi. [Online]. Available: https://www.edisonnext.it/next-journal/comunita-energetiche-rinnovabili-modello-virtuoso/?utm_term=come%20funzionano%20le%20cer&utm_campaign=Blogpost+-+ComunitÃă+energetiche+rinnovabili:+modello+virtuoso&utm_source=adwords&utm_medium=ppc&hsa_acc=4929659778&hsa_cam=22342081167&hsa_grp=171770881610&hsa_ad=739577586614&hsa_src=g&hsa_tgt=kwd-2407683740295&hsa_kw=come%20funzionano%20le%20cer&hsa_mt=b&hsa_net=adwords&hsa_ver=3&gad_source=1&gclid=Cj0KCQjwy46_BhDOARIsAIvmcwMhfhDTu7ShfFDK74oBvVFoVpCSXfL2yOQPpAlrFcOapIJSzr9OqfEaAwcB

[11] S. Verde, N. Rossetto, A. Ferrari, and T. Fonteneau, "The future of renewable energy communities in the eu: an investigation at the time of the clean energy package," 08 2020.

[12] F. Ceglia, P. Esposito, A. Faraudello, E. Marrasso, P. Rossi, and M. Sasso, "An energy, environmental, management and economic analysis of energy efficient system towards renewable energy community: The case study of multipurpose energy community," *Journal of Cleaner Production*, vol. 369, p. 133269, 2022.

[13] G. Macchi Jánica, "Gis, critical gis e storia della cartografia," *Geotema*, vol. 58, pp. 179–187, 2018.

[14] ESRI. History of the gis. [Online]. Available: https://www.esri.com/it-it/what-is-gis/history-of-gis#:~:text=Il%20primo%20GIS&text=Tomlinson%20%C3%A8%20il%20primo%20a,il%20%22padre%20del%20GIS%22.

[15] ENEA. Recon. [Online]. Available: https://recon.smartenergycommunity.enea.it

[16] E. Commission. Pvgis. [Online]. Available: https://re.jrc.ec.europa.eu/pvg_tools/it/#DR

[17] ENEA. Atlante italiano della radiazione solare. [Online]. Available: http://www.solaritaly.enea.it/CalcComune/Calcola.php/

[18] Arera. Analisi dei consumi dei clienti domestici. [Online]. Available: https://www.arera.it/dati-e-statistiche/dettaglio/analisi-dei-consumi-dei-clienti-domestici

[19] R. Emilia-Romagna. Servizio di download db topografico. [Online]. Available: https://geoportale.regione.emilia-romagna.it/download/download-data?type=dbtopo

[20] ISTAT, "Istatdata la banca dati dell'istituto nazionale di statistica," 2023, accessed: 2024-09-25. [Online]. Available: https://esploradati.istat.it/databrowser//

[21] R. Emilia-Romagna. Download uso suolo e legende. [Online]. Available: https://geoportale.regione.emilia-romagna.it/download/dati-e-prodotti-cartografici-preconfezionati/pianificazione-e-catasto/uso-del-suolo/1976-78-coperture-vettoriali-uso-del-suolo-di-dettaglio-edizione-2024

[22] B. Farabegoli, "Analisi delle componenti principali: algoritmi e applicazioni," Master's thesis, Università di Bologna, Facoltà di scienze matematiche, fisiche e naturali, Corso di laurea in Matematica, 2015-2016.

[23] J. A. D. . W. A. Beckman, *Solar Engineering of Thermal Process*. Willey, 1980.

[24] L. Belli, "Irraggiamento solare superficiale: messa a punto di un metodo di calcolo e validazione," Master's thesis, Politecnico di Milano, scuola di architettura e società, corso di laurea magistrale in progettazione architettonica, 2012-2013.

[25] S. Bollanti, D. De Meis, P. Di Lazzaro, A. Fastelli, F. Flora, G. Gallerano, L. Mezi, D. Murra, A. Torre, and D. Vicca, *Calcolo analitico della posizione del sole per l'allineamento di impianti solari ed altre applicazioni*. ENEA-Unità Tecnica Sviluppo di Apllicazioni delle Radiazioni Centro Ricerche Frascati, Roma, 2012.

[26] F. Giunta, "Strumento per la determinazione dell'irraggiamento solare di falde di tetti," Master's thesis, Università degli Studi di Padova, Dipartimento di

Ingegneria dell'Informazione, Corso di Laurea Specialistica in Ingegneria Informatica, 2013-2014.

[27] GSE. Mappa interattiva delle cabine primarie. [Online]. Available: https://www.gse.it/servizi-per-te/autoconsumo/mappa-interattiva-delle-cabine-primarie