KTH ROYAL INSTITUTE OF TECHNOLOGY

Degree in mechanical engineering

Second cycle 30 CFU

Techno-economic Analysis of a Hybrid Particle CSP-PV Plant including Cogeneration

STEFAN MOROSANU

Stockholm, Sweden 2022

Master of Science Thesis

Department of Energy Technology

KTH 2020

KTH Industrial Engineering and Management

Techno-economic analysis of hybrid particle CSP-PV plant including cogeneration

TRITA: ITM-EX 2022:188

Stefan Morosanu

Approved	Examiner	Supervisor
13-01-2022	Laumert Björn	Rafael Eduardo Guedez Mata, Silvia Trevisan, Salvatore Guccione
	Industrial Supervisor	Contact person
	-	Eliodoro Chiavazzo

Abstract

This study aimed at evaluating the economic performance of electricity and heat production from a concentrated solar power system hybridized with PV, working with particles, and using a thermodynamic cycle base on the Brayton one having supercritical CO2. The heat is produced at around 400 ÷ 450 °C and it is intended for industrial application.

The study has been divided into two steps, the first one has been the evaluation of feasibility without the cogeneration, and the second step has been the implementation of the cogeneration and finding for which thermal load the heat production can be cost-competitive.

The analysis of the performance of the systems has been made using MoSES (Modelling of Solar Energy Systems), a new Python-based tool developed in KTH. The techno-economic has been done considering a quasi-steady-state model coupled with an economical model. The leading parameter for the optimization of both systems has been the LCOE, to make the technology economically attractive to the market the system should be able to cover almost 100 % of the thermal load to make the system able to supply heat reliably.

In this analysis has emerged that the systems can cover a high percentage of the electric load with an LCOE of 90 €/MWh, in the case of downsizing of the power block the LCOE is 82 €/MWh but with a small penalization on the load coverage. The plant presents an LCOE lower than 90 €/MWh at a power size of 50 MWe, which is relatively a small size for a power plan. In case of the cogeneration, the LCOE is 95 €/MWh with a lower electrical load coverage but the system is able to produce heat at the same LCOH from natural gas and in a reliable way.

The analysis has shown that the LCOE and the overall performance of the system indicate that its role in the grid could be the covering the slow and predictable fluctuation of the electric demand, it can also produce heat at the same price as natural gas. With the technological progress and the urging need for the reduction of emissions, these technologies will get more and more market traction.

Index

Abstract 3

1 Introduction 6

1.1 Introduction on the heat demand 7

1.2 Aim of the study 7

2 Theoretical background 8

2.1 Types of CSP 8

2.2 Solar Power Tower (SPT) 9

2.3 Ways of making more competitive the SPT 10

2.3.1 Particles 10

2.3.1.1 Direct heating particle 11

2.3.1.2 Indirect heating particle 12

2.3.2 Integration with photovoltaic (PV) 13

2.3.3 Supercritical CO_2 as a working fluid for the power block 14

2.4 Heat demand from the industry 15

2.4.1 CSP for heat generation 16

2.5 Market trend 17

2.5.1 Past and state of the art 17

2.5.2 Future direction 18

3 Methodology 19

3.1 Optimization 20

3.1.1 Design variable 20

3.1.2 Genetic algorithm 22

3.1.2.1 With cogeneration 23

4 Modelling 24

4.1 Modelling the receiver 25

4.2 Modelling of the heat exchanger for heating the sCO2 27

4.3 Modelling the thermal energy storage 30

4.4 Modelling the cogeneration 33

4.5 Modelling the electric grid 35

4.6 Modelling the thermal load 36

4.7 Modelling the economic scenario 37

4.8 Modelling the total land usage 38

5 Analysis of results 39

5.1 Without cogeneration 39

5.1.1 Performance in the scenario 43

5.2 With cogeneration 44

5.2.1 Performance in the scenario 46

5.3 Comparison the LCOE among different technologies 48

5.4 Comparison the LCOH among different technologies 49
5.5 The footprint of the system....50
6 Conclusion....51
6.1 Limitation and future work 52
7 Reference....53
8 Appendix 55
8.1 System model....55
8.2 Receiver model....92
8.3 Particle heat exchanger....94
8.4 Thermal storage (TES) 96
8.5 Cogeneration....97
8.6 Genetic algorithm 98

1 Introduction

Nowadays, the energy demand is increasing and at the same time, there is a need of reducing the emission of greenhouse gases in the atmosphere. In this context, solar energy takes an important role in the energy transition to renewable sources. The solution that is feasible for a large-scale application that has been discussed here is the so-called concentrated solar power (CSP). This type of solution for producing electricity is ideal in the counties that are located in tropical areas where the solar irradiance is high enough to make economically advantageous the conversion.

The main idea of CSP is to use the reflection of the light to concentrate in a relatively small area then, reach a high temperature up to 1000 °C (it depends on the technology) and use this heat source to run a power cycle like the Rankine or Brayton. The leading advantage of this technology is thermal energy storage, which gives the possibility to match the demand and the supply of electricity in a cost-competitive way.

Due to the decreasing price of photovoltaic panels, CSP is integrated with PV to mitigate the high LCOE of CSP alone. On the other hand, the main disadvantage of the photovoltaic panels is the variability of power output, which is intrinsic due to the weather conditions, this drawback of the PV is mitigated by the TES of the CSP. For these reasons, the hybridization with PV has gained market traction (1) due to the combinations of cost-effective energy storage and electricity production.

Moreover, the thermodynamic performance of CSP is increased by using particles because this allows the system of increasing the working temperature, which is beneficial for the overall efficiency of the thermodynamic cycle. The thermodynamic cycle is based on the Brayton cycle, and it has as a working fluid supercritical CO2, which offers better thermodynamic behaviour in that temperature range (2).

1.1 Introduction on the heat demand

A significant percentage of the energy demand is ascribable to the heat demand, **Fig. 1** (3) shows that in 2030 the 48% of the overall energy demand will be as a form of heat demand. More specifically, about 11% of the energy demand will be requested as high-temperature heat, at temperature higher than 400 °C.

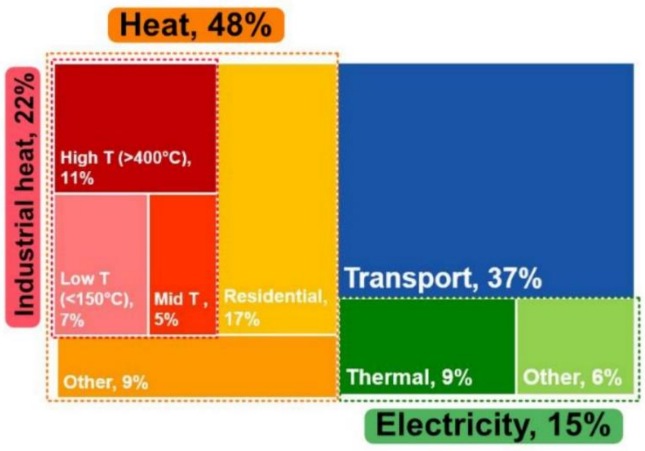


Fig. 1 Total energy demand share in 2030

According to this information, one of the points on which the study focuses is the possibility of these systems producing high-temperature heat reliably in a cost-competitive way because that system could be economically attractive. There are some CSP systems already in the market for heat production mainly parabolic through see paragraph 2.4.1 for more details, the system studied in the present work has the advantage of reaching a higher temperature than parabolic through thus, it can cover a higher variety of thermal loads.

1.2 Aim of the study

The goal of this work is the evaluation of the techno-economic performance of CSP hybridized with PV working with particles and having a Brayton cycle with sCO2 for the power block with cogeneration and without cogeneration.

The performance of the system has been evaluated with the following considerations:

- Calculating the minimum LCOE of the system in the design variable chosen
- Find an application in the electricity market in which these technologies could be competitive considering its KPI, mainly capacity factor (Cf) and LCOE.
- Test the system considering the actual prices of electricity of the geographical position chosen
- Introducing the cogeneration and finding if the system can produce heat at a competitive cost.

2 Theoretical background

This chapter has discounted the most relevant theoretical background for the sake of understanding the system that has been modelled. It has been explained the different types of CSP with a particular focus on solar power tower (SPT) because it is the system that has been used for the studying, after which, it has been discussed the different techniques commonly used for making the SPT more competitive in the market.

2.1 Types of CSP

There are mainly 4 types of configurations of CSP see **Fig. 2** (4). However, only 2 types are used which are parabolic trough and solar tower; The most used nowadays is the parabolic trough in 90% of the cases and for the rest part there is the solar tower, nevertheless, the solar tower technology has gained market traction due to its potential and the decreasing costs.

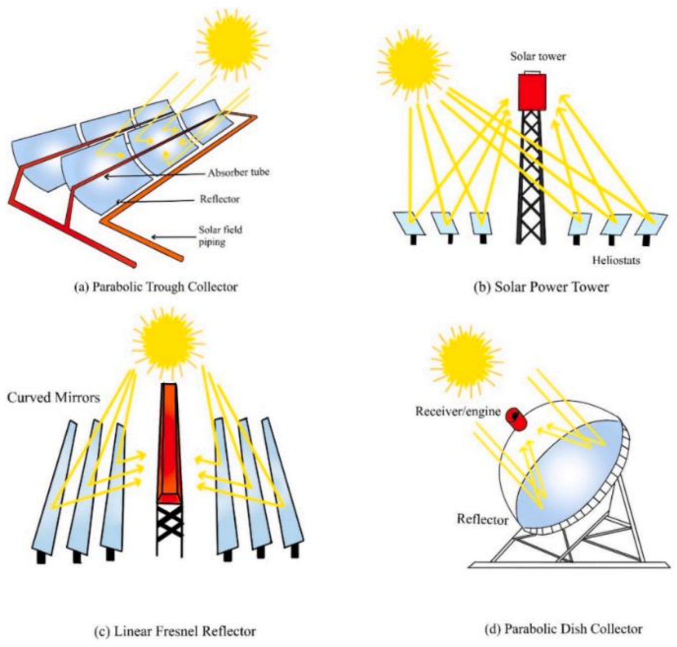


Fig. 2 Scheme of the different types of CSP

- The parabolic trough is the most used nowadays due to its relative simplicity and its reasonable power production. The distinctive feature of this technology is the shape of the mirror, therefore, the shape of the receiver. The mirror has a parabolic shape, and it concentrates the solar irradiance on the pipe, the solar tracking of the mirrors is one dimension, so it tracks the movement of the sun using just one single axis hence, the cosine effectiveness is not always maximum. The solar irradiance is concentrated on the receiver, which is the pipe, the working fluid can be directly water to have a direct steam generation of can be synthetic oil that through a heat exchange warms up the water for the power cycle. The receiver is made of layers, the first layer has the function to reduce the heat losses due to the convection of the external wind and at the same time having the lowest possible reflection, so it is typically transparent glass. The second layer has the function to absorb as much as possible thus, it is a black material with high absorption properties. The main limit of this technology is the low concertation ratio that leads to a low working temperature, therefore, a low thermal efficiency.

- Linear Fresnel reflector plants have a similar design to parabolic trough collect, the receiver is a pipe on which is concentrated the solar radiation by the mirrors, by contrast, the mirrors have a linear profile or a slightly curved shape to descries the optical losses. The structure is closer to the ground than in the case of parabolic trough collection, which leads to a simpler mechanical structure because there are fewer loads applied to the structure. The overall system complexity is relatively low compared to the other types because there are fewer technical difficulties. However, it has a low system efficiency due to the low working temperature because it has a low concentration ratio.
- Dish-Stirling solar power generation concentrates all the solar irradiance on a focal point that allows the system reaching temperatures up to 750 °C (5), the conversion of heat into mechanical power is done by a Stirling cycle. The mirrors of the engine systems have the same requirements as always (e.g. high reflectance, accurate shape), due to the structure of the systems, it has a sun-tracking system to increase the energy harvest. The Dish-Stirling has a high potential due to its high concentration ratio because of the small scale of the engine the losses are significant the overall efficiency is up to 30% (5). The main advantage of this type of technology is modularity. Due to its modular properties, it can be used in remote areas or for bigger power production.

2.2 Solar Power Tower (SPT)

This is the second most used type of CSP, it seems that this technology will take the lead in the future over the other types of CSP (4). This technology has a vastly potential because of its high concentration ratio, which leads to high efficiency and summed up with the fact that it is feasible for large-scale power production this is the technology in which there is a lot of interest.

The working principle of a solar power tower is that all the solar irradiance is concentrated on the one receiver (the tower) see **Fig. 2** (4), the sun concentration must be done with reasonable accuracy, therefore, all the heliostats of the systems have a double-axis tracking of solar zenith and solar elevation and to reflect on the receiver. Furthermore, these plants can reach a relatively high working temperature and compared to the other types of technologies has a short heat transfer path, which leads to small heat losses. The optical-electrical power efficiency is comparatively higher than the other solution.

The different types of solar power towers may differ from each other in the heat transfer fluid (HTF) the fluid could be water, molten salt, or air. In the case of water/steam, the advantage is the heat transfer fluid is also the same fluid for the power cycle and in this case, there is not a need for heat exchange which leads to a decrease in the complexity of the system. In a plant in which the HTF is molten-salt hence, the fluid that receives directly the heat is molten-salt, after this, HTF supplies a steam generator to feed the cycle for the power generation.

The benefit of using molten salt as an HTF is a simplification of the mechanical structure, the heat absorption by the HTF is done with a relatively low-pressure increase, which allows the system to increase the pressure in the power cycle and together with that an increase in the system efficiency. Another advantage of molten-salt plants is the energy storage, in these systems the integration with efficient energy storage is easier. The last type of solar power tower is the one that has only air circulating in the system, these kinds of solutions are using the Joule cycle. The air in the receiver is warmed up to around 700°C (5)], after it, it expands on a gas turbine and produces mechanical power. The benefit of the plants that use air as an HTF is that these systems are water-free and in places where there is a lot of solar irradiation water could be a precious resource for example in the desert, which are places where there is a lot of potential for energy production, but water is difficult to obtain.

2.3 Ways of making more competitive the SPT

One way to increase the thermodynamic performance of the system is by increasing the working temperature in the power cycle, the classic molten salts have problems with chemical stability at temperatures above 600°C, generating corrosive chemical compounds, which lead to significant mass losses (5). To overcome this problem particle receivers have been introduced, they can reach higher working temperatures, particle receivers can be integrated into plants heating directly or indirectly see the next paragraph (2.3.1).

One of the problems of SPT is its low profitability and its high cost of it, to reduce the Levelized Cost of energy produced (LCOE), the integration with PV has been proposed in many scientific papers in the last years, see paragraph 2.3.2 for more details.

SPT uses a power cycle like the Ranking cycle, but the Brayton with supercritical CO2 cycle demonstrates a better performance than the classic Ranking cycle in the common temperature range of SPT. See paragraph 2.3.3 for more details.

2.3.1 Particles

The first benefit of working with a particle receiver is the possibility of increasing the working temperature range, which is the current limitation of using the molten-salt system because of the chemical degradation of the salt. The thermal energy storage costs could be reduced by using a cheap particle like sand (5). Another advantage is the reduction of the mechanical stress in the structure because there is no increase of pressure during the heating of the fluid for the power cycle thus, there is a reduction in the complexity and the costs. The particle receiver can be done in two ways, direct or indirect.

The direct method consists of heating the particles without an intermediate medium, by contrast, the indirect method has an intermediate medium. Both heating methods have been discussed in the next paragraph 2.3.1.1 and 2.3.1.2.

2.3.1.1 Direct heating particle

The basic idea of direct particle heating is that directly concentrates the light on the particle without an intermediate medium to reduce the exegetic losses. There are a few types of direct heating particles, but just the one called Free falling particle (**Fig. 3** (5)) applied on solar power towers seems to be feasible for large-scale applications so just this technology has been discussed in this paragraph. The particles are released above the receiver, creating a thin layer of falling particles through the receiver. To significantly increase the outlet temperature the particles are continuously recirculating, this allows reaching temperatures above 700 °C with a thermal efficiency of around 50-65% (5). The amount of heat absorbed by the particles depends on the time of exposure to concentrated sunlight, which can be increased by recirculating the particles multiple times in the receiver.

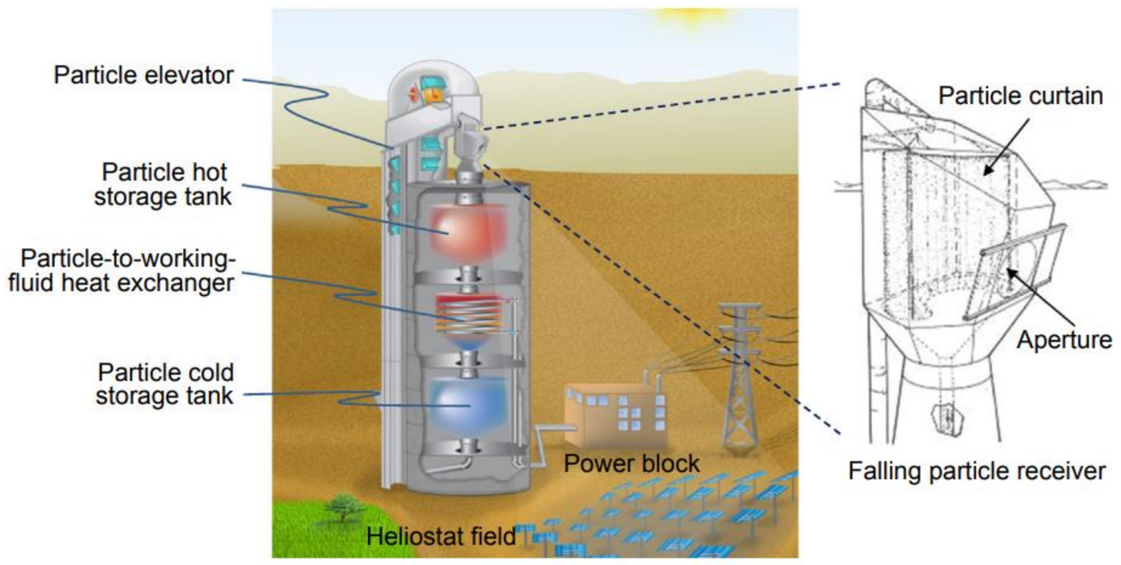

Particle elevator
Particle hot storage tank
Particle-to-working-fluid heat exchanger
Particle cold storage tank
Power block
Heliostat field
Particle curtain
Aperture
Falling particle receiver

Fig. 3 Free falling particle receiver integrated with solar tower and thermal heat storage

2.3.1.2 Indirect heating particle

The indirect heating particles were introduced to avoid the loss of particles through the cavity of the receiver, the main drawback of this technology is the loss in efficiency due to the intermediate medium.

A solution that has been proposed is called flow-through enclosures (5), the intermediate mediums for the heat exchange from the concentrated sunlight to the particles are a kind of tube. On the inner surface of the tubes is concentrated the light, on the external surface there is a flowing of particles that are flowing by the gravity field see **Fig. 4** (5), the heat is transferred by conduction and convection.

According to (5) the tests that have been made the heat exchange seems to be limited due to the loss of contact between the particles and the external surface of the tube, however, it is a technology under study.

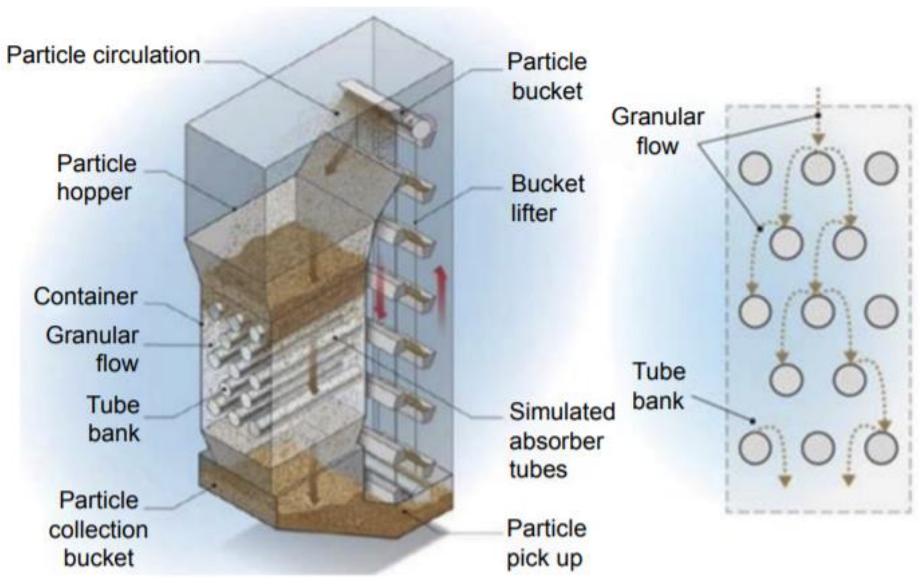


Fig. 4 Indirect particle receiver

2.3.2 Integration with photovoltaic (PV)

The integration with PV allows the system to be more flexible to supply the electricity according to the demand. The basic idea is to produce electricity during the day using PV because it is the most cost-competitive way. During the cloudy periods or overnight, it relies on CSP, which has thermal energy storage. The two systems are coupled with a thermal link (an electrical heater) see **Fig. 5** (1), however, they interact with each just in some cases. When the electricity production from the PV is not enough to supply the electrical demand the power block of CSP is turned on to cover the gap between the demand and the output of PV.

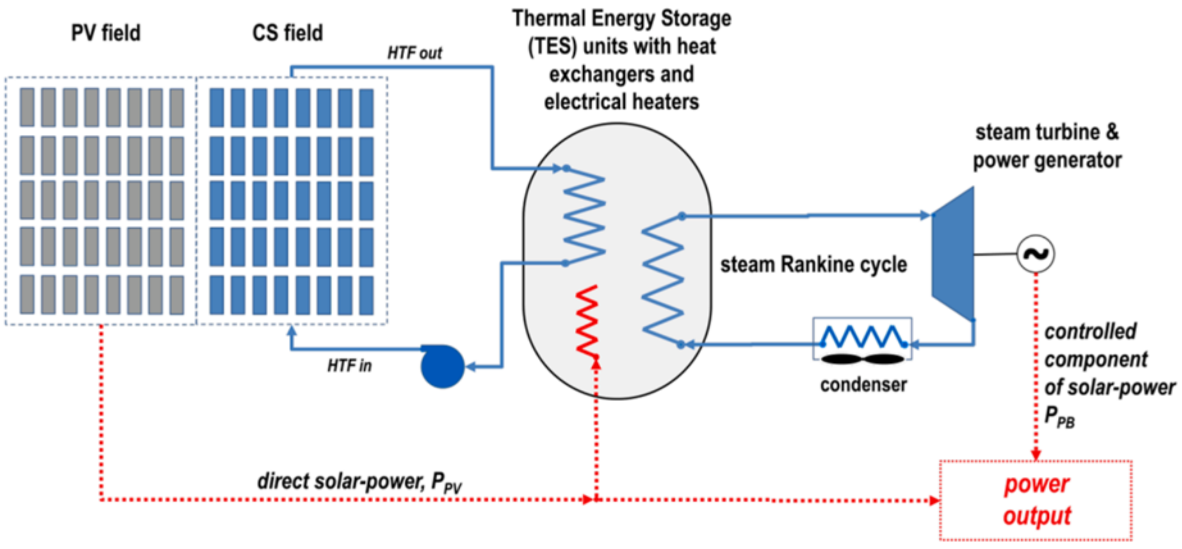

PV field
CS field
HTF out
HTF in
Thermal Energy Storage (TES) units with heat exchangers and electrical heaters
steam turbine & power generator
steam Rankine cycle
condenser
controlled component of solar-power P_{PB}
direct solar-power, P_{PV}
power output

Fig. 5 Scheme of CSP integrated with PV

There are four different scenarios, the first is when the demand of the grid is higher than the PV output in this case if there is enough thermal energy storage the power block of CSP is turned on to cover the difference between the demand and power output of PV. In the second case when the demand of the grid is almost equal to the output of PV, in this case, the CSP accumulates energy in the thermal battery. In the third case, the demand is lower than the output of PV the extra electricity is converted into heat and stored in the thermal battery has space, this aspect is useful to avoid the overload of the grid, however, the convention of electricity into heat that it will be further converted into electricity, and this is a waste of exergy, but it makes sense from an economical perspective. The last case is when the sum of the power output of the power block and PV is not enough to cover the demand, in this case, the load has to be supported by other sources.

The integration of PV with a CSP that has some degree of freedom to adapt according to the demand and a possibility of energy storage, allows the system to increase the energy supply from 30 % of the demand to 60-90% depending on the area (1). The drawback of the integration of PV with CSP is that the LCOE (Levelized Cost of Energy) is almost double that of the one using only PV, the LCOE of PV only is 0.05 €/kWh and with the integration of the two technologies is 0.08-0.11 €/kWh (1).

2.3.3 Supercritical CO_2 as a working fluid for the power block

The idea of running a power cycle with a working fluid of CO2 came up in the late '90, but due to the low price of natural gas, the idea of using CO2 for a closed Brayton (see **Fig.6** (2)) cycle was abandoned until recently [9]. In the last few years, many studies have been done about the topic, there are several proposals in the literature, but it seems that almost all of them agree on the benefits, which are discussed in this paragraph.

The first advantage of Brayton supercritical CO2 is the higher efficiency than the Rankine cycle at the typical temperature of CSP, furthermore, it has a good efficiency in a reasonable range of working temperature, which is very important to increase the off-design performance. Brayton C02 can be used in the solar power tower where the temperature is up to 1000° and in the case of parabolic through due to its flexibility.

The second benefit is that the density of supercritical CO2 is high in the working conditions of CSP, this leads to high power density machines as a direct consequence of a reduced size of the components and a lower footprint of the power block (2). The last significant improvement considering the Rankine is that there is no need for water, this is an important aspect considering the lack of water in some areas where there is a huge amount of solar energy like in a desert.

The last significant improvement considering the Rankine is that there is no need for water, this is an important aspect considering the lack of water in some areas where there is a huge amount of solar energy like in a desert.

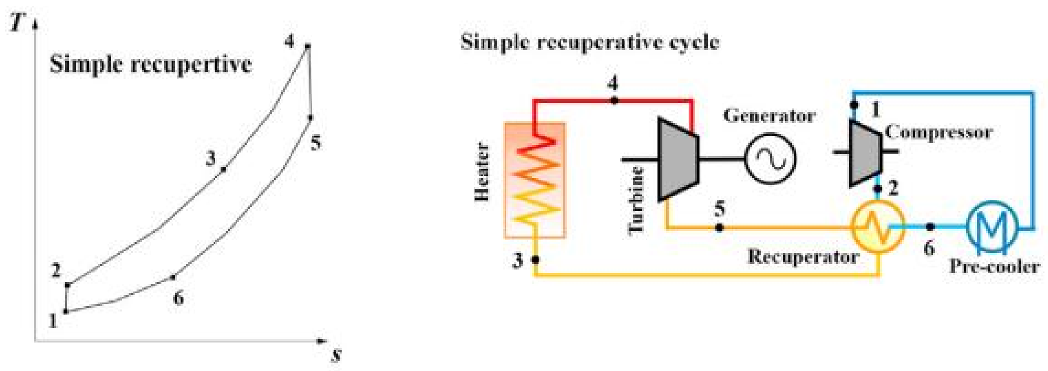


Fig. 6 Thermodynamic cycle on T-s and the scheme of the basic components of the cycle

There are seven types of different configurations, the recompression with reheating reaches the highest efficiency if the load is between 100 % and 57 % of the nominal capacity (2). Under 57 % partial-cooling cycles show higher efficiency than recompression with reheating, but generally speaking the layout with the best thermal performance is recompression with reheating (2).

However, under a techno-economic analysis considering the TES capacity, the layout with the lowest LCOE is partial-cooling (6). Although the recompression with reheating has the highest thermal efficiency it has also the highest cost due to its components.

2.4 Heat demand from the industry

The CSP technologies could be used to generate heat for the industry without relying on fossil fuel technology. Fig. 7 (7) shows the heat demand from the industry based on the sector and the temperature needed in the European Union in 2006. Based on Fig. 7, there are many sectors where the temperature needed could be matched with the capability of the CSP for heat production.

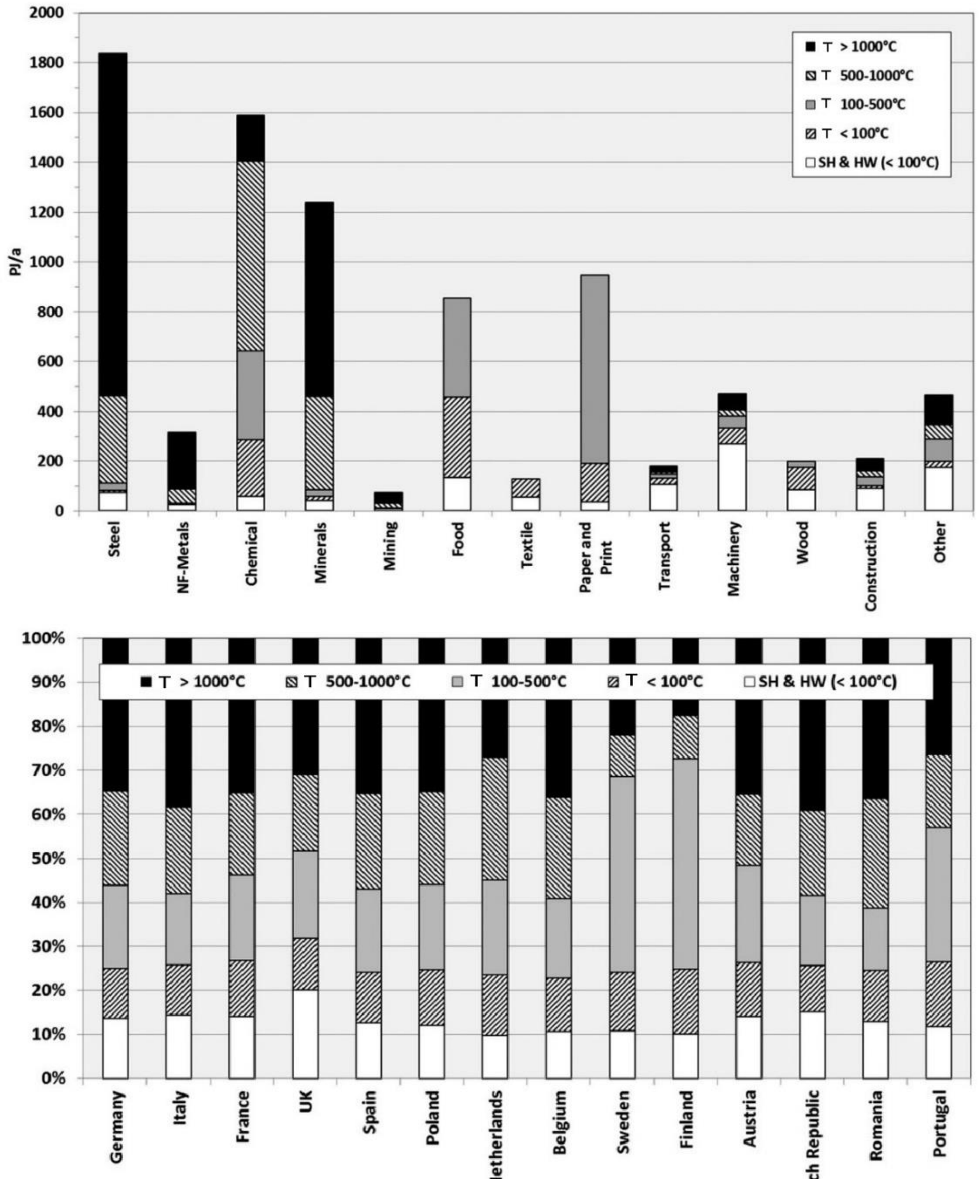


Fig. 7 The top figure shows the heat demand for different sectors, the bottom figure shows the normalized heat demand of each nation for different temperature

The location of the study is Sevilla (Spain 37° N, 6°W, DNI=1800 kWh/m^2 (8)), Figure 7 also indicates the heat demand according to the temperature for some European countries. The percentage of heat demand up to 450°C is around 40% of the overall heat demand in Spain thus, these technologies could be able to produce renewable heat in a cost-competitive way without relying on fossil fuel resources. SH is the acronym for space heat, and HW is the acronym for hot water. The other categories of temperature range are referring to the heat needed for industrial processes.

2.4.1 CSP for heat generation

At the present moment, the systems that have integrated CSP as a heat supply for their need are relatively few. However, the most used CSP type for heat production is the parabolic trough that could reach temperatures up to 300 °C (4). **Fig. 8** (9) shows a typical case of integration of the CSP in the existing plant, in this case, the CSP allows the system to consume significantly less amount of natural gas for its heat needs. The application in which the CSP has been integrated for heat generation purposes are the following: beverage, food processing, textile, district heating, dairy, and chemicals.

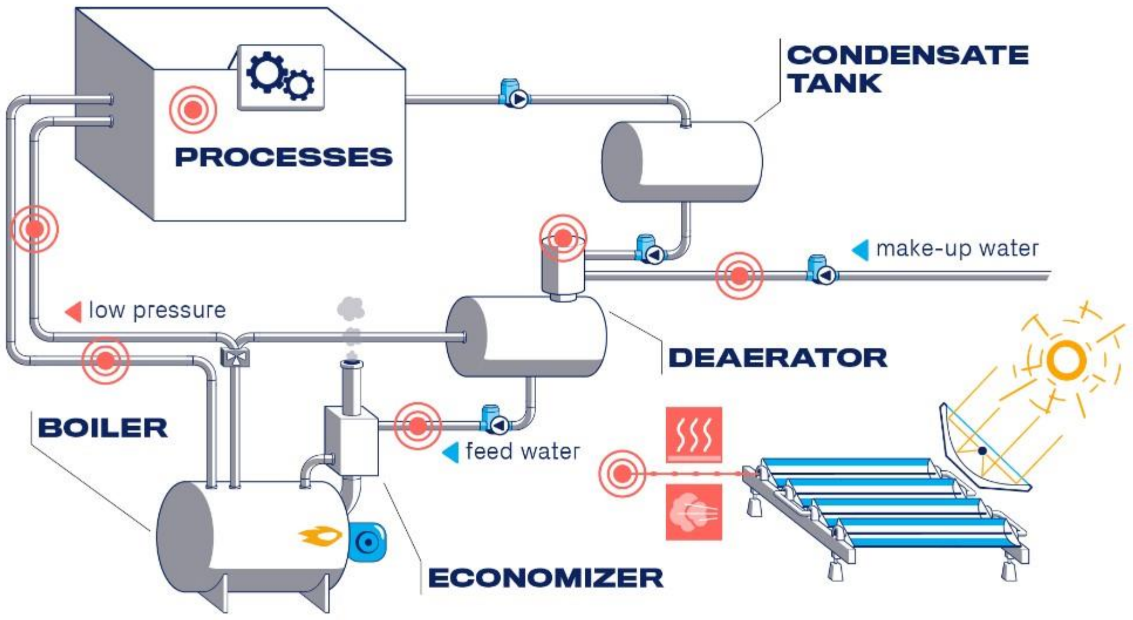


Fig. 8 Possible layout of CSP used as a heat generation for industrial purposes

The main difference between the parabolic trough and the Solar Power Tower for heat generation is the maximum temperature, in the case of SPT the temperature could be up almost to 800 °C. Another difference is SPT requires more infrastructure because of its need for the tower hence, it is not attractive on a small scale.

2.5 Market trend

This chapter summarizes the past and the current state of the art regarding the SPT and it forecasts the future market trend according to the scientific papers that have been found in the literature.

2.5.1 Past and state of the art

In the past years since nowadays, the most implemented CSP technology worldwide is the Parabolic trough collector (PTC) see **Fig. 9** (4). As a direct consequence, it is the most mature technology of CSP and it has the lowest technical and financial risk (10). The first CSP plants have not been integrated with thermal energy storage (TES) because of the costs and complexity of the system, but the benefits of TES are so significant that the integration with it makes the plant more competitive. Since 2015, hardly any projects have been built without a TES, the addition of TES is now a cost-effective way to lower LCOE (11), increasing the flexibility and increasing the capacity factor. Until today, the method of thermal energy storage has been done using Sensible heat storage (SHS) even though phase change energy storage has better thermodynamic properties, from a technological and economical point of view, SHS are competitive (12). The materials most used for TES have been molten salt (60% wt NaNO3 and 40% wt KNO3 (13)) or synthetic oil. The molten salt has the following benefits: higher working temperature, no flammability, and lower toxicity (13). Once the technological challenges have been overcome, the most plant started to use molten salt.

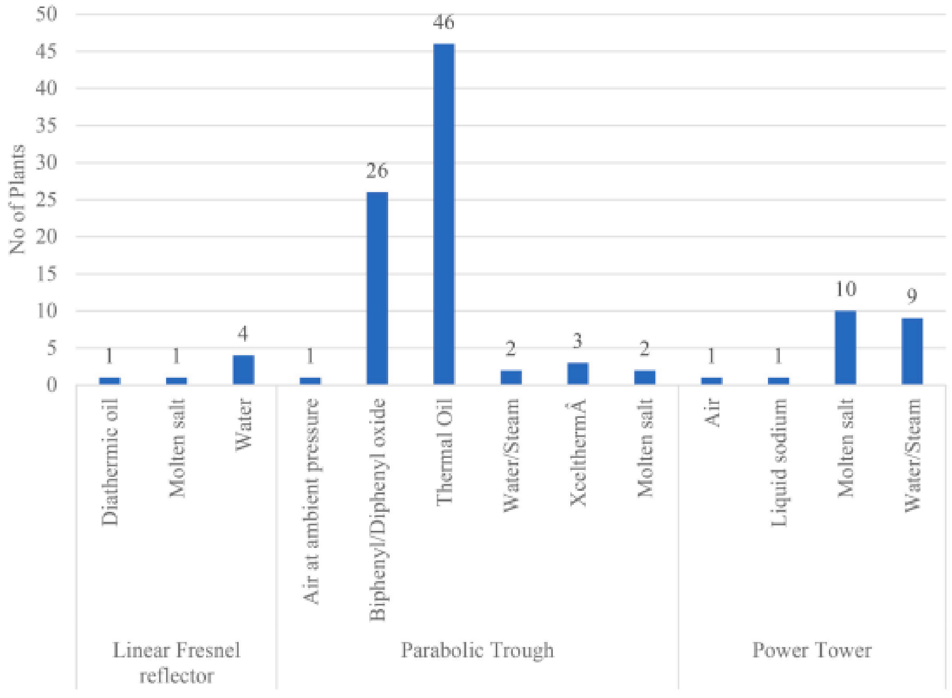


Fig. 9 The global the number of CSP plant and types of plants in the world

2.5.2 Future direction

The first trend of the market relevant to mention is the one of SPT, considering the total capacity under construction the 60% of it is on SPT and 37% of it is PTC (10). The SPT due to its concentration ratio and its properties is more feasible for large scale and it seems that the market is going in this direction. Consequently, there are many papers on improving the efficiency, and the integration with PV and the Brayton-CO2 cycles is almost certain.

However, there are different approaches regarding the heat transfer fluid: molten salt, particles, and air. In this case, is more difficult to predict which HTF will take the lead in the future. The HTF more suitable for the integration with SPT Brayton sCO2 now is molten salt due to its maturity, but in the future, air with a packed bed of rocks and particles will take remarkable importance in the market. Both air and particles receivers have the following advantages (12): higher operating temperature, operating pressure can be close to ambient temperature so there is no need for complex sealing, lower fabrication cost than in the case of molten salts and steam, lower cost of rocks and particles (if properly chosen).

In the next future, thermal heat storage will still be sensible heat storage because of the low maturity of the other technologies. There are some studies regarding the possibility of using latent heat storage or chemical storage, but there are remarkably more expensive despite their higher thermal efficiency these solutions are not ready to be commercialized (14).

Fig.10 shows (14) that the majority of the capacity currently developing is of the types of SPT, this is a clear direction of the market.

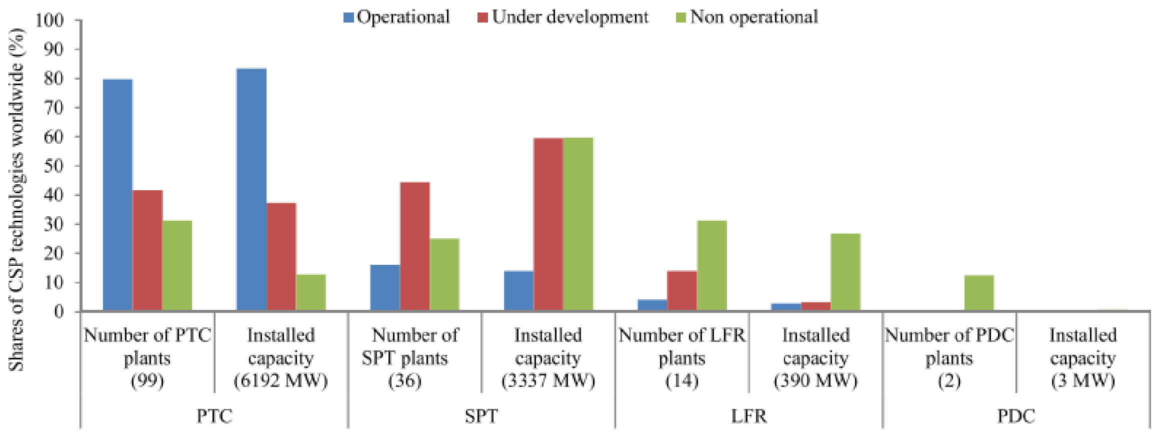


Fig. 10 Status and distribution of each type of CSP

3 Methodology

The starting point has been the literature review to have a better understanding of the topic and find the literature gap to determine the research direction.

The second step was finding in the literature the most suitable component for the conversion from a classic molten salt system to a particle one. The components that changed remarkably are the following: the receiver, the storage system, the heat exchanger that heats the sCO2 and the heat exchanger for the cogeneration. The first part of the study has been conductive non considering the cogeneration to evaluate the system without it.

The third step was building a techno-economical of the components and integrating them into the model, after which the assumptions of the electrical demand and the geographical positions have been made. At this point it has been possible deciding the design variable of the study and run the optimization to obtain the size of each component, the optimization algorithm is from the family of Genetic Algorithms. The optimization of the system has been based on the minimization of the LCOE.

After the first optimization, the cogeneration has been added to the system. To avoid having too many design variables the size of the plant has been fixed at the minimum LCOE of the case without cogeneration. Two types of thermal load profiles have been tried with different peak values and for each thermal load, the system has been optimized. The optimization in the case of the cogeneration has been slightly different, the fitness function of the algorithm is not just the LCOE but a parameter that decreases the rank of the solution if it does not respect the constraint on the reliability. Since the parameters that decrease the rank of the solution are arbitrary, a sensitivity analysis has been done to see the influence on the results.

The fourth step has been introducing a price of electricity from the geographical position chosen and analyzing how the system performs in that scenario, in the case of the system with cogeneration it has been analyzed the possibility to produce heat at the same cost as natural gas.

The last step has been the analysis of the results and understanding of how the different plants could be implemented cost-effectively depending on the relationship with the grid.

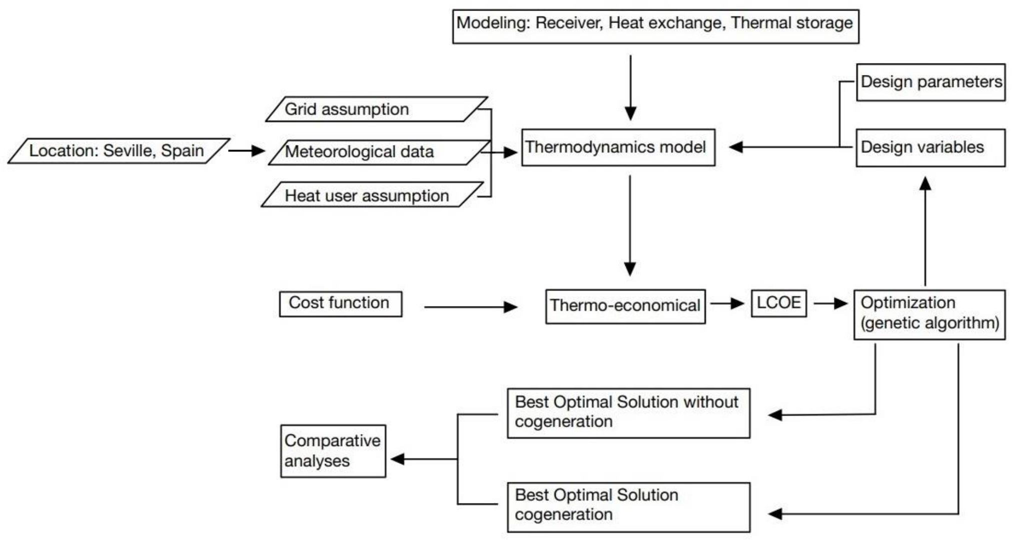


Fig. 11 Methodology scheme

3.1 Optimization

This chapter explains the optimization procedure and the design variables used.

3.1.1 Design variable

In this short paragraph it has been explained the physical meaning of design variable use in this study.

- Maximum power injectable into the grid (P_{max}): it indicates the maximum power injectable in the electric grid, it corresponds to the summer peak.
- Capacity of the CSP (P_{CSP}): it indicates the power capacity of power block, it is connected to P_{max} by the formula 3
- Solar multiple (SM): it indicates the thermal power of the receiver considering the nominal thermal power required by the power block (PB) see formula 1

$$SM = \frac{Q_{receiver}}{Q_{nominal\ PB}} \tag{1}$$

- PV ration (PV): it indicates the ration between the capacity of the CSP and the PV power installed formula 2.

$$PV = \frac{PV}{P_{CSP}} \tag{2}$$

- Thermal storage (t): it indicates the number of hours for which the thermal storage (TES) can support the power block at its nominal capacity
- Alfa (α): it indicates the size of the power block in respect to the peak energy demand see formula 3

$$\alpha = \frac{P_{CSP}}{P_{max}} \tag{3}$$

- Thermal load (Q): it indicates the peak of the thermal load on the plant, its design variable is represented in the formula 4

$$Q = \frac{Thermal\ load\ peak}{P_{max}} \tag{4}$$

After the definition of the design variable, their ranges have been defined table 1 summarized the ranges.

Table 1. Design variables and their range

Design variable	Without cogeneration	With cogeneration
P_{max}	50 ÷ 200 MW	130 MW
PV	1.1 ÷ 2.4	1.6 ÷ 2.7
t	8 h ÷ 12 h	10 h ÷ 14 h
α	0.8 ÷ 1	1
SM	1.1 ÷ 2.4	1.6 ÷ 2.7
Q	0	0.01 ÷ 0.4

The range of the PV ratio, storage time, and solar multiple has been found in the literature similar study case (2), on the other hand for alpha and the thermal load the approach was different. The rage of alpha has been selected to show the beneficial effect on the LCOE thus, values lower than 0.8 has not been considered because there is no benefit on LCOE going lower than values.

The range of thermal load has been selected in such a way that has been possible for the techno-economical thus, large enough to determine the minimum thermal load for which there is an economical benefit.

The lowest power size investigated is 50 MWe because there are some limitations regarding the modelling of some components for example the receiver and the main heat exchanger, while the higher power size investigated has been 200 MWe because the CSP plants with higher capacity are quite rare and not so realistic at the current state of the art of these technologies.

3.1.2 Genetic algorithm

The optimization has been based on the minimization of LCOE, the main issue is that each simulation takes around 90 seconds to be computed and considering all the possible combinations amount all the variables it will take around 120 days of simulations thus, this approach of calculating for finding the minimum LCOE is not efficient. This problem is typical in the field of simulations thus, there are several approaches possible, for this optimization have been implemented a genetic algorithm.

For the calculation of LCOE has been used the formula 5, the parameters used are the following:

- $CAPEX$: capital expenditures
- i: interest rate
- t: lifetime of the pant
- E: annually energy produced
- $OPEX$: operational expenditure

$$LCOE = \frac{CAPEX\left\{\frac{i(1+i)^t}{(1+i)^t-1}\right\}+OPEX}{E} \tag{5}$$

The genetic algorithm is inspired by the natural selection phenomena, the main components of the algorithm are the following:

- Representation of a solution: a solution is represented by a specific combination of the design variables, the vector that identifies the solution is called the genome, for example [i,j,k,l,m...] each number represents the index of the design variable
- Iteration process: the algorithm works in such a way that at each iteration the accuracy of the results increases, at each iteration the program calculates n number of solutions this number is called population size
- Fitness function: it represents the criteria with which the solutions are ranked, in this case, the fitness function is the LCOE
- Selection of the solutions: using the fitness function the best solutions are selected for the generation of the solution of the next generation
- Generation of new solutions: At each iteration, a new set of solutions is generated based on the solution with the higher rank at the iteration before.
- Random mutation: At each iteration, for each solution, a random mutation to a random index in the genome is applied to avoid the algorithm will be blocked in a local minimum

The parameters which have been used are summarized in the table 2.

Table 2. Parameters used for the genetic algorithm

Parameter	Value
Population size	10
Minimum rank	4
Random mutation	1

3.1.2.1 With cogeneration

In the optimization with the cogeneration, the first difference has been the introduction of a penalty function, the fitness function used in this optimization is reported on the formula 6.

$$Fitness\ function = LCOE * Penalty(fd) \tag{6}$$

The penalty function represents the constrain on the reliability of the heat production, the fd defines the fraction of heat supplied over the theoretical one in one year formula 7.

$$fulfilled\ thermal\ demand\ (fd) = \frac{Heat\ supplied}{Theoretical\ heat\ suppliable} \tag{7}$$

Fig.12 shows the penalty function used; it is a linear interpolation between two points summarized in table 3 The goal of the heat production is to supply 0.95 of the yearly thermal demand thus, this is one of the interpolations the other has been set to have fast convergence.

Table 3. Values assumed for defining the penalty function

Fulfilled thermal demand	Penalty function
0	4
0.95	1
1	1

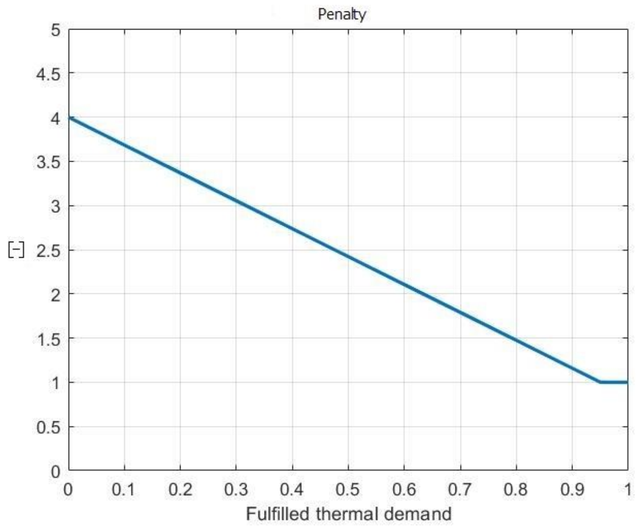


Fig. 12 Penalty function

However, this penalty function is arbitrary and for this, a sensitivity analysis has been done to evaluate the influence of the penalty function on the results, the sensitivity analysis shows that there is not a remarkable difference considering different values. The second difference has been that alpha is considered unitary because of the need to reduce the number of design variables.

4 Modelling

In this paragraph, it has been explained the modeling of the components used in the analysis. The model that has been used is based on a quasi-static analysis without considering the transient phenomenon. Fig. 13 shows the layout of the system that has been modelling.

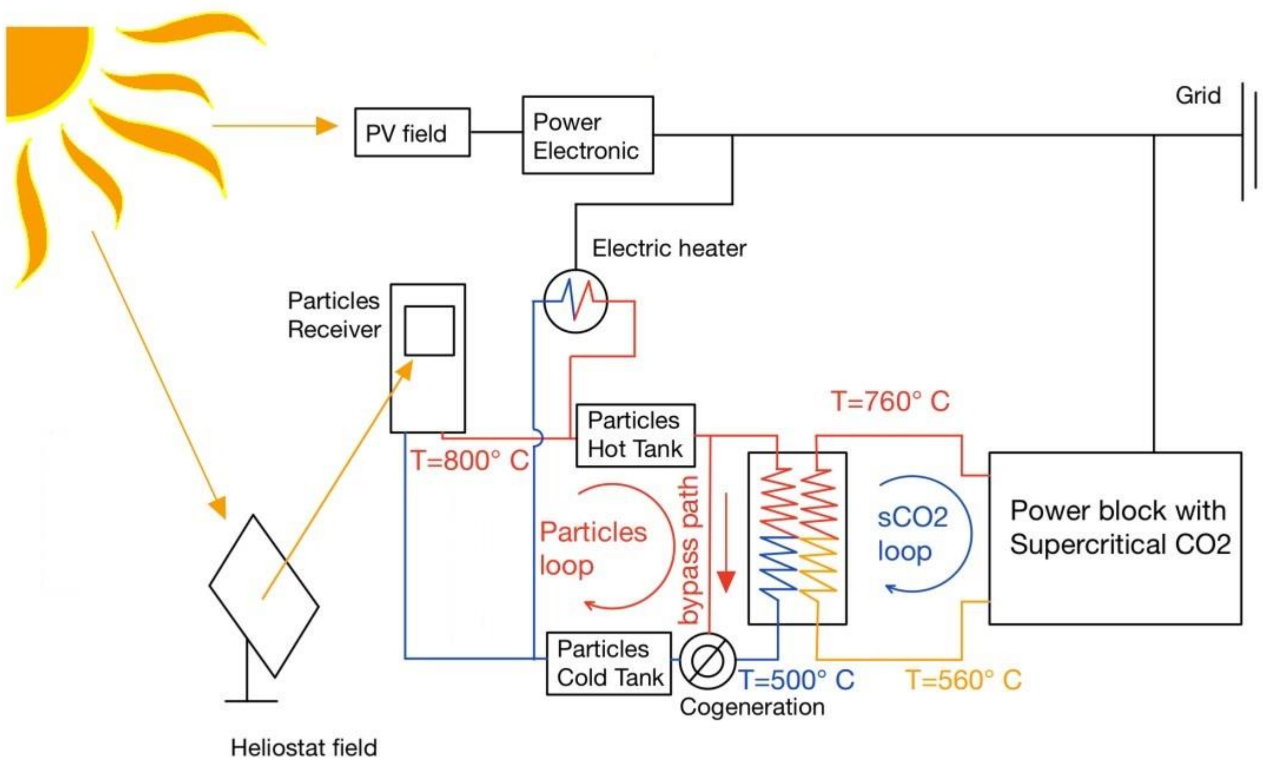


Fig. 13 Scheme of the plant

4.1 Modelling the receiver

The receiver that has been chosen is called a free-falling particle receiver, the reason behind this choice is because of its competitive costs and thermodynamic performance (15). **Fig.14** shows the scheme of it, the idea of this receiver is that the particles are falling from above and particles are heated up from the concentrated solar irradiance from the heliostats field.

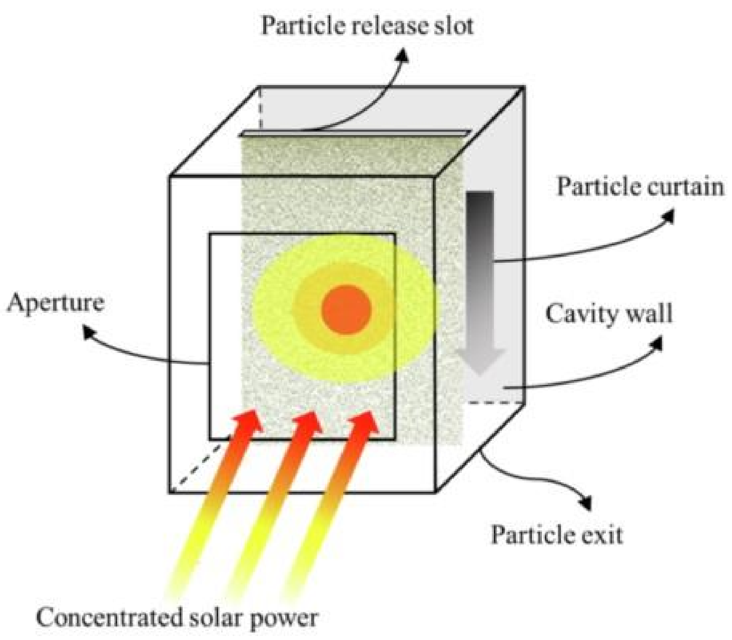


Fig. 14 Scheme of the particle receiver

The techno-economical model of the receiver is based on the following assumptions:

- The thermodynamic efficiency is a function of the DNI and the types of the particles (15)
- The heat losses and mass leakage have been included in the thermodynamic efficiency
- The cost of the receiver depends only on the thermal capacity installed

Considering these assumptions is not possible to evaluate the change in the efficiency for different temperature ranges that is a limitation of the receiver model, furthermore, during the cogeneration implementation, it will not be possible to make a consideration about the changing performance of the receiver.

The efficiency of the receiver is plotted in **Fig. 15** (15), it is specified for this type of receiver and the sand particles.

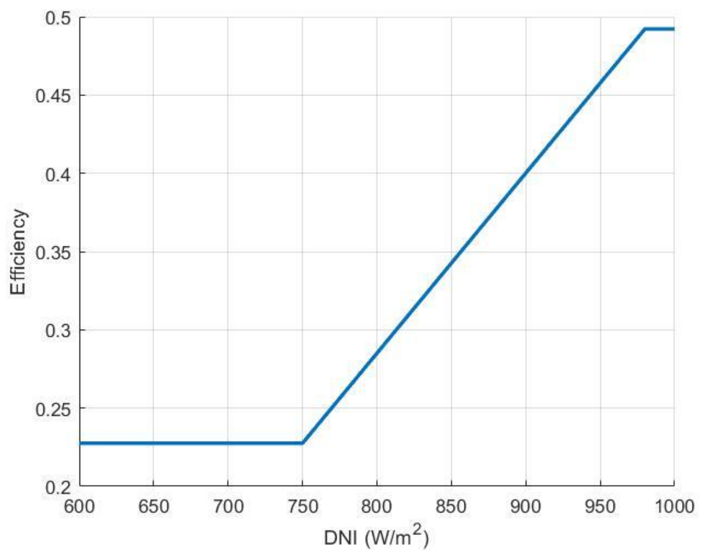


Fig. 15 The efficiency of the receiver as a function of the DNI

Formula 8 is the formula used for calculating the thermal power that the receiver gives to the particles.

$$Q_{receiver} = Q_{heliostat} * \eta_{receiver} \tag{8}$$

- $Q_{receiver}$: Useful thermal power
- $Q_{heliostat}$: Thermal power on the receiver from the heliostat field

The working temperatures of the receiver are listed in table 4 (15), and the output temperature is fixed. However, the temperature of the particles entering the receiver depends on the working conditions of the plants, in the case without cogeneration the temperature is around 500 °C and in the case with cogeneration is around 400 °C because of the heat extraction.

Table 4. Main values for the modelling of the receiver

Parameter	**Value**
T_{out}	800 °C
$T_{in\ minimum}$	300 °C
$\eta_{effective}$	0.2-0.5
$Specific\ cost$	150 €/kWth

- T_{out}: it is the outlet temperature of the particle going out from the receiver to the TES
- $T_{in\ minimum}$: it is the temperature from the cold tank going to the receiver to be heated up, it changes according to the conditions of the cold tank, but it cannot be lower than $T_{in\ minimum}$ because it is outside of the working condition of the receiver.

The economic model is based on a specific price considering the installed capacity of thermal power installed, C=150 €/kWth (15) valid for power installed higher than 100 MWth.

4.2 Modelling of the heat exchanger for heating the sCO2

The heat exchanger that has been used for connecting the particle loop with the supercritical CO2 is shown in **Fig.16** (16), it is the type of indirect heat exchanger because there is no direct contact between the particles and the supercritical CO2. The particles are falling from above the heat exchanger, most of the external pipes have the function of recirculating the air to increase the effectiveness of heat exchange and the remaining ones are used for the circulation of the supercritical CO2.

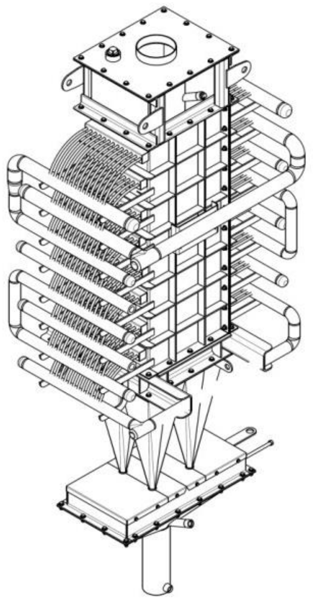


Fig. 16 Shows the particle heat exchanger

The techno-economical model of the heat exchange is based on the following assumptions:

- The thermal power exchanged between the particles and the sCO2 is a function of the NTU
- The heat losses and mass leakage have been neglected
- The pressure drop has been neglected because of their relatively low value compared to the pressure involved (16)
- The cost of the heat exchanger depends only on the thermal capacity installed

Considering these assumptions, the main limitation is that the cost of the heat exchanger depends only on the thermal capacity installed and it is decoupled from the effectiveness, therefore, it is not possible to make considerations about the effectiveness and the cost of the heat exchanger. The thermodynamic modelling has been carried out using the experimental data [16] see **Fig. 17**. The first step for the design of the heat exchange was imposing its working temperature in the design point (16), table 5 summarized the design temperature and the specifics of the heat exchange.

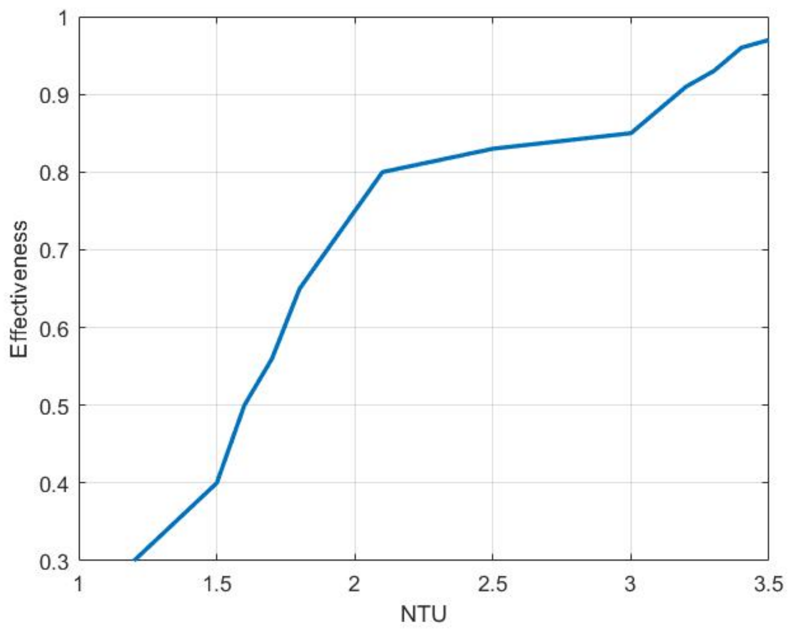


Fig. 17 Effectiveness as a function of NTU experimental data

Table 5. Main values for the modelling of the heat exchanger that provides the thermal power to the power block

Parameter	**Value**
Effectiveness design point	0.85
T in particle	790 °C
T out particle	590 °C
T in sCO2	560 °C
T out sCO2	770 °C
U (Overall Heat Transfer Coefficient)	100 W/K*m^2
$Specific\ cost$	440 €/kWth

In the following list has been explained the parameters used for the modelling of the heat exchanger:

- $Effectiveness\ design\ point$: it is the effectiveness that the heat exchanger has at its nominal working point
- $T_{in\ particle}$: it is the temperature of the particle coming from the hot tank (TES) going to the heat exchanger to heat up the sCO2, it is fixed
- $T_{out\ particle}$: it is the temperature of the particle coming from the heat exchanger and going to the cold tank, it might have a little variation in the off-design of the heat exchanger
- $T_{in\ sCO2}$: it is the temperature of the cold sCO2, it might have a little variation due to the variation of the ambient temperature
- $T_{out\ sCO2}$: it is the temperature of the hot sCO2 that will go to the turbine, it is fixed to have the same working condition for the power block
- $U\ (Overall\ Heat\ Transfers\ Coefficient)$: it is the coefficient that regulates the heat exchanger used to determine the area of the heat exchanger

The area of the heat exchanger has been calculated using the scheme in **Fig. 18**, the first step has been calculating the mass flow rate needed (formula 9-10) to be able to fulfil the thermal power requirement of the power block at the design point, and the second has been combining the information of the effectiveness at the design point and the information about the mass flowrate and determine the NTU, at last, an inverse formula of NTU (formula 11) has been used to calculate the area.

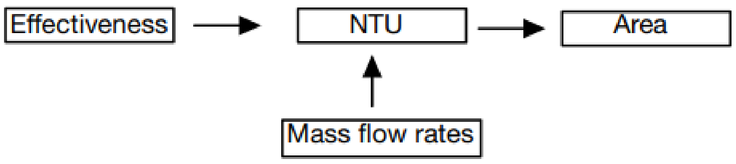


Fig. 18 Scheme of determine the area of the heat exchanger

$$m_{particles\ design} = \frac{Q_{design}}{h_{in\ Particles} - h_{out\ Particles}} \tag{9}$$

$$m_{sCO2\ design} = \frac{Q_{design}}{h_{in\ sCO2} - h_{out\ sCO2}} \tag{10}$$

$$NTU = \frac{U * A_{heat\ exchanger}}{C_{min}} \tag{11}$$

The specific cost of the receiver that has been chosen is 440 €/kWth (16) valid for power size higher than 100 kWth.

4.3 Modelling the thermal energy storage

The technology that has been chosen is the silo type for storing the heated particles see **Fig. 19** (17) because this is the most competitive technology using particles, **Fig. 19** shows the system of transportation of the particles.

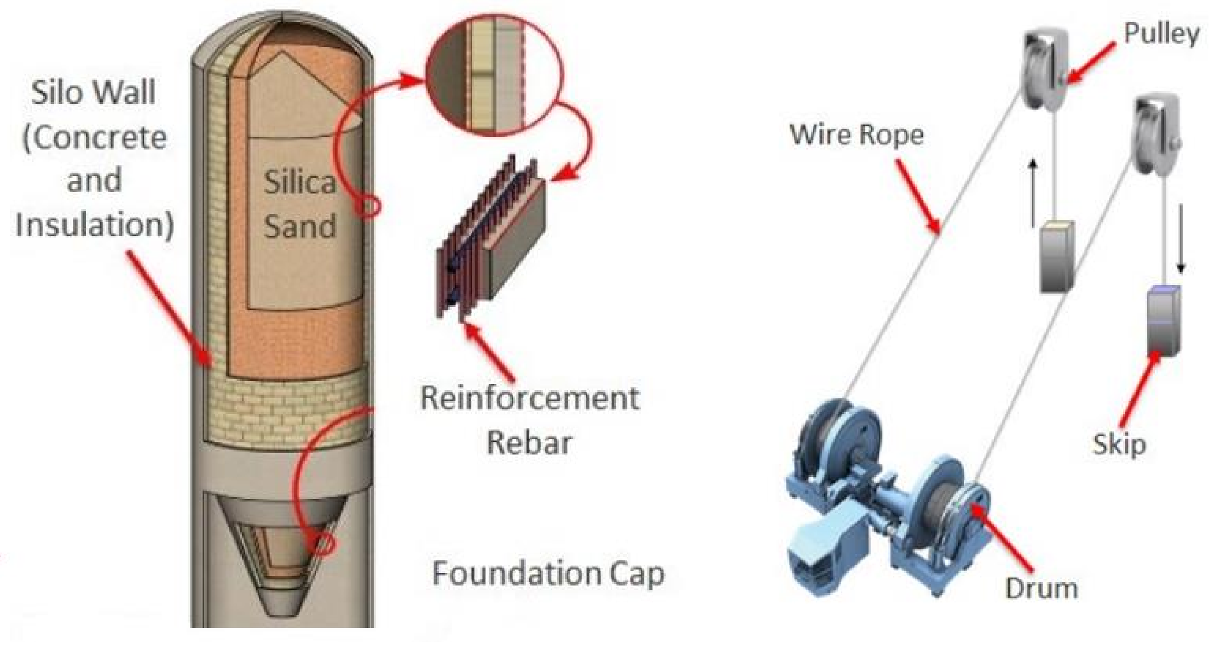


Fig.19 Thermal energy storage technology and the movement system

The techno-economical model of the heat exchange is based on the following assumptions:

- The heat losses coefficient is constant
- The cost of the thermal storage depends on the thermal energy installed

Considering these assumptions, the main limitation is that it is not possible to evaluate the heat losses for different environment condition such a different wind speed.

The thermodynamic model used is quite simple since this type of TES has high efficiency (>95) (17), and the heat losses have been calculated using the electrical analogy formula 12-15. **Fig. 20** (17) shows the equivalent thermodynamic model used

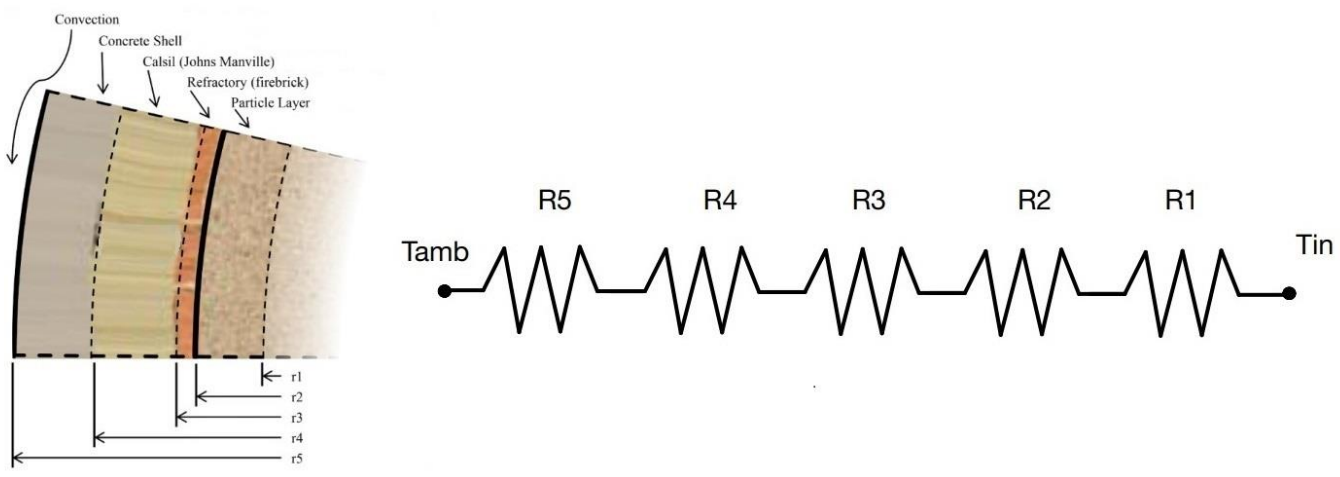


Fig. 20 Thermal coefficient of each layer and the thermal resistance model

$$\emptyset_{losses} = h\,A\,(T_{in} - T_{amb}) = \frac{1}{R_{tot}}(T_{in} - T_{amb}) \tag{12}$$

$$R_{tot} = \sum_{i=1}^{5} R_i \tag{13}$$

$$R_i = \frac{\log\left(\frac{r_{k+1}}{r_k}\right)}{2*\pi*k_k*H} \tag{14}$$

$$R_5 = \frac{1}{A\,h_{air}} \tag{15}$$

In the following list has been explained the parameters used for the modelling the TES:

- $\emptyset_{losses}$: it is the heat losses in the environment
- A: it is the external area of the tank
- h: it is the overall heat transfer coefficient
- T_{in}: it is the average temperature of the tank
- T_{amb}: it is the temperature of the environment
- R_i: it is the thermal resistance that represent a specific layer
- R_5: it is the thermal resistance of the air, it represents an average value through the year

Table 6. summarized the value used for the calculation of the heat losses and modelling the TES

Parameter	Values
Height of the tank	25 m
Conductivity of the layer n°1	0.7 W/m*K
Conductivity of the layer n°2	0.2 W/m*K
Conductivity of the layer n°3	0.1 W/m*K
Conductivity of the layer n°4	0.8 W/m*K
Convection of the air	5 W/m^2*K
Specific cost	25 €/kWh

The second step was defining the size range of the TES depending on the thermal energy to be stored, which depends on the storage time and the capacity of the CSP. On this range has been evaluated that the h coefficient does not change significantly concerning the size and the storage time see **Fig. 21** because of that it has been considered a constant value of h for the whole studying.

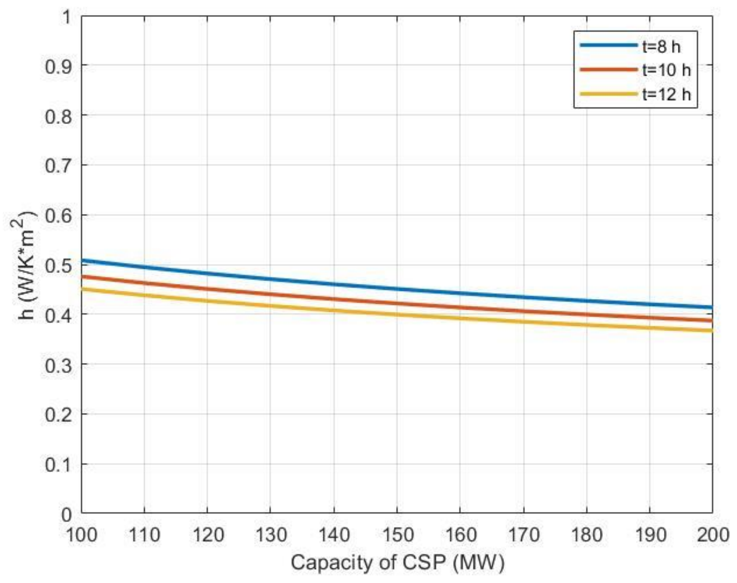


Fig. 21 variation of the h coefficient considering different size of the TES

The cost of the thermal storage has been calculated considering the specific cost of 25 €/kWh (17).

4.4 Modelling the cogeneration

The cogeneration implementation has been made in such a way that the heat production is done reliably therefore, the system has the possibility to bypass the main heat exchangers when the power block does not work. **Fig. 22** show the scheme of the cogeneration implementation. Due to the constrain of the reliability, the thresholds of the TES for the activation of the PB are higher because there is the need of having enough heat for the cogeneration otherwise, the PB would use all the heat for the electric conversion. With this change, the system in case of limited thermal energy will choose to support the thermal load instead of the electric load.

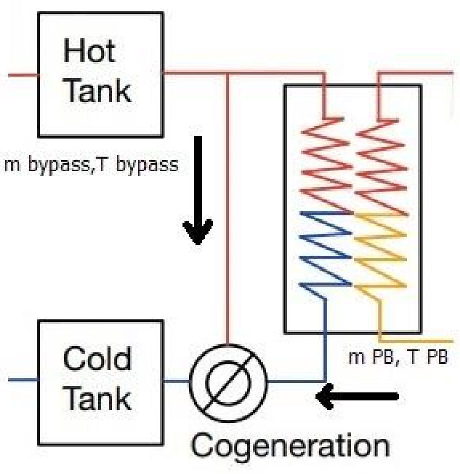


Fig. 22 Scheme of the plant related to the cogeneration

The working temperature of the heat exchanger are summarized on the table 7. The lowest possible temperature that the heat exchanger of the cogeneration brings the particles is Tmin which depends on the receiver.

Table 7. Main values for the modelling of the cogeneration

Parameter	Value
T_{bypass}	790 °C
$T_{power\ block}$	500 °C
T_{min}	300 °C
$T_{average}$	400/450 °C
$Specific\ cost$	110 €/kWth

The formulas that have been used are coming from the direct application of the first principle of thermodynamics.

$$Q_{cogeneration} = m_{particles} * (h_{particles} - h_{min}) \tag{16}$$

$$h_{particles} = \frac{m_{power\ block} * h_{power\ block} + m_{TES} * h_{TES}}{m_{particles}} \tag{17}$$

In the following list has been explained the parameters used for the cogeneration:

- $Q_{cogeneration}$: it is the thermal power going to the heat user
- $m_{particles}$: it is the mass flow rate through the heat exchanger of the cogeneration and going to the cold tank, when the power block is working that mass corresponds to the mass flow rate needed to heat the sCO2 otherwise, the mass flow is taken directly from the TES
- h_{min}: it is the minimum enthalpy at which the cogeneration is working, it is a direct consequential of the receiver minimum inlet temperature
- $h_{particles}$: it is the enthalpy of the particles before the cogeneration when the power block is operating h_p is equal to the outlet enthalpy after heating the sCO2 otherwise the enthalpy of the particles is equal to the outlet enthalpy of the TES

Fig. 23 shows the control unit for the cogeneration. During the operation ration of the PB, the remaining heat from particles heat is used for the cogeneration if that heat is not enough to supply the thermal there is the necessity for heat extraction directly from the TES. When the PB is not working all the heat is extracted from the TES. In the case of PB being off and there is not enough thermal energy in the TES, the supply of heat is partial or null. The specific cost of the heat exchanger in this analysis has been assumed based on a commercial heat exchanger air to water, the specific cost (18) of the commercial heat exchanger has been multiplied by a factor of 1.3 because it uses air instead of particles. The specific cost used for the cogeneration implementation is 110 €/kWth.

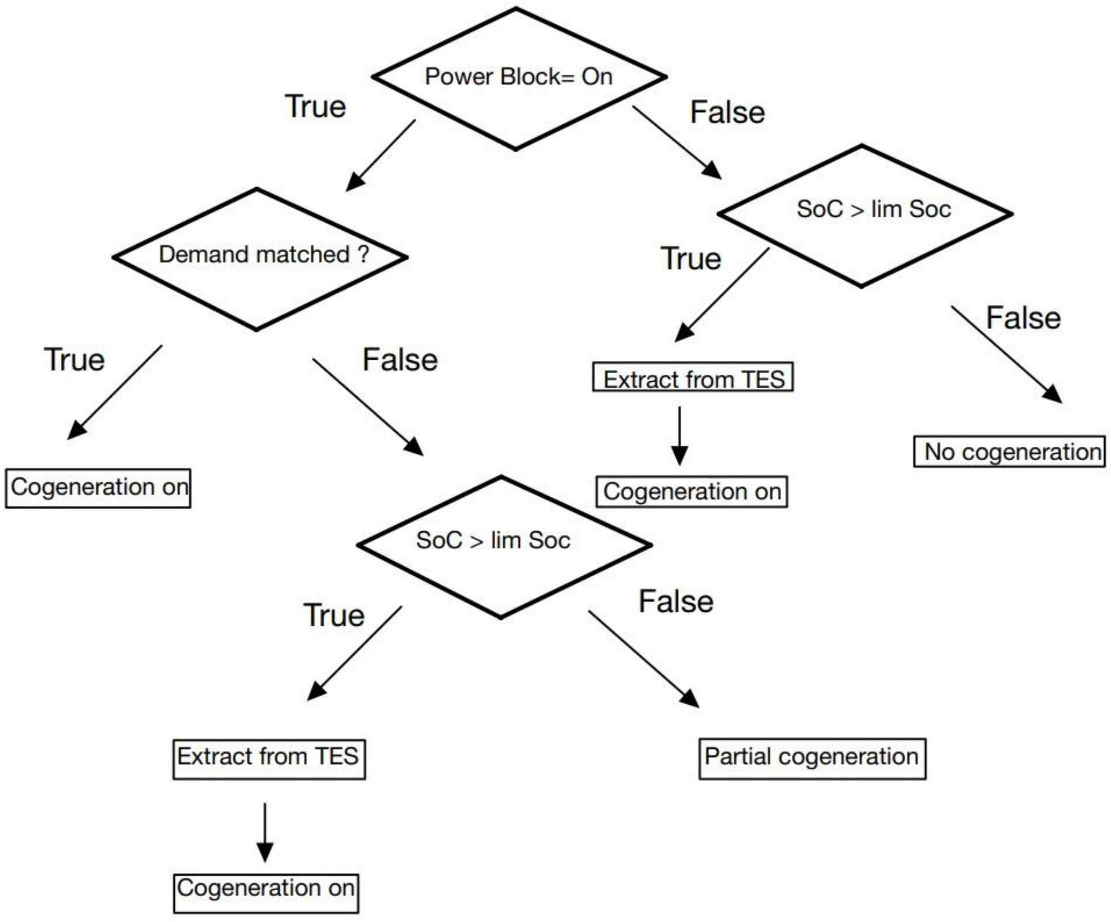


Fig. 23 Flow chart of the control system for the cogeneration

4.5 Modelling the electric grid

To understand the behavior and to evaluate the capacity factor of the system in one year a variable electricity demand has been considered because the electrical load to support changes in each month and for each hour. It has been considered 4 normalized electric loads for each season, see **Fig. 24** (19), the intermediate curve has been evaluated through multiple interpolations, and the result is that for each hour of the year it has been calculated the normalized electricity demand has. The electric load data has been selected from the Spanish database according to the geographical location of the plant. To allow the system to support the load in absence of solar radiation the maximum power injectable into the grid is equal to the CSP capacity, however, the CSP depends on the alpha coefficient which is a design variable.

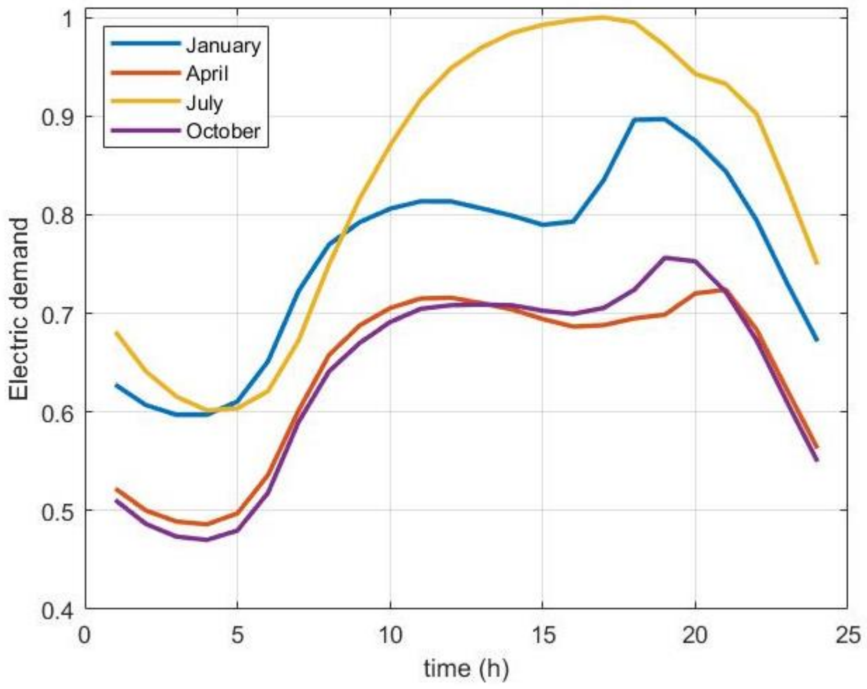


Fig. 24 Electric load for each season

Due to the electric load assumed the capacity factor (see formula 18) is limited to 0.72 because the maximum power injectable in the grid is usually lower than P_{max} thus, even in the best scenario when it is possible to supply all the demand Cf is limited. The Cf limit has been calculated using the formula 18 and substituted in the energy to produce the maximum energy injectable into the grid.

$$Capacity\ factor\ = \frac{Energy\ produced}{24*365*P_{max}} \tag{18}$$

4.6 Modelling the thermal load

The thermal loads considered are assumed to be the typical thermal need for industrial applications because of their temperature, **Fig. 25** shows the thermal load profile.

- The first thermal profile considered has been a continuous load hence, the system must be able to supply heat continuously.
- The second has been a shift base load thus, the starting heat request is at the start of the shift it has been supposed at 8:00 and the ending time is at the end of the shift it has been supposed at 18:00.

The two-load profile have in common the peak power, but the energy that they extract from the system is different by a factor of around 2.4. This difference leads to a higher impact on the system because in the case of continuous load the system has to produce 2.4 more heat, Fig.32 shows that the impact on the LCOE for the same peak thermal power is remarkable higher in the case of continuous load.

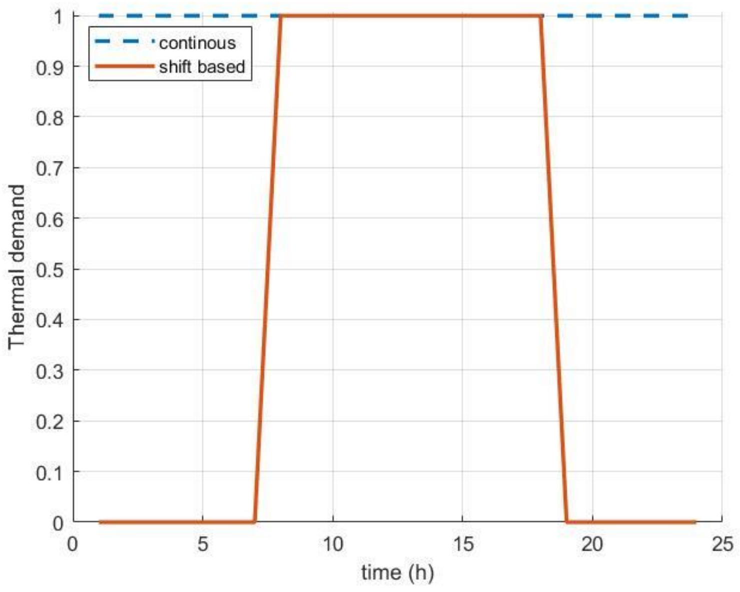


Fig. 25 Normalized thermal load over the time

4.7 Modelling the economic scenario

Most of the study has been based on the minimization of the LCOE, the last step of the analysis has the goal of evaluating the economic performance of the system considering the actual price of the electricity and the heat. The assumption behind this approach is the electricity and the heat are sold at the market price.

In this scenario, a variable price of electricity has been considered (19) the intermediate price of the electricity has been calculated using the same approach used in the maximum electricity injectable into the grid. The price of electricity (**Fig. 26**) is the one for 2019, it has been considered also the present prices (2022) that are strongly affected by the war in Ukraine. The present prices of electricity and natural gas represent a scenario where there is the economic pressure of finding to reach the independence of energy production. The price of electricity for the year 2022 has been considering the trend 2.5 times more than the 2019 (19). The main reason for introducing the variable price is to take into account the variable in the price of the different seasons

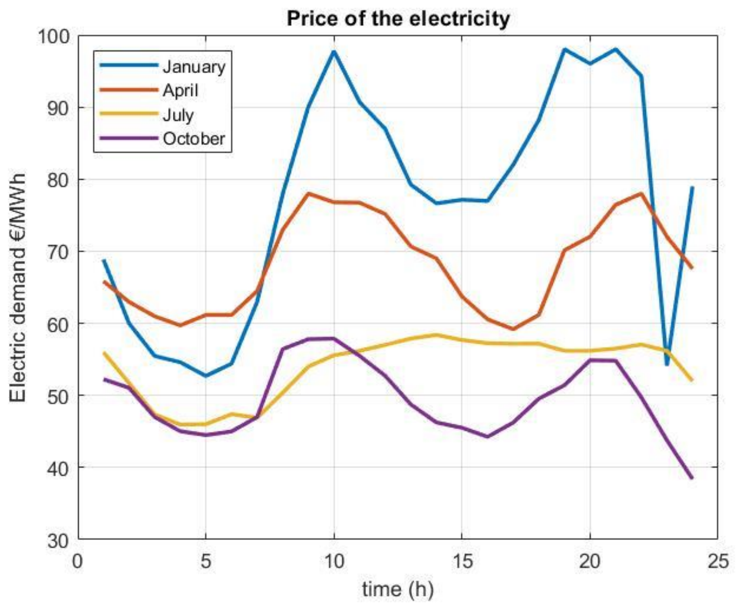


Fig. 26 Price of the electricity for each season

The price of the heat has been considered fixed to the same LCOH of natural gas (20) considering the year 2019 and 2022 (21).

Table 8. The cost of the heat from natural gas in two different years

Year	Price of natural gas (€/MWh)	LCOH (€/MWh)
2019	17	55
2022	80	118

4.8 Modelling the total land usage

An important aspect of the plant is its footprint, the size related to the plant of the system has been estimated based on the similar power installed of PV and CSP operating in the world. The footprints have been calculated for the system without cogeneration and with cogeneration in the best optimal solution, formula 19 has been used for the calculation of the land needed. Table 9 ((22), (23), (24), (25), (26)) indicates the values used for the calculation, these values have been calculated based on the values found for the plant of a similar capacity and storage time of the study case. Cerro Dominador Solar Thermal Plant (Antofagasta, Chile DNI 3200 kWh/m^2 (27)) is the name of the power plant considered as a base case, the plant is hybridized with PV, and it has a similar capacity to the configuration of which the footprint has been estimated. The capacity of the base case is 100 MWe (28) hence, the land usage has been increased by a factor $beta$ (see formula 19) to consider the difference in the power installed. The base case has a land usage of 10 Km^2 considering a capacity of 130 MWe.

$$beta = \frac{Capacity\ considered}{Capacity\ of\ the\ base\ case} = 1.3 \tag{19}$$

The ratio between the size of the receiver has been introduced to take into account the oversize of some components in the case of cogeneration thus, in the case of cogeneration the electric capacity is the same but the size of the heliostat field and the receiver increase. In the case where there is no cogeneration that ratio is unitary otherwise, that ratio is bigger than the unit.

$$Land = \ Specif\ area_{PV} * Capacity_{PV} + \frac{Receiver}{Receiver\ REF} * Specific\ area_{CSP} * Capacity_{CSP} \tag{20}$$

In the following list has been explained the parameters used for the modelling the TES:

- $Specif\ area_{PV}$: it is the specific land usage with respect to the capacity of the photovoltaic installed
- $Specific\ area_{CSP}$: it is the specific land usage with respect to the capacity of the CSP installed
- $Capacity_{PV}$: it is the capacity of the PV installed
- $Capacity_{CSP}$: it is the capacity of the CSP installation
- $Receiver$: it is the thermal capacity of the receiver considering the cogeneration
- $Receiver\ REF$: it is the thermal capacity of the receiver of the base case without the cogeneration at the specific size of the plant.

Table 9. Specific areas calculated from actual plants

Parameter	Value
$Specif\ area_{PV}$	$0.028\ \frac{Km^2}{MW}$
$Specific\ area_{CSP}$	$0.05\ \frac{Km^2}{MW}$

5 Analysis of results

In this chapter, the results of several simulations have been presented. The results have been divided into two paragraphs, one considering without the cogeneration and the anther considering with the cogeneration.

5.1 Without cogeneration

The optimization using the genetic algorithm gave the optimal solution, which is summarized in table 10, from this solution several plots have been obtained (**Fig. 27-29**) fixing all the design variables at their optimal values (minimum of LCOE) expects P max and the specific parameter.

Table 10. The best optimal solution in the case without cogeneration

Design variable	**Value**
P_{max}	130 MW
SM	1.8
t	11 h
α	0.85
PV	2.4

Fig. 27 the plot at the left has summarized the calculation for mapping the minimum of LCOE in the design variable done by the genetic algorithm. The plot is quite scattered, to be sure that the actual minimum of LCOE is the one shown on the left of **Fig. 27** a sensitivity analysis considering the design variables has been done on the minimum LCOE found by the genetic algorithm. The right plot of **Fig. 27** shows that the minimum found by the genetic algorithm is the same that has been found in the sensitivity analysis. Moreover, the black window in **Fig. 27** shows that the system has several solutions that have an LCOE of around 82 €/MWh with different CAPEX, which offers some flexibility to the system because it can have the same LCOE but if a lower capital investment. The configuration with the lowest power size (50 MWe) presents an LCOE of 88 with almost the lowest CAPEX in all configurations simulated.

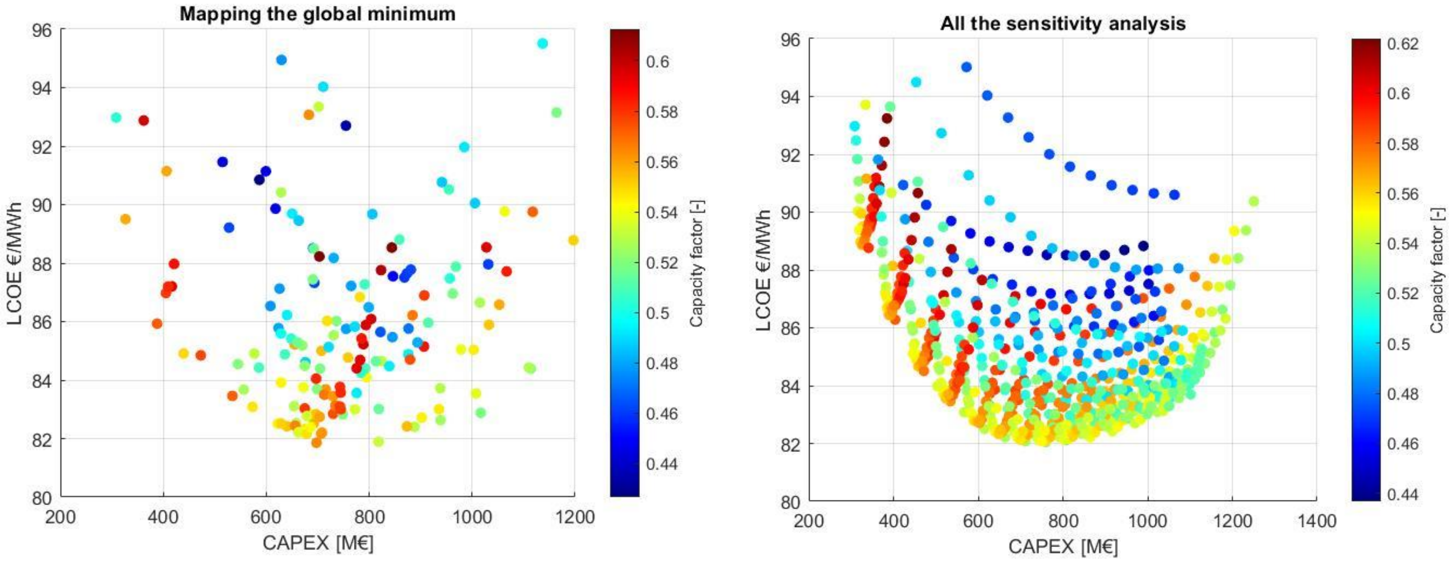
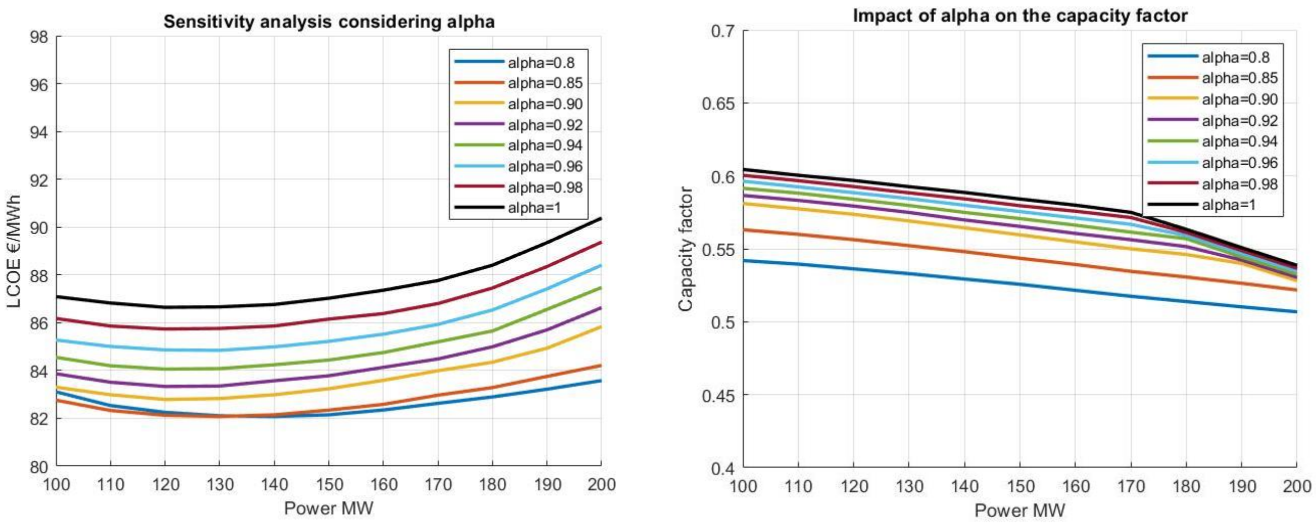


Fig. 27 Mapping the global minimum using the genetic algorithm and the sensitivity analyses on the minimum found in the mapping step

In **Fig. 28** it is possible to see the effect of the alpha it has a beneficial effect on the LCOE with the downside of reducing the capacity factor, the choice of the specific alpha depends on the agreement with the electric grid, it the plant should supply a baseload alpha can be lower to reduce the size and the cost of the power block. It is possible to see those values close to the unit have remarkable benefits on LCOE with a limited influence on the capacity factor. An important observation is that the $Capacity\ factor$ is limited at 0.72 with the electric load chosen (see paragraph 4.5), which is a value like the one obtained for the different values of alpha considered. This shows that the system can reliably produce electricity in absence of sunlight and that is a



high-value asset.

Fig. 28 The influence of alpha on the LCOE and Cf on the system

Fig. 29 shows the effect of the solar multiple, PV ratio and the storage time on the LCOE. It has been plotted the most significant values to make the representation clear and comprehensive. The design variable that affects the most the LCOE is the storage time because it has a strong impact on the electricity of the power block, it is possible to notice that lower values of t, which is an indication of the capacity of the TES, have a remarkably higher value of LCOE.

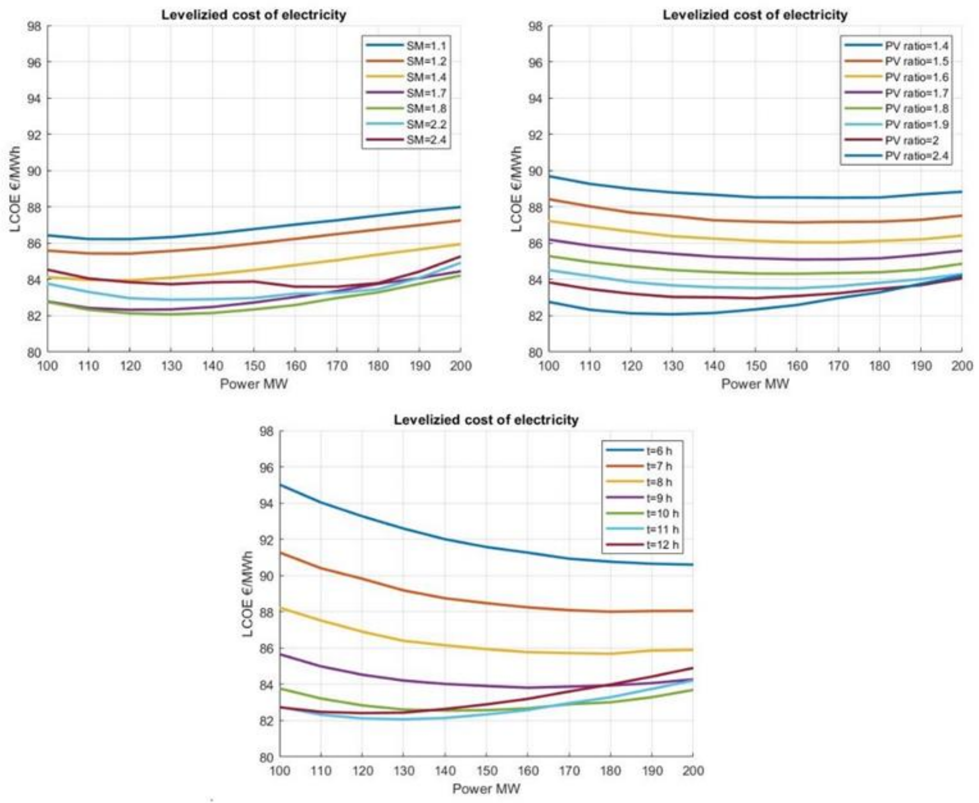


Fig. 29 The effect on LCOE considering PV ration, storage time and solar multiple

The behavior of the system considering two typical days one in summer and one in winter is shown in **Fig. 30,** it shows that the influence of alpha does not significantly affect the electricity production in winter because the electricity demand is lower thus, the system can cover almost the same percentage of the demand with a lower cost of the components. **Fig. 30** shows the working principle of the system, during the sunlight the plant produces electricity using the PV, while at night-time it produces the electricity using the CSP (power block). Generally, the PV and the power block are working in opposite phases thus, they do not work at the same time for the following reasons:

- The power block minimum working load is $0.4\ P_{CSP}$
- There is no reason not to produce electricity with PV when it is possible, the PV field is oversized this allows the system to cover the electrical demand normally in the daytime

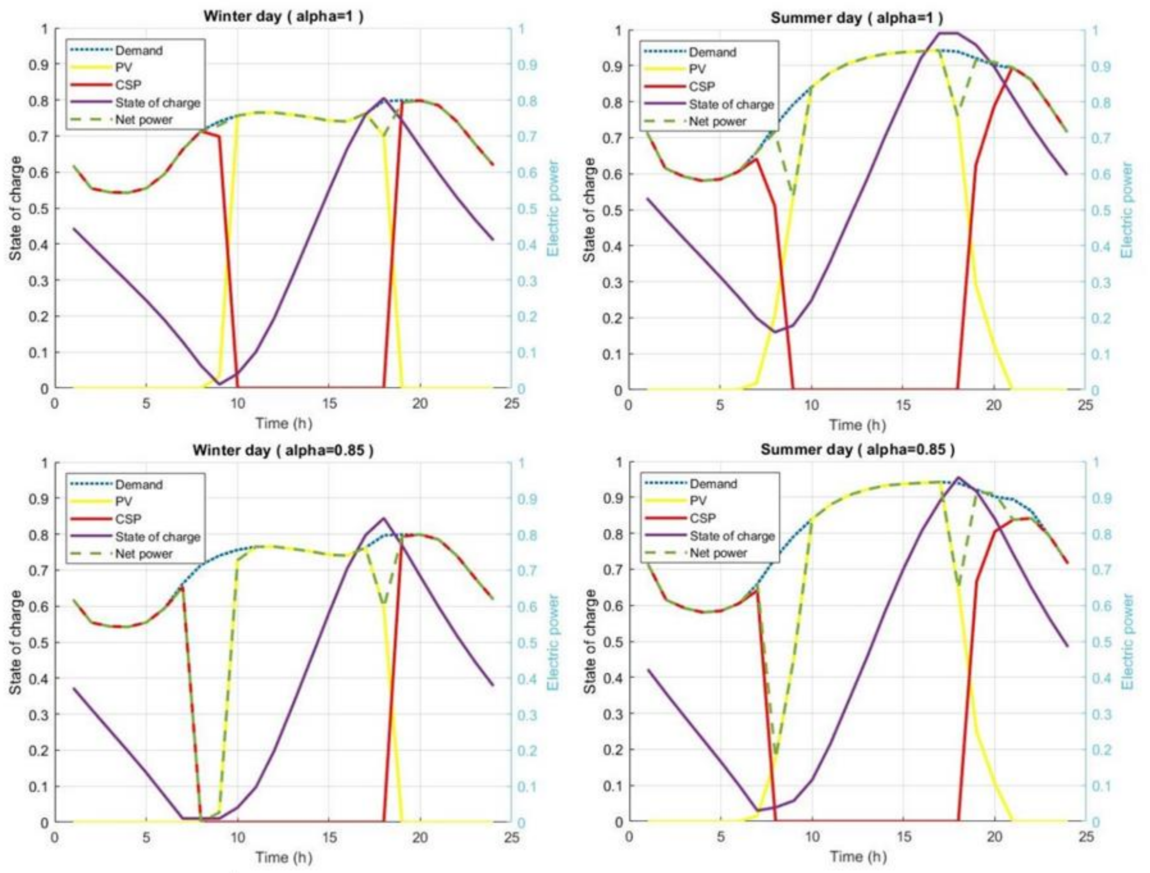


Fig. 30 shows the behavior of the plant and the influence of alpha

5.1.1 Performance in the scenario

The payback time has been evaluated to show that the system has a lower payback time than its lifetime thus, the technology could be profitable and have some market attraction. The analysis that has been done is the one for the evaluation of the payback time of the system without cogeneration, formula 19 has been used for finding the payback time. Fig. 31 shows the point of NVP=0, which corresponds to the payback time, for two different values of alpha. The alpha has a beneficial impact also in the CAPEX without impacting remarkably on the revenues. The two systems have a similar payback time because there is compensation between a lower CAPEX and lower revenue. It has been considered 1 year for the construction of the plant. In this context, the payback time is an indication that the system can compete in the energy market considering the actual prices without any government substitutes. Fig. 31 shows the difference of the NPV over time considering two different prices scenario, due to the Ukrainian conflict the price of electricity has increased significantly, which leads to a lower payback time.

$$NetPresentValue\ (NPV) = Revenues - Cost \qquad (21)$$

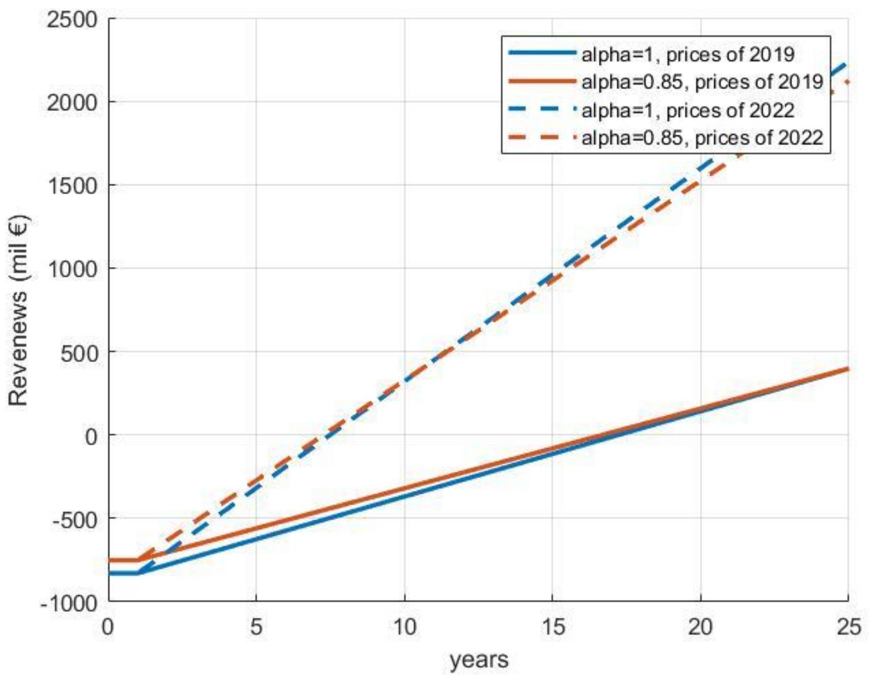


Fig. 31 NPV considering two prices scenario

Table 11 summarizes the payback time considering the different conditions, the big difference in the payback time is due to the dependence between the price of the electricity and the cost of the natural gas, which have been increased in 2022 due to the war in Ukraine.

Table 11. Payback time considering two different price scenarios

Year considered	Payback time (year)
2019	7 years
2022	17 years

5.2 With cogeneration

The design variables that have been considered for the cogeneration analysis are summarized in table 1.

The LCOE of the systems increases because of the following reasons:

- The lower amount of heat available for the power block thus, the amount electricity produced is lower
- Due to the constraint on the reliability of the heat generation, some components are oversized to introduce more heat into the system hence, the total costs are increasing
- The system produces less electricity due to the changing of the minimum threshold of the TES to activates the power block

Fig. 32 shows the changes in the LCOE and the different sizes of the components, the reason for the increasing size of the components is due to the increasing heat requested by the system to operate properly. The overall trend of the Cf and the component size are intuitive, a lower electricity production leads to a lower Cf and a higher heat request in the system leads to a higher components size. The impact on the system with a continuous thermal load to supply is higher because it extracts higher heat from the system.

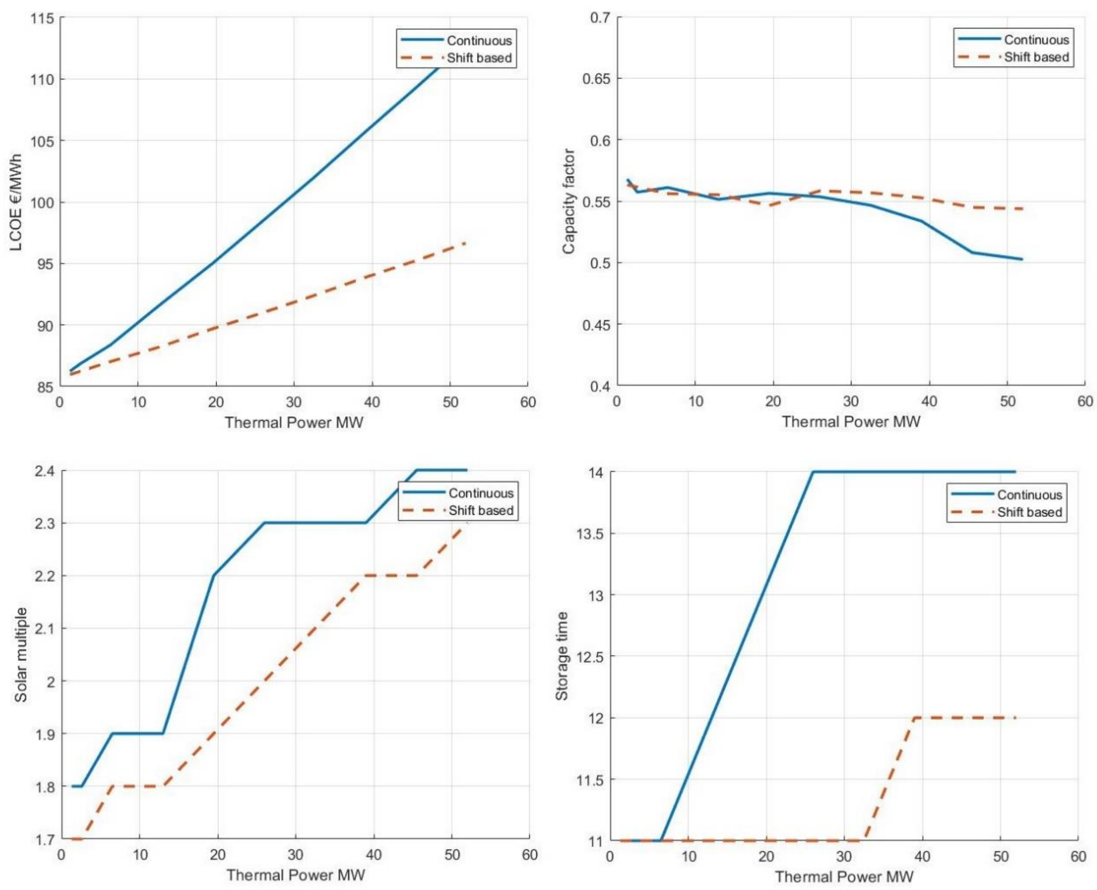


Fig. 32 shows the changes in the LCOE and in the components of the plant for different thermal load

Fig. 33 shows the behavior of the system considering cogeneration, the thermal load affects the electricity production of the CSP most in the winter due to the lower solar resource, while in the summer period the electricity production is nearly not affected by the thermal load.

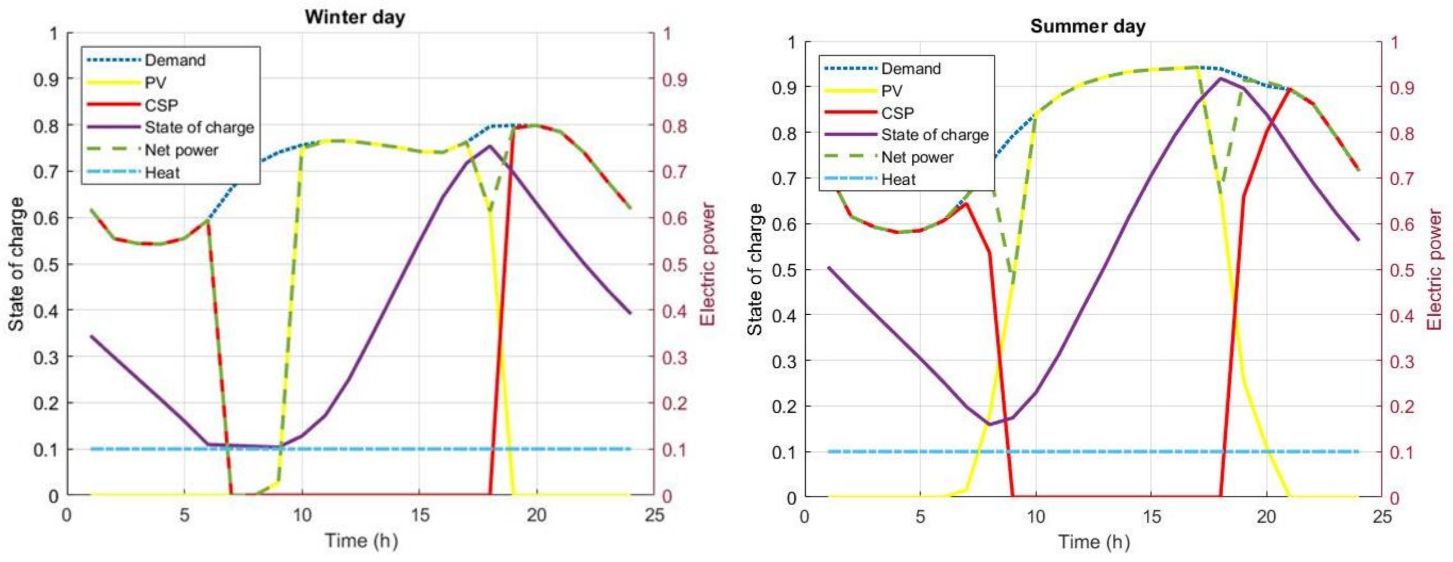


Fig. 33 The behavior of the system in case of the cogeneration in a representative winter and summer day

5.2.1 Performance in the scenario

The system without cogeneration and the one with is compared in the scenario where the price of electricity and natural gas is the one in 2019 because the comparison has been done based on the relative difference in the revenues (from the system without cogeneration to the one with cogeneration). Considering the year 2022 would lead to higher revenues in the system but this study has analyzed the relative difference in the revenues between the two plants.

To compare properly the two systems, it has been calculated the minimum price for selling the heat to have the same revenues as the system without cogeneration for the different thermal loads. For lower thermal load the price of the heat needed is extremely high because the system produces less amount of heat, but it is modified from the optimal case to be able of producing heat reliably. **Fig. 34** it shows the minimum price at which the system has to sell the heat to have the same revenues as the plant without cogeneration, the plant with cogeneration has lower revenues from the electricity production because it produces less electricity for the reason above mentioned. **Fig. 34** has two plots, the one on the left is considering all the domains studied while the right one is a zoom on the most economically interesting part. The price of the natural gas has been used to determine the minimum thermal load which has the same revenues as the system without cogeneration. Formula 22 has been used for calculating the cost of the heat.

$$Cost\ of\ heat = \frac{Revenues\ eletricity_{without\ cogeneration} - Revenues\ eletricity_{with\ cogeneration}}{Thermal\ energy\ produced} \tag{22}$$

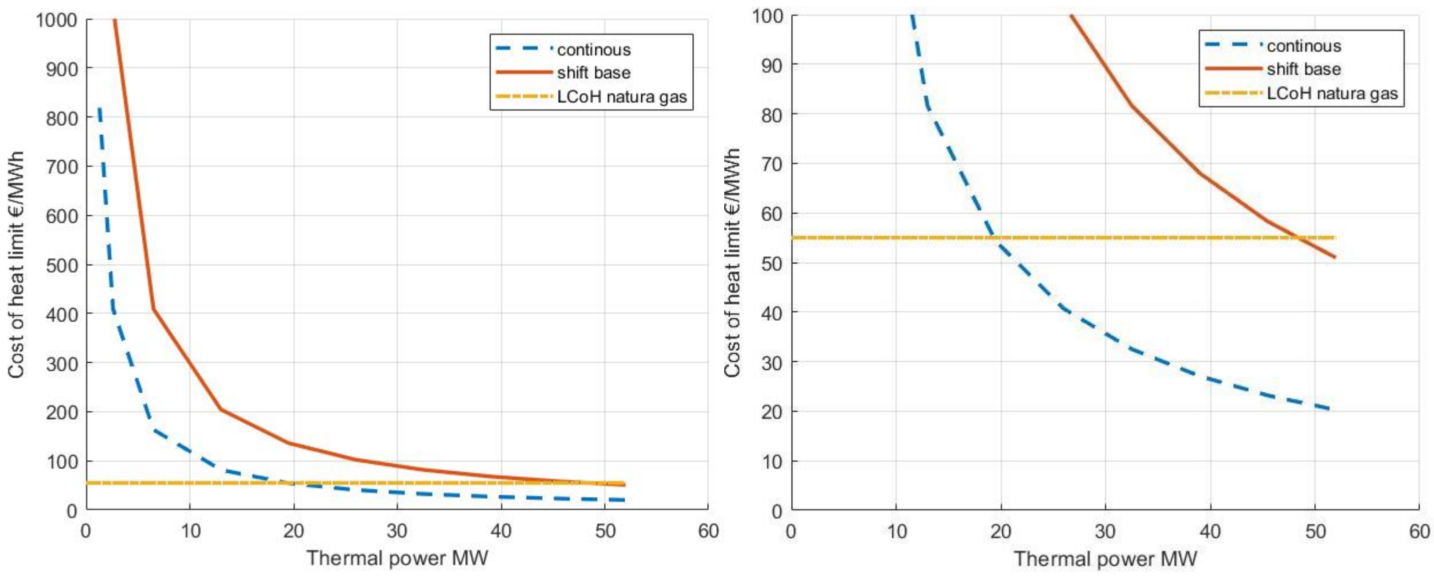


Fig. 34 The minimum price of the heat to have the same revenues for different thermal load

The thermal loads that have been selected for the calculation of the payback time are those that have a cost of heat limit equal to the LCOH of the natural gas, table 12 summarized the two systems with the minimum thermal load.

Table 12. The minimum thermal load to get the same cost of the heat of the natural gas

Type of thermal load	Minimum load
Continuous	19.5 MW
Shift base	45.5 MW

Fig. 35 shows the ratio of energy production from the three sources, the PV and CSP produce electricity and the cogeneration produces heat, although from an energetic point of view they should not be directly compared it is important to understand the amount of heat produced for the cogeneration with respect to the electricity produced. In the case of cogeneration, the electricity production from the CSP decreases because of the less amount of heat in the system a way of mitigating the reduction of the revenues from the CSP could be the changing the dispatch strategy thus, producing the electricity with the CSP when the price of it is highest.

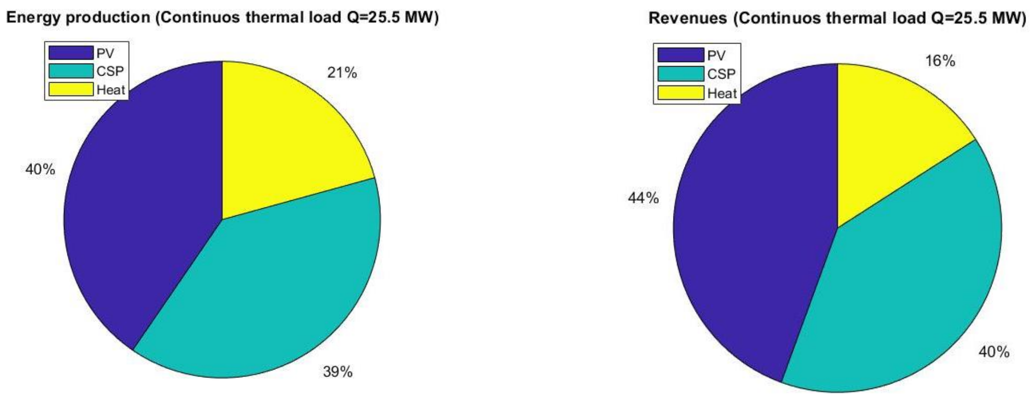


Fig. 35 Pie chart of the energy production considering a thermal load that has the same revenues as the system without cogeneration and sells heat at the LCOH of natural gas

The evaluation of system with cogeneration has been evaluated using formula 15. Fig. 36 shows the payback time for two different thermal loads, considering the price of natural gas in 2019. The type of thermal load affects in a negligible way the payback time because the revenues from the heat generation are less than 20 % of the total revenues.

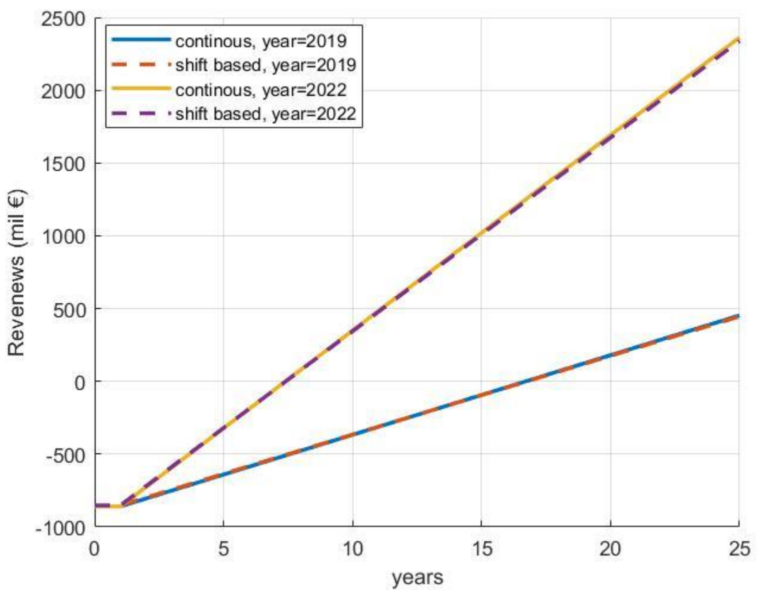


Fig. 36 NPV considering two prices scenario

5.3 Comparison the LCOE among different technologies

Fig. 37 ((29), (30), (31), (32)) shows the comparison among some CSP technologies, according to the data found in the literature the system that shows the lower LCOE is the one using SPT with molten salt. The system studied in this analysis is in the lower range of other CSP technologies. However, the systems studied have a wide range of improvements because they have a higher thermodynamic potential due to their higher working temperature and they have not reached their maturity.

Fig. 37 shows the LCOE from a combined cycle using natural gas as a heat source for different years, normally the price of natural gas is that the CSP are less competitive in the energy market. However, in the case when the price of natural gas significantly increases the CSP are more cost-competitive, in the past something similar happen around the 70' during first the oil crisis and these technologies got some interest in the market but, after the decreasing prices of fossil fuels they almost all the market attraction. Nowadays, in the context of urging or reducing the greenhouse emission and energy independence is less likely that it will happen something like in the 70'.

The systems studied could have a similar relationship with the grid to one of the combined cycle plants because the systems have in common the thermodynamic cycle and due to the TES the CSP can produce electricity to meet the demand.

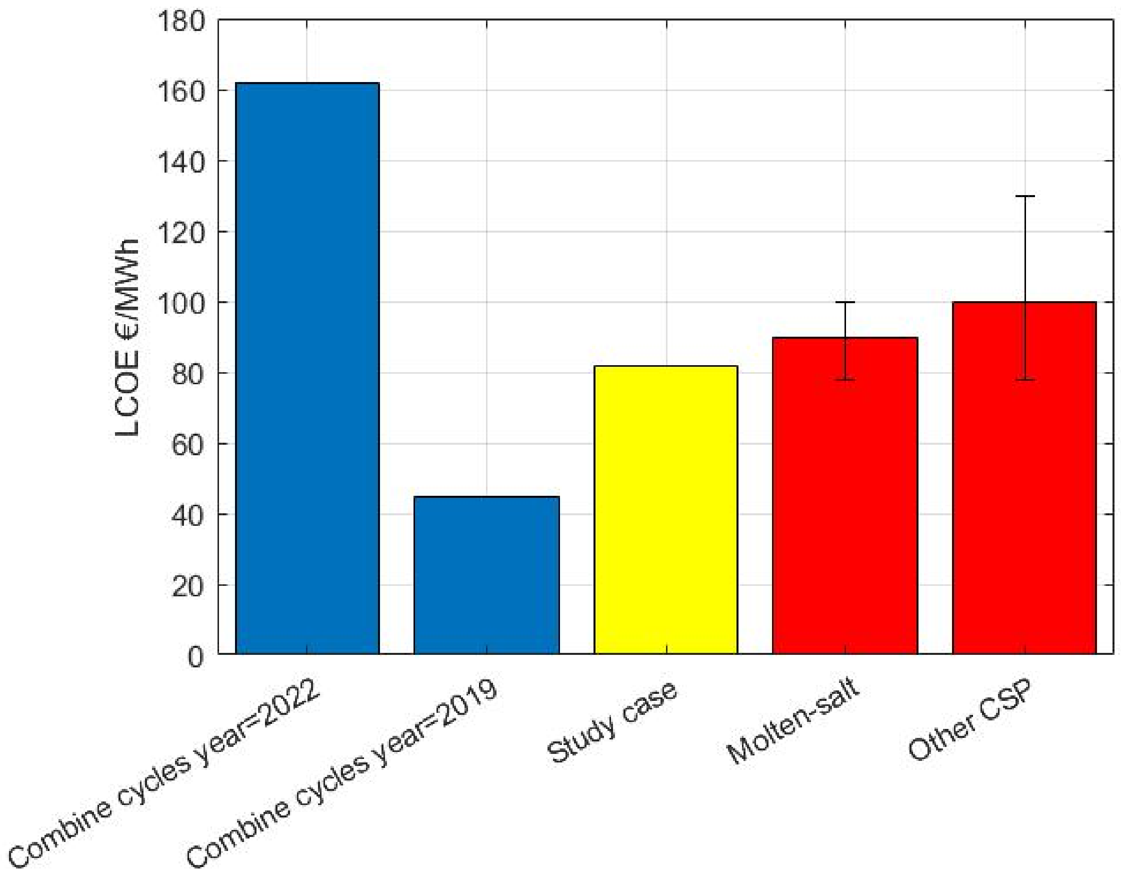


Fig. 37 Comparison of LCOE among different CSP and combined cycle in different years

5.4 Comparison the LCOH among different technologies

Fig. 38 ((20), (21)) shows the comparison between the cost of the heat from the study case compared to the LCOH from natural gas considering two different scenarios for the gas price.

The cost of the heat depends on the thermal load, for a continuous thermal load higher than 20 MW the system can produce heat at the same LCOH of natural gas in 2019. The minimum cost of heat found is 20 €/MWh which is the asymptotic value of the LCOH produced using the system.

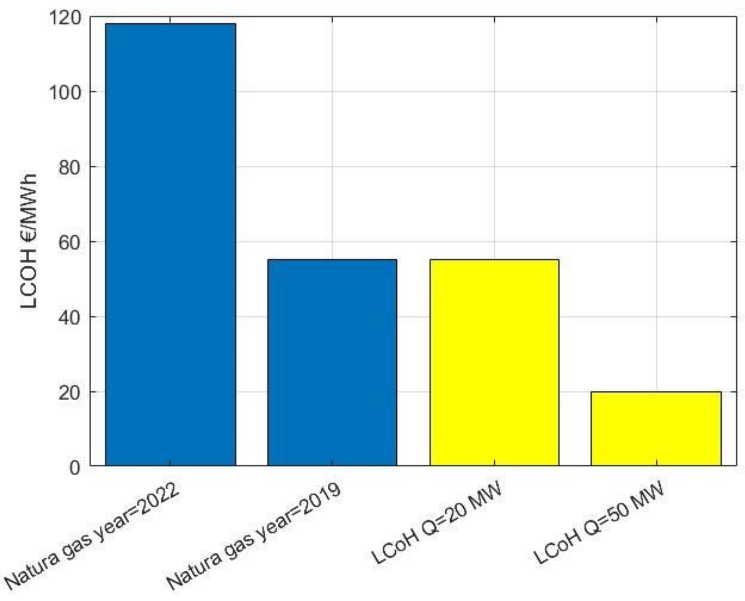


Fig. 38 Comparison of LCOH from natural gas in two different years to the LCOH of the study case

5.5 The footprint of the system

Table 13 summarizes the footprint of different configurations of the system. It is possible to appreciate the effect the downsizing of the power block has some advantages on land use. The main reduction is coming from the lower heliostats field. In case of the cogeneration, the bigger contribution to the increase of the footprint of the system is coming from the higher heliostats field needed to introduce more heat into the system. The comparison among the footprint of different configurations should be done between the base and the system optimized with the downsizing without the cogeneration. The reason is that the base case system considered (Cerro Dominador Solar Thermal Plant) does not have a cogeneration feature, the main contribution of the difference in the land usage is because of the different locations the base case location has a DNI significantly higher than the one of Sevilla.

Table 13. Land used of different configurations analyses and the base case used for the estimation of the footprint

Configuration	Footprint
Best Optimal solution without downsized of the PB	14 Km^2
Best Optimal solution with downsized of the PB	12.8 Km^2
Base case (see 4.8)	10 Km^2
Best Optimal solution having a continuous thermal load of 20 MW	15 Km^2

6 Conclusion

These systems studied show a higher capacity factor between 0.5-0.6 which compared to other renewable technology is a competitive value, they also show an LCOE between 82-120 €/MWh and considering that the system has some degree of flexibility for the electricity production this LCOE is competitive in the energy market. The systems studied could have a similar relationship with the grid to one of the combined cycle plants. The payback time of the system considering the prices of 2019 are around 16 years for both systems, which is a positive indication since the systems can make a profit within their lifetime, on the other hand considering that these technologies are relatively new building a system of this size (130 MWe) with the risk of being obsolete in some years it is not a negligible aspect.

- Without cogeneration: The base case system shows the minimum LCOE of 82 €/MWh which is a competitive value considering the possibility of electricity when is needed, alpha coefficient shows that is possible changing the size of the power block according to the relationship with the electric grid that is a variable feature
- With cogeneration: The modified system shows the possibility of selling the heat at the same cost as the LCOH of natural gas in a reliable way, which gives market attraction to this system. The system with some changes would be able of producing heat at temperatures up to 760 °C, that is the main difference between this technology and the parabolic trough for producing high-temperature heat

The possibility of producing high-temperature heat for industrial purposes could boost the performance of these technologies significantly in the case of some government action in the direction of reducing greenhouse emissions. The main limitation of these technologies is their location, which must have specific characteristics, however, an implementation would surely lead to a decrease of emissions from the energy sector and the industry. The present work indicates the possibility of producing renewable heat at a cost like the one of natural gas hence, the overall trend of the energy market seems to go in the favor of these technologies.

6.1 Limitation and future work

The limitation of this analysis can be summarized in the following lists:

- The model is a quasi-static model which does not consider the transient phenomena, it has been used for evaluating the performance of a load which varies every 1 hour hence, the quasi-static assumption has been used at its limit. The model used gives a reasonably precise indication of the overall energy production of the systems, but it cannot evaluate the losses of the changing working condition of the CSP.
    - A possible future work could be the introduction of electric batteries and analysis of their impact on the LCOE, the electric battery could overcome the problem of the transient behavior of the thermodynamic cycle hence, the system will be able to cover the electric demand when there is not solar resource availability. The electric battery could impact a positive way on the LCOE and on the exegetic performance because the electric heater will be used less.
- The assumption on the thermal used is optimistic thus, considering the heat user in a long-distance so introducing the heat losses the system could not be able to produce heat at the same LCOH of natural gas.
    - A possible future work could be analyzing a specific thermal user and considering the distance between the heat generation and the heat user and evaluating if it is still possible to produce heat in a cost-competitive way.
- The economic advantage of using this type of technology in a context where the population is more sensitive to the greenhouse reduction could have a positive impact perceived image of the firm that implements it, therefore the costumers are willing to a higher price this would increase the economical values
- The system footprint and the constraints on the location are strong limits on the application of these technologies thus, the CSP for obvious reasons needs an area where the DNI has high enough value to make the plant cost-competitive in the energy market. In the case of cogeneration, the system must have a near industry which needs high-temperature heat to operate, with the cogeneration the conditions in the location increase thus, it could applicable only in some areas. The effect of the high footprint could mitigate acting the downsize of the power thus, reducing the alpha parameter.
    - A possible future work could be of integrating the cogeneration with the Carnot battery to avoid the limitation of having an area with high DNI, however, the system will be drastically different as it will have only the TES and the power block.

7 Reference

1. **Alberto Giaconia, Roberto Grena.** A model of integration between PV and thermal CSP technologies. *ScienceDirect.* [Online] 2021. https://www.sciencedirect.com/science/article/pii/S0038092X21004138.

2. **Jingze Yang, Zhen Yang,Yuanyuan Duan.** Load matching and techno-economic analysis of CSP plant with S–CO2 Brayton cycle in CSP-PV-wind hybrid system. *ScienceDirect.* [Online] 2021. https://www.sciencedirect.com/science/article/pii/S0360544221002656.

3. **Trevisan, Silvia.** *Renewable Heat on Demand: High-temperature thermal energy storage: a comprehensive study from material investigation to system analysis via innovative component design.* 2022.

4. **Naman Goyal, Akshansh Aggarwal, Anil Kumar.** Concentrated solar power plants: A critical review of regional dynamics and operational parameters. *ScienceDirect.* [Online] 2022. https://www.sciencedirect.com/science/article/abs/pii/S2214629621004230.

5. **Wang, Zhifeng.** Design of Solar Thermal Power Plants. *ScienceDirect.* [Online] 2019. https://www.sciencedirect.com/science/article/pii/B9780128156131000018.

6. **Francesco Crespi, Giacomo Gavagnin, David Sánchez, Gonzalo S. Martínez.** Supercritical carbon dioxide cycles for power generation: A review. [Online] 2017. https://www.sciencedirect.com/science/article/pii/S0306261917301915.

7. **Tobias Naegler, Sonja Simon, Martin Klein, Hans Christian Gils.** *Quantification of the European industrial heat demand.* 2015.

8. **Solargis.** Spain. *Solargis.* [Online] https://solargis.com/maps-and-gis-data/download/spain.

9. **ABSOLICON.** [Online] https://www.absolicon.com/solar-applications-markets/.

10. **IRENA.** International Renewable Energy Agency. [Online] 2019. https://www.irena.org/publications/2020/Jun/Renewable-Power-Costs-in-2019.

11. **Harmeet Singh, R.P. Saini, J.S. Saini.** A review on packed bed solar energy storage systems. *Science Direct.* [Online] 2009. https://www.sciencedirect.com/science/article/pii/S1364032109002536.

12. **Markus Hänchen, Sarah Brückner, Aldo Steinfeld.** High-temperature thermal storage using a packed bed of rocks e Heat transfer analysis and experimental validation. *Science Direct.* [Online] 2010. https://reader.elsevier.com/reader/sd/pii/S1359431111001062?token=76B2E09A219E77E93746211405F0924D319204A90290F1EAD5BD9E96E3EEABACB4493EDA9C1F9344FFDC945942DC2B1E&originRegion=eu-west-1&originCreation=20220216121247.

13. **Alberto Giaconia, Gaetano Iaquaniello, Amr Amin Metwally, Giampaolo Caputo, Irena Balog.** Experimental demonstration and analysis of a CSP plant with molten salt. *Direct Science.* [Online] 2020. https://www.sciencedirect.com/science/article/pii/S0038092X20310501.

14. **O. Achkari, A. El Fadar.** Latest developments on TES and CSP technologies – Energy and environmental issues, applications and research trends. *ScienceDirect.* [Online] 2020. https://www.sciencedirect.com/science/article/pii/S1359431118363269.

15. *A new methodology of thermal performance improvement and numerical analysis of free-falling particle receiver.* **Rui Jiang, Ming-Jia Li, Wen-Qi Wang.** 2021, DirectScience.

16. *High-Temperature Particle Heat* . **Matthew D.Carlson, Kevin J. Albrecht, Clifford K. Ho, Hendrick F. Laubscher, Francisco Alvarez.** 2020, Sandia National Laboratories.

17. *Design analysis of particle-base thermal energy storage system for concentrating solar power or grid energy storage.* **Zhiwen Ma, Patrick Davenport, Ruichong Zhang.** 2020.

18. ***ATO.* [Online] 2022. https://www.ato.com/plate-heat-exchanger-50-60-plate.**

19. Omie. [Online] 2019. https://www.omie.es/en/market-results/monthly/daily-market/daily-market-price?scope=monthly&year=2021&month=5.

20. *Standardized benchmarking establishes degree of competitiveness for thermal storage solutions.* McK/LDES.

21. Trading economic. [Online] https://tradingeconomics.com/commodity/eu-natural-gas.

22. Wikipedia. Solar power tower. [Online] https://en.wikipedia.org/wiki/Solar_power_tower#:~:text=Generally%2C%20installations%20use%20from%20150,hectares%20(3%2C200%2C000%20m2).

23. —. Cerro Dominador Solar Thermal Plant. [Online] https://en.wikipedia.org/wiki/Cerro_Dominador_Solar_Thermal_Plant.

24. —. Khi Solar One. [Online] https://en.wikipedia.org/wiki/Khi_Solar_One.

25. Photovoltaic power station. *Wikipedia.* [Online] https://en.wikipedia.org/wiki/Photovoltaic_power_station.

26. Golmud Solar Park. [Online] https://en.wikipedia.org/wiki/Huanghe_Hydropower_Golmud_Solar_Park.

27. Solargis. Chile. [Online] https://solargis.com/maps-and-gis-data/download/chile.

28. Cerro Dominador Solar Thermal Plant. *Wikipedia.* [Online] https://en.wikipedia.org/wiki/Cerro_Dominador_Solar_Thermal_Plant.

29. Energy Intelligence. [Online] https://www.energyintel.com/0000017e-fc9d-d1a7-affe-fcff7cfe0000.

30. The Cost of Electricity. *Direct Science.* [Online] https://www.sciencedirect.com/science/article/pii/B9780128238554000097.

31. Qiliang Wanga, Gang Peib, Hongxing Yanga. *Direct Science.* [Online] 2020. https://www.sciencedirect.com/science/article/pii/S096014812031884X.

32. Guccione, Salvatore. Design and Optimization of a Sodium-Molten Salt Heat Exchanger for Concentrating Solar Power applications. *Diva.* [Online] 2020. http://kth.diva-portal.org/smash/record.jsf?pid=diva2%3A1461996&dswid=7396.

33. *A Methodology to Identify the Most Promising Concentrating Solar .* Guccione, Salvatore. 2022.

34. Wikipedia. Ouarzazate Solar Power Station. [Online] https://en.wikipedia.org/wiki/Ouarzazate_Solar_Power_Station.

8 Appendix

In this paragraph, there is the code used for the analysis. The code was written by Salvatore Guccione for simulating the techno-economical performance of a system having molten salt, I adapted to be able to simulate the behavior of a system working with particles.

8.1 System model

```
import Models.HybridPlant.PV as PV
import Models.ControlSystems.CS_ElectricHeater as EHCS
import Models.HybridPlant.ElectricHeater as EH
import Models.Media.Particle as Medium1
import Models.Media.sCO2 as Medium3
import Models.SolarField.Sun as SUN
import Models.SolarField.HeliostatField as HF
import Models.Utilities.U_SolarField.SolarFunctions as SFun
import Models.ControlSystems.CS_DirectReceiverS as RECS
import Models.Receivers.ParticleReceiverS as REFREC
import Models.Utilities.U_Receiver.ThermalLosses_MoltenSaltReceiver as TLMSREC
import Models.Storages.TankStorage as TES
import Models.ControlSystems.CS_PowerBlock as PBCS
import Models.HeatExchangers.HX_sCO2PowerBlockS as PBHX
import Models.Utilities.U_PowerBlock.Mixer as MIX
import Models.Utilities.U_ControlSystem.Tank2Logic as T2LU
import Models.Utilities.U_CostModels.CM_MoltenSaltsCO2System as CM
import Models.Utilities.KPI as KPI
import Models.Utilities.GeneralUtilities as GU
import Models.Utilities.ExportExcel as EE
from pathlib import Path
import Data.ElectricityDemand.Demand as DE
import Models.HeatExchangers.HX_cogeneration as HX_coge
import Models.ControlSystems.CS_Cogeneration as CS_coge
import math as MA
import numpy as np
import pandas as pd
import time
import openpyxl


def MoltenSaltsCO2(

    Model_Name = 'MoltenSaltsCO2',                               # [-]       -
Name of the model

    # ------------------------------      Location Inputs
    state_name = 'Spain',                                        # [-]       -
Name of the state where the plant is located
    w_file_path = r'./Data/Weather/Spain/',                      # [-]       -
Path to the weather file
    w_file_name = 'ESP_AN_Sevilla.AP.083910_TMYx.epw',           # [-]       -
Weather file name
    day_des = 172,                                               # [-]       -
Design day [1-365]
    hour_des = 12,                                               # [-]       -
Design hour (in solar time) [1-24]
    lat = 37.367,                                                # [deg]     -
Latitude of the location
```

```
    lon = -6.000,                                                    # [deg]     -
Longitude of the location
    time_zone = 1,                                                   # [h]       -
Time zone of the location
    Wspd_des = 5,                                                    # [m/s]     -
Wind speed at the design point
    elev_location = 2,                                               # [m]       -
Elevation of the methereological tower
    alpha_wind = 0.16,                                               # [-]       -
Scaling coefficient for wind speed - assumed based on:
https://doi.org/10.1016/j.rser.2021.111411

    # ------------------------------       PV Inputs
    P_max = 25e6,                                                    # [W]       -
Maximum Electric Power that can be injected to the grid
    P_AC=50e6,                                                       # [W]       -
AC nameplate system capacity
    r_DCAC=1.2,                                                      # [-]       -
DC-to-AC ratio
    module_type = 0,                                                 # [-]       -
Module type [0, 1, 2] - [Standard,Premium,Thin film]
    array_type = 0,                                                  # [-]       -
Array type [0, 1, 2, 3, 4] - [Fixed Rack, Fixed Roof, 1Axis, Backtracked, 2Axis]
    tilt=35,                                                         # [deg]     -
Array tilt angle (if no tracking)
    azimuth = 180,                                                   # [deg]     -
Array azimuth angle [deg] - Options: N=0, E=90,S=180,W=270
    enable_battery = 0,                                              # [-]       -
Boolean to enable battery
    GCR = 0.4,                                                       # [-]       -
Ground coverage ratio - MIN=0.01,MAX=0.99
    eta_inv_input = 98,                                              # [%]       -
Inverter efficiency at rated power - MIN=90,MAX=99.5
https://www.nrel.gov/docs/fy19osti/72399.pdf
    P_single_PV = 325,                                               # [W]       -
Single PV module DC power output - Based on the YGE 72 CELL SERIES 2 - P = 325 W
    A_single_PV = 1.96*0.99,                                         # [m2]      -
Single PV module area - Based on the YGE 72 CELL SERIES 2 - P = 325 W

    # ------------------------------       Electric Heater Inputs
    P_name_EH = 25e6,                                                # [W]       -
Nominal electric heater capacity (P_AC - P_max)
    eta_heater_design = 0.95,                                        # [-]       -
Electric heater efficiecny

    # ------------------------------       Heliostat Field Inputs
    SM = 2.4,                                                        # [-]       -
Solar Multiple
    use_SolarPilot=True,                                             # [-]       -
Boolean to decide to use SolarPILOT
    helio_width=12.2,                                                # [m]       -
Width of the heliostat
    helio_height=12.2,                                               # [m]       -
Height of the heliostat
    DNI_des=850,                                                     # [W/m2]    -
Direct Normal Irradiance at design point
    excl_fac=0.97,                                                   # [-]       -
Exclusion factor (mirror density)
    he_av_design = 0.99,                                             # [-]       -
Helisotats availability
```

```
    helio_reflectance = 0.9,                                    # [-]        -
Heliostats reflectance
    Optimize_SF = True,                                         # [-]        -
Boolean to run optimization of the solar field
    input_helio_map = False,                                    # [-]        -
Boolean to decide if the heliostat map is given as an input
    optical_path = r'./Data/Optical/',                          # [-]        -
Path to the heliostat map
    helio_map_name = 'Heliostats_Position_Map.xlsx',            # [-]        -
Heliostat map file name
    land_max = 9.5,                                             # [-]        -
Maximum land multiplier
    land_min = 0.75,                                            # [-]        -
Minimum land multiplier
    W_track=0.055e3,                                            # [W]        -
Tracking power for a single heliostat
    optical_file_name = 'OpticalEfficiency.txt',                # [-]        -
Name of the optical file

    # ------------------------------       Tower and Receiver Inputs
    FlatPlate = False,                                          # [-]        -
Boolean to decide to use a flat plate receiver or not
    input_eff = False,                                          # [-]        -
Boolean to decide to use a fixed receiver efficiency
    eta_rec_input = 0.5,                                       # [-]        -
Input of a constant receiver efficiency
    D_rec_input = 17.65,                                        # [m]        -
Input of the receiver diameter
    ar_rec_input = 51.6/17.65,                                  # [-]        -
Receiver aspect ratio (height over width)
    N_pa_rec = 20,                                              # [-]        -
Number of panels in receiver
    N_fl = 1,                                                   # [-]        -
Number of parallel flow-paths
    t_tb_rec = 1.25e-3,                                         # [m]        -
Thickness of the receiver tube wall
    D_tb_rec = 40e-3,                                           # [m]        -
Receiver tube outer diameter
    H_tower_input = 190,                                        # [m]        -
Input of the tower height
    rec_absorptance = 0.95,                                     # [-]        -
Receiver coating absorptance
    rec_emissivity = 0.84,                                      # [-]        -
Receiver coating emissivity
    k_material = 16,                                            # [W/(mK)]   -
Conductivity of the receiver material (SS)
https://doi.org/10.1680/ensu.2007.160.4.167
    rec_hl_perm_guess = 30,                                     # [kW/m2]    -
Receiver design heat loss
    T_cold_set_REC_input = GU.from_degC(290),                   # [K]        -
Cold Receiver design temperature - Ref. https://elib.dlr.de/141315/
    T_hot_set_REC = GU.from_degC(800),                          # [K]        -
Hot Receiver design temperature - Ref. https://elib.dlr.de/141315/

    # ------------------------------       Storage Inputs
    t_storage = 10,                                             # [h]        -
Hours of storage
    H_storage = 12,                                             # [m]        -
Height of the storage tank
```

```
    T_hot_start_TES = GU.from_degC(790),                            # [K]        -
Hot tank starting temperature (T_hot_start_TES = T_hot_set_REC)
    T_cold_aux_set = GU.from_degC(290),                             # [K]        -
Cold tank auxiliary heater set-point temperature
    T_hot_aux_set = GU.from_degC(785),                              # [K]        -
Hot tank auxiliary heater set-point temperature
    alpha = 0.4,                                                    # [W/(m2K)] -
Tank constant heat transfer coefficient with ambient
    k_loss_cold = 0.15e3,                                           # [J/kg]     -
Cold tank parasitic power coefficient
    k_loss_hot = 0.55e3,                                            # [J/kg]     -
Hot tank parasitic power coefficient
    k_loss_rec = 0.21e3,                                            # [J/kg]     -
Cold tank parasitic power coefficient
    W_heater_hot = 30e6,                                            # [W]        -
Hot tank heater capacity
    W_heater_cold = 15e6,                                           # [W]        -
Cold tank heater capacity
    tank_ar = 38.8/12,                                              # [-]        -
Storage tank aspect ratio
    e_ht= 0.99,                                                     # [-]        -
Tank Heater Efficiency
    dt = 3600,                                                      # [s]        -
Number of samples to generate in one hour in the ODE to simulate the storage
behaviour
    # ------------------------------      Heat Exchanger

    U_HX_design=100,                                                 #[W/K*m^2]
    T_source_cold_design=595+273.15,
    eff_HX_coge=1,
    L_min_HX=0.3,
    Q_de=10*1e3,

    # ------------------------------       Control Inputs
    t_ramping = 0.5*3600,                                           # [s]        -
Power block startup delay: 0.5 hour
    t_standby = 2*3600,                                             # [s]        -
Power block standby delay: 2 hour
    ele_min = 0.13962634015955,                                     # [rad]      -
Heliostat stow deploy angle
    Wspd_max = 15,                                                  # [m/s]      -
Wind stow speed
    nu_start = 0.2,                                                 # [-]        -
Minimum energy start-up fraction to start the receiver
    nu_min_sf = 0.2,                                                # [-]        -
Minimum turn-down energy fraction to stop the receiver
    nu_min_EH = 0.2,                                                # [-]        -
Minimum energy fraction to start/stop the electric heater
    L_start = 50,                                                   # [%]        -
Hot Tank initial level
    hot_tnk_empty_lb = 6,                                           # [%]        -
Hot tank empty trigger lower bound - Level (below which) to stop disptach
    hot_tnk_empty_ub = 11,                                          # [%]        -
Hot tank empty trigger upper bound - Level (above which) to start disptach
    hot_tnk_full_lb = 93,                                           # [%]        -
Hot tank full trigger lower bound
    hot_tnk_full_ub = 98,                                           # [%]        -
Hot tank full trigger upper bound
    cold_tnk_defocus_lb = 6,                                        # [%]        -
Cold tank empty trigger lower bound# - Level (below which) to stop disptach
```

```
    cold_tnk_defocus_ub = 11,                                       # [%]       -
Cold tank empty trigger upper bound# - Level (above which) to start disptach
    cold_tnk_crit_lb = 5,                                           # [%]       -
Cold tank critically empty trigger lower bound# - Level (below which) to stop
disptach
    cold_tnk_crit_ub = 10,                                          # [%]       -
Cold tank critically empty trigger upper bound# - Level (above which) to start
disptach



    # ------------------------------       sCO2 Power Block Inputs
    P_gross=115e6,                                                  # [W]       -
Power Cycle Gross Output
    use_eta_net_blk=True,                                           # [-]       -
Boolean to decide to use a fixed net-to-gross efficiecny or calculate parasitic
losses
    eta_net_blk = 0.95,                                             # [-]       -
Gross-to-net power conversion factor at the power block - Ref.
https://doi.org/10.17185/duepublico/73961
    TIT = GU.from_degC(775),                                        # [K]       -
Turbine Inlet Temperature at design
    T_in_Compr = GU.from_degC(32),                                  # [K]       -
Compressor inlet temperature at design
    eta_gen = 0.98,                                                 # [-]       -
Mechanical-to-Electrical Efficiency - Ref.
https://doi.org/10.17185/duepublico/73961
    Reheat = False,                                                 # [-]       -
Boolean to decide to include Reheating in the sCO2 power block
    Recompression = True,                                           # [-]       -
Boolean to decide to include Recompression in the sCO2 power block
    Intercooling = True,                                            # [-]       -
Boolean to decide to include Intercooling in the sCO2 power block
    p_high_blk = 25e6,                                              # [Pa]      -
Power block operating high pressure - 250 bar
    p_int_blk= 16.19e6,                                             # [Pa]      -
Power block operating intermediate pressure - 161.9 bar (ONLY if Reheat is True)
    p_low_blk = 7.38e6,                                             # [Pa]      -
Power block operating low pressure - 73.8 bar
    p_incooler_blk = 10e6,                                          # [Pa]      -
Power block operating intercooler pressure - 100 bar (ONLY if Intercooling is True)
    T_in_inter_Compr = GU.from_degC(32),                            # [K]       -
Compressor inlet temperature at design
    SR = 0.7,                                                       # [-]       -
sCO2 mass flow split ratio (ONLY if Recompression is True)
    eta_HTR = 0.95,                                                 # [-]       -
Effectiveness High-Temperature Recuperator
    eta_LTR = 0.95,                                                 # [-]       -
Effectiveness Low-Temperature Recuperator (ONLY if Recompression is True)
    DT_recuperator = 10,                                            # [K]       -
Min Pich-point temperature recuperators
    PB_load_min = 0.4,                                              # [-]       -
Min load at which the PB can operate
    eta_PBHX=1,                                                     # [-]       -
Salt-to-sCO2 HX efficiency
    DT_pinch_TESsCO2=15,                                            # [K]       -
Initial Temperature difference for the primary HX(s)
    T_in_air_cooler_des = GU.from_degC(28),                         # [K]       -
Ambient temperature at design point for power block
```

```
    par_fix_fr = 0.0055,                                          # [-]        -
Fixed parasitics as fraction of gross rating

    # ------------------------------       Cost Data Inputs
    currency_name = "EUR",                                        # [-]        -
Name of the currency adopted
    M_conv_currency_to_USD = 0.84,                                # [EUR/USD] -
The currency rate from USD to currecny adopted
    r_disc = 0.07,                                                # [-]        -
Discount rate
    r_i = 0.025,                                                  # [-]        -
Inflation rate
    t_life = 30,                                                  # [years]    -
Lifetime of plant - Based on Downselect Criteria, Table 2
    f_contingency_CSP = 0.07,                                     # [-]        -
Contingency costs share - CSP plant - Ref. SAM Default Value
    f_contingency_PV = 0.03,                                      # [-]        -
Contingency costs share - PV plant - Ref. SAM Default Value
https://www.nrel.gov/docs/fy19osti/72399.pdf
    f_decommissioning = 0,                                        # [-]        -
Decommissioning costs share - It shoudl be 0.05 #Decommissioning costs share
    f_EPC_CSP = 0.13,                                             # [-]        -
Engineering, procurement and construction(EPC) and owner costs costs share - CSP
plant - Ref. SAM Default Value
    f_EPC_PV = 0.1,                                               # [-]        -
Engineering, procurement and construction(EPC) and owner costs costs share - PV
plant - Ref. SAM Default Value https://www.nrel.gov/docs/fy19osti/72399.pdf
    f_Subs = 0,                                                   # [-]        -
Subsidies on initial investment costs

    # ------------------------------       Characterization of the System
Optimization
    LCOE_OF = True,                                               # [-]        -
Boolean to select LCOE as an objective function for the system optimization
    CAPEX_OF = False,                                             # [-]        -
Boolean to select CAPEX as an objective function for the system optimization
    CF_OF = False,                                                # [-]        -
Boolean to select CF as an objective function for the system optimization
    AEY_OF = False,                                               # [-]        -
Boolean to select AEY as an objective function for the system optimization
    AF_OF = False,                                                # [-]        -
Boolean to select AF as an objective function for the system optimization
    c_block_OF = False,                                           # [-]        -
Boolean to select c_block as an objective function for the system optimization
    ASCO2eq_OF = False,                                           # [-]        -
Boolean to select ASCO2eq as an objective function for the system optimization
    TEW_share_OF = False,                                         # [-]        -
Boolean to select TEW_share as an objective function for the system optimization
    f_AEY_CSP_OF = False,                                         # [-]        -
Boolean to select f_AEY_CSP as an objective function for the system optimization
    CF_CSP_OF = False,                                            # [-]        -
Boolean to select CF_CSP as an objective function for the system optimization
    CF_PV_OF = False,                                             # [-]        -
Boolean to select CF_PV as an objective function for the system optimization
    CAPEX_CSP_OF = False,                                         # [-]        -
Boolean to select CAPEX_CSP as an objective function for the system optimization
    CAPEX_PV_OF = False,                                          # [-]        -
Boolean to select CAPEX_PV as an objective function for the system optimization
    EH_UF_OF = False,                                             # [-]        -
Boolean to select EH_UF as an objective function for the system optimization
```

```
    TES_PV_fraction_OF = False,                                         # [-]        -
Boolean to select TES_PV_fraction as an objective function for the system
optimization

    # ------------------------------       Handling of the Outputs
    identification_folder_summary = '',                                 # [-]        -
Identification of the folder for the summary of the results
    identification_summary = '',                                        # [-]        -
Identification of the summary of results file
    identification_folder_results = '',                                 # [-]        -
Identification of the folder for the results
    identification_results = '',                                        # [-]        -
Identification of the results files
    save_all_files = False,                                             # [-]        -
Boolean to decide if to save all the results files or only the summary
    erase_old_summary = False,                                          # [-]        -
Boolean to decide if to erase the old results in the summary file

    # ------------------------------       Other Inputs
    Short_Simulation = False,                                           # [-]        -
Boolean to decide to limit the simulation at a short time interval
    shortime_start = 0,                                                 # [-]        -
Index of the start of the time interval
    shortime_end = 3,                                                    # [-]        -
Index of the end of the time interval
    #---------------------------------- Simulation with multiple parameters
    Flag=0,
    Co_i=1,
    Co_j=1,
    Co_k=1,
    Co_l=1,
    Co_m=1,
    number_sim=1
):

    start_time = time.time()
    print(u"\u2192 Design of the plant started")

# ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Design  ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
    PD=DE.load(P_max)
    #print(max(PD))
#
__________________________________________________________________________________
__________________________________________________________________________________
_____
#                                                                     PV Field
Calculated Parameters
#
__________________________________________________________________________________
__________________________________________________________________________________
_____
    part1 = time.time()
    P_DC=r_DCAC*P_AC                                                    # [W]        -
System size - DC nameplate
    W_heater_min = P_name_EH * nu_min_EH                                # [W]        -
Minimum Electric Heater Power
    w_file = '%s%s' %(w_file_path, w_file_name)                         # [-]        -
Weather file
```

```
                                                                          # Calculation
of the PV Field
    [PV_rows, A_land_PV_tot, A_PV_field, PV_EnergyYield, PV_CF, PV_P_AC_inv_annual,
PV_Annual_Energy, PV_P_DC_inv, PV_P_AC_inv, W_net_PV, W_heater_PV_raw, EH_on,
PV_share, W_wasted_PV_plus]=PV.Simple_PVPlant(
        w_file, tilt, azimuth, array_type, module_type, P_single_PV, A_single_PV,
r_DCAC, P_DC, GCR, enable_battery, eta_inv_input, P_max, P_name_EH, W_heater_min,
PD)
    #print(len(PV_share))
    print(u"\u2192 Design of the PV Field completed (Duration: %s s)"
%(np.around(time.time() - part1, decimals=0)))

#
__________________________________________________________________________________
__________________________________________________________________________________
_____
#                                                                     Power Block
Calculated Parameters
#
__________________________________________________________________________________
__________________________________________________________________________________
_____
    part1 = time.time()
                                                                    # Selecting the
PB Type and defining the design
    if Recompression==False and Reheat==False and Intercooling==False:
        import Models.PowerBlocks.sCO2_PB_Simple as PBS
        PB_S=True
        PB_S_Inter=False
        PB_R=False
        PB_R_Inter=False
        PB_Recomp=False
        PB_Recomp_Inter=False
        PB_RR=False
        PB_RR_Inter=False
    if Recompression==False and Reheat==False and Intercooling==True:
        import Models.PowerBlocks.sCO2_PB_Simple_Intercooler as PBSInter
        PB_S=False
        PB_S_Inter=True
        PB_R=False
        PB_R_Inter=False
        PB_Recomp=False
        PB_Recomp_Inter=False
        PB_RR=False
        PB_RR_Inter=False
    if Recompression==False and Reheat==True and Intercooling==False:
        import Models.PowerBlocks.sCO2_PB_Reheat as PBR
        PB_S=False
        PB_S_Inter=False
        PB_R=True
        PB_R_Inter=False
        PB_Recomp=False
        PB_Recomp_Inter=False
        PB_RR=False
        PB_RR_Inter=False
    if Recompression==False and Reheat==True and Intercooling==True:
        import Models.PowerBlocks.sCO2_PB_Reheat_Intercooler as PBRInter
        PB_S=False
        PB_S_Inter=False
        PB_R=False
        PB_R_Inter=True
        PB_Recomp=False
        PB_Recomp_Inter=False
        PB_RR=False
        PB_RR_Inter=False
    if Recompression==True and Reheat==False and Intercooling==False:
        import Models.PowerBlocks.sCO2_PB_Recompr as PBRecomp
        PB_S=False
        PB_S_Inter=False
        PB_R=False
        PB_R_Inter=False
        PB_Recomp=True
        PB_Recomp_Inter=False
        PB_RR=False
        PB_RR_Inter=False
    if Recompression==True and Reheat==False and Intercooling==True:
        import Models.PowerBlocks.sCO2_PB_Recompr_Intercooler as PBRecompInter
        PB_S=False
        PB_S_Inter=False
        PB_R=False
        PB_R_Inter=False
        PB_Recomp=False
        PB_Recomp_Inter=True
        PB_RR=False
        PB_RR_Inter=False
    if Recompression==True and Reheat==True and Intercooling==False:
        import Models.PowerBlocks.sCO2_PB_Recompr_plus_Reheat as PBRR
        PB_S=False
        PB_S_Inter=False
        PB_R=False
        PB_R_Inter=False
        PB_Recomp=False
        PB_Recomp_Inter=False
        PB_RR=True
        PB_RR_Inter=False
    if Recompression==True and Reheat==True and Intercooling==True:
        import Models.PowerBlocks.sCO2_PB_Recompr_plus_Reheat_Intercooler as
PBRRInter
        PB_S=False
        PB_S_Inter=False
        PB_R=False
        PB_R_Inter=False
        PB_Recomp=False
        PB_Recomp_Inter=False
        PB_RR=False
        PB_RR_Inter=True

    if PB_Recomp_Inter:
        identification_PB='_PB_Recomp_Inter_'
        [m_flow_sCO2_des, eta_blk_des, T_MH_sCO2_cold, T_max_HTR_sCO2,
T_max_LTR_sCO2, W_HPT_des, eta_HPT_des, W_MC1_des, eta_MC1_des, W_MC2_des,
eta_MC2_des, W_RC_des, eta_RC_des, UA_MH_des, UA_HTR_des, UA_LTR_des,
UA_cooler_des, UA_intercooler_des, Q_Cooler_des, DT_pinch_cooler,
Q_Intercooler_des, DT_pinch_intercooler, Q_HTR_des, Q_LTR_des,
thermodynamic_cycle_des,
f_prop_des]=PBRecompInter.Design_sCO2_PB_Recompr_Intercooler(
            P_gross, TIT, T_in_Compr, T_in_inter_Compr, p_high_blk, p_low_blk,
p_incooler_blk, SR, eta_gen, eta_HTR, eta_LTR, T_hot_set_REC, T_cold_set_REC_input,
DT_pinch_TESsCO2, T_in_air_cooler_des, DT_recuperator
            )
        [TS_1_des, TS_2_des, TS_3_des, TS_4_des, TS_5_des, TS_6_des, TS_7_des,
TS_8_des, TS_9_des, TS_9_prime_des, TS_9_second_des, TS_10_des] =
thermodynamic_cycle_des
        T_sCO2_1_des=GU.to_degC(Medium3.temperature(TS_1_des))
        T_sCO2_2_des=GU.to_degC(Medium3.temperature(TS_2_des))
        T_sCO2_3_des=GU.to_degC(Medium3.temperature(TS_3_des))
        T_sCO2_4_des=GU.to_degC(Medium3.temperature(TS_4_des))
        T_sCO2_5_des=GU.to_degC(Medium3.temperature(TS_5_des))
        T_sCO2_6_des=GU.to_degC(Medium3.temperature(TS_6_des))
        T_sCO2_7_des=GU.to_degC(Medium3.temperature(TS_7_des))
        T_sCO2_8_des=GU.to_degC(Medium3.temperature(TS_8_des))
        T_sCO2_9_des=GU.to_degC(Medium3.temperature(TS_9_des))
        T_sCO2_9_des_K = Medium3.temperature(TS_9_des)
        T_sCO2_9_prime_des=GU.to_degC(Medium3.temperature(TS_9_prime_des))
        T_sCO2_9_second_des=GU.to_degC(Medium3.temperature(TS_9_second_des))
        T_sCO2_10_des=GU.to_degC(Medium3.temperature(TS_10_des))
        #print(T_sCO2_10_des)
        #print("332")
        T_high_h = f_prop_des[1]
        T_incooler_h = f_prop_des[5]
        T_low_T = f_prop_des[9]
        h_high_T = f_prop_des[0]
        h_incooler_T = f_prop_des[4]
        h_low_T = f_prop_des[8]
        m_flow_sCO2_MC = SR * m_flow_sCO2_des
        m_flow_sCO2_RC = (1-SR) * m_flow_sCO2_des


                                                                      # sCO2
Thermodynamic States
    state_MH_hot_sCO2 = Medium3.setState_pTX(p_high_blk, TIT)
    state_MH_cold_sCO2 = Medium3.setState_pTX(p_high_blk, T_MH_sCO2_cold)
    #print(T_MH_sCO2_cold) #Temperatura 825,75
    #print("347")
    h_MH_sCO2_hot_des = Medium3.specificEnthalpy(state_MH_hot_sCO2)
    h_MH_sCO2_cold_des = Medium3.specificEnthalpy(state_MH_cold_sCO2)
    if PB_RR or PB_R or PB_RR_Inter or PB_R_Inter:
        state_RH_hot_sCO2 = Medium3.setState_pTX(p_int_blk, TIT)
        state_RH_cold_sCO2 = Medium3.setState_pTX(p_int_blk, T_RH_sCO2_cold)
        h_RH_sCO2_hot_des = Medium3.specificEnthalpy(state_RH_hot_sCO2)
        h_RH_sCO2_cold_des = Medium3.specificEnthalpy(state_RH_cold_sCO2)
                                                                      # Design
thermal power to the power Block
    #Q_flow_MH_des = m_flow_sCO2_des*(h_MH_sCO2_hot_des-
h_MH_sCO2_cold_des)/eta_PBHX
    Q_flow_MH_des=P_gross/eta_blk_des   #The requirement thermal power needed for
the PB at the desing point
    if PB_RR or PB_R or PB_RR_Inter or PB_R_Inter:
        Q_flow_RH_des = m_flow_sCO2_des*(h_RH_sCO2_hot_des-
h_RH_sCO2_cold_des)/eta_PBHX
        Q_flow_des=Q_flow_MH_des+Q_flow_RH_des                        # It can be
calculated also as P_gross/eta_blk_des

    else:
        Q_flow_des=Q_flow_MH_des                                      # It can be
calculated also as P_gross/eta_blk_des

    W_base_blk = par_fix_fr * P_gross                                 # Power
consumed at all times in power block
```

```
    P_net = eta_net_blk * P_gross                                  # Power block
net rating at design point
    P_name = P_net + P_AC                                          # Nominal Power
of the system
    print(u"\u2192 Design of the PB completed (Duration: %s s)"
%(np.around(time.time() - part1, decimals=0)))

#
__________________________________________________________________________________
__________________________________________________________________________________
_____
#                                                                  Control of the
Power Block Calculated Parameters
#
__________________________________________________________________________________
__________________________________________________________________________________
_____
    part1 = time.time()
                                                                   # Molten Salt-
side Thermodynamic States
    T_hot_set_TES = T_hot_set_REC
    state_hot_set_TES = Medium1.setState_pTX(Medium1.p_default, T_hot_set_TES)
    h_hot_set_TES = Medium1.specificEnthalpy(state_hot_set_TES)
    h_hot_set_REC = h_hot_set_TES
    state_cold_set_TES_input = Medium1.setState_pTX(Medium1.p_default,
T_cold_set_REC_input)
    h_cold_set_TES_input = Medium1.specificEnthalpy(state_cold_set_TES_input)
    state_min_cold_TES_MH = Medium1.setState_pTX(Medium1.p_default,
T_MH_sCO2_cold+DT_pinch_TESsCO2)
    h_cold_min_TES_MH =  Medium1.specificEnthalpy(state_min_cold_TES_MH)

    h_cold_set_TES_MH = max(h_cold_set_TES_input, h_cold_min_TES_MH)
    state_cold_TES_MH = Medium1.setState_phX(Medium1.p_default, h_cold_set_TES_MH)
    T_cold_set_TES_MH = Medium1.temperature(state_cold_TES_MH)
    m_flow_blk_MH = Q_flow_MH_des / (h_hot_set_TES - h_cold_set_TES_MH) # Mass flow
rate to power block MH at design point


    m_flow_startup_MH = m_flow_blk_MH/2                            # Mass flow
rate to power block at startup
    m_flow_standby = 0                                             # Mass flow
rate to power block at standby
    m_flow_off = 0                                                 # Mass flow
rate to power block during no operation
    if PB_RR or PB_R or PB_RR_Inter or PB_R_Inter:
        state_min_cold_TES_RH = Medium1.setState_pTX(Medium1.p_default,
T_RH_sCO2_cold+DT_pinch_TESsCO2)
        h_cold_min_TES_RH =  Medium1.specificEnthalpy(state_min_cold_TES_RH)
        T_cold_min_TES_RH = Medium1.temperature(state_min_cold_TES_RH)
        h_cold_set_TES_RH = max(h_cold_set_TES_input, h_cold_min_TES_RH)
        state_cold_TES_RH = Medium1.setState_phX(Medium1.p_default,
h_cold_set_TES_RH)
        T_cold_set_TES_RH = Medium1.temperature(state_cold_TES_RH)
        m_flow_blk_RH = Q_flow_RH_des / (h_hot_set_TES - h_cold_set_TES_RH) # Mass
flow rate to power block RH at design point
        m_flow_startup_RH = m_flow_blk_RH/2                        # Mass flow
rate to power block at startup
    print(u"\u2192 Design of the Controller completed (Duration: %s s)"
%(np.around(time.time() - part1, decimals=0)))
```

```
#
____________________________________________________________________________________
____________________________________________________________________________________
_____
#                                                                     Storage
Calculated Parameters
#
____________________________________________________________________________________
____________________________________________________________________________________
_____
    part1 = time.time()
    if PB_RR or PB_R or PB_RR_Inter or PB_R_Inter:                     # Cold salt
specific enthalpy at design
        h_cold_set_TES = MIX.Mixer(m_flow_blk_MH, h_cold_set_TES_MH, m_flow_blk_RH,
h_cold_set_TES_RH)
    else:
        h_cold_set_TES = h_cold_set_TES_MH
    state_cold_set_TES = Medium1.setState_phX(Medium1.p_default, h_cold_set_TES) #
Cold salt thermodynamic state at design
    T_cold_set_TES = Medium1.temperature(state_cold_set_TES)         # Cold salt
specific enthalpy at design
    T_cold_set_REC = T_cold_set_TES
    T_cold_start_TES = T_cold_set_TES
    DT_TES = T_hot_set_REC - T_cold_set_TES                            # Design DT
storage
    #print(T_cold_set_TES)
    #print("423")
    state_hot_set_TES = Medium1.setState_pTX(Medium1.p_default, T_hot_set_REC) #
Hold salt thermodynamic state at design
    h_hot_set_TES = Medium1.specificEnthalpy(state_hot_set_TES)      # Hot salt
specific enthalpy at design
    E_max = t_storage * 3600 * Q_flow_des                              # [J] - Maximum
tank stored energy
    rho_cold_set = Medium1.density(state_cold_set_TES)                 # Cold salt
density at design
    rho_hot_set = Medium1.density(state_hot_set_TES)                   # Hot salt
density at design
    m_max = E_max / (h_hot_set_TES - h_cold_set_TES)                   # Max salt mass
in tanks
    V_max = m_max / ((rho_hot_set + rho_cold_set) / 2)                 # Max salt
volume in tanks
    tank_min_l = 1.8                                                   # Storage tank
fluid minimum height - Based on NREL Gen3 SAM model v14.02.2020
    D_storage = (4*V_max/(MA.pi*(H_storage - tank_min_l)))**0.5
    V_tank = (H_storage*MA.pi*D_storage**2)/4                          # Tank Volume
    A_surf_TES=MA.pi*D_storage*H_storage + MA.pi*D_storage**2/4      # [m2]
    HT_Design=[V_tank, D_storage, H_storage, alpha, W_heater_hot, T_hot_aux_set,
e_ht]
    CT_Design=[V_tank, D_storage, H_storage, alpha, W_heater_cold, T_cold_aux_set,
e_ht]
    state_cold_start_TES = Medium1.setState_pTX(Medium1.p_default,
T_cold_start_TES) # Cold salt thermodynamic state at design
    state_hot_start_TES = Medium1.setState_pTX(Medium1.p_default, T_hot_start_TES)
# Hold salt thermodynamic state at design
    h_cold_start_TES = Medium1.specificEnthalpy(state_cold_start_TES) # Cold salt
specific enthalpy at design
    h_hot_start_TES = Medium1.specificEnthalpy(state_hot_start_TES) # Hot salt
specific enthalpy at design
    rho_cold_start = Medium1.density(state_cold_start_TES)            # Cold salt
density at design
```

```
    rho_hot_start = Medium1.density(state_hot_start_TES)            # Hot salt
density at design
    print(u"\u2192 Design of the TES completed (Duration: %s s)"
%(np.around(time.time() - part1, decimals=0)))

#
_______________________________________________________________________________
_______________________________________________________________________________
_____
#                                                                Electric Heater
Calculated Parameters
#
_______________________________________________________________________________
_______________________________________________________________________________
_____
    Q_name_EH = P_name_EH*eta_heater_design
    state_inlet_EH_des = Medium1.setState_pTX(Medium1.p_default, T_cold_set_REC)
    state_outlet_EH_des = Medium1.setState_pTX(Medium1.p_default, T_hot_set_REC)
    h_cold_set_EH = Medium1.specificEnthalpy(state_inlet_EH_des)
    h_hot_set_EH = Medium1.specificEnthalpy(state_outlet_EH_des)
    m_flow_EH_des = Q_name_EH/(h_hot_set_EH-h_cold_set_EH)          # Design
Electric Heater mass flow rate

#
_______________________________________________________________________________
_______________________________________________________________________________
_____
#                                                                Heliostat Field
and Receiver Calculated Parameters
#
_______________________________________________________________________________
_______________________________________________________________________________
_____
    part1 = time.time()
    helio_map_file = '%s%s' %(optical_path, helio_map_name)
    optical_file = '%s%s' %(optical_path, optical_file_name)
    Q_rec_des = Q_flow_des * SM                                     # Heat from
receiver at design
    [_, tower_fixed_cost, tower_exp] = CM.Cost_Tower(M_conv=M_conv_currency_to_USD,
H_tower=1)
    #[_, C_receiver_ref, A_receiver_ref, rec_cost_exp] =
CM.Cost_Receiver(M_conv=M_conv_currency_to_USD, A_receiver = 1)
    [_, pri_land] = CM.Cost_Land(M_conv = M_conv_currency_to_USD, A_land_tot = 1)
    [_, pri_site] = CM.Cost_SiteImprovements(M_conv=M_conv_currency_to_USD, A_SF=1)
    [_, pri_field] = CM.Cost_SolarField(M_conv=M_conv_currency_to_USD, A_SF=1)
    geom_inputs=[N_pa_rec, D_tb_rec, t_tb_rec, N_fl]
    [A_SF, n_heliostat, A_single_heliostat, A_land_base, A_land_CSP_tot, H_tower,
H_rec, ar_rec, D_rec, D_tower, C_ratio, Map_Helio,
SF_Efficiency]=HF.Design_HeliostatField(
            use_SolarPilot = use_SolarPilot,
            DNI_des = DNI_des,
            helio_width = helio_width,
            helio_height = helio_height,
            excl_fac = excl_fac,
            he_av_design = he_av_design,
            FlatPlate = FlatPlate,
            D_rec_input = D_rec_input,
            ar_rec_input = ar_rec_input,
            Q_rec_des = Q_rec_des,
            w_file = w_file,
            helio_reflectance = helio_reflectance,
            Optimize_SF = Optimize_SF,
            H_tower_input = H_tower_input,
            rec_absorptance = rec_absorptance,
            rec_hl_perm_guess = rec_hl_perm_guess,
            input_helio_map = input_helio_map,
            helio_map_file = helio_map_file,
            land_max = land_max,
            land_min = land_min,
            tower_fixed_cost = tower_fixed_cost,
            tower_exp = tower_exp,
            #C_receiver_ref = C_receiver_ref,
            #A_receiver_ref = A_receiver_ref,
            #rec_cost_exp = rec_cost_exp,
            site_spec_cost = pri_site,
            heliostat_spec_cost = pri_field,
            land_spec_cost = pri_land,
            contingency_rate = f_contingency_CSP,
            cost_sf_fixed = 0,
            sales_tax_frac = 100,
            sales_tax_rate = 0
            )
                                                                    # Receiver
Design
    T_amb_des = T_in_air_cooler_des
    Wspd_des_Ht=Wspd_des*(H_tower/elev_location)**alpha_wind

    [Rec_design, eta_rec_des, eta_rec_th_des, Q_loss_rec_des, R_des,
m_flow_rec_des]=REFREC.Design_Receiver(
        Q_rec_des, T_hot_set_REC, T_cold_set_REC,eta_rec_input)
    #print(T_cold_set_REC)
    #print("508")
    #H_rec = Rec_design[1]
    #A_receiver = Rec_design[3]
    #A_out_losses_rec = Rec_design[4]
    #A_in_losses_rec = Rec_design[5]
    #A_cs_rec = Rec_design[6]
    #N_tubes_rec = Rec_design[7]
    #D_in_tb_rec = Rec_design[9]
    #w_pa = Rec_design[11]
    #N_tb_pa = Rec_design[12]
    #V_rec = Rec_design[14]
    #rec_hl_perm=(R_des-Q_rec_des)/(A_receiver)/1000                 # [kW/m2]
Receiver design heat loss
    m_flow_max_REC = 1.5 * m_flow_rec_des                           # Maximum mass
flow rate from/to receiver
    m_flow_start_REC = m_flow_rec_des                               # Initial or
guess value of mass flow rate from/to heat exchanger in the feedback controller
    Q_field_design = (R_des - Q_rec_des + Q_flow_des)
    time_des=(24*(day_des-1)+hour_des)*3600
    [hra_des, dec_des] = SFun.SolarPosition(time_des, time_zone, lon)
    azi_deg180_des = GU.to_deg(SFun.azimuthAngle(dec_des, hra_des, lat))+180
    ele_deg_des = GU.to_deg(SFun.elevetionAngle(dec_des, hra_des, lat))
    eff_SF_des=SF_Efficiency(azi_deg180_des, ele_deg_des)
    print(u"\u2192 Design of the Receiver and SF completed (Duration: %s s)"
%(np.around(time.time() - part1, decimals=0)))
                                                                # Heat exchange
Design
```

```
#[NTU_UC_d,NTU_fu_d,Fact_c,Area_HX,eta_HX_de,m_flow_sCO2_design,m_flow_p_design,Del
ta_1]=PBHX.Desing_HeatExchanger( T_hot_start_TES-273.15,T_source_cold_design-
273.15,TIT-273.15,T_MH_sCO2_cold-273.15, Medium1.h_T(T_hot_start_TES-
273.15),Medium3.h_T(p_high_blk,T_MH_sCO2_cold),Medium1.h_T(T_source_cold_design-
273.15),Medium3.h_T(p_high_blk,TIT),Q_flow_des,U_HX_design,p_high_blk
    #)

[NTU_UC,Area_HX,m_flow_sCO2_design,m_flow_p_design,eta_de_HX]=PBHX.Desing_HeatExcha
nger( T_hot_start_TES,T_source_cold_design, TIT,T_MH_sCO2_cold,
Medium1.h_T(T_hot_start_TES),Medium3.h_T(p_high_blk,T_MH_sCO2_cold),Medium1.h_T(T_s
ource_cold_design),Medium3.h_T(p_high_blk,TIT),Q_flow_des,U_HX_design,p_high_blk)
    #print(m_flow_sCO2_design) #test
    #print(m_flow_sCO2_des)   #
    #print(m_flow_p_design)
    #print(m_flow_blk_MH)
    #print("522")


#
____________________________________________________________________________________
____________________________________________________________________________________
_____
#                                                                    Calculated
Costs
#
____________________________________________________________________________________
____________________________________________________________________________________
_____
    part1 = time.time()
                                                                     # PV - Costs
    C_modules_PV = CM.Cost_PV_Field(M_conv_currency_to_USD, P_DC)
    C_BoS_PV = CM.Cost_PV_BoS(M_conv_currency_to_USD, P_DC)
    [C_site_PV, _] = CM.Cost_SiteImprovements(M_conv_currency_to_USD, A_PV_field) #
Site improvements cost
    C_inverter_PV = CM.Cost_Inverter(M_conv_currency_to_USD, P_AC)
    [C_land_PV, _] = CM.Cost_Land(M_conv_currency_to_USD, A_land_PV_tot) # PV Land
cost
    C_year_PV = CM.Cost_OperationAndMaintenence_PV_Fixed(M_conv_currency_to_USD,
P_AC)
                                                                     # Hybrid
Component
    C_heater = CM.Cost_ElectricalHeater(M_conv_currency_to_USD, Q_name_EH) #
Electric Heater cost
                                                                     # CSP Section
    [C_land_CSP, _] = CM.Cost_Land(M_conv_currency_to_USD, A_land_CSP_tot) # CSP
Land cost
    [C_site_CSP, _] = CM.Cost_SiteImprovements(M_conv_currency_to_USD, A_SF) # Site
improvements cost
    [C_field, _] = CM.Cost_SolarField(M_conv_currency_to_USD, A_SF) # Field cost
    [C_tower, _, _] = CM.Cost_Tower(M_conv_currency_to_USD, H_tower) # Tower cost
    #A_receiver=100 #
    C_receiver = CM.Cost_Receiver(M_conv_currency_to_USD, Q_rec_des) # Receiver
cost
    C_storage = CM.Cost_ThermalEnergyStorage(M_conv_currency_to_USD, E_max, DT_TES,
m_max) # Storage cost
    #C_cogeneration=CM.Cost_Cogeneretion(M_conv_currency_to_USD,Q_de)
    C_HX_part=CM.Cost_HX(M_conv_currency_to_USD,Q_flow_des)
                                                                     # Power Block
cost
```

```
    if PB_S:
        C_MC1 = CM.Cost_Compressor_PB(M_conv_currency_to_USD, W_MC1_des)
        C_MC2 = 0
        C_RC = 0
        C_HPT = CM.Cost_Turbine_PB(M_conv_currency_to_USD, TIT, W_HPT_des)
        C_LPT = 0
        C_MH = CM.Cost_Heater_PB(M_conv_currency_to_USD, UA_MH_des)
        C_RH = 0
        C_HTR = CM.Cost_Recuperator_PB(M_conv_currency_to_USD, T_max_HTR_sCO2, 
UA_HTR_des)
        C_LTR = 0
        C_cooler = CM.Cost_Cooler_PB(M_conv_currency_to_USD, UA_cooler_des)
        C_intercooler = 0
        W_net_turbomachinery_des = W_HPT_des-W_MC1_des
        C_gearbox = CM.Cost_Gearbox_PB(M_conv_currency_to_USD, 
W_net_turbomachinery_des)
        C_generator = CM.Cost_Generator_PB(M_conv_currency_to_USD, 
W_net_turbomachinery_des)
    if PB_S_Inter:
        C_MC1 = CM.Cost_Compressor_PB(M_conv_currency_to_USD, W_MC1_des)
        C_MC2 = CM.Cost_Compressor_PB(M_conv_currency_to_USD, W_MC2_des)
        C_RC = 0
        C_HPT = CM.Cost_Turbine_PB(M_conv_currency_to_USD, TIT, W_HPT_des)
        C_LPT = 0
        C_MH = CM.Cost_Heater_PB(M_conv_currency_to_USD, UA_MH_des)
        C_RH = 0
        C_HTR = CM.Cost_Recuperator_PB(M_conv_currency_to_USD, T_max_HTR_sCO2, 
UA_HTR_des)
        C_LTR = 0
        C_cooler = CM.Cost_Cooler_PB(M_conv_currency_to_USD, UA_cooler_des)
        C_intercooler = CM.Cost_Cooler_PB(M_conv_currency_to_USD, 
UA_intercooler_des)
        W_net_turbomachinery_des = W_HPT_des-W_MC1_des-W_MC2_des
        C_gearbox = CM.Cost_Gearbox_PB(M_conv_currency_to_USD, 
W_net_turbomachinery_des)
        C_generator = CM.Cost_Generator_PB(M_conv_currency_to_USD, 
W_net_turbomachinery_des)
    if PB_R:
        C_MC1 = CM.Cost_Compressor_PB(M_conv_currency_to_USD, W_MC1_des)
        C_MC2 = 0
        C_RC = 0
        C_HPT = CM.Cost_Turbine_PB(M_conv_currency_to_USD, TIT, W_HPT_des)
        C_LPT = CM.Cost_Turbine_PB(M_conv_currency_to_USD, TIT, W_LPT_des)
        C_MH = CM.Cost_Heater_PB(M_conv_currency_to_USD, UA_MH_des)
        C_RH = CM.Cost_Heater_PB(M_conv_currency_to_USD, UA_RH_des)
        C_HTR = CM.Cost_Recuperator_PB(M_conv_currency_to_USD, T_max_HTR_sCO2, 
UA_HTR_des)
        C_LTR = 0
        C_cooler = CM.Cost_Cooler_PB(M_conv_currency_to_USD, UA_cooler_des)
        C_intercooler = 0
        W_net_turbomachinery_des = W_HPT_des+W_LPT_des-W_MC1_des
        C_gearbox = CM.Cost_Gearbox_PB(M_conv_currency_to_USD, 
W_net_turbomachinery_des)
        C_generator = CM.Cost_Generator_PB(M_conv_currency_to_USD, 
W_net_turbomachinery_des)
    if PB_R_Inter:
        C_MC1 = CM.Cost_Compressor_PB(M_conv_currency_to_USD, W_MC1_des)
        C_MC2 = CM.Cost_Compressor_PB(M_conv_currency_to_USD, W_MC2_des)
        C_RC = 0
        C_HPT = CM.Cost_Turbine_PB(M_conv_currency_to_USD, TIT, W_HPT_des)
        C_LPT = CM.Cost_Turbine_PB(M_conv_currency_to_USD, TIT, W_LPT_des)
        C_MH = CM.Cost_Heater_PB(M_conv_currency_to_USD, UA_MH_des)
        C_RH = CM.Cost_Heater_PB(M_conv_currency_to_USD, UA_RH_des)
        C_HTR = CM.Cost_Recuperator_PB(M_conv_currency_to_USD, T_max_HTR_sCO2,
UA_HTR_des)
        C_LTR = 0
        C_cooler = CM.Cost_Cooler_PB(M_conv_currency_to_USD, UA_cooler_des)
        C_intercooler = CM.Cost_Cooler_PB(M_conv_currency_to_USD,
UA_intercooler_des)
        W_net_turbomachinery_des = W_HPT_des+W_LPT_des-W_MC1_des-W_MC2_des
        C_gearbox = CM.Cost_Gearbox_PB(M_conv_currency_to_USD,
W_net_turbomachinery_des)
        C_generator = CM.Cost_Generator_PB(M_conv_currency_to_USD,
W_net_turbomachinery_des)
    if PB_Recomp:
        C_MC1 = CM.Cost_Compressor_PB(M_conv_currency_to_USD, W_MC1_des)
        C_MC2 = 0
        C_RC = CM.Cost_Compressor_PB(M_conv_currency_to_USD, W_RC_des)
        C_HPT = CM.Cost_Turbine_PB(M_conv_currency_to_USD, TIT, W_HPT_des)
        C_LPT = 0
        C_MH = CM.Cost_Heater_PB(M_conv_currency_to_USD, UA_MH_des)
        C_RH = 0
        C_HTR = CM.Cost_Recuperator_PB(M_conv_currency_to_USD, T_max_HTR_sCO2,
UA_HTR_des)
        C_LTR = CM.Cost_Recuperator_PB(M_conv_currency_to_USD, T_max_LTR_sCO2,
UA_LTR_des)
        C_cooler = CM.Cost_Cooler_PB(M_conv_currency_to_USD, UA_cooler_des)
        C_intercooler = 0
        W_net_turbomachinery_des = W_HPT_des-W_MC1_des-W_RC_des
        C_gearbox = CM.Cost_Gearbox_PB(M_conv_currency_to_USD,
W_net_turbomachinery_des)
        C_generator = CM.Cost_Generator_PB(M_conv_currency_to_USD,
W_net_turbomachinery_des)
    if PB_Recomp_Inter:
        C_MC1 = CM.Cost_Compressor_PB(M_conv_currency_to_USD, W_MC1_des)
        C_MC2 = CM.Cost_Compressor_PB(M_conv_currency_to_USD, W_MC2_des)
        C_RC = CM.Cost_Compressor_PB(M_conv_currency_to_USD, W_RC_des)
        C_HPT = CM.Cost_Turbine_PB(M_conv_currency_to_USD, TIT, W_HPT_des)
        C_LPT = 0
        C_MH = CM.Cost_Heater_PB(M_conv_currency_to_USD, UA_MH_des)
        C_RH = 0
        C_HTR = CM.Cost_Recuperator_PB(M_conv_currency_to_USD, T_max_HTR_sCO2,
UA_HTR_des)
        C_LTR = CM.Cost_Recuperator_PB(M_conv_currency_to_USD, T_max_LTR_sCO2,
UA_LTR_des)
        C_cooler = CM.Cost_Cooler_PB(M_conv_currency_to_USD, UA_cooler_des)
        C_intercooler = CM.Cost_Cooler_PB(M_conv_currency_to_USD,
UA_intercooler_des)
        W_net_turbomachinery_des = W_HPT_des-W_MC1_des-W_MC2_des-W_RC_des
        C_gearbox = CM.Cost_Gearbox_PB(M_conv_currency_to_USD,
W_net_turbomachinery_des)
        C_generator = CM.Cost_Generator_PB(M_conv_currency_to_USD,
W_net_turbomachinery_des)
    if PB_RR:
        C_MC1 = CM.Cost_Compressor_PB(M_conv_currency_to_USD, W_MC1_des)
        C_MC2 = 0
        C_RC = CM.Cost_Compressor_PB(M_conv_currency_to_USD, W_RC_des)
        C_HPT = CM.Cost_Turbine_PB(M_conv_currency_to_USD, TIT, W_HPT_des)
        C_LPT = CM.Cost_Turbine_PB(M_conv_currency_to_USD, TIT, W_LPT_des)
        C_MH = CM.Cost_Heater_PB(M_conv_currency_to_USD, UA_MH_des)
        C_RH = CM.Cost_Heater_PB(M_conv_currency_to_USD, UA_RH_des)
        C_HTR = CM.Cost_Recuperator_PB(M_conv_currency_to_USD, T_max_HTR_sCO2,
UA_HTR_des)
        C_LTR = CM.Cost_Recuperator_PB(M_conv_currency_to_USD, T_max_LTR_sCO2,
UA_LTR_des)
        C_cooler = CM.Cost_Cooler_PB(M_conv_currency_to_USD, UA_cooler_des)
        C_intercooler = 0
        W_net_turbomachinery_des = W_HPT_des+W_LPT_des-W_MC1_des-W_RC_des
        C_gearbox = CM.Cost_Gearbox_PB(M_conv_currency_to_USD,
W_net_turbomachinery_des)
        C_generator = CM.Cost_Generator_PB(M_conv_currency_to_USD,
W_net_turbomachinery_des)
    if PB_RR_Inter:
        C_MC1 = CM.Cost_Compressor_PB(M_conv_currency_to_USD, W_MC1_des)
        C_MC2 = CM.Cost_Compressor_PB(M_conv_currency_to_USD, W_MC2_des)
        C_RC = CM.Cost_Compressor_PB(M_conv_currency_to_USD, W_RC_des)
        C_HPT = CM.Cost_Turbine_PB(M_conv_currency_to_USD, TIT, W_HPT_des)
        C_LPT = CM.Cost_Turbine_PB(M_conv_currency_to_USD, TIT, W_LPT_des)
        C_MH = CM.Cost_Heater_PB(M_conv_currency_to_USD, UA_MH_des)
        C_RH = CM.Cost_Heater_PB(M_conv_currency_to_USD, UA_RH_des)
        C_HTR = CM.Cost_Recuperator_PB(M_conv_currency_to_USD, T_max_HTR_sCO2,
UA_HTR_des)
        C_LTR = CM.Cost_Recuperator_PB(M_conv_currency_to_USD, T_max_LTR_sCO2,
UA_LTR_des)
        C_cooler = CM.Cost_Cooler_PB(M_conv_currency_to_USD, UA_cooler_des)
        C_intercooler = CM.Cost_Cooler_PB(M_conv_currency_to_USD,
UA_intercooler_des)
        W_net_turbomachinery_des = W_HPT_des+W_LPT_des-W_MC1_des-W_MC2_des-W_RC_des
        C_gearbox = CM.Cost_Gearbox_PB(M_conv_currency_to_USD,
W_net_turbomachinery_des)
        C_generator = CM.Cost_Generator_PB(M_conv_currency_to_USD,
W_net_turbomachinery_des)
                                                              # PB Equipment
Costs
    C_equipment_PB = C_MC1 + C_MC2 + C_RC + C_HPT + C_LPT + C_MH + C_RH + C_HTR +
C_LTR + C_cooler + C_intercooler + C_generator + C_gearbox + C_HX_part
    #C_equipment_PB= C_equipment_PB+ C_cogeneration
    C_piping_PB = CM.Cost_Piping_PB(C_equipment_PB, Reheat)         # PB Piping
Cost
    C_block = C_equipment_PB + C_piping_PB                          # Power block
cost
    C_bop = CM.Cost_BalanceOfPlant(M_conv_currency_to_USD, P_gross) # Balance of
plant cost
    C_site = C_site_CSP
                                                              # Direct
capital cost subtotal - i.e. purchased equipment costs
    C_Cap_CSP = (C_field + C_site + C_tower + C_receiver + C_storage + C_block +
C_bop + C_heater)
    C_Cap_PV = (C_modules_PV + C_BoS_PV + C_site_PV + C_inverter_PV)
    C_year_CSP = CM.Cost_OperationAndMaintenence_CSP_Fixed(M_conv_currency_to_USD,
P_net) # Fixed O&M cost per year
    C_year = C_year_CSP + C_year_PV
    c_OM_CSP = CM.Cost_OperationAndMaintenence_CSP(M_conv_currency_to_USD)
    print(u"\u2192 Definition of the Costs completed (Duration: %s s)"
%(np.around(time.time() - part1, decimals=0)))
    design_time = time.time()
    print(u"\u2192 Design of the plant completed (Duration: %s s)"
%(np.around(design_time - start_time, decimals=0)))
```

```
# ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Operation
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
#
_________________________________________________________________________________
_________________________________________________________________________________
_____
#                                                                  Sun Operation
#
_________________________________________________________________________________
_________________________________________________________________________________
_____
    [time_vec_original, hra_deg, dec_deg, ele_deg, azi_deg, zen_deg, DNI_original,
Tamb, Wspd_original]=SUN.SolarCalculations(w_file)
    if Short_Simulation==True:
        time_vec=time_vec_original[shortime_start:shortime_end]
        DNI=DNI_original[shortime_start:shortime_end]
        Wspd = Wspd_original[shortime_start:shortime_end]
    else:
        time_vec=time_vec_original
        DNI=DNI_original
        Wspd = Wspd_original
    ele=np.array(ele_deg)*MA.pi/180
    if use_SolarPilot:
        azi_deg180=np.array(azi_deg)+180*np.ones(len(azi_deg))
        eff_field_SP=SF_Efficiency(azi_deg180, ele_deg)
    else:
        eff_field_SP=0*np.ones(len(azi_deg))
#
_________________________________________________________________________________
_________________________________________________________________________________
_____
#                                                                  Definition and
Initialization of the variables
#
_________________________________________________________________________________
_________________________________________________________________________________
_____
    # 1) Heliostat Field
    Q_in_SF = np.empty(len(time_vec))
    Q_raw_SF = np.empty(len(time_vec))
    Q_out_SF = np.empty(len(time_vec))
    Q_wasted_startup = np.empty(len(time_vec))
    Q_wasted_defocus = np.empty(len(time_vec))
    Q_wasted_CSP = np.empty(len(time_vec))
    eta_opt_SF = np.empty(len(time_vec))
    SF_on=np.empty(len(time_vec))
    Wspd_Ht=Wspd*(H_tower/elev_location)**alpha_wind
    W_loss_HF = np.empty(len(time_vec))
    defocus=np.empty(len(time_vec))
    defocus1=np.empty(len(time_vec))
    defocus2=np.empty(len(time_vec))
    Q_flow_defocus = np.empty(len(time_vec))
    # 2) Receiver
    Q_out_rec = np.empty(len(time_vec))
    eta_rec = np.empty(len(time_vec))
    eta_th_rec = np.empty(len(time_vec))
    Q_loss_rec=np.empty(len(time_vec))
    h_REC_hot = np.empty(len(time_vec))
    h_REC_cold = np.empty(len(time_vec))
```

```
T_REC_cold = np.empty(len(time_vec))
T_REC_hot = np.empty(len(time_vec))
m_flow_rec = np.empty(len(time_vec))
# 3) Electric Heater
m_flow_heater = np.empty(len(time_vec))
W_heater_PV = np.empty(len(time_vec))
Q_missing_defocus = np.empty(len(time_vec))
Q_out_heater = np.empty(len(time_vec))
h_EH_hot = np.empty(len(time_vec))
h_EH_cold = np.empty(len(time_vec))
T_EH_cold = np.empty(len(time_vec))
T_EH_hot = np.empty(len(time_vec))
eta_heater = np.empty(len(time_vec))
Q_loss_heater = np.empty(len(time_vec))
waste_extra_PV = np.empty(len(time_vec))
W_wasted_PV_tot = np.empty(len(time_vec))
W_wasted_PV_defocus = np.empty(len(time_vec))
# 4) Thermal Energy Storage
h_TES_cold=np.empty(len(time_vec))
h_TES_hot=np.empty(len(time_vec))
h_toTES_cold = np.empty(len(time_vec))
h_toTES_hot = np.empty(len(time_vec))
T_TES_cold = np.empty(len(time_vec))
T_TES_hot = np.empty(len(time_vec))
T_toTES_cold = np.empty(len(time_vec))
T_toTES_hot = np.empty(len(time_vec))
cold_tank_ready=np.empty(len(time_vec))
L_HT=np.empty(len(time_vec))
h_HT=np.empty(len(time_vec))
m_HT=np.empty(len(time_vec))
Q_losses_HT=np.empty(len(time_vec))
W_loss_HT=np.empty(len(time_vec))
L_CT=np.empty(len(time_vec))
h_CT=np.empty(len(time_vec))
m_CT=np.empty(len(time_vec))
Q_losses_CT=np.empty(len(time_vec))
W_loss_CT=np.empty(len(time_vec))
HT_on_discharge=np.empty(len(time_vec))
HT_on_charge=np.empty(len(time_vec))
Q_flow_PB = np.empty(len(time_vec))
m_flow_PB = np.empty(len(time_vec))
m_flow_TES_cold = np.empty(len(time_vec))
m_flow_toTES_cold = np.empty(len(time_vec))
m_flow_toTES_hot = np.empty(len(time_vec))
m_flow_TES_hot = np.empty(len(time_vec))
# 5) Power Block
F_prod = np.empty(len(time_vec))
F_prod_off = np.empty(len(time_vec))
PB_on = np.empty(len(time_vec))
PB_ramp = np.empty(len(time_vec))
t_on_PB=np.empty(len(time_vec))
t_off_PB=np.empty(len(time_vec))
rampUP_PB=np.empty(len(time_vec))
t_ramp_start=np.empty(len(time_vec))
t_ramp_end=np.empty(len(time_vec))
Q_flow_MH = np.empty(len(time_vec))
m_flow_MH = np.empty(len(time_vec))
h_cold_MH = np.empty(len(time_vec))
T_toTES_MH_cold = np.empty(len(time_vec))
Q_sCO2_PB_MH = np.empty(len(time_vec))
```

```python
m_flow_sCO2_MH = np.empty(len(time_vec))
h_sCO2_hot_PB_MH = np.empty(len(time_vec))
T_cold_MH=np.empty(len(time_vec))
#T_source_hot=np.empty(len(time_vec))
h_sCO2_cold_PB_MH = np.empty(len(time_vec))
T_sCO2_hot_MH = np.empty(len(time_vec))
T_sCO2_cold_MH = np.empty(len(time_vec))
W_T1 = np.empty(len(time_vec))
W_MC1 = np.empty(len(time_vec))
T_sCO2_1 = np.empty(len(time_vec))
T_sCO2_2 = np.empty(len(time_vec))
T_sCO2_3 = np.empty(len(time_vec))
T_sCO2_4 = np.empty(len(time_vec))
T_sCO2_5 = np.empty(len(time_vec))
T_sCO2_6 = np.empty(len(time_vec))
W_loss_PB_pumps = np.empty(len(time_vec))
parasities_blk = np.empty(len(time_vec))
W_net_CSP=np.empty(len(time_vec))
W_net_tot=np.empty(len(time_vec))
eff_PB = np.empty(len(time_vec))
Q_cond = np.empty(len(time_vec))
Q_HTR = np.empty(len(time_vec))
# Heat exchanger
NTU_UC=np.empty(len(time_vec))
eta_HXopp=np.empty(len(time_vec))
T_sCO2_hot_pb_check=np.empty(len(time_vec))
T_sCO2_cold_pb_check=np.empty(len(time_vec))
T_source_hot_pb_check=np.empty(len(time_vec))
T_source_cold_pb_check=np.empty(len(time_vec))

HX_co=np.empty(len(time_vec))
Q_co_HX=np.empty(len(time_vec))
T_coldtoTES_HX=np.empty(len(time_vec))


if PB_Recomp_Inter:
    Q_LTR = np.empty(len(time_vec))
    Q_intercooler = np.empty(len(time_vec))
    W_MC2 = np.empty(len(time_vec))
    W_RC = np.empty(len(time_vec))
    T_sCO2_7 = np.empty(len(time_vec))
    T_sCO2_8 = np.empty(len(time_vec))
    T_sCO2_9_prime = np.empty(len(time_vec))
    T_sCO2_9_second = np.empty(len(time_vec))
    T_sCO2_9 = np.empty(len(time_vec))
    T_sCO2_10 = np.empty(len(time_vec))


#Start Values for Variables
# 1) Heliostat Field
Q_in_SF[0] = 0
Q_raw_SF[0] = 0
Q_out_SF[0] = 0
Q_wasted_startup[0] = 0
Q_wasted_defocus[0] = 0
Q_wasted_CSP[0] = 0
eta_opt_SF[0]=0
SF_on[0] = False
W_loss_HF[0] = 0
defocus1[0] = False
```

```
defocus2[0] = False
defocus[0] = GU.Or(defocus1[0], defocus2[0])
Q_flow_defocus[0] = 0
# 2) Receiver
Q_out_rec[0] = 0
eta_rec[0] = 0
eta_th_rec[0] = 0
Q_loss_rec[0] = 0
h_REC_hot[0] = h_hot_set_REC
h_REC_cold[0] = h_REC_hot[0]
T_REC_hot[0] = GU.to_degC(T_hot_set_REC)
T_REC_cold[0] = GU.to_degC(T_cold_set_REC)
m_flow_rec[0] = 0
# 4) Electric Heater
m_flow_heater[0] = 0
W_heater_PV[0] = 0
Q_missing_defocus[0] = 0
Q_out_heater[0] = 0
h_EH_hot[0] = Medium1.specificEnthalpy(state_hot_set_TES)
h_EH_cold[0] = h_EH_hot[0]
T_EH_hot[0] = GU.to_degC(Medium1.temperature(state_hot_set_TES))
T_EH_cold[0] = T_EH_hot[0]
eta_heater[0] = 0
Q_loss_heater[0] = 0
waste_extra_PV[0] = 0
W_wasted_PV_tot[0] = 0
W_wasted_PV_defocus[0] = 0
# 5) Thermal Energy Storage
h_TES_cold[0] = h_cold_set_TES
h_TES_hot[0] = h_hot_set_TES
h_toTES_cold[0] = h_cold_set_TES
h_toTES_hot[0] = h_hot_set_TES
T_TES_cold[0] = GU.to_degC(T_cold_set_REC)
T_TES_hot[0] = GU.to_degC(T_hot_set_REC)
T_toTES_cold[0] = GU.to_degC(T_cold_set_REC)
T_toTES_hot[0] = GU.to_degC(T_hot_set_REC)
cold_tank_ready[0] = True
L_HT[0] = L_start
h_HT[0]=h_hot_start_TES
m_HT[0]= L_HT[0]/100*m_max
Q_losses_HT[0] = 0
W_loss_HT[0] = 0
L_CT[0] = 100-L_start
h_CT[0]=h_cold_start_TES
m_CT[0]=L_CT[0]/100*m_max
Q_losses_CT[0] = 0
W_loss_CT[0] = 0
HT_on_discharge[0] = (L_HT[0]>hot_tnk_empty_ub) and (L_HT[0]>hot_tnk_empty_lb)
HT_on_charge[0] = m_flow_rec[0]>0
Q_flow_PB[0] = 0
m_flow_PB[0] = 0
m_flow_TES_cold[0] = 0
m_flow_toTES_cold[0] = 0
m_flow_toTES_hot[0] = 0
m_flow_TES_hot[0] = 0
# 5) Power Block
F_prod[0] = 1
F_prod_off[0] = 1
PB_ramp[0] = 0
PB_on[0] = False
```

```
    t_ramp_start[0] = 0
    t_ramp_end[0] = 3600
    t_on_PB[0] = 0
    t_off_PB[0] = 0
    rampUP_PB[0]=False
    Q_flow_MH[0] = 0
    m_flow_MH[0] = 0
    h_cold_MH[0] = h_TES_hot[0]
    T_toTES_MH_cold[0] = GU.to_degC(T_hot_set_REC)
    Q_sCO2_PB_MH[0] = 0
    m_flow_sCO2_MH[0] = 0
    h_sCO2_hot_PB_MH[0] = h_MH_sCO2_cold_des
    h_sCO2_cold_PB_MH[0] = h_MH_sCO2_cold_des
    state_cold_PB_sCO2=state_MH_cold_sCO2
    T_sCO2_hot_MH[0] = GU.to_degC(Medium3.temperature(state_cold_PB_sCO2))
    T_sCO2_cold_MH[0] =GU.to_degC(Medium3.temperature(state_cold_PB_sCO2))   #
possibile problema nel valore iniziale
    eff_PB[0] = 0
    Q_cond[0] = 0
    Q_HTR[0] = 0
    W_T1[0] = 0
    W_MC1[0] = 0
    T_sCO2_1[0]=GU.to_degC(Medium3.temperature(state_cold_PB_sCO2))
    T_sCO2_2[0]=GU.to_degC(Medium3.temperature(state_cold_PB_sCO2))
    T_sCO2_3[0]=GU.to_degC(Medium3.temperature(state_cold_PB_sCO2))
    T_sCO2_4[0]=GU.to_degC(Medium3.temperature(state_cold_PB_sCO2))
    T_sCO2_5[0]=GU.to_degC(Medium3.temperature(state_cold_PB_sCO2))
    T_sCO2_6[0]=GU.to_degC(Medium3.temperature(state_cold_PB_sCO2))
    W_loss_PB_pumps[0] = 0
    parasities_blk[0] = 0
    W_net_tot[0] = 0
    W_net_CSP[0] = 0
    HX_co[0]=0


    if PB_Recomp_Inter:
        Q_LTR[0] = 0
        Q_intercooler[0] = 0
        W_MC2[0] = 0
        W_RC[0] = 0
        T_sCO2_7[0]=GU.to_degC(Medium3.temperature(state_cold_PB_sCO2))
        T_sCO2_8[0]=GU.to_degC(Medium3.temperature(state_cold_PB_sCO2))
        T_sCO2_9_prime[0]=GU.to_degC(Medium3.temperature(state_cold_PB_sCO2))
        T_sCO2_9_second[0]=GU.to_degC(Medium3.temperature(state_cold_PB_sCO2))
        T_sCO2_9[0]=GU.to_degC(Medium3.temperature(state_cold_PB_sCO2))
        T_sCO2_10[0]=GU.to_degC(Medium3.temperature(state_cold_PB_sCO2))

    TES_time = 0
    PB_time = 0
    PBControl_time = 0
    HF_time = 0
    RecEH_time = 0
    ControlRecEH_time = 0
    Temperature_time = 0
    print(u"\u2192 Initialization of the plant variables completed (Duration: %s
s)" %(np.around(time.time() - design_time, decimals=0)))

    for tt in range(1, len(time_vec)):
    #
```

```
____________________________________________________________________________________
_____
    #                                                                       Hourly
Helisotat Field Operation
    #
____________________________________________________________________________________
____________________________________________________________________________________
_____

        time0 = time.time()
        F_prod[tt] = (max(0,min(1,1 - PV_share[tt])))*(PD[tt])/(P_max)
        cold_tank_ready[tt] = T2LU.Tank2Logic(L_CT[tt-1], cold_tnk_crit_ub,
cold_tnk_crit_lb, cold_tank_ready[tt-1])
        if not cold_tank_ready[tt]:
            defocus_HF = True
        else:
            defocus_HF = defocus[tt-1]
        Q_flow_defocus[tt] = (R_des - Q_rec_des + Q_flow_des*F_prod[tt])
        [Q_in_SF[tt], Q_raw_SF[tt], Q_out_SF[tt], Q_wasted_startup[tt],
Q_wasted_defocus[tt], Q_wasted_CSP[tt], SF_on[tt], eta_opt_SF[tt], W_loss_HF[tt]] =
HF.Operating_HeliostatFIeld(
            A_SF, DNI[tt], Q_field_design, Q_flow_defocus[tt], Q_loss_rec_des,
nu_start, nu_min_sf, ele[tt], ele_min, Wspd[tt], Wspd_max, defocus_HF, SF_on[tt-1],
use_SolarPilot, eff_field_SP[tt], optical_file, hra_deg[tt], dec_deg[tt],
n_heliostat, W_track, he_av_design
        )
        DTime = time.time() - time0
        HF_time = HF_time + DTime

    #
____________________________________________________________________________________
____________________________________________________________________________________
_____
    #                                                                       Hourly PV
Operation
    #
____________________________________________________________________________________
____________________________________________________________________________________
_____
        # The following variables are calculated in PV Calculated parameters:
        #W_net_PV, W_heater_PV_raw, EH_on, PV_share
        Q_missing_defocus[tt] = max(0, Q_flow_defocus[tt]-Q_out_SF[tt]-
R_des+Q_rec_des)
        if waste_extra_PV[tt-1] or (not cold_tank_ready[tt]):
            W_wasted_PV_defocus[tt] = max(0, (W_heater_PV_raw[tt] -
Q_missing_defocus[tt]/eta_heater_design))
        else:
            W_wasted_PV_defocus[tt] = 0
        W_heater_PV[tt] = max(0,(W_heater_PV_raw[tt] - W_wasted_PV_defocus[tt]))
        W_wasted_PV_tot[tt] = W_wasted_PV_defocus[tt] + W_wasted_PV_plus[tt]

    #
____________________________________________________________________________________
____________________________________________________________________________________
_____
    #                                                                       Hourly
Receiver Control Output
    #
____________________________________________________________________________________
```

```
____________________________________________________________________________________
_____
        time0 = time.time()
        h_REC_cold[tt] = h_TES_cold[tt-1]
        [m_flow_rec[tt], defocus1[tt]]=RECS.ReferenceREC_ControlSystem(
            SF_on[tt], L_CT[tt-1], cold_tnk_defocus_lb, cold_tnk_defocus_ub,
defocus[tt-1], Q_out_SF[tt], h_REC_cold[tt], T_hot_set_REC, eta_rec_input,
input_eff, DNI[tt]
        )
    #
____________________________________________________________________________________
____________________________________________________________________________________
_____
    #                                                                      Hourly
Electric Heater Control Output
    #
____________________________________________________________________________________
____________________________________________________________________________________
_____
        h_EH_cold[tt] = h_TES_cold[tt-1]
        [m_flow_heater[tt], waste_extra_PV[tt]]=EHCS.ReferenceEH_ControlSystem(
            Medium1, EH_on[tt], L_CT[tt-1], cold_tnk_defocus_lb,
cold_tnk_defocus_ub, waste_extra_PV[tt-1], eta_heater_design, W_heater_PV[tt],
h_EH_cold[tt], T_hot_set_REC)
        DTime = time.time() - time0
        ControlRecEH_time = ControlRecEH_time + DTime

    #
____________________________________________________________________________________
____________________________________________________________________________________
_____
    #                                                                      Hourly
Receiver Opration
    #
____________________________________________________________________________________
____________________________________________________________________________________
_____
        time0 = time.time()
        [Q_out_rec[tt], h_REC_hot[tt], eta_rec[tt], eta_th_rec[tt],
Q_loss_rec[tt]]=REFREC.Operating_Receiver(
            DNI[tt], Q_out_SF[tt], h_REC_cold[tt], m_flow_rec[tt], SF_on[tt],
input_eff, eta_rec_input
        )

    #
____________________________________________________________________________________
____________________________________________________________________________________
_____
    #                                                                      Hourly
Electric Heater Opration
    #
____________________________________________________________________________________
____________________________________________________________________________________
_____
        [Q_out_heater[tt], h_EH_hot[tt], eta_heater[tt], Q_loss_heater[tt]]=
EH.Operating_ElectricHeater(
            W_heater_PV[tt], h_EH_cold[tt-1], m_flow_heater[tt], EH_on[tt],
eta_heater_design)
        m_flow_toTES_hot[tt] = m_flow_rec[tt]+ m_flow_heater[tt]
        h_toTES_hot[tt] = MIX.Mixer(m_flow_rec[tt], h_REC_hot[tt],
m_flow_heater[tt], h_EH_hot[tt])
        m_flow_TES_cold[tt] = m_flow_toTES_hot[tt]
        DTime = time.time() - time0
        RecEH_time = RecEH_time + DTime

    #
__________________________________________________________________________________
__________________________________________________________________________________
_____
    #                                                                   Hourly
Power Block Control Output
    #
__________________________________________________________________________________
__________________________________________________________________________________
_____
        time0 = time.time()
        T_cold_control_MH =GU.from_degC(T_sCO2_cold_MH[tt-1]) + DT_pinch_TESsCO2
        #print(T_sCO2_cold_MH[tt-1])
        #print("1034")

        state_cold_control_MH = Medium1.setState_pTX(Medium1.p_default,
T_cold_control_MH)
        h_cold_control_MH = Medium1.specificEnthalpy(state_cold_control_MH)
        if PB_on[tt-1]>0:
            if Tamb[tt]>T_in_air_cooler_des:
                F_prod_off[tt] = 1+(eta_blk_des-eff_PB[tt-1])/eta_blk_des
            else:
                F_prod_off[tt] = 1
        else:
            F_prod_off[tt] = 1
        if PB_S or PB_Recomp or PB_S_Inter or PB_Recomp_Inter:
            # m_flow_blk_MH = Q_flow_MH_des / (h_TES_hot[tt-1] - h_cold_set_TES_MH)
#Mass flow rate to power block MH at design point
            m_flow_blk_MH = Q_flow_MH_des / (h_TES_hot[tt-1] - h_cold_control_MH)
#Mass flow rate to power block MH at design point take in to account change it to
the design of the HX
            #print(Q_flow_MH_des)
            #print(h_TES_hot[tt-1])
            #print(h_cold_control_MH)
            #print(m_flow_blk_MH)
            [m_flow_MH[tt], PB_on[tt], PB_ramp[tt], defocus2[tt], W_loss_HT[tt],
HT_on_discharge[tt], HT_on_charge[tt], t_on_PB[tt], t_off_PB[tt], rampUP_PB[tt],
t_ramp_start[tt], t_ramp_end[tt]]=PBCS.PB_ControlSystem_MH(
                F_prod[tt], F_prod_off[tt], PB_load_min, m_flow_toTES_hot[tt],
L_HT[tt-1], time_vec[tt], m_flow_blk_MH, m_flow_blk_MH/2, m_flow_standby,
m_flow_off, hot_tnk_full_ub, hot_tnk_full_lb, hot_tnk_empty_ub, hot_tnk_empty_lb,
t_ramping, t_standby, t_ramping, k_loss_hot, defocus2[tt-1], HT_on_discharge[tt-1],
HT_on_charge[tt-1], t_on_PB[tt-1], t_off_PB[tt-1], rampUP_PB[tt-1],
t_ramp_start[tt-1], t_ramp_end[tt-1]
            )
            m_flow_PB[tt]=m_flow_MH[tt]



        if PB_R or PB_RR or PB_R_Inter or PB_RR_Inter:
            m_flow_blk_MH = Q_flow_MH_des / (h_TES_hot[tt-1] - h_cold_control_MH)
#Mass flow rate to power block MH at design point
            m_flow_blk_RH = Q_flow_RH_des / (h_TES_hot[tt-1] - h_cold_set_TES_RH)
#Mass flow rate to power block RH at design poin
```

```
            [m_flow_MH[tt], m_flow_RH[tt], PB_on[tt], PB_ramp[tt], defocus2[tt],
W_loss_HT[tt], HT_on_discharge[tt], HT_on_charge[tt], t_on_PB[tt], t_off_PB[tt],
rampUP_PB[tt], t_ramp_start[tt], t_ramp_end[tt]]=PBCS.PB_ControlSystem_MH_and_RH(
                F_prod[tt], F_prod_off[tt], PB_load_min, m_flow_toTES_hot[tt],
L_HT[tt-1], time_vec[tt], m_flow_blk_MH, m_flow_blk_RH, m_flow_blk_MH/2,
m_flow_blk_RH/2, m_flow_standby, m_flow_off, hot_tnk_full_ub, hot_tnk_full_lb,
hot_tnk_empty_ub, hot_tnk_empty_lb, t_ramping, t_standby, t_ramping, k_loss_hot,
defocus2[tt-1], HT_on_discharge[tt-1], HT_on_charge[tt-1], t_on_PB[tt-1],
t_off_PB[tt-1], rampUP_PB[tt-1], t_ramp_start[tt-1], t_ramp_end[tt-1]
            )
            print("1051 no")
            m_flow_PB[tt]=m_flow_MH[tt]+m_flow_RH[tt]
        m_flow_TES_hot[tt] = m_flow_PB[tt]
        m_flow_toTES_cold[tt] = m_flow_TES_hot[tt]
        defocus[tt]=GU.Or(defocus1[tt], defocus2[tt])
        DTime = time.time() - time0
        PBControl_time = PBControl_time + DTime

    #
__________________________________________________________________________________
__________________________________________________________________________________
_____
    #                                                                        Hourly TES
- Hot Tank Operation
    #
__________________________________________________________________________________
__________________________________________________________________________________
_____
        time0 = time.time()
        [L_HT[tt], h_HT[tt], m_HT[tt], Q_losses_HT[tt],
W_loss_HT[tt]]=TES.Storage_Tank(
            Medium1, m_flow_toTES_hot[tt], h_toTES_hot[tt], m_HT[tt-1], h_HT[tt-1],
m_flow_TES_hot[tt], Tamb[tt], time_vec[tt-1], time_vec[tt], dt, HT_Design, m_max
        )
        h_TES_hot[tt]=h_HT[tt]
        Dtime1 = time.time() - time0

    #
__________________________________________________________________________________
__________________________________________________________________________________
_____
    #                                                                        Primary
Heat Exchanger(s) Operation
    #
__________________________________________________________________________________
__________________________________________________________________________________
_____
        time0 = time.time()
        if PB_S or PB_Recomp or PB_S_Inter or PB_Recomp_Inter:          # Only Main
Heater
            #[Q_sCO2_PB_MH[tt], m_flow_sCO2_MH[tt], h_cold_MH[tt],
h_sCO2_hot_PB_MH[tt], T_cold_MH[tt], T_source_hot[tt]
]=PBHX.Operating_HeatExchanger(
            #    PB_on[tt], m_flow_MH[tt], h_TES_hot[tt], h_sCO2_cold_PB_MH[tt-1],
h_cold_control_MH, h_MH_sCO2_hot_des, eta_PBHX
            #)
            #Dum_var=Medium3.T_h(p_high_blk,h_sCO2_cold_PB_MH[tt-1])-273.15
            #Dum_var1=Medium1.T_h(h_TES_hot[tt])
            [Q_sCO2_PB_MH[tt], m_flow_sCO2_MH[tt],
h_cold_MH[tt],h_sCO2_hot_PB_MH[tt],T_cold_MH[tt],NTU_UC[tt],eta_HXopp[tt]]=PBHX.Ope
rating_HeatExchanger(
                PB_on[tt], m_flow_MH[tt], h_TES_hot[tt],h_sCO2_cold_PB_MH[tt-
1],h_MH_sCO2_hot_des, U_HX_design, Area_HX, p_high_blk)
            #if PB_on[tt]:
            #    print(m_flow_sCO2_MH[tt])
            #    print(Q_sCO2_PB_MH[tt])




            T_sCO2_hot_pb_check[tt]=Medium3.T_h(p_high_blk,h_sCO2_hot_PB_MH[tt])
            T_sCO2_cold_pb_check[tt]=Medium3.T_h(p_high_blk,h_sCO2_cold_PB_MH[tt-
1])
            T_source_hot_pb_check[tt]=Medium1.T_h(h_TES_hot[tt])
            T_source_cold_pb_check[tt]=Medium1.T_h(h_cold_MH[tt])

            Q_flow_MH[tt] = m_flow_MH[tt]*(h_TES_hot[tt]-h_cold_MH[tt])

            #print(m_flow_MH[tt])
            #print(h_cold_MH[tt])
            #print(HX_co[tt])

[HX_co[tt],Q_co_HX[tt]]=CS_coge.Control_Cogeneration(L_HT[tt]/100,L_min_HX,T_cold_s
et_REC_input,T_source_cold_pb_check[tt],Q_de,m_flow_MH[tt],PB_on[tt])

[T_coldtoTES_HX[tt],h_cold_MH[tt],Q_co_HX[tt]]=HX_coge.HX_cogeneration(HX_co[tt],ef
f_HX_coge,T_source_cold_pb_check[tt],Q_co_HX[tt],m_flow_MH[tt])
            #print(HX_co[tt])
            #print(m_flow_MH[tt])
            #print(h_cold_MH[tt])

            h_toTES_cold[tt] = h_cold_MH[tt]
            Q_flow_PB[tt]=Q_flow_MH[tt]
            W_loss_PB_pumps[tt]=k_loss_hot*(m_flow_MH[tt])
        if PB_R or PB_RR or PB_R_Inter or PB_RR_Inter:                  # Main
Heater + Reheater
            #Main Heater
            [Q_sCO2_PB_MH[tt], m_flow_sCO2_MH[tt], h_cold_MH[tt],
h_sCO2_hot_PB_MH[tt]]=PBHX.Operating_HeatExchanger(
                PB_on[tt], m_flow_MH[tt], h_TES_hot[tt], h_sCO2_cold_PB_MH[tt-1],
h_cold_control_MH, h_MH_sCO2_hot_des, eta_PBHX
            )
            Q_flow_MH[tt] = m_flow_MH[tt]*(h_TES_hot[tt]-h_cold_MH[tt])
            #Re-Heater
            [Q_sCO2_PB_RH[tt], m_flow_sCO2_RH[tt], h_cold_RH[tt],
h_sCO2_hot_PB_RH[tt]]=PBHX.Operating_HeatExchanger(
                PB_on[tt], m_flow_RH[tt], h_TES_hot[tt], h_sCO2_cold_PB_RH[tt-1],
h_cold_set_TES_RH, h_RH_sCO2_hot_des, eta_PBHX
            )
            Q_flow_RH[tt] = m_flow_RH[tt]*(h_TES_hot[tt]-h_cold_RH[tt])
            Q_flow_PB[tt]=Q_flow_MH[tt]+Q_flow_RH[tt]
            h_toTES_cold[tt]=MIX.Mixer(m_flow_MH[tt], h_cold_MH[tt], m_flow_RH[tt],
h_cold_RH[tt])
            W_loss_PB_pumps[tt]=k_loss_hot*(m_flow_MH[tt]+m_flow_RH[tt])
            print("1117 no")
        parasities_blk[tt]=W_loss_CT[tt-
1]+W_loss_HT[tt]+W_loss_PB_pumps[tt]+W_loss_HF[tt]
```

```
    #
________________________________________________________________________________
________________________________________________________________________________
_____
    #                                                                Power Block
Operation
    #
________________________________________________________________________________
________________________________________________________________________________
_____

        if PB_Recomp_Inter:
            [W_net_CSP[tt], h_sCO2_cold_PB_MH[tt], eff_PB[tt], Q_cond[tt],
Q_intercooler[tt], Q_HTR[tt], Q_LTR[tt], W_T1[tt], W_MC1[tt], W_MC2[tt], W_RC[tt],
cycle_temperatures]=PBRecompInter.sCO2_PB_Recompr_Intercooler(
                m_flow_sCO2_MH[tt], TIT, T_in_Compr, T_in_inter_Compr,
T_sCO2_9_des_K, SR, eta_HPT_des, eta_MC1_des, eta_MC2_des, eta_RC_des, eta_HTR,
eta_LTR, eta_gen, use_eta_net_blk, eta_net_blk, Tamb[tt], DT_pinch_cooler,
DT_pinch_intercooler, W_base_blk, parasities_blk[tt], DT_recuperator, f_prop_des
            )
            [T1, T2, T3, T4, T5, T6, T7, T8, T9, T9_prime, T9_second, T10] =
cycle_temperatures
            T_sCO2_1[tt]=GU.to_degC(T1)
            T_sCO2_2[tt]=GU.to_degC(T2)
            T_sCO2_3[tt]=GU.to_degC(T3)
            T_sCO2_4[tt]=GU.to_degC(T4)
            T_sCO2_5[tt]=GU.to_degC(T5)
            T_sCO2_6[tt]=GU.to_degC(T6)
            T_sCO2_7[tt]=GU.to_degC(T7)
            T_sCO2_8[tt]=GU.to_degC(T8)
            T_sCO2_9[tt]=GU.to_degC(T9)
            T_sCO2_9_prime[tt]=GU.to_degC(T9_prime)
            T_sCO2_9_second[tt]=GU.to_degC(T9_second)
            T_sCO2_10[tt]=GU.to_degC(T10)

        W_net_tot[tt] = min(W_net_CSP[tt] + W_net_PV[tt], P_max)
        DTime = time.time() - time0
        PB_time = PB_time + DTime

    #
________________________________________________________________________________
________________________________________________________________________________
_____
    #                                                                Hourly TES
- Cold Tank Operation
    #
________________________________________________________________________________
________________________________________________________________________________
_____
        time0 = time.time()
        [L_CT[tt], h_CT[tt], m_CT[tt], Q_losses_CT[tt],
W_loss_CT[tt]]=TES.Storage_Tank(
            Medium1, m_flow_toTES_cold[tt], h_toTES_cold[tt], m_CT[tt-1], h_CT[tt-
1], m_flow_TES_cold[tt], Tamb[tt], time_vec[tt-1], time_vec[tt], dt, CT_Design,
m_max
        )
        h_TES_cold[tt]=h_CT[tt]
        DTime = time.time() - time0
        TES_time = TES_time + DTime + Dtime1
    #
                                            Hourly Temperatures Calculations
________________________________________________________________________________
        time0 = time.time()
        #Molten Salt - Receiver
        state_output_REC = Medium1.setState_phX(Medium1.p_default, h_REC_hot[tt])
        state_input_REC = Medium1.setState_phX(Medium1.p_default, h_REC_cold[tt])
        T_REC_hot[tt] = GU.to_degC(Medium1.temperature(state_output_REC))
        T_REC_cold[tt] = GU.to_degC(Medium1.temperature(state_input_REC))
        #Electric Heater
        state_output_EH = Medium1.setState_phX(Medium1.p_default, h_EH_hot[tt])
        T_EH_hot[tt] = GU.to_degC(Medium1.temperature(state_output_EH))
        state_input_EH = Medium1.setState_phX(Medium1.p_default, h_CT[tt])
        T_EH_cold[tt] = GU.to_degC(Medium1.temperature(state_input_EH))
        # Molten Salt - TES
        state_toTES_hot = Medium1.setState_phX(Medium1.p_default, h_toTES_hot[tt])
        state_cold_TES = Medium1.setState_phX(Medium1.p_default, h_TES_cold[tt])
        T_TES_cold[tt] = GU.to_degC(Medium1.temperature(state_cold_TES))
        T_toTES_hot[tt] = GU.to_degC(Medium1.temperature(state_toTES_hot))
        #Molten Salt  - from TES to/from Power Block
        state_hot_TES = Medium1.setState_phX(Medium1.p_default, h_TES_hot[tt])
        state_toTES_cold = Medium1.setState_phX(Medium1.p_default,
h_toTES_cold[tt])
        state_toTES_MH_cold = Medium1.setState_phX(Medium1.p_default,
h_cold_MH[tt])
        T_TES_hot[tt] = GU.to_degC(Medium1.temperature(state_hot_TES))
        T_toTES_cold[tt] = GU.to_degC(Medium1.temperature(state_toTES_cold))
        T_toTES_MH_cold[tt] = GU.to_degC(Medium1.temperature(state_toTES_MH_cold))
        if PB_R or PB_RR or PB_R_Inter or PB_RR_Inter:
            state_toTES_RH_cold = Medium1.setState_phX(Medium1.p_default,
h_cold_RH[tt])
            T_toTES_RH_cold[tt] =
GU.to_degC(Medium1.temperature(state_toTES_RH_cold))
        #sCO2 - Power Block - Main Heater
        T_sCO2_hot_MH[tt] = GU.to_degC(T_high_h(h_sCO2_hot_PB_MH[tt]))
        T_sCO2_cold_MH[tt] = GU.to_degC(T_high_h(h_sCO2_cold_PB_MH[tt])) # non
capisco che è sta roba però è solo il plot
        #sCO2 - Power Block - Re-Heater
        if PB_R or PB_RR or PB_R_Inter or PB_RR_Inter:
            T_sCO2_hot_RH[tt] = GU.to_degC(T_int_h(h_sCO2_hot_PB_RH[tt]))
            T_sCO2_cold_RH[tt] = GU.to_degC(T_int_h(h_sCO2_cold_PB_RH[tt]))
        DTime = time.time() - time0
        Temperature_time = Temperature_time + DTime

    operation_time = time.time()
    print(u"\u2192 Operation of Heliostat Field (Duration: %s s)"
%(np.around(HF_time, decimals=1)))
    print(u"\u2192 Control of Receiver and EH  (Duration: %s s)"
%(np.around(ControlRecEH_time, decimals=1)))
    print(u"\u2192 Operation of Receiver and EH (Duration: %s s)"
%(np.around(RecEH_time, decimals=1)))
    print(u"\u2192 Operation of TES (Duration: %s s)" %(np.around(TES_time,
decimals=1)))
    print(u"\u2192 Operation of PB Control (Duration: %s s)"
%(np.around(PBControl_time, decimals=1)))
    print(u"\u2192 Operation of PB (Duration: %s s)" %(np.around(PB_time,
decimals=1)))
    print(u"\u2192 Evaluation of system temperatures (Duration: %s s)"
%(np.around(Temperature_time, decimals=1)))
```

```
    print(u"\u2192 Operation of the plant completed (Duration: %s s)"
%(np.around(operation_time - design_time, decimals=0)))

# ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Results
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
#
__________________________________________________________________________________
_________________________________________________________________
#                                                                   - Results -
# _______________________________________________________________   KPIs
Calculation  ______________________________________________________________
    time0 = time.time()
    AEY=KPI.sumEnergy(W_net_tot) #[GWh]
    AEY_CSP=KPI.sumEnergy(W_net_CSP) #[GWh]
    AEY_PV=KPI.sumEnergy(W_net_PV) #[GWh]
    CF=KPI.CapacityFactor(AEY, P_net) #[%]
    CAPEX_CSP=float(KPI.CapitalExpenditure(C_Cap_CSP, C_land_CSP,
f_contingency_CSP, f_EPC_CSP, f_decommissioning, f_Subs)) #[currency]
    CAPEX_PV=float(KPI.CapitalExpenditure(C_Cap_PV, C_land_PV, f_contingency_PV,
f_EPC_PV, f_decommissioning, f_Subs)) #[currency]
    CAPEX = CAPEX_CSP + CAPEX_PV
    OPEX=KPI.OperationExpenditure(C_year, c_OM_CSP, AEY_CSP) #[currency/year]
    LCOE=float(KPI.LevelizedCostofElectricity(CAPEX, OPEX, AEY, r_disc, r_i,
t_life)) #[currency/MWh]
    AF=KPI.AvailabilityFactor(W_net_tot)
    EH_UF=KPI.ElectricHeaterUtilizationFactor(W_heater_PV, P_name_EH) #[%]
    TES_PV_fraction=KPI.PVSharechargingStorage(Q_out_heater, Q_out_rec) #[%]
    c_block = KPI.PowerBlockSpecificCost(C_block, P_net) #[currency/kWe]
    ASCO2eq = KPI.AnnualSavingCO2eq(state_name, AEY) #[Mtons/year]
    QW_startup_CSP = KPI.sumEnergy(Q_wasted_startup) #[GWh]
    QW_defocus_CSP = KPI.sumEnergy(Q_wasted_defocus) #[GWh]
    QW_CSP = KPI.sumEnergy(Q_wasted_CSP) #[GWh]
    QSF_CSP = KPI.sumEnergy(Q_out_SF) #[GWh]
    EW_PV = KPI.sumEnergy(W_wasted_PV_tot) #[GWh]
    EEH_PV = KPI.sumEnergy(W_heater_PV) #[GWh]
    QW_tot = QW_CSP + EW_PV
    Q_prod_tot = QSF_CSP + EEH_PV
    TEW_share = KPI.ShareofEnergyWasted(Q_prod_tot, QW_tot) #[%]
    CSP_wasting_share = KPI.ShareofEnergyWasted(QSF_CSP, QW_CSP) #[%]
    PV_wasting_share = KPI.ShareofEnergyWasted(EEH_PV, EW_PV) #[%]
    CF_tot=KPI.CapacityFactor(AEY, P_name) #[%]
    CF_CSP=KPI.CapacityFactor(AEY_CSP, P_net) #[%]
    CF_PV=KPI.CapacityFactor(AEY_PV, P_AC) #[%]
    f_AEY_CSP = AEY_CSP/AEY*100 #[%]
    f_AEY_PV = AEY_PV/AEY*100 #[%]
    df_KPIs = { 'currency' : currency_name,
                'AEY' : AEY,
                'CF' : CF,
                'LCOE' : LCOE,
                'CAPEX' : CAPEX,
                'OPEX' : OPEX,
                'AF' : AF,
                'EH_UF' : EH_UF,
                'TES_PV_fraction' : TES_PV_fraction,
                'ASCO2eq' : ASCO2eq,
                'c_block' : c_block,
                'eta_blk_des' : eta_blk_des*100,
                'f_AEY_CSP' : f_AEY_CSP,
                'f_AEY_PV' : f_AEY_PV,
```

```
                'TEW_share' : TEW_share,
                'CSP_wasting_share' : CSP_wasting_share,
                'PV_wasting_share' : PV_wasting_share,
                'QW_startup_CSP' : QW_startup_CSP,
                'QW_defocus_CSP' : QW_defocus_CSP,
                'QSF_CSP' : QSF_CSP,
                'EW_PV' : EW_PV,
                'EEH_PV' : EEH_PV,
                'CF_tot' : CF_tot,
                'CF_CSP' : CF_CSP,
                'CF_PV' : CF_PV,
                }
    df1=pd.DataFrame(df_KPIs, index=[0])
    exporting_KPIs = time.time() - time0
    print(u"\u2192 Calculation and Exporting of the KPIs (Duration: %s s)"
%(np.around(exporting_KPIs, decimals=1)))

#
__________________________________________________________________________________
________________________________________________________________
#                                                                    - Results -
# ___________________________________________________________   Exporting Cost
Results   ____________________________________________________________
    time0 = time.time()
    df_Cost = { 'C_tower' : C_tower,
                'C_receiver' : C_receiver,
                'C_field' : C_field,
                'C_site_CSP' : C_site_CSP,
                'C_storage' : C_storage,
                'C_block' : C_block,
                'C_bop' : C_bop,
                'C_heater' : C_heater,
                'C_land_CSP' : C_land_CSP,
                'C_year_CSP' : C_year_CSP,
                'C_OM_CSP' : c_OM_CSP*AEY_CSP,
                'C_cap_CSP' : C_Cap_CSP,
                'C_modules_PV' : C_modules_PV,
                'C_BoS_PV' : C_BoS_PV,
                'C_site_PV' : C_site_PV,
                'C_inverter_PV' : C_inverter_PV,
                'C_land_PV' : C_land_PV,
                'C_year_PV' : C_year_PV,
                'C_cap_PV' : C_Cap_PV,
                'C_MC1' : C_MC1,
                'C_MC2' : C_MC2,
                'C_RC' : C_RC,
                'C_HPT' : C_HPT,
                'C_LPT' : C_LPT,
                'C_MH' : C_MH,
                'C_RH' : C_RH,
                'C_HTR' : C_HTR,
                'C_LTR' : C_LTR,
                'C_cooler' : C_cooler,
                'C_intercooler' : C_intercooler,
                'C_piping_PB' : C_piping_PB,
                'C_generator' : C_generator,
                'C_gearbox' : C_gearbox,
                'C_HX_part': C_HX_part
                #'C_cogeneration': C_cogeneration
    }
    df2=pd.DataFrame(df_Cost, index=[0])
    exporting_Cost = time.time() - time0
    print(u"\u2192 Exporting of the Costs (Duration: %s s)"
%(np.around(exporting_Cost, decimals=1)))
#
__________________________________________________________________________________
_________________________________________________________________
#                                                                  - Results -
# ________________________________________________________   Exporting Solar Field
Results  _________________________________________________________
    time0 = time.time()
    nptsx, nptsy = 400, 100
    x_in=np.linspace(0, 360, nptsx)
    y_in=np.linspace(0, 90, nptsy)
    xg, yg = np.meshgrid(x_in, y_in)
    zg = SF_Efficiency(xg, yg)*100
    SF_Eff=np.matrix(zg)
    Map_Helio=np.matrix(Map_Helio)
    df3=pd.DataFrame(SF_Eff)
    df4=pd.DataFrame(Map_Helio.transpose())
    exporting_SF = time.time() - time0
    print(u"\u2192 Exporting of the SF Design (Duration: %s s)"
%(np.around(exporting_SF, decimals=1)))
#
__________________________________________________________________________________
_________________________________________________________________
#                                                                  - Results -
# ___________________________________________________________   Exporting Design
Results  ___________________________________________________________
    time0 = time.time()
    if PB_Recomp_Inter:
        df_Design = {   'Q_field_des' : R_des,
                        'A_SF' : A_SF,
                        'A_land_CSP_tot' : A_land_CSP_tot,
                        'eff_SF_des' : eff_SF_des,
                        'n_heliostat' : n_heliostat,
                        'SM' : SM,
                        'H_tower' : H_tower,
                        'Q_rec_des' : Q_rec_des,
                        'H_rec' : H_rec,
                        'D_rec' : D_rec,
                        'rec_eff_design' : eta_rec_des,
                        'rec_eff_th_design' : eta_rec_th_des,
                        #'A_receiver' : A_receiver,
                        'm_flow_rec' : m_flow_rec_des,
                        'T_hot_set_REC/EH' : T_hot_set_REC,
                        'T_cold_set_REC/EH' : T_cold_set_REC,
                        'Q_name_EH' : Q_name_EH,
                        'EH_eff_design' : eta_heater_design,
                        'm_flow_EH_des' : m_flow_EH_des,
                        'P_name_EH' : P_name_EH,
                        'EH_W_min' : W_heater_min,
                        't_storage' : t_storage,
                        'H_storage' : H_storage,
                        'D_storage' : D_storage,
                        'A_surf_TES' : A_surf_TES,
                        'm_max' : m_max,
                        'Q_flow_des' : Q_flow_des,
                        'Q_Cooler_des' : Q_Cooler_des,
                        'eta_blk_des' : eta_blk_des,
                        'm_flow_sCO2_des' : m_flow_sCO2_des,
                        'SR' : SR,
                        'm_flow_sCO2_MC' : m_flow_sCO2_MC,
                        'm_flow_sCO2_RC' : m_flow_sCO2_RC,
                        'W_MC1_des' : W_MC1_des,
                        'eta_MC1_des' : eta_MC1_des,
                        'W_MC2_des' : W_MC2_des,
                        'eta_MC2_des' : eta_MC2_des,
                        'W_RC_des' : W_RC_des,
                        'eta_RC_des' : eta_RC_des,
                        'Q_HTR_des' : Q_HTR_des,
                        'Q_LTR_des' : Q_LTR_des,
                        'W_HPT_des' : W_HPT_des,
                        'eta_HPT_des' : eta_HPT_des,
                        'TIT' : TIT,
                        'T_sCO2_1_des' : T_sCO2_1_des,
                        'T_sCO2_2_des' : T_sCO2_2_des,
                        'T_sCO2_3_des' : T_sCO2_3_des,
                        'T_sCO2_4_des' : T_sCO2_4_des,
                        'T_sCO2_5_des' : T_sCO2_5_des,
                        'T_sCO2_6_des' : T_sCO2_6_des,
                        'T_sCO2_7_des' : T_sCO2_7_des,
                        'T_sCO2_8_des' : T_sCO2_8_des,
                        'T_sCO2_9_prime_des' : T_sCO2_9_prime_des,
                        'T_sCO2_9_second_des' : T_sCO2_9_second_des,
                        'T_sCO2_9_des' : T_sCO2_9_des,
                        'T_sCO2_10_des' : T_sCO2_10_des,
                        'UA_MH_des' : UA_MH_des,
                        'UA_HTR_des' : UA_HTR_des,
                        'UA_LTR_des' : UA_LTR_des,
                        'UA_cooler_des' : UA_cooler_des,
                        'UA_intercooler_des' : UA_intercooler_des,
                        'P_gross' : P_gross,
                        'P_net' : P_net,
                        'PV_P_AC' : P_AC,
                        'PV_P_DC' : P_DC,
                        'PV_r_DCAC' : r_DCAC,
                        'A_PV_field' : A_PV_field,
                        'A_land_PV_tot' : A_land_PV_tot,
                        'PV_GCR' : GCR,
                        'PV_rows' : PV_rows,
                        'PV_azimuth' : azimuth,
                        'eta_inv' : eta_inv_input,
                        'P_max' : P_max,
                        'Area_HX':Area_HX,
                        'eta_de_HX':eta_de_HX

                    }
    df5=pd.DataFrame(df_Design, index=[0])
    exporting_design = time.time() - time0
    print(u"\u2192 Exporting of the System Design (Duration: %s s)"
%(np.around(exporting_design, decimals=1)))

#
____________________________________________________________________________________
__________________________________________________________________
#                                                                   - Results -
# __________________________________________________________  Exporting Operating
Results  ___________________________________________________________
    time0 = time.time()
```

```
if PB_Recomp_Inter:
    df_Operation = {'DNI' : DNI,
                    'defocus' : defocus,
                    'eta_opt_SF' : eta_opt_SF,
                    'Q_in_SF' : Q_in_SF,
                    'Q_out_SF' : Q_out_SF,
                    'eta_rec' : eta_rec,
                    'eta_th_rec' : eta_th_rec,
                    'Q_raw_SF' : Q_raw_SF,
                    'Q_wasted_CSP': Q_wasted_CSP,
                    'Q_wasted_defocus' : Q_wasted_defocus,
                    'Q_wasted_startup' : Q_wasted_startup,
                    'W_loss_HF' : W_loss_HF,
                    'SF_on' : SF_on,
                    'Wspd_Ht' : Wspd_Ht,
                    'Q_flow_defocus' : Q_flow_defocus,
                    'Q_rec' : Q_out_rec,
                    'T_REC_hot' : T_REC_hot,
                    'T_REC_cold' : T_REC_cold,
                    'm_flow_rec' : m_flow_rec,
                    'waste_extra_PV' : waste_extra_PV,
                    'W_heater_PV_raw' : W_heater_PV_raw,
                    'W_heater_PV' : W_heater_PV,
                    'W_wasted_PV_defocus' : W_wasted_PV_defocus,
                    'W_wasted_PV_plus' : W_wasted_PV_plus,
                    'W_wasted_PV' : W_wasted_PV_tot,
                    'Q_missing_defocus' : Q_missing_defocus,
                    'Q_out_heater' : Q_out_heater,
                    'eta_heater' : eta_heater,
                    'Q_loss_heater' : Q_loss_heater,
                    'EH_on' : EH_on,
                    'm_flow_heater' : m_flow_heater,
                    'T_EH_hot' : T_EH_hot,
                    'T_EH_cold' : T_EH_cold,
                    'm_flow_toTES_hot' : m_flow_toTES_hot,
                    'T_toTES_hot' : T_toTES_hot,
                    'T_TES_hot' : T_TES_hot,
                    'm_flow_TES_hot' : m_flow_TES_hot,
                    'm_flow_toTES_cold' : m_flow_toTES_cold,
                    'T_toTES_cold' : T_toTES_cold,
                    'T_TES_cold' : T_TES_cold,
                    'm_flow_TES_cold' : m_flow_TES_cold,
                    'cold_tank_ready' : cold_tank_ready,
                    'L_HT' : L_HT,
                    'L_CT' : L_CT,
                    'Q_losses_HT' : Q_losses_HT,
                    'W_loss_HT' : W_loss_HT,
                    'Q_losses_CT' : Q_losses_CT,
                    'W_loss_CT' : W_loss_CT,
                    'Q_flow_PB' : Q_flow_PB,
                    'm_flow_PB' : m_flow_PB,
                    'Q_flow_MH' : Q_flow_MH,
                    'm_flow_MH' : m_flow_MH,
                    'T_toTES_MH_cold' : T_toTES_MH_cold,
                    'Q_sCO2_PB_MH' : Q_sCO2_PB_MH,
                    'm_flow_sCO2_MH' : m_flow_sCO2_MH,
                    'T_sCO2_hot_MH' : T_sCO2_hot_MH,
                    'T_sCO2_cold_MH' : T_sCO2_cold_MH,
                                 'T_sCO2_1' : T_sCO2_1,
                                 'T_sCO2_2' : T_sCO2_2,
                                 'T_sCO2_3' : T_sCO2_3,
                                 'T_sCO2_4' : T_sCO2_4,
                                 'T_sCO2_5' : T_sCO2_5,
                                 'T_sCO2_6' : T_sCO2_6,
                                 'T_sCO2_7' : T_sCO2_7,
                                 'T_sCO2_8' : T_sCO2_8,
                                 'T_sCO2_9' : T_sCO2_9,
                                 'T_sCO2_9_prime' : T_sCO2_9_prime,
                                 'T_sCO2_9_second' : T_sCO2_9_second,
                                 'T_sCO2_10' : T_sCO2_10,
                                 'W_loss_PB_pumps' : W_loss_PB_pumps,
                                 'parasities': parasities_blk,
                                 'Q_cond' : Q_cond,
                                 'Q_intercooler' : Q_intercooler,
                                 'Q_HTR' : Q_HTR,
                                 'Q_LTR' : Q_LTR,
                                 'W_T1' : W_T1,
                                 'W_MC1' : W_MC1,
                                 'W_MC2' : W_MC2,
                                 'W_RC' : W_RC,
                                 'PB_on' : PB_on,
                                 'F_prod' : F_prod,
                                 'W_net_CSP' : W_net_CSP,
                                 'W_net_PV' : W_net_PV,
                                 'W_net_tot' : W_net_tot,
                                 'PD':PD,
                                 'eff_PB' : eff_PB,
                                 'T_cold_MH':T_cold_MH,
                                 #'T_source_hot':T_source_hot,
                                 'NTU_UC':NTU_UC,
                                 'eta_HXopp':eta_HXopp,
                                 'T_sCO_hot':T_sCO2_hot_pb_check,
                                 'T_sCO_cold':T_sCO2_cold_pb_check,
                                 'T_source_hot':T_source_hot_pb_check,
                                 'T_source_cold':T_source_cold_pb_check,
                                 'HX_co': HX_co,
                                 'Q_co_HX': Q_co_HX,
                                 'T_coldtoTES': T_coldtoTES_HX
                                 }

    df6=pd.DataFrame(df_Operation)
    exporting_design = time.time() - time0
    print(u"\u2192 Exporting of the System Design (Duration: %s s)"
%(np.around(exporting_design, decimals=1)))

#
______________________________________________________________________________________
_________________________________________________________________
#                                                                        - Results -
# _______________________________________________________  Exporting Summary of the
Results   ________________________________________________________
    summary_results = pd.concat([df1, df2, df5], axis=1)
    Outputs_max = np.array([LCOE, CAPEX, CF, AEY, AF, c_block, ASCO2eq, TEW_share,
f_AEY_CSP, CF_CSP, CF_PV, CAPEX_CSP, CAPEX_PV, EH_UF, TES_PV_fraction])
    ObjectiveFunction_index = np.array([LCOE_OF, CAPEX_OF, CF_OF, AEY_OF, AF_OF,
c_block_OF, ASCO2eq_OF, TEW_share_OF, f_AEY_CSP_OF, CF_CSP_OF, CF_PV_OF,
CAPEX_CSP_OF, CAPEX_PV_OF, EH_UF_OF, TES_PV_fraction_OF])
    index_outputs = np.where(ObjectiveFunction_index == True)
    Outputs = Outputs_max[index_outputs]
    path_summary_results = './Outputs/%s/Source/%s' %(Model_Name,
identification_folder_summary)
    Path(path_summary_results).mkdir(parents=True, exist_ok=True)
    filename_summary = '%s/SummaryResults%s.xlsx' %(path_summary_results,
identification_summary)
    EE.export_to_excel(filename_summary, summary_results, erase_old_summary)
    if save_all_files:
        path_results = './Outputs/%s/Source/%s' %(Model_Name,
identification_folder_results)
        path_KPIs = '%s/KPIs' %(path_results)
        path_Cost = '%s/Cost' %(path_results)
        path_SF1 = '%s/SF1' %(path_results)
        path_SF2 = '%s/SF2' %(path_results)
        path_Design = '%s/Design' %(path_results)
        path_Operation = '%s/Operation' %(path_results)
        Path(path_results).mkdir(parents=True, exist_ok=True)
        Path(path_KPIs).mkdir(parents=True, exist_ok=True)
        Path(path_Cost).mkdir(parents=True, exist_ok=True)
        Path(path_SF1).mkdir(parents=True, exist_ok=True)
        Path(path_SF2).mkdir(parents=True, exist_ok=True)
        Path(path_Design).mkdir(parents=True, exist_ok=True)
        Path(path_Operation).mkdir(parents=True, exist_ok=True)
        df1.to_excel('%s/KPIs%s.xlsx' %(path_KPIs, identification_results))
        df2.to_excel('%s/Cost%s.xlsx' %(path_Cost, identification_results))
        df3.to_excel('%s/SF1%s.xlsx' %(path_SF1, identification_results))
        df4.to_excel('%s/SF2%s.xlsx' %(path_SF2, identification_results))
        df5.to_excel('%s/Design%s.xlsx' %(path_Design, identification_results))
        df6.to_excel('%s/Operation%s.xlsx' %(path_Operation,
identification_results))
    print(u"\u2192 Simulation completed (Duration: %s s)" %(np.around(time.time() -
start_time, decimals=0)))


    if Flag==1:

        wb = openpyxl.load_workbook("History.xlsx")
        ws = wb['Sheet1']
        ws.cell(row = number_sim, column = 1, value = LCOE) # Writes the content of
totalcost in A1
        ws.cell(row = number_sim, column = 2, value = CAPEX)
        ws.cell(row = number_sim, column = 3, value = Co_i)
        ws.cell(row = number_sim, column = 4, value = Co_j)
        ws.cell(row = number_sim, column = 5, value = Co_k)
        ws.cell(row = number_sim, column = 6, value = Co_l)
        ws.cell(row = number_sim, column = 7, value = Co_m)

        ws.cell(row = number_sim, column = 8, value = P_max)
        ws.cell(row = number_sim, column = 9, value = t_storage)
        ws.cell(row = number_sim, column = 10, value = SM)
        ws.cell(row = number_sim, column = 11, value = P_AC)
        ws.cell(row = number_sim, column = 12, value = Q_de)
        ws.cell(row = number_sim, column = 13, value = number_sim)

        wb.save("History.xlsx")
        print('Simulation'+ ' ('+ str(Co_i)+','+str(Co_j)+','+str(Co_k)+','+
str(Co_l)+ ')'+','+ str(Co_m)+' #'+ str(number_sim))

    return Outputs
```

8.2 Receiver model

```
from attr import define
from sympy import Q
import Models.Media.Particle as Medium1
import math as MA
import numpy as np


def Design_Receiver(
    Q_rec_des,                   # [W]       - Nominal output thermal power
    #w_curtain,                  # [m]       - Width of the receiver
    #ar_rec,                     # [-]       - Receiver aspect ratio (height over 
width)
    #T_amb_des,                  # [K]       - Ambient temperature at the design 
point
    T_max_particle_des,          # [K]       - Outlet HTF temperature at the design 
point
    T_min_particle_des,          # [K]       - Inlet HTF temperature at the design 
point
    #d_particle,                 # [m]       - Particle diameter
    #vf_par,                     # [%]       - Volume fraction of particles
    #ab_rec,                     # [-]       - Receiver coating absorptance
    #em_rec,                     # [-]       - Receiver coating emissivity
    eta_rec_des
    ):

    eta_rec_des=0.25             #the design but be calculated on the average 
efficiency
    Rec_design=1
    eta_rec_th_des=eta_rec_des
    Q_loss_rec_des=Q_rec_des*(1- eta_rec_des)
    Q_in_rec_des=Q_rec_des
    h_cold_in=Medium1.h_T(T_min_particle_des)
    h_hot_out=Medium1.h_T(T_max_particle_des)

    m_flow_des=max(1e-3,((Q_rec_des*eta_rec_des)/(h_hot_out-h_cold_in)))

    return (    Rec_design,         # [-]       - Complete geometrical design of 
the receiver
                eta_rec_des,        # [-]       - Design total receiver efficiecny
                eta_rec_th_des,     # [-]       - Design thermal receiver 
efficiecny
                Q_loss_rec_des,     # [W]       - Design receiver losses
                Q_in_rec_des,       # [w]       - Design receiver input power
                m_flow_des)         # [kg/s]    - Design receiver mass flow rate

def Operating_Receiver(
    in_DNI,                         #[W/m^2]    - Direct normal irradice for the 
efficiency
#    Medium,                         # [-]       - HTF utilized in the receiver
#    TLREC,                          # [-]       - Thermal Losses Receiver 
Calculation - Model
    Q_in_rec,                       # [W]       - Incoming receiver power
    h_in,                           # [J/kg]    - Input specific enthalpy
    m_flow,                         # [kg/s]    - Receiver mass flow rate
    SF_on,                          # [-]       - Boolean to indicate if solar 
field is on
#    T_amb,                          # [K]       - Ambient temperature
#     u_wind,                         # [m/s]      - Wind speed
#     T_out_des,                      # [K]        - Set point output temperature
#     Rec_design,                     # [-]        - Receiver geometrical design
#     rec_absorptance,                # [-]        - Receiver absorptance
#     rec_emissivity,                 # [-]        - Receiver emissivity
    input_eff,                      # [-]        - Boolean to indicate if a fixed
efficiency shoudl be used
    eta_rec_input                   # [-]        - Input receiver efficiency (if
input_eff is True)
    ):

    if SF_on:
        if input_eff:
            eta_rec = eta_rec_input
            Q_loss_rec = Q_in_rec*(1-eta_rec)
        else:
            delta_eta=eta_rec_input-0.5 # 0.5 is the reference receiver efficency

            if in_DNI < 205:
                eta_rec=0.2 + delta_eta
            if in_DNI > 480:
                eta_rec=0.5 + delta_eta
            if in_DNI <= 480 and in_DNI >= 205:
                coef_C1=0.0011
                coef_C0=-0.0230
                eta_rec=coef_C1*in_DNI+coef_C0+delta_eta
            Q_loss_rec = Q_in_rec*(1-eta_rec)

        #Efficiency

        #  eta_rec = max(0,min(1,((Q_in_rec - Q_loss_rec)/(1e-3+Q_in_rec)))) in my
case does not have any meaning


        eta_th_rec=eta_rec
    else:
        # Advection Losses
        Q_adv_loss = 0

        #Radiative Losses
        Q_rad_loss = 0

        # Reflective Losses
        Q_ref_loss = 0

        # Total Losses
        Q_loss_th_rec = Q_rad_loss + Q_adv_loss
        Q_loss_rec = Q_loss_th_rec + Q_ref_loss
        eta_th_rec = 0
        eta_rec = 0
    Q_out_rec=eta_rec*Q_in_rec
    h_out=h_in+Q_out_rec/max(1e-6,m_flow)

    return (    Q_out_rec,          # [W]        - Output receiver power
                h_out,              # [J/kg]     - Output specific enthalpy
                eta_rec,            # [-]        - Receiver total efficiency
                eta_th_rec,         # [-]        - Receiver thermal efficiency
                Q_loss_rec)         # [-]        - Receiver thermal losses
```

8.3 Particle heat exchanger

```
from ast import Del
from cmath import log
import numpy as np
import Models.Media.Particle as Medium1
import Models.Media.sCO2 as Medium3
import Models.Utilities.finterpolation as linear

def Operating_HeatExchanger(
    PB_on,                    # [-]       - Boolean to indicate if the power block is
on
    m_flow_hot_source,        # [kg/s]    - Mass flow rate hot-side
    h_source_hot,             # [J/kg]    - Specific enthalpy hot source
    h_sCO2_cold,              # [J/kg]    - Specific enthalpy cold sCO2
    h_sCO2_hot_des,           # [J/kg]    - Set point specific enthalpy hot sCO2
    U_HX_design,               # [W/m^2K]   - General heat transfer of the HX
    Area,
    pressure
    ):
    NTU_cc=2 #test

    x=[1.2, 1.5, 1.6, 1.7, 1.8, 1.9, 2, 2.1, 2.5, 3, 3.1, 3.2, 3.3]
    y=[0.3, 0.4, 0.5, 0.56, 0.65, 0.7, 0.75, 0.8, 0.81, 0.85, 0.9, 0.91, 0.95]
    T_sco2_cold=Medium3.T_h(pressure,h_sCO2_cold)
    T_source_hot=Medium1.T_h(h_source_hot)
    if PB_on:

        Q_in_HX=((T_source_hot-T_sco2_cold)*Medium1.cp_T(1)*m_flow_hot_source)
        NTU_cc=(U_HX_design*Area)/(m_flow_hot_source*Medium1.cp_T(1))
        if NTU_cc <= 3.5:
            eta_HX=linear.linear(y,x,NTU_cc)
        else:
            eta_HX=0.97

        Q_sCO2_PB = eta_HX*Q_in_HX                  # Q_max
        h_source_cold = h_source_hot - Q_sCO2_PB/max(1e-3,m_flow_hot_source)
        m_flow_sCO2 = Q_sCO2_PB/(h_sCO2_hot_des-h_sCO2_cold)
        h_sCO2_hot = h_sCO2_cold + Q_sCO2_PB/max(1e-3,m_flow_sCO2)



    else:
        h_source_cold = h_source_hot
        h_sCO2_hot = h_sCO2_cold
        Q_sCO2_PB = 0
        m_flow_sCO2 = 0
        NTU_cc=2 #test
        eta_HX=0.8
    T_cold_MH=Medium1.T_h(h_source_cold)
    return (    Q_sCO2_PB,                   # [W]        - Thermal power to the
sCO2 power block
                m_flow_sCO2,                 # [kg/s]     - sCO2 power block mass
flow rate
                h_source_cold,               # [J/kg]     - Specific enthalpy output
heat source #per me questo è l'output in design
                h_sCO2_hot,                  # [J/kg]     - Specific enthalpy output
hot sCO2
                T_cold_MH,
                NTU_cc,
                eta_HX
                )

def Desing_HeatExchanger(
    T_source_hot,           # [K]
    T_source_cold,          # [K]
    T_sco2_hot,             # [K]
    T_sco2_cold,            # [K]
    h_source_hot_des,       # [J/kg]    - Specific enthalpy hot source desing
    h_sco2_cold_des,        # [J/kg]    - Specific enthalpy cold sCO2 desing
    h_source_cold_des,      # [J/kg]    - Set point specific enthalpy cold source
design
    h_sco2_hot_des,         # [J/kg]    - Set point specific enthalpy hot sCO2
design
    Q_flow_des,             # [W]       - Heat in the PB at the design point
    U_HX_design,            # [W/m^2K]  - General heat transfer of the HX
    pressure_design,        # [Pa]      - Pressure
    ):

    # parametri dello scambiatore di calore
    x=[1.2, 1.5, 1.6, 1.7, 1.8, 1.9, 2, 2.1, 2.5, 3, 3.1, 3.2]
    y=[0.3, 0.4, 0.5, 0.56, 0.65, 0.7, 0.75, 0.8, 0.81, 0.85, 0.9, 0.91]

    del_Tlog=((T_source_hot-T_sco2_hot)-(T_source_cold-
T_sco2_cold))/(np.log((T_source_hot-T_sco2_hot)/(T_source_cold-T_sco2_cold)))
    m_flow_p_design=Q_flow_des/(h_source_hot_des-h_source_cold_des)
    m_flow_sco2_design=Q_flow_des/(h_sco2_hot_des-h_sco2_cold_des)
    eta_de=Q_flow_des/((T_source_hot-
T_sco2_cold)*(min(Medium1.cp_T(1)*m_flow_p_design,m_flow_sco2_design*Medium3.cp_T(p
ressure_design,T_sco2_hot),m_flow_sco2_design*Medium3.cp_T(pressure_design,T_sco2_c
old))))
    NTU_fu=linear.linear(x,y,eta_de)
    #Area_U=Q_flow_des/(U_HX_design*del_Tlog)

Area=NTU_fu*(min(Medium1.cp_T(1)*m_flow_p_design,m_flow_sco2_design*Medium3.cp_T(pr
essure_design,T_sco2_hot),m_flow_sco2_design*Medium3.cp_T(pressure_design,T_sco2_co
ld)))/U_HX_design

#NTU_Uc=(U_HX_design*Area)/(min(Medium1.cp_T(1)*m_flow_p_design,m_flow_sco2_design*
Medium2.cp_T(pressure_design,T_sco2_hot),m_flow_sco2_design*Medium2.cp_T(pressure_d
esign,T_sco2_cold)))
    #Fact_c= NTU_fu/NTU_Uc
    #Delta_1=del_Tlog/eta_de


    return (
        NTU_fu,
        Area,
        m_flow_sco2_design,
        m_flow_p_design,
        eta_de
    )
```

8.4 Thermal storage (TES)

```
import Models.Utilities.GeneralUtilities as GU
import Models.Utilities.U_Storage.ODEs_Storage as ODE
from scipy.integrate import odeint
import numpy as np
import math as MA

def Storage_Tank(
    Medium,                         # [-]       - HTF utilized in the TES
    m_flow_in,                      # [kg/s]    - Inlet mass flow rate in tank
    h_in,                           # [J/kg]    - Inlet specific enthalpy
    m_prev,                         # [kg]      - Previous value of estimated mass
in tank
    h_prev,                         # [J/kg]    - Previous value of average
specific enthalpy
    m_flow_out,                     # [kg/s]    - Outlet mass flow rate from tank
    T_amb,                          # [K]       - Ambient temperature
    t_in,                           # [s]       - Initial time step
    t_out,                          # [s]       - Final time step
    dt,                             # [s]       - Duration time step
    Tank_Design,                    # [-]       - Tank design parameters (Tank
volume, diameter, height, thermal losses coefficiecnt, auxiliary heater power, set
point temperature HTF, efficiency auxiliary heater)
    m_max_des                       # [kg]      - Maximum mass - design
    ):

    V_t=Tank_Design[0] # [m^3]
    D=Tank_Design[1] # [m]
    H=Tank_Design[2] # [m]
    alpha=Tank_Design[3]
    W_max=Tank_Design[4]
    T_set=Tank_Design[5]
    e_ht=Tank_Design[6]

    #Calculated Parameters
    state_medium=Medium.setState_phX(Medium.p_default, h_prev)
    t_storage = np.linspace(t_in, t_out, num=dt)
    m_time = odeint(ODE.mass_balance, m_prev, t_storage, args=(m_flow_in,
m_flow_out)) #[kg]
    m_calc = m_time[-1]
    dmdt = ODE.mass_balance(0, 0, m_flow_in, m_flow_out)
    max_mass = m_max_des*0.99+1e-5
    min_mass = m_max_des*0.01-1e-5


    if m_calc > min_mass and m_calc < max_mass:
        m = m_calc
    elif m_calc >= max_mass:
        m = max_mass
    else:
        m = min_mass

    L=100*m/m_max_des # [%]
    rho = Medium.density(state_medium) # [kg/m^3]
    V=m/rho
    A=MA.pi*D*H*(V/V_t) + MA.pi*D**2/4 # [m^2]

    T=Medium.temperature(state_medium)
    Q_losses=A*alpha*(T-T_amb)
    if T<T_set:
        W_net = min(Q_losses, W_max)
    else:
        W_net = 0

    if m<1:
        h_time = h_prev
        h = h_prev
    else:
        h_time = odeint(ODE.energy_balance, h_prev, t_storage, args=(m_flow_in,
m_flow_out, h_in, m, dmdt, W_net, Q_losses))
        h = h_time[-1]

    W_loss=W_net/e_ht

    return (    L,                  # [%]        - Tank level
                h,                  # [J/kg]     - Averge spefic enthalpy
                m,                  # [kg]       - Mass of HTF stored in tank
                Q_losses,           # [W]        - Thermal losses
                W_loss)             # [W]        - Parasitic losses due to heat
tracing
```

8.5 Cogeneration

```
import Models.Media.Particle as Medium1

def HX_cogeneration (
    HX_c,                     # boolean HX on/off
    eff_HX_co,                 # [-]   Effectivness of HX
    T_cold_MH,                 # [K]   Temperature particle after heat exchage
    Q_out,                         # [Wth] Thermal power
    m_particle                # [kg/s]Mass flow rate of the particle
    #T_min,                     # [K]   Minimun Temperature in the TES
):
    if HX_c==1:

        if eff_HX_co==1:
            T_out=T_cold_MH-Q_out/(max((m_particle*Medium1.cp_T(1)),1e-3))
        else:
            print("NoHXCO")
    else:
        Q_out=0
        T_out=T_cold_MH
    return (
        T_out,
        Medium1.h_T(T_out),
        Q_out
    )
```

8.6 Genetic algorithm

```
from SystemModels.MoltenSaltsCO2 import MoltenSaltsCO2
import numpy as np
import pandas as pd
import Generation as GE
import openpyxl
import random

# ------------------------------       Design Variables
P_max = 100e6                                              # [W]        - Maximum
Electric Power that can be injected to the grid
P_gross = 100e6/0.9                                        # [W]        - Power Cycle
Gross Output
Q_de=0                                                     # [W]        - Therma power
output
P_AC = 200e6                                               # [W]        - AC nameplate
system capacity
P_name_EH = P_AC - P_max                                   # [W]        - Electric
heater nominal capacity
SM_value = 2                                               # [-]        - Solar
multiple
TES_value = 10                                             # [h]        - Thermal
energy storage capacity
H_tower_input = 190                                        # [m]        - Input of the
tower height
Optimize_SF = False                                        # [-]        - Boolean to
run optimization of the solar field
Reheat = False                                             # [-]        - Boolean to
decide to include Reheating in the sCO2 power block
Recompression = True                                       # [-]        - Boolean to
decide to include Recompression in the sCO2 power block
Intercooling = True                                        # [-]        - Boolean to
decide to include Intercooling in the sCO2 power block

P_max=np.array([50e6, 75e6, 100e6, 110e6, 120e6, 130e6, 140e6, 150e6, 160e6, 170e6,
180e6, 190e6, 200e6 ])
P_gross=np.multiply(P_max,1.11111)
#TES_value=np.array([8, 10, 12])
TES_value=np.array([6, 7, 8, 9, 10, 11, 12])
SM_value=np.array([ 1.2, 1.3, 1.4, 1.5, 1.6, 1.7, 1.8, 1.9, 2, 2.1, 2.2, 2.3, 2.4,
2.5])
#SM_value=np.array([1.5, 2, 2.5])
r_P_Ac=np.array([1.2, 1.3, 1.4, 1.5, 1.6, 1.7, 1.8, 1.9, 2, 2.1, 2.2, 2.3, 2.4,
2.5])

Flag=1



God_S=False
First_time=False
Size_population=10
Min_rank=5
CC=np.empty([2,Size_population])
S_start=1
number_row=2

Size_population=10
```

```
Number_generation=3
Max_size=np.array([len(P_max),len(TES_value),len(SM_value), len(r_P_Ac)])
Max_gen=len(Max_size)
Index=np.array([1, 5])

for k_gen in range(0,Number_generation):
    if God_S:
        Generation=1
        number_sim=1#
        Generation_data=GE.generation_start(Max_size,Max_gen,Size_population)
        All_data=np.empty([Size_population,12])
        All_data[:,Index[0]:Index[1]]=Generation_data

        God_S=False
    else:
        path="History.xlsx"
        df=pd.read_excel(path)
        #dt = pd.DataFrame(df, columns=
['LCOE','i','j','k','l','m','P_CSP','Storage Time','Solar Time','PV','Number',
'Gen','Number Give', 'Gen Give' ])
        dt = pd.DataFrame(df, columns=
['LCOE','i','j','k','l','m','Number','P_CSP','Storage Time','Solar
multipe','PV','Heat'])

        Index_number=6 #keep track
        All_data=np.array(dt)
        number_row=int(max(All_data[:,Index_number]))+1
        A=np.array(dt)
        A = A[A[:, 0].argsort()]    #rank the solutions
        #In=len(A)-Size_population
        #St=In+Size_population
        In=0
        St=Size_population-1

        Generation_data_raw=A[In:St,:]  #extraction
        print(Generation_data_raw[:,Index[0]:Index[1]])
        print('cosa entra')

[Generation_data,CC,ty]=GE.reproduction(Generation_data_raw,Max_size,Size_populatio
n,Index,Min_rank,Max_gen)
        print(Generation_data)
        print(CC)
        #print('end reprod')

    for i_spec in range(0,Size_population):

        Co_i_c=int(Generation_data[i_spec,0])
        Co_j_c=int(Generation_data[i_spec,1])
        Co_k_c=int(Generation_data[i_spec,2])
        Co_l_c=int(Generation_data[i_spec,3])
        #Co_m_c=int(Generation_data[number_row,4])
        Co_m_c=2
        print(Generation_data)
        # ------------------------------       Handling of the Outputs
        identification_folder_summary =
'SingleSimulation'+str(Co_i_c)+str(Co_j_c)+str(Co_k_c)+str(Co_l_c)+str(Co_m_c)
        identification_summary =''

        identification_folder_results =
'SingleSimulation'+str(Co_i_c)+str(Co_j_c)+str(Co_k_c)+str(Co_l_c)+str(Co_m_c)
```

```
        identification_results =''

        save_all_files = True
        erase_old_summary = True


[check_v,Results]=GE.clone(All_data,Generation_data[i_spec,:],Index,First_time)

        if check_v:
            #export
            print('Y')
            LCOE_c=float(Results[0])
            CAPEX_c=float(Results[0])
            Co_i_c=int(Results[1])
            Co_j_c=int(Results[2])
            Co_k_c=int(Results[3])
            Co_l_c=int(Results[4])
            Co_m_c=int(Results[5])
            P_CSP_c=float(Results[7])
            t_sto_c=float(Results[8])
            SM_c=float(Results[9])
            r_c=float(Results[10])
            Q_de_c=float(Results[11])

#[_]=EX.export_data(number_row,LCOE_c,CAPEX_c,Co_i_c,Co_j_c,Co_k_c,Co_l_c,Co_m_c,P_
CSP_c,t_sto_c,SM_c,r_c,Q_de_c)


            wb = openpyxl.load_workbook("History.xlsx")
            ws = wb['Sheet1']
            ws.cell(row = number_row, column = 1, value = LCOE_c) # Writes the
content of totalcost in A1
            ws.cell(row = number_row, column = 2, value = CAPEX_c)
            ws.cell(row = number_row, column = 3, value = Co_i_c)
            ws.cell(row = number_row, column = 4, value = Co_j_c)
            ws.cell(row = number_row, column = 5, value = Co_k_c)
            ws.cell(row = number_row, column = 6, value = Co_l_c)
            ws.cell(row = number_row, column = 7, value = Co_m_c)
            ws.cell(row = number_row, column = 8, value = P_CSP_c)
            ws.cell(row = number_row, column = 9, value = t_sto_c)
            ws.cell(row = number_row, column = 10, value = SM_c)
            ws.cell(row = number_row, column = 11, value = r_c)
            ws.cell(row = number_row, column = 12, value = Q_de_c)
            ws.cell(row = number_row, column = 13, value = number_row)

            wb.save("History.xlsx")
            print('Simulation'+ ' ('+
str(Co_i_c)+','+str(Co_j_c)+','+str(Co_k_c)+','+ str(Co_l_c)+','+ str(Co_m_c)+
')'+' #'+ str(number_row))
        else:
# ------------------------------       Simulation

            Co_i_c=int(Generation_data[i_spec,0])
            Co_j_c=int(Generation_data[i_spec,1])
            Co_k_c=int(Generation_data[i_spec,2])
            Co_l_c=int(Generation_data[i_spec,3])
            #Co_m_c=int(Generation_data[number_row,4])
            Co_m_c=2
            P_CSP_c=P_max[Co_i_c]
            t_sto_c=TES_value[Co_j_c]
```

```
    SM_c=SM_value[Co_k_c]
    r_c=r_P_Ac[Co_l_c]
    Q_de_c=Q_de[Co_m_c]

    MoltenSaltsCO2(
    P_max = P_CSP_c,     #per avere tutto in un solo vettore
    P_gross = P_CSP_c/0.9,
    Q_de=Q_de_c,
    P_AC = r_c*P_CSP_c,
    P_name_EH = P_name_EH,
    SM = SM_c,
    t_storage = t_sto_c,
    Optimize_SF = Optimize_SF,
    Recompression = Recompression,
    Reheat = Reheat,
    Intercooling = Intercooling,
    identification_folder_summary = identification_folder_summary,
    identification_folder_results = identification_folder_results,
    save_all_files = save_all_files,
    Co_i=Co_i_c,
    Co_j=Co_j_c,
    Co_k=Co_k_c,
    Co_l=Co_l_c,
    Co_m=Co_m_c,
    number_sim=number_row,
    Flag=Flag
    )
number_row=number_row+1
```