POLITECNICO DI MILANO

Scuola di Ingegneria Industriale e dell Informazione

Corso di Laurea Magistrale in Ingegneria Energetica

CFD simulation of a small scale solar tower receiver operated with liquid sodium

Relatore: Prof. Luca Davide Marocco
Correlatore: Prof. Andrea Giostri

Tesi di Laurea Magistrale di:
Alessandro GALLI Matr. 863010

Anno Accademico 2019/2020

Alessandro Galli: *CFD simulation of a small scale solar tower receiver operated with liquid sodium* | Tesi di laurea magistrale in Ingegneria Energetica, Politecnico di Milano.

Politecnico di Milano:
www.polimi.it
Scuola di Ingegneria Industriale e dell'Informazione:
www.ingindinf.polimi.it

Ringraziamenti

Prima di procedere con la trattazione, vorrei dedicare qualche riga a tutti coloro che mi sono stati vicini in questo percorso di crescita personale e professionale.

Un sentito grazie al mio relatore Luca Davide Marocco e al mio correlatore Andrea Giostri, per la guida e i consigli dati nel corso di questa tesi.

Un grazie infinito ai miei genitori, senza il cui supporto morale non sarei mai potuto arrivare fin qui. Grazie per esserci sempre stati soprattutto nei momenti di sconforto.

Ringrazio infine i miei amici e colleghi, compagni di avventura in questi anni di universitá, grazie per tutte le risate e i bei momenti passati insieme.

Contents

Abstract

The use of Concentrating Solar Power (CSP) for electricity production promises to be one of the most viable options to replace fossil fuel with a renewable power source [1]. Research and development activities on this type of technologies been booming rapidly since 1980s, with the objective to make them competitive with conventional thermal plants. In particular, solar tower power plants are the most promising in terms of reducing their capital costs and improving their performances [2].

Selection of an appropriate Heat Transfer Fluid (HTF) is crucial for minimizing costs of solar receivers, thermal storage and heat exchangers, and to improve overall performances of the plant [3]. Heat transfer mechanism in liquid sodium differs greatly from that of conventional Solar Salts, because it has very low Prandtl number and high thermal conductivity. Those characteristics result in higher heat transfer coefficients. Moreover, it can be employed in a much larger temperature range.

These superior properties could lead to a downsizing of a tower plant components, reducing greatly investment and maintenance costs. One of the most promising projects regarding liquid sodium is Vast Solar CSP located at Jemalong, New South Wales (AUS). It is a modular plant composed by 5 small scale solar receivers ($1.2MW_{th}$ each), but to date detailed data about its performances are not available.

The purpose of this work is to perform a CFD simulation of a small scale receiver operated with liquid sodium and designed similarly to the one in Vast Solar plant, in order to highlight strengths and weaknesses of this configuration.

Since the aim is to obtain the most realistic condition possible, the heat flux on the receiver has to be non-uniform both circumferentially and longitudinally. For this reason, also the solar field has been reconstructed with the software SolarPilot, and a ray-tracing simulation will be performed with SolTrace. In this way, it has been possible to obtain the exact distribution of sun rays over absorbing tubes surface.

Finally, the numerical simulation has been done with the open-source CFD code OpenFOAM-5.x. The case is a steady-state, conjugate heat transfer problem that involves heat transfer between solid tubes and a fluid in turbulent flow.

An accurate analysis has been done on thermal losses, heat transfer coefficients and temperature distribution, proving the viability of liquid sodium as an alternative to solar salts in solar applications.

Keywords: Liquid sodium, CFD, Conjugate heat transfer, Solar tower, Non-uniform heat flux, Turbulent convection

Sommario

L'utilizzo di sistemi a concentrazione solare rappresenta una delle migliore alternative per sostituire le fonti fossili con energia rinnovabile. A partire dagli anni '80, ricerche e studi su questo tipo di tecnologie sono cresciuti rapidamente con l'obiettivo di renderle competitive con le centrali termiche convenzionali.In particolare, le centrali a torre solare sono le piú promettenti in termini di riduzione dei costi e miglioramento delle prestazioni.

La selezione di un fluido termovettore adeguato ha un enorme impatto sulla perfomance di ricevitori, scambiatori di calore e sistemi di stoccaggio. I metalli liquidi, come il sodio liquido, hanno un meccanismo di trasmissione del calore atipico, caratterizzato da un numero di Prandtl molto basso ed elevatissima conduttivitá termica. Di conseguenza, il coefficiente di scambio termico risulta molto piú elevato rispetto ai sali fusi. Inoltre possono essere impiegati in un range di temperature maggiore. Queste proprietá consentirebbero di ridurre le dimensioni dei componenti delle torri solari, con costi di investimento e manutenzione inferiori. Uno dei progetti piú promettenti con il sodio liquido é la centrale Vast Solar a Jemalong, nel New South Wales (AUS). Si tratta di una centrale modulare composta da 5 ricevitori di piccola taglia ($1.2 MWth$ ciascuno), ma ad oggi non sono disponibili dati dettagliati sulle sue prestazioni.

Lo scopo di questo lavoro é di eseguire una simulazione CFD su un ricevitore solare centralizzato di piccola taglia a sodio liquido progettato su modello di uno dei moduli di Vast Solar, al fine di verificare punti di forza e limiti di questa configurazione. Essendo il flusso termico reale sul ricevitore non uniforme, é stato necessario ricostruire il campo solare al fine di ottenere una condizione al contorno realistica. Per ottenere una simulazione realistica, il flusso termico sul ricevitore deve essere non uniforme sia longitudinalmente sia lungo la circoferenza dei tubi. Per questo motivo anche il campo solare é stato simulato con il software SolarPilot, e la distribuzione dei raggi solari sul ricevitore tracciata con SolTrace.

La simulazione numerica é stata eseguita con OpenFOAM-5.x, un codice open source. Si tratta di un caso stazionario, con un fluido in moto turbolento e scambio di calore solido/liquido.

Un'analisi accurata é stata condotta sulle perdite di calore, coefficienti di scambio termico e distribuzione di temperatura, dimostrando come il sodio sia una alternativa promettente ai sali fusi nelle applicazioni solari.

Parole chiave: Sodio liquido, CFD, Scambio termico coniugato, Torre solare, Flusso termico non uniforme, Convezione turbolenta

Extended Abstract

Introduction

Solar tower plants are one of the most promising technology among Concentrating Solar Power (CSP) systems. In a solar tower plant, a field of sun-tracking mirrors (heliostats) concentrate the solar radiation onto a receiver placed on top of a tower. The incident solar energy heats up an Heat Transfer Fluid (HTF), which is then used to sustain a power cycle. Compared to other configurations such as parabolic through, Fresnel and dish collectors, tower thermal plants are expected to reach lower levelized electricity costs [**2**,**3**]. Being the receiver a key component for the plant, the choice of a proper HTF is crucial to improve thermal performances. The most common fluids used in current generation plants are binary Solar Salts (mass specific composition 60% $NaNO_3$ and 40% KNO_3) and Hitec (mass specific composition 53% KNO_3, 40% $NaNO_2$ and 7% $NaNO_3$). The biggest drawback of these fluids is the limited temperature range: melting point is at 220°C (SS) and 142°C (Hitec). Also, they become chemically unstable above 565°C [**7**]. This results in great expense in auxiliary heating for piping and no possibility to couple the solar tower with an USC steam cycle ($\sim 600-620°C$) or other advanced power cycles (e.g. sCO_2 Brayton cycle, 650°C+). A promising alternative to Solar Salts are liquid metals (sodium, tin, LBE mixtures): in particular liquid sodium is the most technologically ready candidate as there is extensive experience from nuclear industry. Sodium shows a larger temperature range, with a melting temperature of 97.7°C and a boiling temperature of 873°C [**11**]. Moreover, as all liquid metals, is characterized by very high thermal conductivity and low Prandtl number.

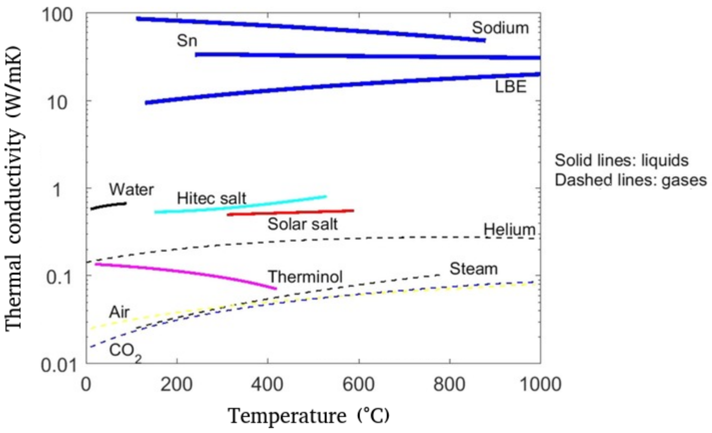


Figure 1: *Thermal conductivity as a function of temperature for three selected liquid metals and other common heat transfer fluids. Conductivity is plotted only in viable temperature range for each fluid*

Prandtl number, defined as $Pr = \nu/\alpha$, is a non-dimensional parameter and can be seen as the ratio between momentum and thermal diffusivity. In this work, Pr number variates in a range between 0.0045 and 0.0065 depending on temperature, meaning that thermal diffusivity is predominant and thermal boundary layer has a bigger thickness [**10**]. As a result, liquid sodium shows very high heat transfer coefficients which allow to operate with higher solar heat flux while maintaining an acceptable temperature difference between the absorber inner surface and the fluid. Another way to exploit these superior heat transfer characteristics is to reduce receiver absorbing area [**10**] and dimension of the solar field, with lower investment and operating costs. In addition, a small external surface means also a reduction of thermal losses, with possibility of more efficient receivers.

During the 80s some tests have been carried out with sodium receivers [**18**], but they were eventually disbanded due to sodium safety issues (violent reaction with air humidity in case of leakage). In recent years, the only significant project involving sodium is a pilot plant by Vast Solar in New South Wales (AUS) [**23**]. The 3500 heliostats facility is modular, composed by five small array-modules concentrating solar radiation on a dedicated thermal receiver tower 27 meters in height. Since detailed performances of that plant are still not available in literature, in this work will be performed a CFD simulation of a small scale solar receiver operated with sodium, using a single module of the Vast Solar plant as a reference for geometrical features. The aim is to analyze all thermodynamic aspects of the receiver such as temperature within wall thickness and in bulk fluid, Nu numbers, heat transfer coefficients and heat losses, highlighting strengths and weaknesses of this particular configuration.

Solar field layout modelling

In order to reproduce realistically the non-uniform heat flux distribution onto the receiver, the solar field layout has been simulated with the software SolarPilot. Geometric features and location of the simulated plant are designed taking as a reference a single module of the Vast Solar Plant and they are summarized in Table 1.

Solar Field		**Receiver**	
N. of heliostats	700	Height from ground	$27m$
Single heliostat area	$3.6m^2$	Receiver lenght	$3.2m$
Tracking	2-axis	Receiver width	$0.9m$
Reflective surface ratio	0.97	Absorbing area	$2.25m^2$
Mirror reflectivity	0.95	Number of tubes	20
Soiling factor	0.97	Tube outer diameter	$25mm$
Average cosine efficiency	0.87	Tube inner diameter	$21.25mm$
Average intercept factor	0.85	Tube thickness	$1.875mm$
Average optical efficiency	0.64	Tube lenght	$2.5m$
		Headers diameter	$0.15m$

Table 1: *Geometric data of heliostats (left) and receiver (right)*

The plant is located in Jemalong, New South Wales (AUS). Exact coordinates are $-33.4°$ latitude and 147.7°E longitude. Design Sun position is chosen during australian

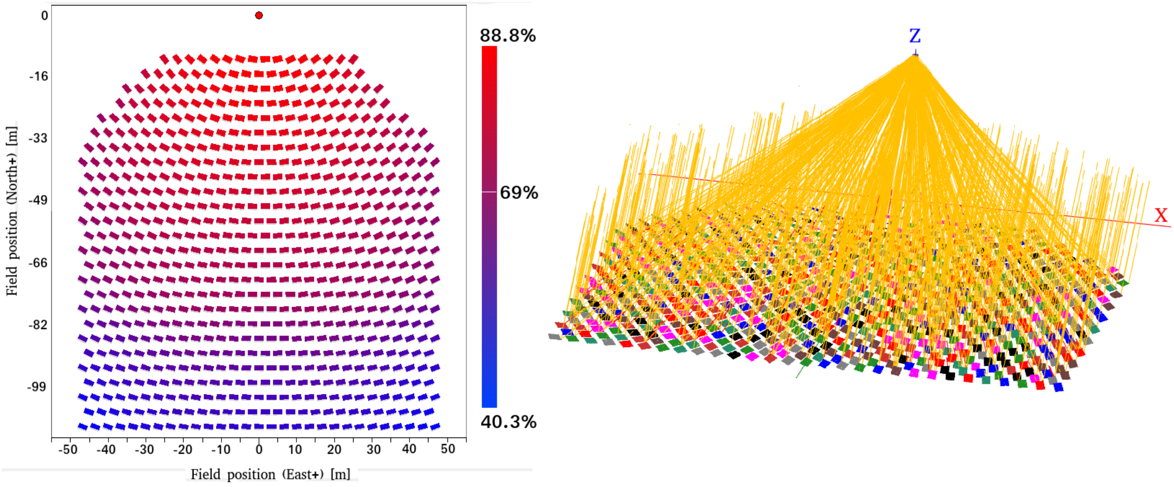


Figure 2: *Solar field layout generated with SolarPilot; colouring shows optical efficiency (left). Path of 800 sun rays traced with SolTrace(right)*

Summer Solstice (22 December): at 13:00 (UTC+11) solar elevation is 76.35° and azimuth angle is 352.25°. Design Direct Normal Irradiation (DNI) is set at 950 W/m^2. The resultant mirrors disposition is shown in Figure 2. Heliostats are placed all in front of and not around the tower to obtain high cosine and intercept factor: since the number of mirrors is low it is important to maximize the contribution of each one of them. Design heat flux from that field shows a peak value of 1.5 MW/m^2, with an average of 0.76 MW/m^2.

With the generated solar field layout, it is possible to trace sun rays. Solar position defined in SolTrace is the design point used also in SolarPilot. This choice is made to generate a symmetrical solar heat flux, which will be useful later to reduce computational time. Nominal DNI is again 950 W/m^2, with 1 million of generated rays. The software provides as an output a series of CSV tables, one for each absorbing tube, containing spatial coordinates of intersection points and heat flux value (W/m^2) for each one of those points.

OpenFOAM case setup

In this simulation the whole receiver needs to be meshed. The geometry is not trivial, so the application *snappyHexMesh* has been used to generate a mesh from a CAD 3D model. Thanks to the assumption of simmetrical heat flux, only half receiver has been meshed in order to halve the number of cells. The total number (half receiver) is 11510505 cells. In Figure 3, an overview of the geometry is shown, while in Figure 4 it is possible to see cross section area of one absorbing tube.

This is a conjugate heat transfer case, so both solid tube walls and fluid region have been meshed. The solver used is *chtMultiRegionSimpleFoam*, which is a multiregion, steady-state solver. Turbolence model is *komega-SST*, a two-equations eddy-viscosity model. Inlet condition for liquid sodium are $T = 543\ K$ and a mass flow rate of $\dot{m} = 3.272\ kg/s$.
Regarding solid domains (walls of absorbing tubes) there are two different conditions on the two sides of each tube: on the half not exposed to solar radiation, external

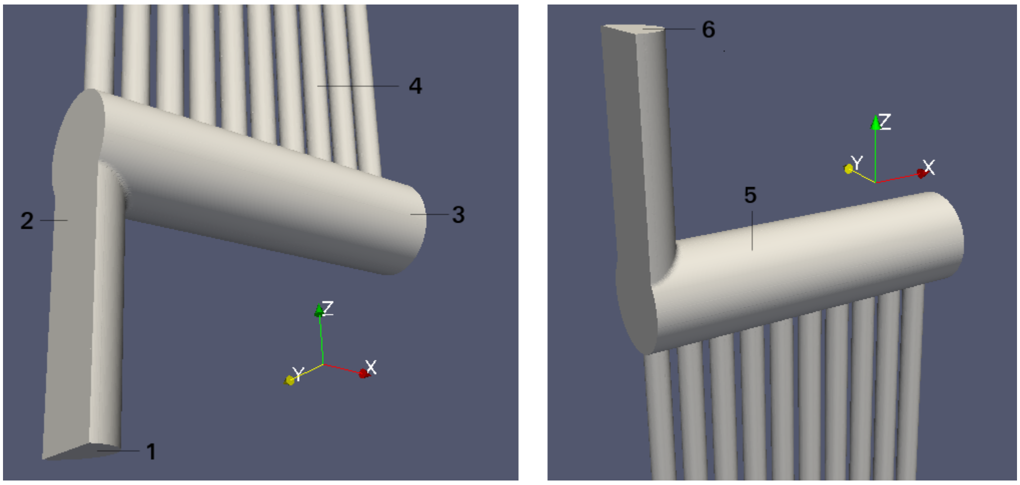


Figure 3: *Simulated receiver: (1) inlet patch; (2) symmetry plane; (3) inlet header; (4) absorbing tubes; (5) outlet header; (6) outlet patch*

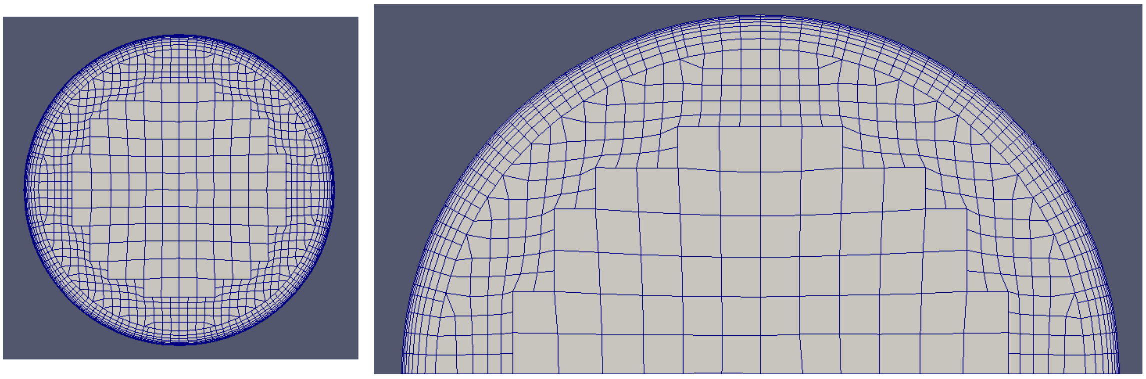


Figure 4: *Cross section of fluid domain of a single tube. On the right it is possible to see more clearly the near-wall boundary layer made with added cells*

surfaces can be considered as adiabatic. On external surfaces hit by solar radiation, the boundary condition is written in (1), an expression that takes into account also radiative and convective losses:

$$\lambda\frac{\partial T}{\partial n} = \alpha q''(x,r,\theta) + h_{conv}(T_\infty - T_s) + h_{rad}(T_{sky} - T_s) \tag{1}$$

Non-unifrm heat flux ($q''(x,r,\theta)$) is read from tables generated with SolTrace, where α is absorptance factor, taken at 95% [**10**]. Convective term is defined by a convective heat transfer coefficient (h_{conv}) and by the temperature difference between tubes external surfaces (T_s) and environment (T_∞). Convective heat transfer coefficient is dependent from wind speed; for this simulation a reasonable value [**11**] of $5W/m^2K$ is considered. The last term, radiative loss, has been linearized through a radiative heat transfer coefficient, defined as:

$$h_{rad} = \sigma\epsilon(T_s^{*2} + T_{sky}^2)(T_s^* + T_{sky}) \tag{2}$$

where T^* is temperature evaluated at the previous time step. Regarding emissivity factor ϵ, a research of 2014 [**12**] studied its impact on radiative losses of solar receivers

considering different values between 0.2 and 0.8. In order to consider the worst-case scenario, a value of 0.8 has been used in this simulation. Also, view factors between absorbing tubes have been neglected for simplicity, with a consequent slight overestimation of radiative loss.
To impose Equation 6 as a boundary condition in OpenFOAM an additional library is needed, called *swak4Foam*, which allows to write custom expressions **[9]**.
Thermophysical properties (ρ, k, c_p, μ) of liquid sodium are given as a function of temperature in polynomial form.

Results

The first aspect that will be studied is temperature distribution on the receiver. Heat flux is strongly non-uniform and only one side of tubes is exposed to solar radiation. As a consequence also temperature variates both circumferentially and longitudinally, with great differences between tubes, as shown in Figure 5. Peak temperatures on external surfaces are found in central tubes, where there is maximum solar radiation intensity: here temperatures overcome $700°C$, with a maximum value of 762°. Tubes with higher incident heat flux also show greater temperature difference within walls thickness, with peak differences of $90°C$ between outer and inner surfaces. Again in Figure 5, it is possible to appreciate the higher temperature drop in high-flux tubes. Metal alloys used in high-temperature applications can handle those peak temperatures: Incoloy 800HT is the material chosen in this receiver, which is declared stabled up to $899°C$ **[13]**. Great importance to understand thermal performances is the temperature distribution inside fluid domain. In Figure 6 is shown bulk temperature profile inside each tube. At the end of outlet header the bulk temperature is $505°C$, with a difference of $\Delta T = 235°C$ with respect to inlet temperature. This result is slightly lower with respect to conventional large tower plant ($\sim 550/560°C$), but still very promising considering the very reduced heat exchanging area in the small simulated receiver.

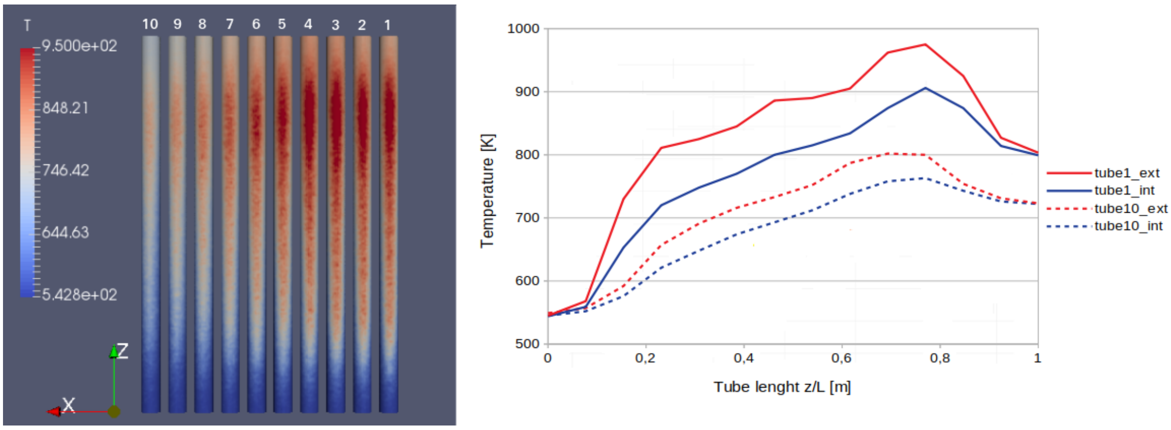


Figure 5: *Temperature (K) distribution on tubes external surfaces (left); on the right temperature profile is plotted in internal and external surfaces of tube1 (hit by highest heat flux intensity) and tube10 (lowest heat flux)*

Regarding heat transfer coefficient, it has been calculated according to:

$$h = \frac{\lambda \, Nu}{L_c} \tag{3}$$

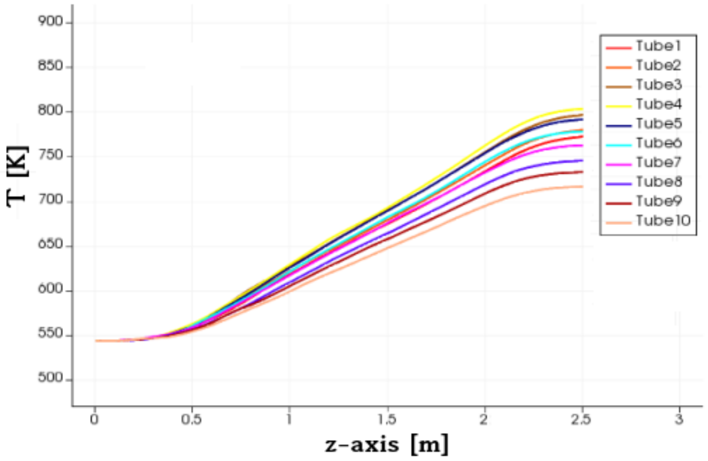


Figure 6: *Bulk temperature inside each tube plotted along z-axis*

where Nusselt number has been estimated as the dimensionless temperature gradient normalized with a characteristic temperature difference [**14**]. Since both temperature gradient and wall temperature variate along inner tube circumference, they have been integrated over internal surface in order to obtain average values:

$$Nu = \frac{-\overline{\frac{\partial T}{\partial x}}\, L_c}{T_{bulk} - \overline{T_{wall}}} \tag{4}$$

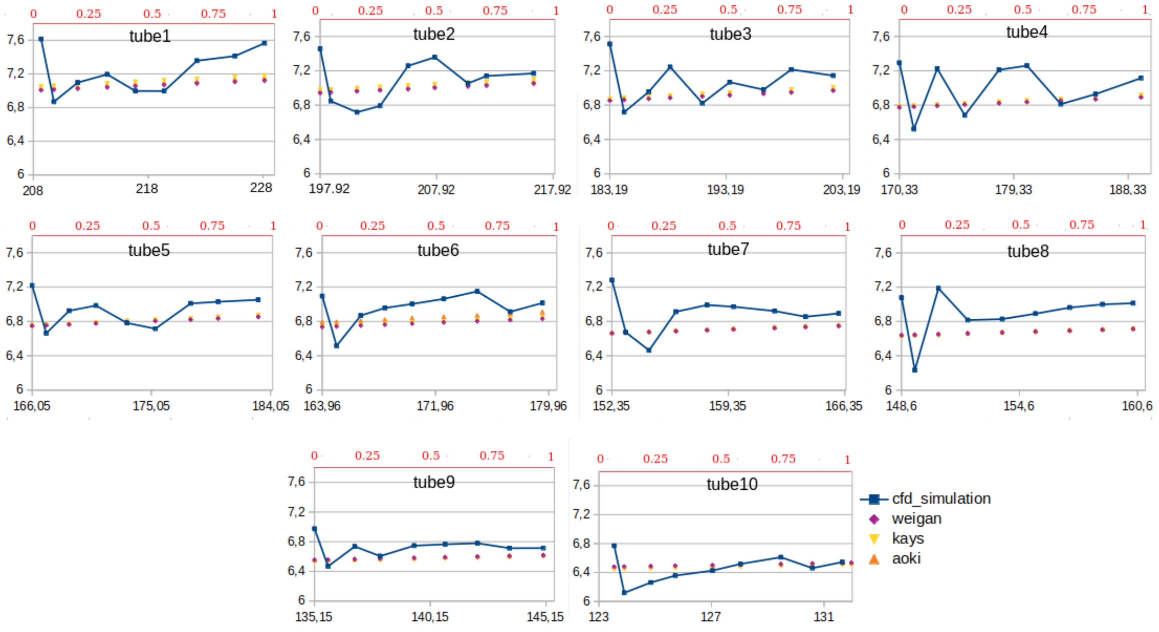


Figure 7: *Nusselt numbers in each tube plotted versus Peclet (black x-axis). Red x-axis show dimensionless tube length z/L. Dots show values obtained empirical correlations*

To understand better the meaning of Nu number, starting from Equation (3) it's possible to write:

$$Nu = \frac{L_c h}{k} = \frac{h \Delta T}{k \frac{\Delta T}{L_c}} = \frac{q_{conv}}{q_{cond}} \tag{5}$$

So the physical meaning of Nusselt number is the ratio of convective to conductive heat transfer across a boundary. Values close to unity mean that conductive heat transfer mechanism is preponderant with respect to convection, such as in the case of liquid metals. In Figure 7, Nu numbers obtained from CFD simulation are compared to the ones obtained with empirical correlations; in liquid metals Nu can be written as a function of Peclet number [8] , where $Pe = RePr$. The correlation for Nusselt may, therefore, can be written in the following form:

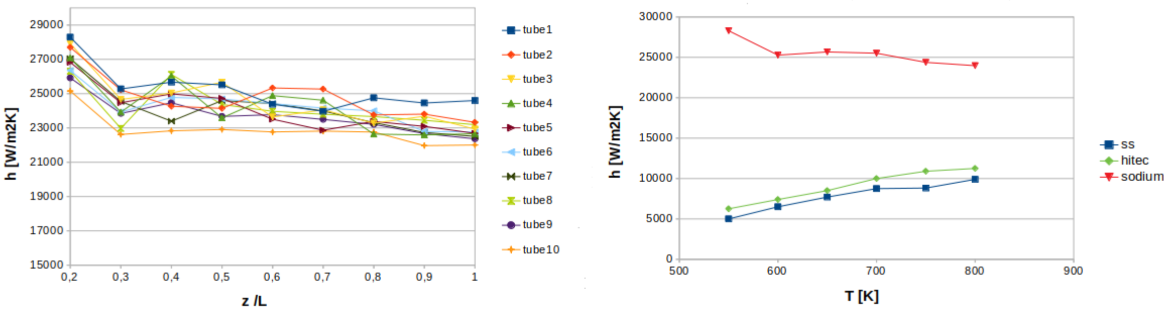


Figure 8: *Heat transfer coefficients plotted along z-axis of each tube (left); comparison at same temperatures between heat transfer coefficients found in tube1 and a test pipe operated with Solar Salts and Hitec with* $\dot{m} = 1kg/s$ *(right)*

$$Nu = x_1 + x_2 Pe^{x_3} \tag{6}$$

where x_i are coefficients according to Weigan-Ferguson, Aoki and Kays models [8]. Notice that those correlations are given with the assumption of an uniform heat flux, so it is expected a certain difference with numerical values. The comparison is still useful as a reference to understand if numerical values are reasonable. Nusselt numbers inside the receiver are comprised within a range of 6 to 8: they are small as expected in a heat transfer process dominated by conduction.

Expectations on heat transfer coefficients are also confirmed: inside absorbing tubes they are very high, in the range between 22000 and 26000 W/m^2K. Looking at Figure 8, a rough comparison is done with a test pipe run with Solar Salts and Hitec: sodium heat transfer coefficients are from 2.5 to 5 times greater.

The last part of this work is focused on heat losses and efficiency analysis. The net power entering the fluid in half volume is 408.93 kW (817.86 kW for the whole receiver) with a total thermal efficiency of 90.24%. Efficiency has been calculated according the following definition:

$$\eta_{th} = \frac{\dot{Q}_{net}}{\dot{Q}_{gross}} = \frac{\alpha \dot{Q}_{gross} - \dot{Q}^{loss}_{conv} - \dot{Q}^{loss}_{rad}}{\dot{Q}_{gross}} \tag{7}$$

Let us now see in detail each type of thermal loss. The first one is related to non-perfect absorptance of tube surfaces. Tubes are coated with thin layers of a special paint, called Pyromark, which increase absorptance factor up to 95%: a 5% loss with

	Net power [W]	Gross power [W]	Efficiency
Tube1	53901.3	59131.2	0.9115
Tube2	52618.2	57813.3	0.9101
Tube3	51000.6	56221.6	0.9071
Tube4	48421.2	53527.8	0.9045
Tube5	44260.3	49036.1	0.9026
Tube6	40245.8	44676.8	0.9008
Tube7	36074.7	40169.3	0.8998
Tube8	31262.4	34961.7	0.8941
Tube9	27796.3	31210.8	0.8905
Tube10	23346.2	26406.8	0.8841
Half rec.	408926.8	453155.5	0.9024

Table 2: *Heat balance and efficiency on each tube and on the whole receiver*

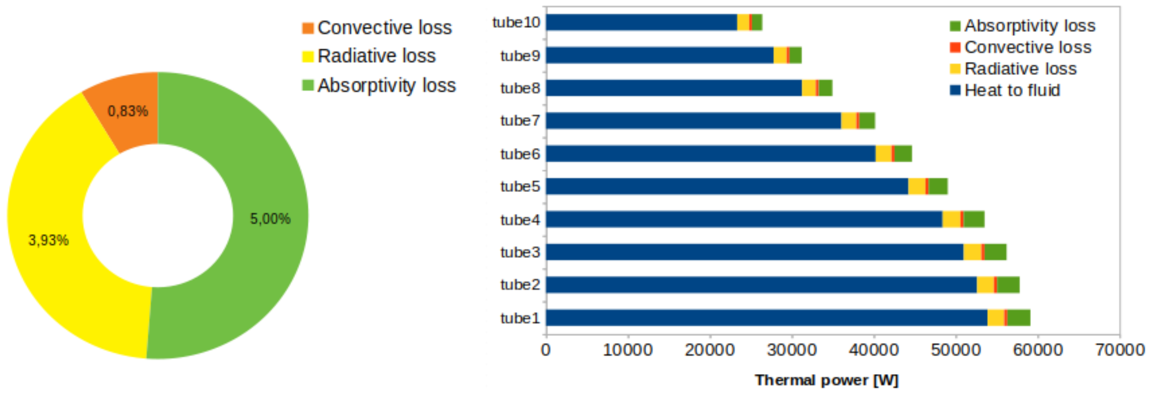


Figure 9: *Weight of each loss in the whole receiver (left); power balance in each tube (right)*

respect to total incident power has to be considered. Loss related to failed absorption is the biggest one and it is proportional to heat flux intensity, so it has a heavier impact on central tubes, as can be seen in Figure 9. The second loss is due to convection. It is responsible of less than 1% of the total heat loss. Being the receiver external area very small, also the magnitude of this loss is limited. Also, the considered geographic area is characterized by a low average wind speed, resulting in small h_{conv}. Finally, the last loss is related to radiative emission from receiver to environment. This loss is highly dependent on temperature, since the total radiant heat power emitted from a surface is proportional to the fourth power of its absolute temperature. It is clear that also in this case tubes hit by higher heat flux show a greater loss. Both convective and radiative losses depends on exchanging area, so a small receiver allows to contain those losses and improve efficiency.

From the analysis, high-flux tubes experience the highest total thermal loss, but still they show better thermal efficiencies (Table 2), stressing how high solar flux concentration is crucial to exploit in full sodium properties. Same high-flux tubes are crossed by higher mass flow rates as a consequence of the inlet header geometry, leading to a greater level of turbulence, which is positive for heat transfer coefficients: this provides an effective cooling of the tubes, avoiding dangerous local peaks of temperature. To enhance turbulence different design solutions could be employed. For example, at fixed mass flow rate, it is better to have more tubes but with smaller diameter, in order to increase bulk flow velocity (and Re number) in each one of them.

Conclusions

A thermal and efficiency analysis of a small scale solar tower plant has been completed, considering a 700 heliostats solar field and a receiver with 2 m^2 of absorbing area operated with liquid sodium. The heliostat field has been reproduced and used to apply a realistic non-uniform heat flux condition through a ray-tracing software on half of the outer wall surfaces, while the other half has been considered adiabatic. The boundary condition has been imposed specifying spatial coordinates of intersection points between traced rays and the receiver, with an heat flux value (W/m^2) assigned to each point. According to the author's knowledge, there are presently no investigations for small size CSP systems with liquid metals subjected to these particular conditions. The CFD simulation made with the software OpenFOAM showed the advantages of employing liquid sodium as a HTF. Heat transfer coefficients found inside absorbing tubes are from 2.5 to 5 times higher with respect to Solar Salts, making downsized solar receivers in solar tower a feasible choice. The system produces 0.82 MW_{th} with a global thermal efficiency above 90%, an encouraging result considering also that harsh conditions have been used during heat loss estimation, with a slight overestimation of radiative losses. Output temperature is $505°C$, a remarkable result considering the very reduced heat exchanging surface in the receiver. The analysis shows how tubes hit by higher heat flux intensity have also the highest thermal efficiency, highlighting that optimization of solar field performances is crucial to exploit in full liquid sodium characteristics. This suggest that sodium could be successfully employed also in large scale towers and high-flux applications.

However, it should be remembered that liquid sodium presents safety issues in case of leakage, in particular at high temperature, so operating and maintenance procedures have to be specifically designed for sodium plants. Moreover it is not an ideal fluid for storage purposes, having a small energy density. In conclusion, liquid sodium has proven himself as a viable and promising alternative to Solar Salts in solar applications, even though research and investments are still needed to solve mentioned limits.

Chapter 1

Introduction to Concentrated Solar

1.1 CSP technology

Global energy consumption, and in particular, electricity consumption are increasing rapidly in the last years due to the growth in population, industrialization, and urbanization [1]. Increasing the share of electricity produced from renewable sources means both to tackle the climate change and to reduce the dependence on fossil fuels; for this reasons, over the past decade, huge investments in renewable technologies have been done by Europe, in particular solar and wind.

Among all renewable sources Concentrated Solar Power (CSP) represent one of the most promising technology due to the possibility of producing dispatchable electric energy [2]. These systems generate solar power by using mirrors or lenses to concentrate solar radiation onto a small area (the receiver).

CSP plants are classified according to the way in which rays are focused. Typical categories are parabolic trough, solar tower (or central receiver), linear Fresnel and dish Stirling. Parabolic through and linear Fresnel are based on long and continuous mirrors that concentrate radiation along a line; on the other hand dish Stirling and solar tower are point-focusing systems [3][15]. In order to reach high energy density a very important parameter to consider when studying the performances of a CSP system is the concentration ratio (CR):

$$CR = \frac{ApertureArea}{AbsorberArea} = \frac{A_p}{A_a} \tag{1.1}$$

Central receiver systems are the most promising for large scale power production [9], because they can reach the highest flux intensity.

In central receiver systems, solar energy concentration and collection is based on a field of individually sun-tracking mirrors (heliostats) that reflect the incident sun rays to a receiver placed at the top of a tower. Typically hundreds or even thousands of mirrors are installed in order to maximize CR and reach very high energy density, since A_p is proportional to the total mirrors surface. This power from the sun is used to heat up the fluid pumped inside the receiver. The heated fluid (or steam) returns down the tower and then to a thermal demand such as a thermal electrical power plant or an industrial process requiring heat, as shown in Figure 1.1.

Thanks to the high CR, tower systems have the potential to reach very high heat flux

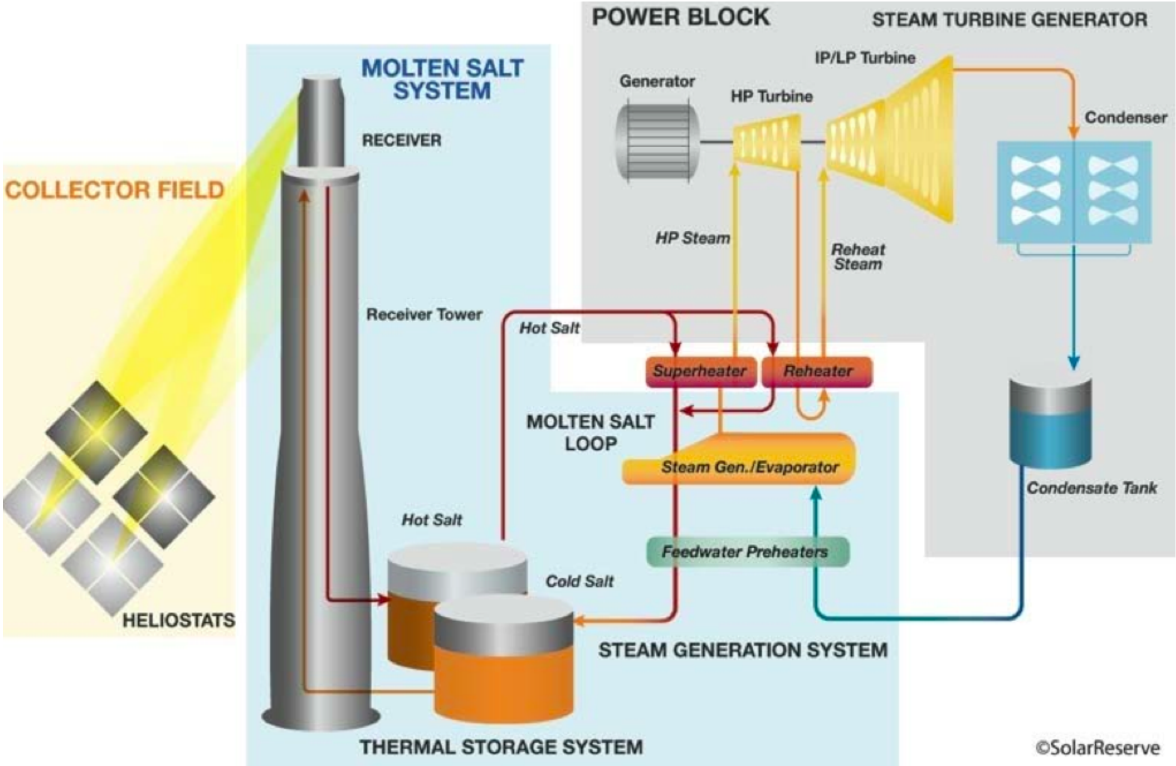


Figure 1.1: *Scheme of a solar plant with tower receiver and two tanks direct storage system*

intensity and temperatures. From an efficiency perspective, this is a huge advantage: in general, the overall conversion efficiency of a CSP system can be written as:

$$\eta_{CSP} = \eta_{th}\eta_{cycle} \tag{1.2}$$

In Equation (1.2) η_{cycle} is the efficiency of the thermodynamic cycle adopted in the power block. Remembering the expression of Carnot efficiency, $\eta_{carnot} = 1 - \frac{T_{amb}}{T_{max}}$, it is clear how increasing max temperature can improve thermodynamic performances. The other term, η_{th}, is the thermal efficiency and express how well the receiver can absorb solar radiation. The estimation of this factor will be the scope of this work.
The high heat flux achievable, together with the possibility of having thermal storage, make central receiver particularly suited for large power levels (1 to 500 MW) and high temperatures [4][5].
As already mentioned, dispatchability is ensured by the presence of a storage system, which is able to sustain the power block for a certain time span (from tens of minutes to few hours) and to compensate the intrinsic variability of a renewable source. Storage can be direct, if the same fluid in the receiver is stored inside insulated tanks to minimize heat losses, or indirect, when the stored fluid is different from the heat transfer fluid. Of course in the second case, an heat exchanger between the two fluids will be needed [6].

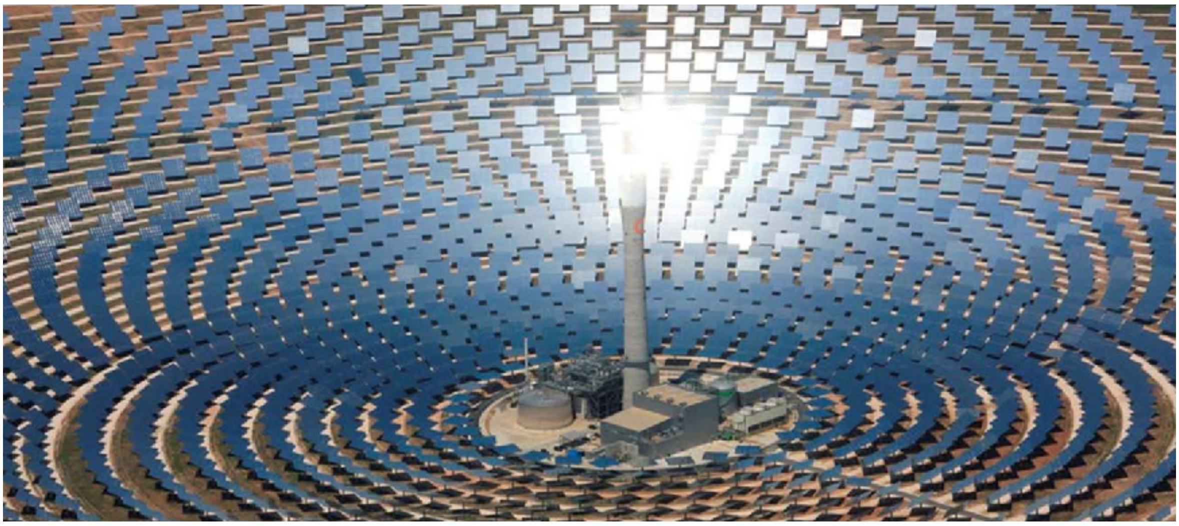


Figure 1.2: *View of Gemasolar plant in Spain. With 2650 heliostats, it has 17 MW of electrical power and a net electric production of 80 GWh/year*

1.2 Heat transfer fluid

Several different fluids can be employed as HTF [15], and this choice affect heavily all the design aspects of the plant [7]. In general, the best fluid for this application should fulfill the following requirements [8]:

- Low melting temperature: freezing of the fluid should be avoided, so lower melting point means lower auxiliary heat consumption
- High upper temperature limit: typically the limit is the boiling temperature or the point in which the fluid loses chemical stability
- Large thermal conductivity λ [W/mK]: more efficient heat transfer
- Low viscosity μ [kg/ms]: lower pressure drops
- Large specific heat capacity c_p [J/kgK]: possibility to have direct storage system
- Safe fluid: means non-polluting and non-flammable in case of leakage
- Easy to operate: means non-corrosive fluid
- Low cost

Most of the operative CSP plants at the present day employ molten salts technology as HTF [26]: binary salts (mass specific composition 60% $NaNO_3$ and 40% KNO_3) and Hitec (mass specific composition 53% KNO_3, 40% $NaNO_2$ and 7% $NaNO_3$). These fluids are cheap, they don't have particular safety issues and their high heat capacity allows to have direct thermal storage [51]. However, the limited temperature range represent their main drawback: solar salts melts at 220°C and become chemically unstable above 565°C [8][10][44]; this results in great expense in auxiliary heating for

	Solar Salts	**Hitec**	**Liquid Sodium**
Freezing point (°C)	220	142	97.7
Upper temperature (°C)	565	535	873
Density (kg m^{-3})	1899	1640	880
Viscosity (Kg m^{-1} s^{-1})	3.26	3.16	3.47
Heat capacity (J Kg^{-1} K^{-1})	1490	1560	1308
Cost ($/kg)	0.49	0.93	2

Table 1.1: *Comparison between main HTF. Properties taken at 300° C,1bar*

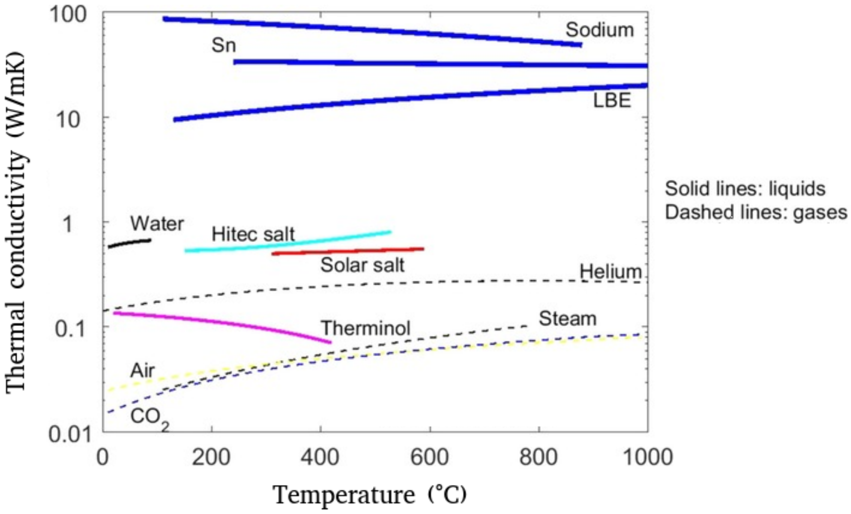


Figure 1.3: *Thermal conductivity as a function of temperature for three selected liquid metals and other common heat transfer fluids*

the piping and no possibility to couple the solar tower with an USC steam cycle ($\sim 600-620°C$) or other advanced power cycles (e.g. sCO_2 Brayton cycle, $650°C+$). Pacio and Wetzel performed a screening of liquid metals as HTF for solar systems, concluding that sodium is one of the most promising candidates [8]. Main characteristics of sodium and molten salts are compared in Table 1.1. As described by chemistry manuals and engineering handbooks [11][12] sodium has a lower melting temperature with respect to solar salts, (only $97.7°C$ which means a sensibly lower amount of auxiliary heat needed), and a boiling point of $873°C$. Moreover, as it's possible to see in Figure 1.4, sodium has a thermal conductivity that is more or less two orders of magnitude bigger with respect to common solar salts and this is the most interesting aspect in employing liquid metals. Figure 1.3 sums up the previously mentioned characteristics, showing thermal conductivity as a function of operative temperature range of many different fluid in heat transfer applications.

This huge conductivity affects the heat transfer coefficient as well: in particular, liquid metals have unique heat transfer mechanism in which conduction is predominant with respect to convection. A deeper discussion of these aspects can be found in the next chapter.

Figure 1.5 shows a comparison of h between sodium and Hitec: sodium heat transfer coefficient is approximately an order of magnitude greater with respect to Hitec, giving and idea of its potential in this type of applications. In a recent comparison [8] between

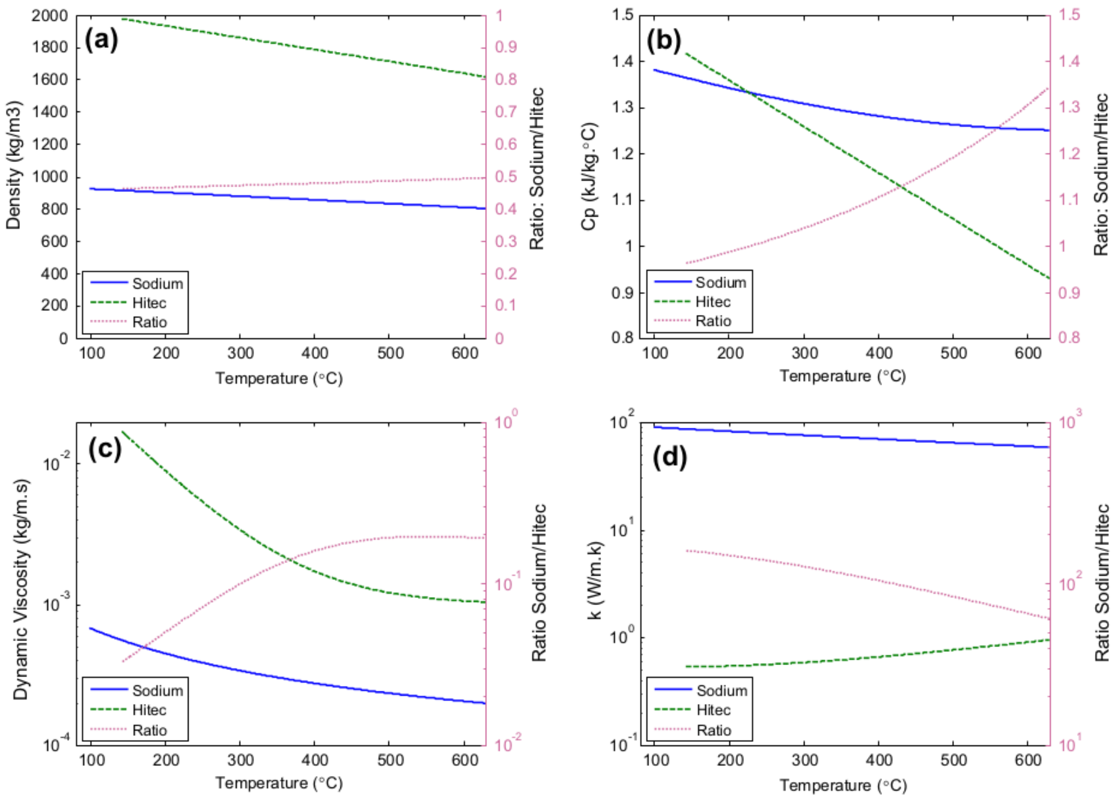


Figure 1.4: *Properties of sodium and Hitec as a function of temperature: (a) density [kg/m^3], (b) specific heat capacities at constant pressure [kJ/kg K], (c) dynamic viscosity [kg/m s], (d) thermal conductivity [W/m K]*

sodium and salts it has been demonstrated that a reduction of 57% in the receiver area is possible; furthermore better heat transfer coefficient prevent overheating of the tubes, reducing thermal stresses on the structure [30].

Unfortunately two big drawbacks still prevent the diffusion of this technology:

- **Safety issues**: sodium produce a very dangerous reaction when put in contact with water [45]:

$$2Na(s) + 2H_2O \rightarrow 2NaOH(aq) + H_2(g) \tag{1.3}$$

  The products are sodium hydroxide (caustic soda), which is very corrosive, and hydrogen gas. The reaction is strongly exothermic and produce enough heat to ignite the hydrogen; at high temperatures even the atmospheric humidity is enough to produce the so-called sodium fires in case of leakage [13][14]. One of the most important sodium fire accident occurred in 1986, at Plataforma Solar de Almeria, Spain: during the maintenance of a valve about 14 tons of sodium has been spilled in a time span of half a hour. However, no fire propagation occurred and only the structures close to the leakage were damaged.
  Also Vast Solar Pilot plant had an incident in 2015 [52]: a small leakage of sodium occurred in the cold tank piping system. Liquid Sodium at 105°C ignited in contact with air, resulting in a fire lasted 8 hours. However, the fire was fully contained and there never was any risk of injury.

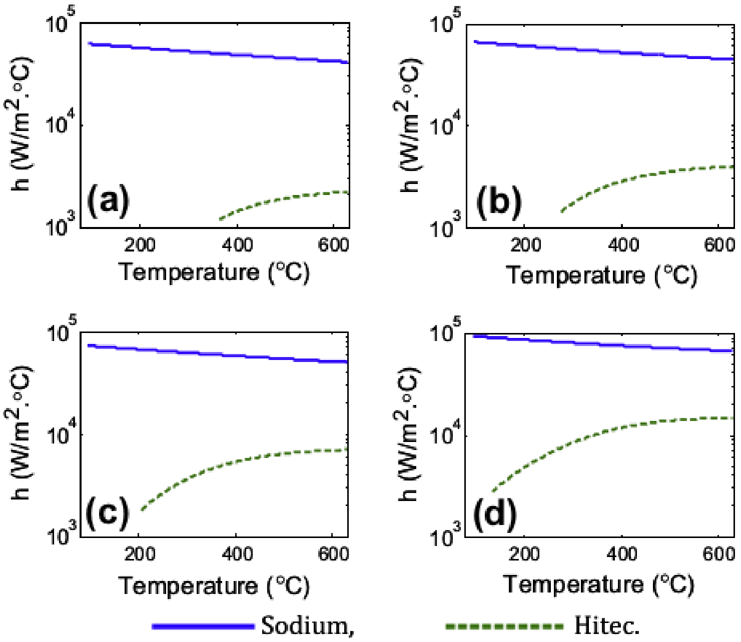


Figure 1.5: *heat transfer coefficient for different mass flow rates: (a) 0.05 kg/s, (b) 0.1 kg/s, (c) 0.2 kg/s, (d) 0.5 kg/s, stainless steel*

- **Storage system design** : sodium has slightly lower specific heat capacity with respect to common solar salts, and definitely a lower density. A lower energy density means a bigger size of the storage system, with an increase in plant investment costs. Moreover, even though storage tanks are well insulated, a certain amount of heat loss is unavoidable. Both convective and radiative losses are proportional to the tank area, so bigger sizes means lower storage efficiency.

1.3 Heat transfer in liquid metals

Liquid metals, such as liquid sodium, are characterized by extremely high thermal conductivity $\lambda($ W/mK), and that's the reason why they are so interesting in heat transfer applications [16][32].

The common characteristic of all liquid metals is an extremely low Prandtl number, which is typically much smaller than 1:

$$Pr = \frac{c_p \mu}{\lambda} = \frac{\nu}{\alpha} \ll 1 \tag{1.4}$$

so they belong to the so-called category of low-Prandtl number fluids, (for a more comprehensive description of low-Prandtl number fluids thermal hydraulics see [41]). Prandtl number is a non-dimensional parameter and can be seen as the ratio between momentum and thermal diffusivity. To understand better the effect of a low Pr number on the fluid it could be useful to start from conservation equations

First of all, liquid metals are Newtonian fluids that below their boiling point are almost incompressible [41] ($\rho = const.$). In general, the rate of change of density of a material fluid element is given by the material derivative:

$$\frac{D\rho}{Dt} = \frac{\partial \rho}{\partial t} + u_j \frac{\partial \rho}{\partial x_j} \tag{1.5}$$

From the continuity equation we have:

$$\frac{D\rho}{Dt} = -\rho \frac{\partial u_j}{\partial x_j} \tag{1.6}$$

Hence the flow is incompressible if the divergence of the velocity field is identically zero. Note that the density field need not be uniform in an incompressible flow. All that is required is that the density of a fluid element should not change in time as it moves through space. Assuming also properties constant with the temperature, mass (1.7) momentum (1.8) and energy (1.9) conservation equations take the form [42]:

$$\vec{\nabla}\vec{U} = 0 \tag{1.7}$$

$$\frac{D\vec{U}}{Dt} = \nu \nabla^2 \vec{U} - \frac{1}{\rho}\vec{\nabla}p + \vec{f} \tag{1.8}$$

$$\rho c_p \frac{DT}{Dt} = \lambda \nabla^2 T + \mu \Phi \tag{1.9}$$

$$\Phi = 2\left[\left(\frac{\partial u}{\partial x}\right)^2 + \left(\frac{\partial v}{\partial y}\right)^2 + \left(\frac{\partial w}{\partial z}\right)^2\right] + \left(\frac{\partial v}{\partial x} + \frac{\partial u}{\partial y}\right)^2 + \left(\frac{\partial w}{\partial y} + \frac{\partial v}{\partial z}\right)^2 + \left(\frac{\partial u}{\partial z} + \frac{\partial w}{\partial z}\right)^2 \tag{1.10}$$

This set of equations can be shifted into dimensionless form using the following set of parameters:

$$x^* = \frac{x}{L_c} \ ; \ u^* = \frac{u}{u_c} \ ; \ T^* = \frac{T - T_c}{\Delta T_c} \ ; \ p^* = \frac{p - p_c}{\rho u_c^2} \tag{1.11}$$

where $L_c, u_c, \Delta T_c$ and p_c are characteristic parameters chosen arbitrarily. Substituting all variables with their dimensionless, a new set of conservation equation is found:

$$\vec{\nabla}^* \vec{U}^* = 0 \tag{1.12}$$

$$\frac{D\vec{U}^*}{Dt} = \frac{1}{Re}\nabla^{*2}\vec{U}^* - \vec{\nabla}^* p \tag{1.13}$$

$$\frac{DT^*}{Dt} = \frac{1}{RePr}\nabla^{*2}T^* + \frac{Ec}{Re}\Phi^* \tag{1.14}$$

Ec is Eckert number, it's the ratio between the flow kinetic energy and enthalpy variation. It is typically very small for incompressible fluids, so it can be considered $= 0$ without big errors.

$$Ec = \frac{V^2}{c_p \Delta T} \tag{1.15}$$

Re is the Reynolds number, and its physical meaning can be seen as the ratio between the inertial forces in a fluid and the viscous forces:

$$Re = \frac{\rho V D}{\mu} \tag{1.16}$$

Neglecting pressure gradient and Eckert number, the dimensionless momentum equation (1.22) and energy equation (1.23) assume the same form, with the exception of Pr.When Pr is close to unity the solutions of the two equations are similar, and also boundary layer thicknesses of velocity and temperature are almost the same.
But in liquid metals $Pr << 1$, so thermal diffusivity is predominant with respect to momentum diffusivity, and the thermal boundary layer has a bigger thickness (Figure 1.6).
The differences existing between dimensionless profile of velocity and temperature in liquid metals point out that their heat transfer mechanism differs from that of other common HTF. Another important consequence is that Reynolds Analogy cannot hold for low-Pr fluids [19][17], because momentum and energy have completely different turbulent scales.

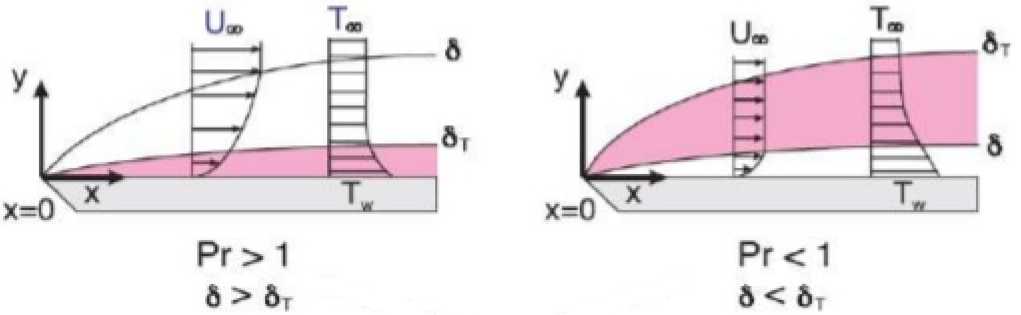


Figure 1.6: *Comparison between velocity (δ) and thermal (δ_t) boundary layer thickness as a result of different Pr numbers*

1.4 RANS approach to turbolence

The goal of a CFD simulation is to solve conservation equations in order to obtain velocity and temperature fields. These equations presents non-linear terms, so they can be solved analytically only on few cases under strong assumptions.
Typically, CFD analysis are based on numerical solution for the conservation equations. In order to obtain a numerical solution time and space must be discretized,a time step must be chosen and a mesh of the geometry is needed.
Turbulent flows are characterized by the presence of rotational flow structures, called turbulent eddies, which produce high frequency fluctuations in the properties. Time steps have to be small enough to catch those fluctuations and spacial discretisation have to be fine enough to resolve even the smallest eddies [34]. These scales of time and space are called Kolmogorov micro scales and their order of magnitude is inversely proportional to the Re number. In flows with high Reynolds an extremely fine discretisation is required: this means an enormous computational efforts.
For this reason, conservation equation are not solved in their original form, but in an averaged form (RANS: Reynolds Averaged Navier Stokes) [28]. The flow properties are separated into a mean and a fluctuating term:

$$\phi(\vec{x},t) = \overline{\phi}(\vec{x}) + \phi'(\overline{x},t) \tag{1.17}$$

$$\overline{\phi}(\vec{x}) = \frac{1}{\Delta t}\int_{t}^{t+\Delta t} \phi(\vec{x}, t) \tag{1.18}$$

The averaged conservation equations assume the form:

$$\frac{\partial \overline{U}_i}{\partial x_i} = 0 \tag{1.19}$$

$$\frac{\partial \overline{U}_i}{\partial t} + \overline{U}_j \frac{\partial \overline{U}_i}{\partial x_j} = -\frac{1}{\rho}\frac{\partial \overline{p}}{\partial x_i} + \frac{\partial}{\partial x_j}\left(\nu \frac{\partial \overline{U}_i}{\partial x_j} - \overline{u_i' u_j'}\right) \tag{1.20}$$

$$\frac{\partial \overline{T}}{\partial t} + \overline{U}_i \frac{\partial \overline{T}}{\partial x_i} = \frac{\partial}{\partial x_j}\left(\alpha \frac{\partial \overline{T}}{\partial x_j} - \overline{u_j' T}\right) \tag{1.21}$$

These equations can be solved with coarser time and space discretisation. However, new terms appears in the equations, they are Reynolds (or turbulent) stresses (eq (1.22)) and turbulent heat flux (eq. (1.23))

$$\tau_{ij}^{t} = -\rho \overline{u_i' u_j'} \tag{1.22}$$

$$q_t'' = -\rho c_p \overline{u_j' T} \tag{1.23}$$

Turbulence models are used to evaluate these terms and solve the averaged conservation equations. Many turbulence models are based on the Boussinesq hypothesis: turbulent stresses are proportional to the mean rates of deformation and the turbulent heat flux is proportional to the gradient of the mean temperature ((1.24)(1.25)). The constants of proportionality are turbulent viscosity ν_t and turbulent thermal diffusivity α_t. They are computed by turbulence model.

$$\frac{\tau_{ij}^{t}}{\rho} = \nu_t \left(\frac{\partial \overline{U}_i}{\partial x_j} + \frac{\partial \overline{U}_j}{\partial x_i}\right) - \frac{2}{3} k \delta_{ij} \tag{1.24}$$

$$\frac{q_{jt}''}{\rho c_p} = \alpha_t \frac{\partial \overline{T}}{\partial x_j} \tag{1.25}$$

As previously said, Reynolds analogy holds when momentum and energy dissipation have similar scale, so that turbulent Prandtl number is close to unity.

$$Pr_t = \frac{\nu_t}{\alpha_t} \tag{1.26}$$

In fact for common fluids, such as water or air, Pr_t is in the range of between 0.85 and 1. This is not true for liquid metals, in which it's typically greater than 1, as a consequence of the different heat transfer mechanism. In order to solve heat transfer problems, Pr_t has to be estimated form experimental data, as shown Figure 1.7.

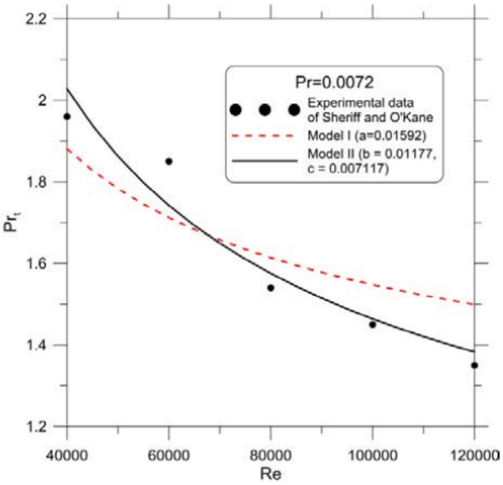


Figure 1.7: *Turbulent Prandtl number for liquid sodium as a function of Reynolds number- approximation of experimental data of Sheriff and O'Kane [17]*

1.5 Review of sodium CSP plants

History of technologies based on liquid sodium is quite limited, mainly due the problems related to safety. The first application dates back in early '80, when liquid metals were used as coolant in nuclear sector.

Regarding solar applications, one of the very first project was Central Receiver Test Facility (CRFT) in Albuquerque, New Mexico, in 1979. It was a demonstrative solar receiver (3 meters high and 1.2 width) made of three panels with 21 tubes each, with the objective to measure thermal efficiency of liquid sodium [18].

The plant was run only for few days, but the estimated efficiency in the various tests was between 90 and 96%.

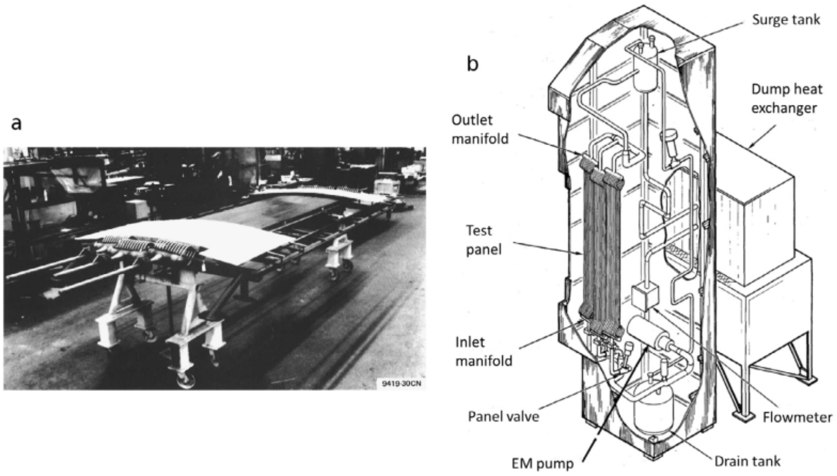


Figure 1.8: *(a) Assembled Rockwell test panel (b) Schematic drawing of the receiver and sodium test loop (Rockwell International, 1983)*

The second experience with sodium was made in 1986 at "Plataforma Solar" in Almeria, Spain. This was an international project established by IEA (International Energy

Agency) with the objective of investigating concentrating solar power generation at small power levels. Two different types of receiver were tested here, one was cavity-type and the second billboard (or external) type. The first receiver was designed to operate at 270/530°C inlet/outlet temperatures, and a peak flux inside the cavity of 0.63 MW/m^2 and total power 2.8 MW_t. The receiver performed reliably but with lower thermal efficiency than expected: the best thermal efficiency reported by was 76.0%, based on daily energy averages [8].
The billboard type receiver was designed to operate in the same range of temperatures, a peak flux of 1.4 MW/m^2 and total power 2.5 MW_t, and accumulated 880 h operation over almost 500 days under these conditions during 1983/84. In this case thermal efficiency was 91% based on daily energy averages, so it's clear that the second configuration performed better [8][9]. The billboard receiver was tested also in a second configuration, with the heliostat field refocused in order to increase the peak heat flux up to 2.5 MW/m^2, in order to exploit better the excellent heat transfer properties of sodium. In this configuration, sodium peak temperature, measured with an IR camera, reached 620°C, but unfortunately the test has been interrupted due to a sodium leakage that lead to a fire. This incident is one of the reason why liquid sodium has not been tested very often in recent years: the possible benefits in terms of efficiency and bigger temperature range are not considered enough to compensate the safety risk.

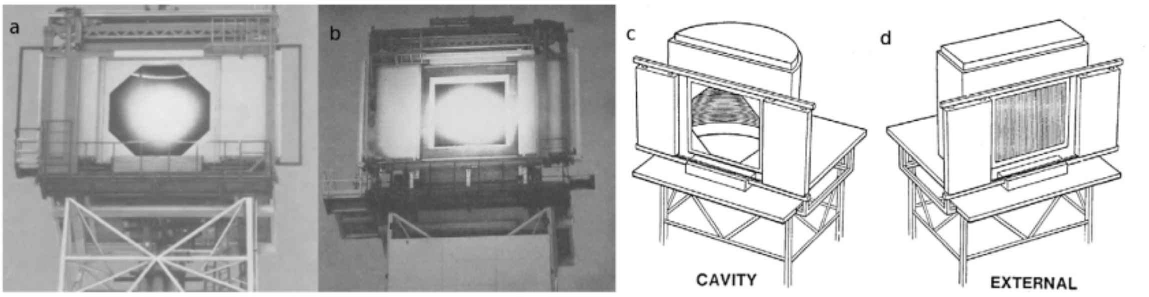


Figure 1.9: *Photos and drawings of cavity receiver (a)(c) and external receiver (b)(d) in Almeria's solar plant*

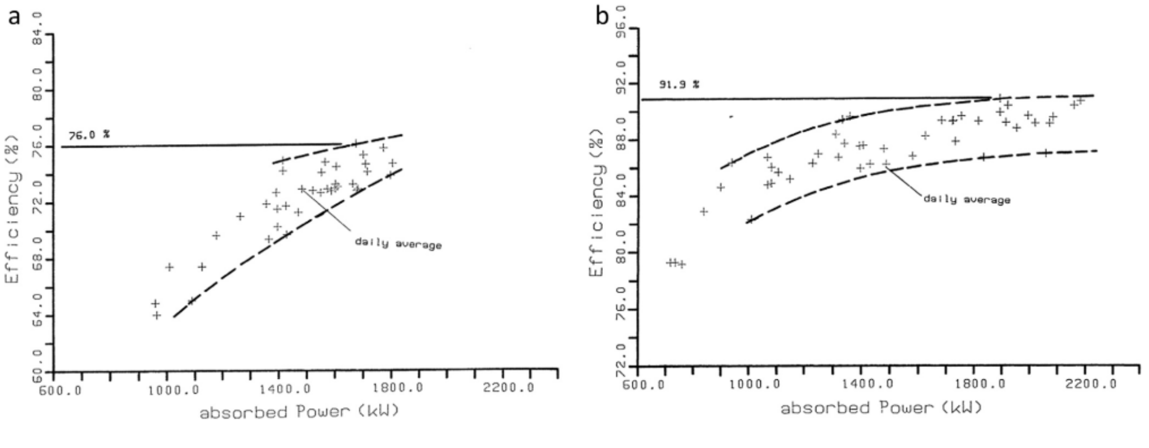


Figure 1.10: *Measured efficiency of (a) the semi-cavity receiver and (b) the external receiver versus absorbed power (IEA, 1986a). Each point is the average of one day of testing*

After many years without any significative test with sodium receivers, in 2012 a new pilot plant was commissioned by Vast Solar in New South Wales, Australia. The facility represents a single full-scale module of a multi-tower power plant.
To the author's knowledge, this is the only project of a central receiver with liquid sodium as HTF. For this reason, this plant will be taken as example to design the simulation in this work. A detailed description of the plant is provided in the next chapter.

	PSA,Spain	**PSA,Spain**	**CRTF,Usa**
Test period	1981-1983	1983-1986	1981-1982
Test time (hours)	1885	n.a.	70
Max thermal power (MW)	2.5	3.5	2.5
Measured efficiency	88%	92%	90%
Sodium temperature (C)	270-530	270-530	288-593
Storage medium	sodium	sodium	none
Storage capacity (MWh_{th})	5	5	-
Absorber area (m^2)	17	8.32	3.6

Table 1.2: *Comparison among the three sodium receivers presented in this section*

Chapter 2

Solar field

2.1 Jemalong pilot plant description

The solar field and receiver that will be simulated in this work is designed on the basis of the informations available on Jemalong solar plant.
In 2012 Vast Solar commenced its 1.2 MW Performance Validation Project which was supported by funding from the then Australian Renewable Energy Agency (ARENA). This involved a single module of Vast Solars low-cost, high-performance concentrating solar thermal (CST) power system a solar field of 700 mirrors focused on a single tower with a thermal energy receiver. ARENA provided funding for Vast Solar to construct a $6MW_{th}$ CST power generation pilot-scale plant comprising five modules and including three equivalent hours of thermal energy storage [20].

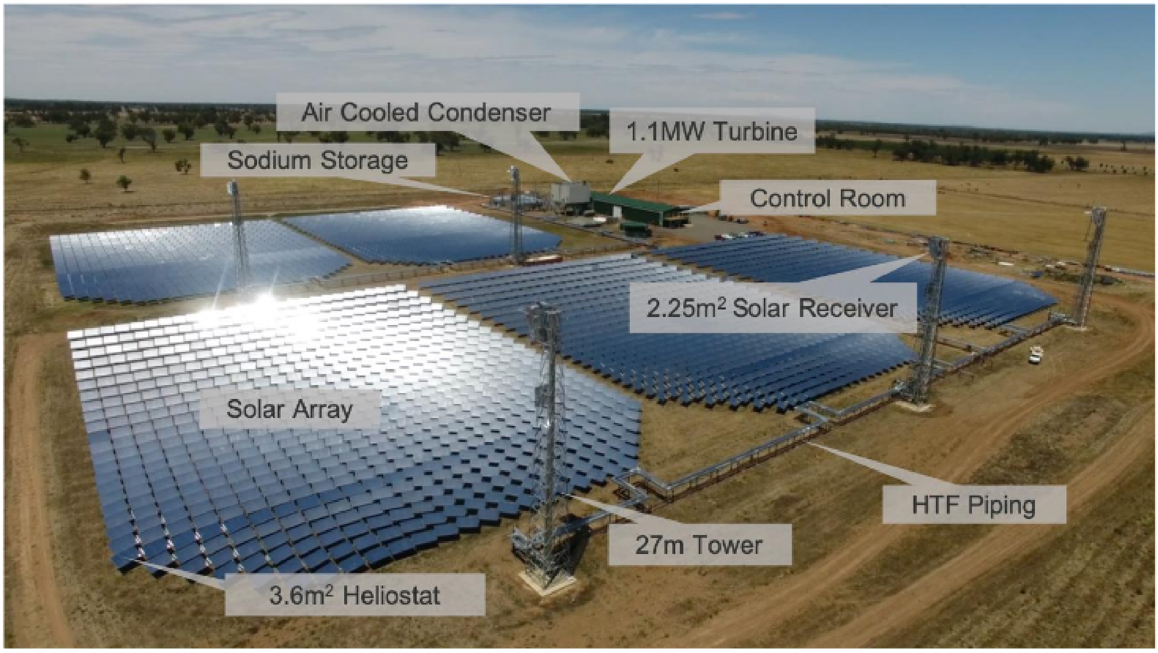


Figure 2.1: *Aerial view of Vast solar plant with key components labeled*

The aim of the project was to demonstrate the reliability and safe operation of a sodium based system, gather performance and cost data, refine operational controls and maintenance processes in order to advance the technology towards commercialization. This showcase plant was meant to reach a cost of electricity (COE) around 100$ per MWh -a level comparable to wind energy, with the added value of reliable dispatchability of power [21][22].

A single module of Vast Solar plant will be taken as a reference to design the small scale solar plant simulated in this work. The exploitation of liquid sodium, thanks to its great heat transfer properties, is what make possible to scale down the plant size while maintaining high values of outlet temperature and overall thermal efficiency. The final purpose is to perform a realistic simulation of both heliostats field and sodium receiver, in order to understand performances and potential of modular arrays and liquid sodium.

Solar Field	
Heat transfer fluid	Liquid sodium
N. of heliostats	3500
Heliostat area	3.6m^2
Tracking	2-axis
Area	15000 m^2
Tower height	27 m
Receiver inlet T	543 K
Receiver outlet T	833 K
Power block	
Gross turbine power	1.1 MW_e
Plant type	Steam Rankine
Cooling method	Dry cooling
Storage	
Type	2-tank direct
Capacity	3 hours
Fluid	Liquid sodium

Table 2.1: *Nominal data of Vast Solar pilot plant project [22]*

In Vast Solar plant, each heliostat has a simple rectangular shape with 3.6m^2 of total area and a 2-axis tracking system, making them able to adjust their position in order to follow the Sun's position in time. The tracking system is controlled by a central computer and the orientation is decided by the algorithm with respect to Sun's Azimuth and Elevation angle.

With regards to the receiver, it's an external type with a 2 m^2 area. The geometry is very simple, with a row of parallel tubes as absorbing element in which liquid sodium can flow. Two collectors (or headers) are placed at the top and bottom to distribute the fluid inside each tube, and to collect it at the outside. Thanks to the very reduced receiver dimension, the plant is able to reach an high concentration ratio ($CR = 1260$) even with a small solar field.

Pipes and headers materials are not specified in the author's available data, but it's possible to make assumptions based on manuals [4]. The chosen material is Incoloy 800HT, a steel alloy with high percentage of nickel and chromium typically used in high temperature applications such as heat exchangers.

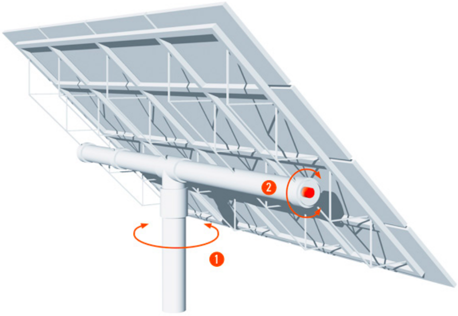


Figure 2.2: *A 2-axis tracking heliostat has two degree of freedom, one to follow Azimuth angle (1) and one for Sun's elevation (2)*

Also many other details about geometry are not specified in literature, so again some reasonable assumptions need to be done. In Table 2.2 is reported a summary of the main features.

Feature	
Height from ground (m)	27
Receiver length (m)	3.2
Receiver width (m)	0.9
Area (m2)	2.25
Number of tubes	20
Tube outer diameter (mm)	25
Tube inner diameter (mm)	21.25
Tube thickness (mm)	1.875
Tube length (m)	2.5
Headers diameter (m)	0.15

Table 2.2: *Geometrical data of the receiver used in this simulation*

In addition, a layer of Pyromark 2500 is applied on receiver external surfaces. Pyromark is a special coating, applied like a paint, used to increase thermal performances. The advantage is to increase tubes absorptance α: in Figure 2.4 it's shown that even a very thin layer of $5\mu m$ it's enough to overcome 95% of absorptance in solar radiation wavelength [23].

The plant is equipped with a 2-tanks direct storage system which is able to sustain the power block for 3 hours, with a hot tank at 560°C and the cold one at 270°C. Stored heat is then used to sustain the power block, where a steam generator feed a turbine with a gross power output of 1.1 MW_e. An interesting detail is the MACCSol air cooled condenser, a completely dry condenser specifically designed for CSP plants deployed in water-scarce locations [53].

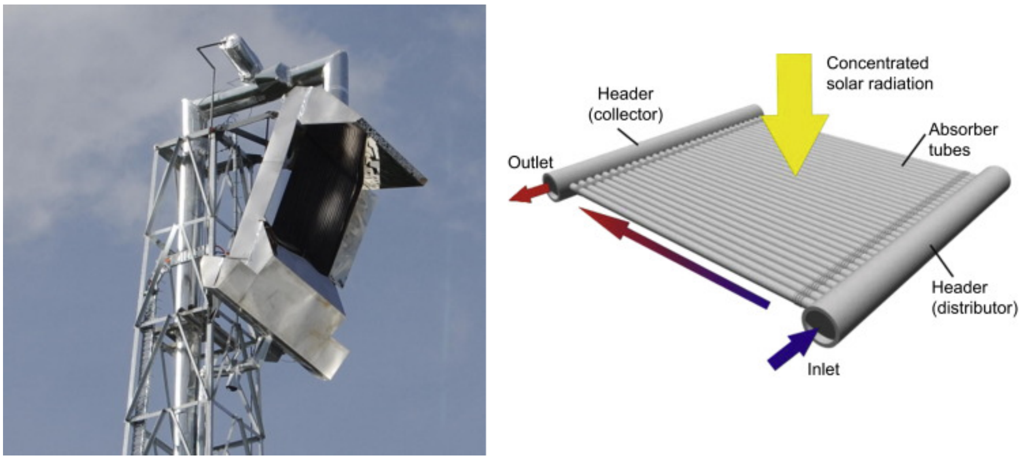


Figure 2.3: *(left) Jemalong plant receiver (right) scheme of a flat tubular receiver*

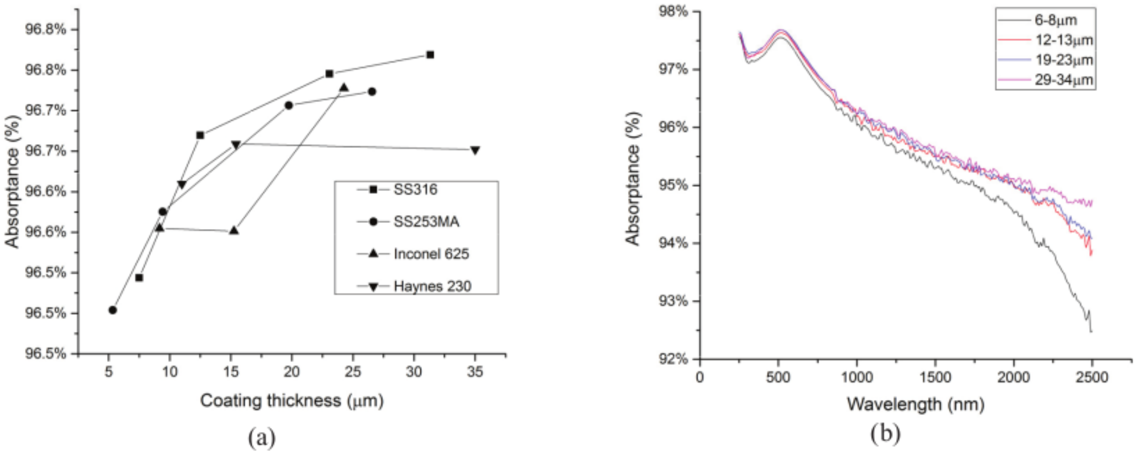


Figure 2.4: *(a) Solar-weighted absorptance for different substrates at a range of coating thicknesses in pristine state, and (b) spectral absorptance for Pyromark on SS316L, averaged from data in the thickness ranges indicated [24]*

2.2 Solar field layout design

The first step in this work is to generate the solar field that will be later used as reference for ray-tracing. This task is performed by SolarPilot, a free source software by NREL that provides layout, characterization, parametric simulation, plotting, and optimization capabilities [37]. For the scope of this work, the program is used to generate a field layout that satisfy constraints imposed by the user: each heliostat is placed in order to maximize power delivered to the receiver and reducing shadowing effects.

The very first input needed by the software are indications on plant location (the software has a database with insolation data for many places all over the world) or alternatively on DNI time series (Direct Normal Irradiation W/m2).

The plant is located in New South Wales, 300 km north of Canberra (latitude $-33.4°$ and longitude 147.7°E). The area is characterized by high values of annual average irradiation [36], as can be seen in Figure 2.5. From climate data it's possible to define design values for DNI and nominal thermal power delivered by the solar field.

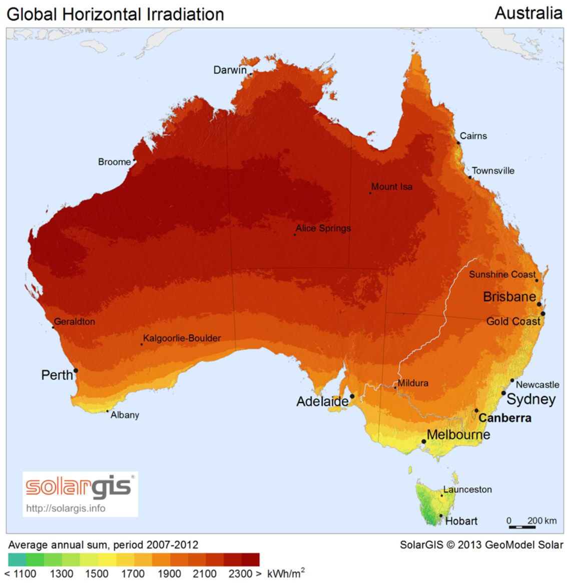


Figure 2.5: *Global horizontal irradiation in Australia*

The next input needed by SolarPilot is a definition of Sun's position in the sky, and this means to find solar altitude and solar azimuth. Solar elevation (complementary to solar Zenith angle) is defined as the angle between the horizon and the center of Sun's disk (in Figure 2.6 represented by the blue arrow) [43]. On the other hand, solar Azimuth is the angle from due north in a clockwise direction (red arrow in Figure 2.4). According to our convention, North direction is 0°, East is 90°, South is 180° and West is 270°.

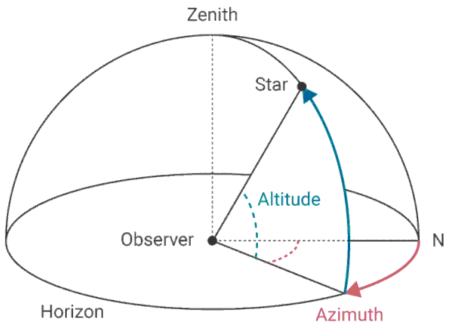


Figure 2.6: *Definition of solar elevation and azimuth angles*

For this simulation, Sun position during Summer Solstice in Australia (22 December) is taken as reference. Solar elevation at 13:00 (UTC+11) is set at 76.35° and solar azimuth at 352.25°. Nominal value for DNI is set at 950 W/m^2, a reasonable value during summer in Australia.

Once the design Sun position is defined, the software need geometric constraints to generate the field, such as the ground space available, details about heliostats and receiver. The total area available for all 5 modules is 15000 m^2, so the area for a single module is more or less 3000 m^2.

Heliostats consists in simple rectangular geometry with very high reflectivity. Details and performances are reported in Table 2.3. A major factor determining an optimum heliostat field layout is the cosine efficiency. The heliostat is positioned by the tracking mechanism so that its surface normal bisects the angle between the suns rays and a line from the heliostat to the tower. The effective reflection area of the heliostat is reduced by the cosine of one-half of this angle [4]. This may be visualized by considering heliostats at two positions in a field as shown on Figure 2.7.

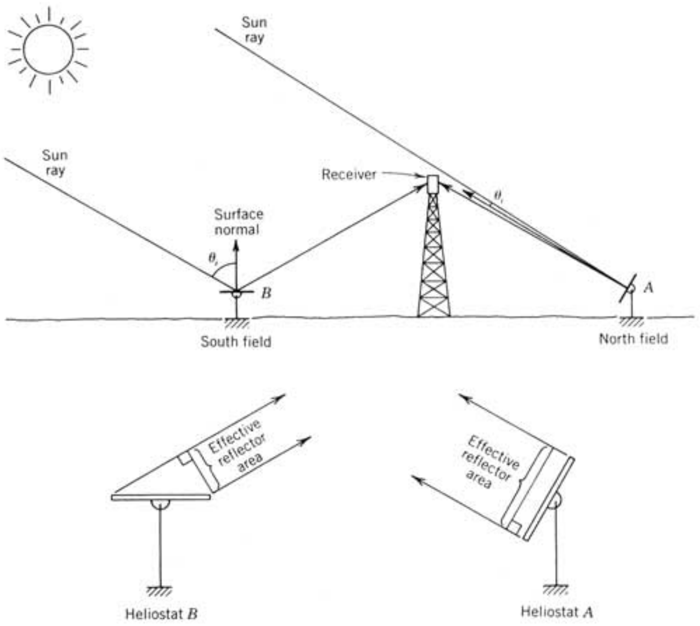


Figure 2.7: *Cosine effect for two heliostats in opposite directions from the tower.For the noontime sun condition shown, heliostat A has a much greater cosine efficiency than does heliostat B*

Note that the most efficient heliostats are located opposite the sun, and this explain the solar field layout (Figure 2.8). Since the total reflective area is very limited in a small solar field, it is important to maximize the contribution of each mirror with an high average cosine factor (values are all above 0.82). Number and spacing between mirrors are decided by the program in order to match the nominal power and to minimize blocking or shadowing effects.

Number of heliostats	700
Mirror height (m)	2.32
Mirror width (m)	1.54
Reflective surface ratio	0.97
Mirror reflectivity	0.95
Soiling factor	0.97
Average cosine efficiency	0.87
Blocking efficiency	0.92
Average Intercept factor	0.85
Average optical efficiency	0.64

Table 2.3: *Details about geometry and performances of heliostats. Values taken at design Sun position*

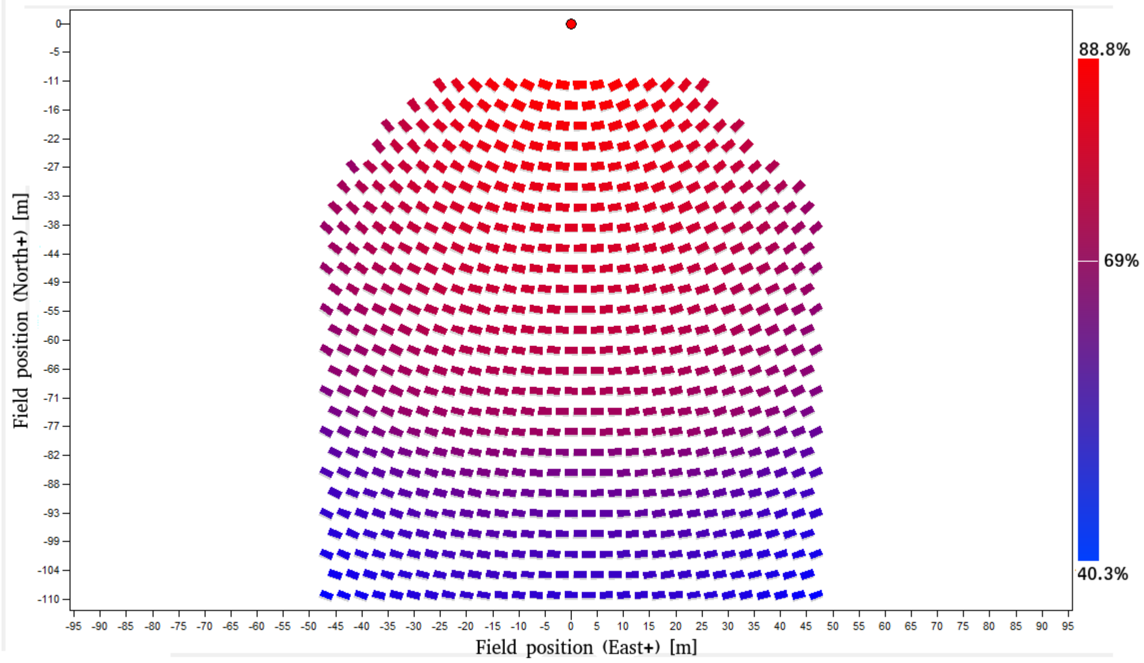


Figure 2.8: *Solar field layout. Colours indicate optical efficiency (%)*

Another extremely important factor is intercept factor, defined as the percentage of rays that intersect the absorber tube against the total of all the reflected rays. This parameter is found after the ray tracing simulation described in the following Section 2.3, but it is mentioned here to give a complete overview of heliostats characteristics. When the Sun is in design position, intercept factors go from 66% for mirrors far from the tower to 99.9 for the closest ones.

The last parameter to be discussed is optical efficiency of the field, which sum up performances of a solar field and depends on all the other factors previously mentioned. It is defined as the ratio between energy reflected to the receiver to the energy incident on heliostat aperture. Optical efficiency of heliostats is between 0.89 (for mirrors close to the tower) and 0.40 (mirrors far from the target, where cosine and intercept factors are lower), as shown by colors in Figure 2.8.

Now that Sun position, solar field and receiver are completely defined, it is possible to run a parametric simulation to see the heat flux distribution on the target.

Distribution of heat flux on the target it is expected to be non-uniform. In Figure 2.9 it's illustrated the actual flux intensity reaching the receiver: in the middle area peak flux reaches a value of almost $1.5MW/m^2$, while the average flux is $0.76MW/m^2$. In general the distribution depends also on the aiming strategy chosen by the user for heliostats. After the parametric simulation, SolarPilot generates a CSV file called "aim.csv" containing the optimal aiming points for the receiver.

These points will be useful as an input for SolTrace, the software used to perform ray tracing. The operation is described in the next section.

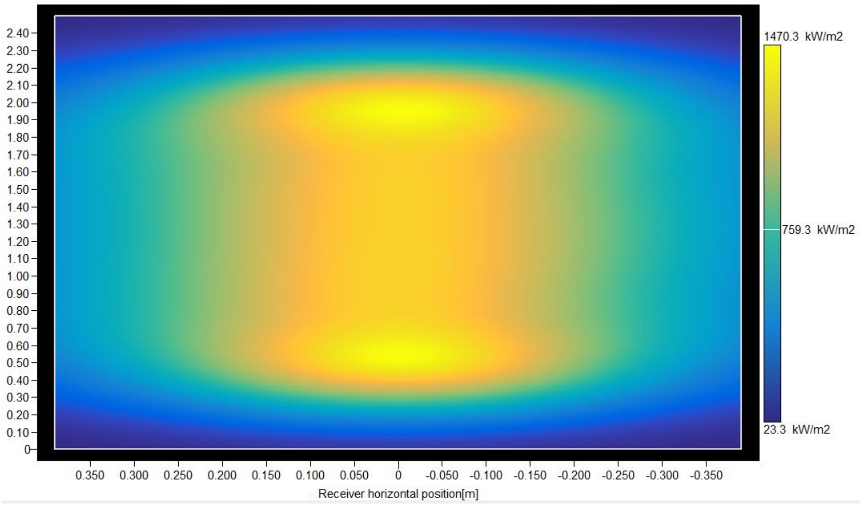


Figure 2.9: *Flux on receiver area simulated with SolarPilot*

2.3 Ray Tracing with SolTrace

Now that SolarPilot has generated the solar field layout and the aiming points for heliostats, it's possible to import everything in SolTrace and to perform an optical simulation.

Soltrace is a code that uses Monte-Carlo ray-tracing methodology [38]. The user selects a given number of rays to be traced and then each ray is traced through the system while encountering various optical interactions. Some of these interactions are probabilistic in nature (e.g., selection of sun angle from sun angular intensity distribution) while others are deterministic (e.g., calculation of ray intersection with an analytically described surface and resultant redirection).

The software needs three category of inputs:

- Sun position and shape
- optical elements definition. This include the solar field layout and optical characteristics of each heliostat (reflectivity, reflective surface ratio, soiling factor and mirrors pointing)
- receiving elements definition (geometry and absorptance)

Two characteristics completely define the sun as the light source: the angular intensity distribution of light across the suns disk, referred to as the sun shape, and the suns position. There are two options for defining the sun position. The first one is to define a particular site latitude and time (day of year and local solar hour). From this information, the sun direction is determined assuming the z-axis of the global coordinate system points due north, the y-axis points towards zenith, and the x-axis points due west.

The other option is to define a point in the global coordinate system such that a vector from this point to the global coordinate system origin defines the sun direction. In equations from (2.1) to (2.7) is described how to define global coordinates $x - y - z$ given latitude L (+north, -south), Julian day of year D, and hour of day H in local solar time [38].

$$\delta = arcsin(0.39795cos(0.98563(D - 173))) \tag{2.1}$$

$$\omega = 15(H - 12) \tag{2.2}$$

$$\alpha = arcsin(sin\delta sinL + cos\delta cos\omega cosL) \tag{2.3}$$

$$\gamma = arccos\left(\frac{sin\delta cosL - cos\delta cos\omega sinL}{cos\alpha}\right) \tag{2.4}$$

$$x = -sin\gamma cos\alpha \tag{2.5}$$

$$y = sin\alpha \tag{2.6}$$

$$z = cos\gamma cos\alpha \tag{2.7}$$

Sun position is the same as the one chosen as plant design point, so that solar radiation to the receiver can be considered symmetrical.
To complete the sun definition, the software needs a sun shape, defined as the angular distribution of normalized intensity across the sun disk. One of the most common model used to define sun shape is the one proposed by Buie [56], based on a parameter (circumusolar ratio CSR) that take into account attenuation effects at Sun's edge (Figure 2.10).

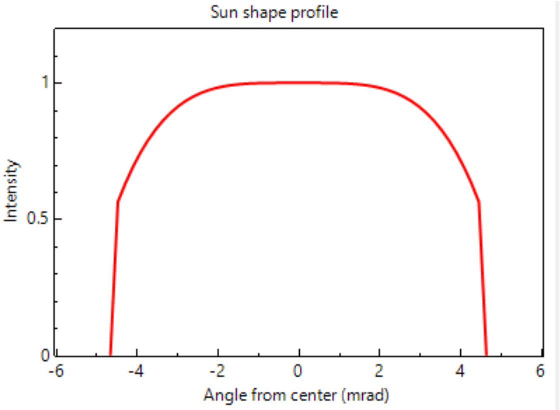


Figure 2.10: Sun shape profile according to CSR at 10%

Data on mirrors disposition and performances are imported from SolarPilot. Regarding receiving element, for ray-tracing purposes it's enough to represent it as a row of parallel tubes: 2.5 m length, 25 mm diameter and 10mm of spacing between them.

Now it's possible to start the simulation. The user has to specify the number of rays to trace (a very high number will give a more accurate result, but of course it will take more time to compute) and a DNI value that will define power-per-ray value. The simulation is run with 1 million rays and $DNI = 950W/m^2$. In Figure 2.11 is shown the receiver geometry with numbered tubes from 1 to 10 (the ones that will be simulated with OpenFOAM) and the distribution of intersection points with traced rays. Aiming strategy has been chosen in order to distribute sun rays over the entire absorbing area avoiding excessive concentrations of heat flux in some points. The path of 800 traced rays is shown in Figure 2.12.

The software provides as an output a window with detailed informations on the flux on each tube, such as average and peak flux, number of rays hit, power per ray,

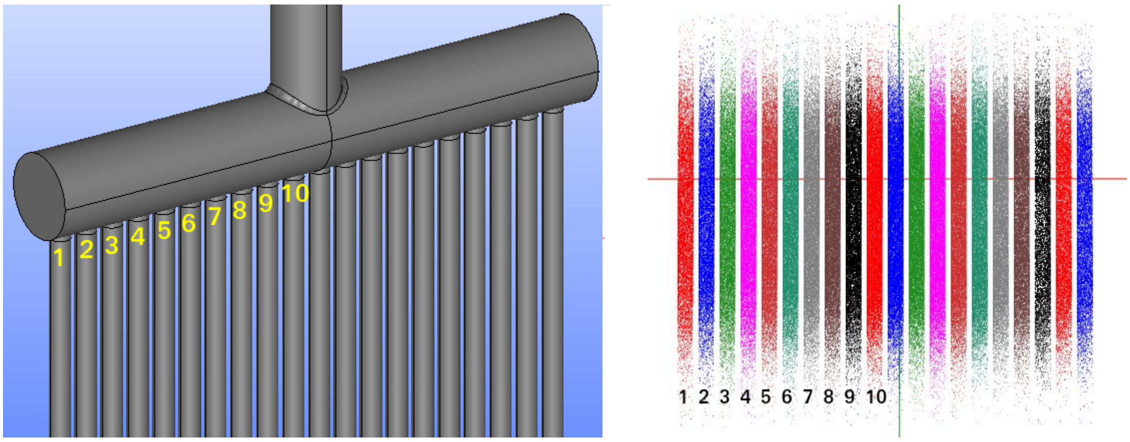


Figure 2.11: *Plot of intersection points between traced solar rays with receiver tubes. Points density is higher in the middle section of tubes and decrease at extremities*

heatmaps of flux intensity. Even though the tube is obviously cylindrical, the software represent the flux on a plane by "unrolling" the surface like a sheet of paper (Figure 2.12 shows the heat flux on each tube). The output of the software is then translated by a script into CSV files, one for each absorbing tube, containing the spatial coordinates of the points where sun rays hit the receiver and the heat flux in each of those points. Those information will be used as a heat boundary condition for CFD simulation, as described in next chapters.

However, dealing hundreds of thousands of points could be uncomfortable, so a larger discretisation is needed. That's why a proper script has been used to solve the problem. This script is run together with the simulation, and according to user specification each receiving surface is splitted into small areas (in our case, 30 steps along x *axis* and 200 along y). Heat flux values are interpolated so that a single value is associated with the central point of each small piece.

The final result is a series of tables, one for each absorbing tubes, with heat flux values given as a function of spatial coordinates x and y. These tables will be used as a boundary condition for the CFD software.

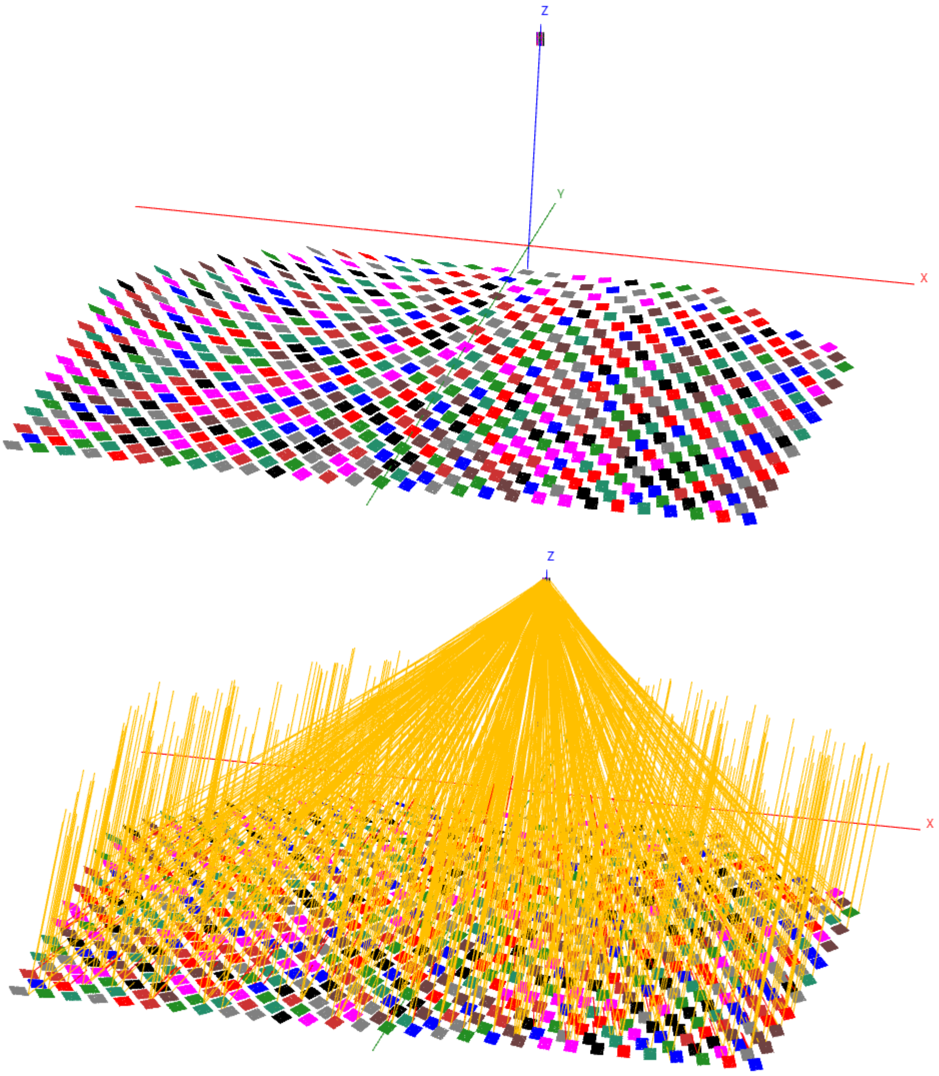


Figure 2.12: *(top) view of the solar field with all the 700 heliostats and the receiver located at 27m above ground; (bottom) path of 800 traced rays*

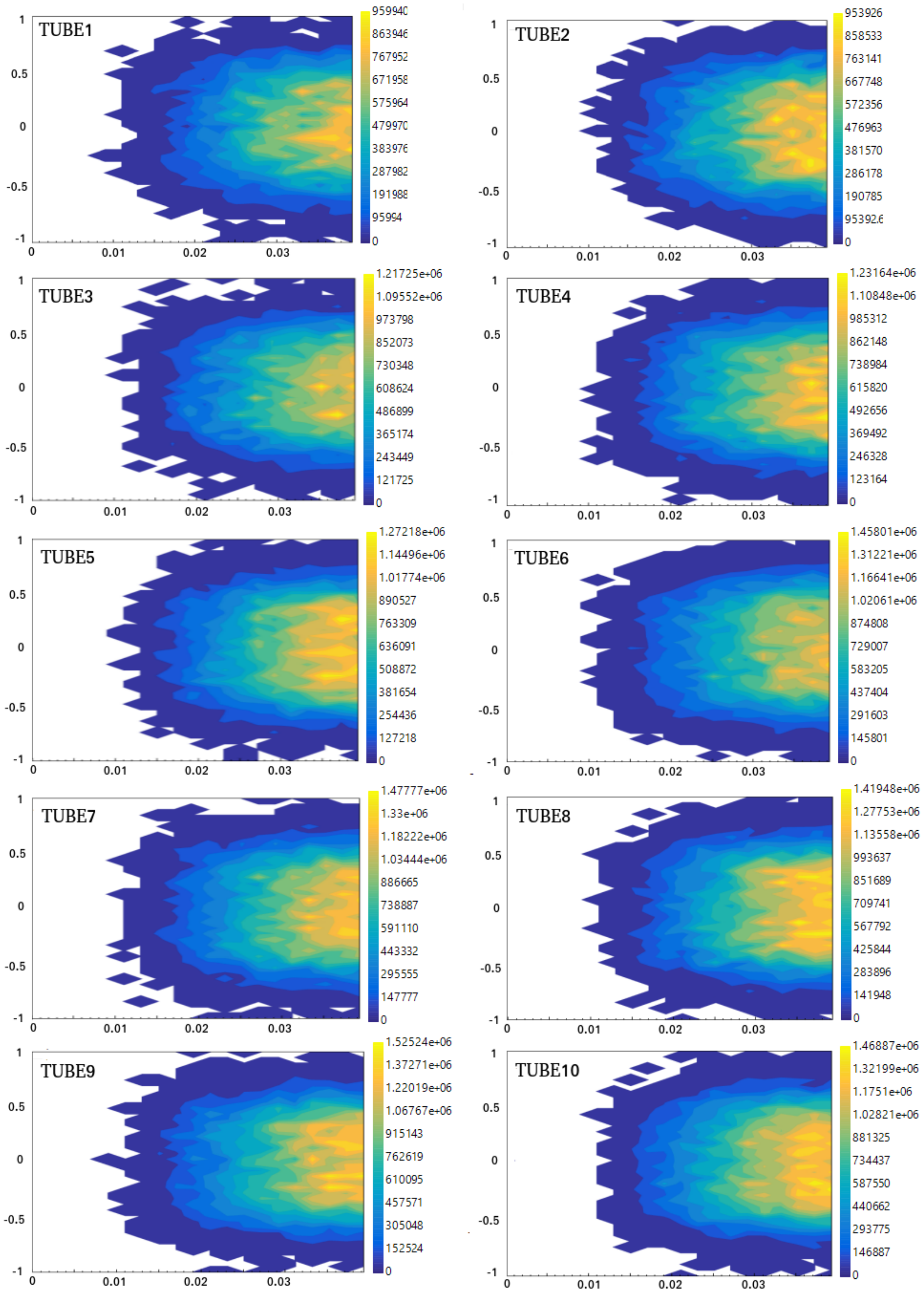


Figure 2.13: *Heat flux (W/m^2) distribution obtained with ray-tracing on each one of the 10 tubes, numbering according to Figure 2.11*

Chapter 3

Mesh generation with OpenFOAM

3.1 Software description

OpenFOAM (Open source Field Operation And Manipulation) is a C++ toolbox for the development of customized numerical solvers, and pre/post-processing utilities for the solution of continuum mechanics problems, including computational fluid-dynamics (CFD) based on the finite volume method (FVM). The code is released as free and open source software under the GNU General Public License.

More deeply, the applications fall into two categories: solvers, that are each designed to solve a specific problem in continuum mechanics; and utilities, that are designed to perform tasks that involve data manipulation. The OpenFOAM distribution contains numerous solvers and utilities covering a wide range of problems such as incompressible or compressible flow, multiphase flow, heat transfer and buoyancy-driven flows,particle-tracking flows, combustion, stress analysis of solids [40].

The basic directory structure for an OpenFOAM case, that contains the minimum set of files required to run an application, is shown in Figure 3.1.

The *System* directory is to set parameters associated with the solution procedure it-

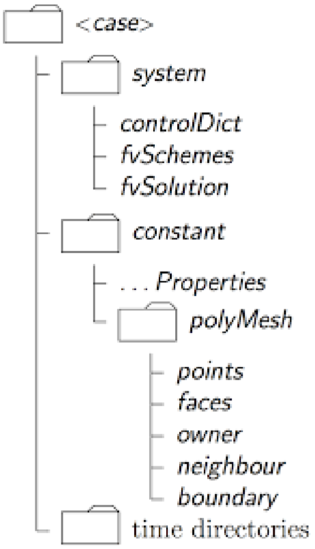


Figure 3.1: *OpenFOAM case structure*

self. It contains at least the following 3 files:*controlDict* where run control parameters are set including start/end time, time step and parameters for data output; *fvSchemes* where discretisation schemes used in the solution may be selected at run-time, and *fvSolution* where the equation solvers, tolerances and other algorithm controls are set for the run. As regards to *constant* directory, it contains a full description of the case mesh in a subdirectory polyMesh and files specifying physical properties for the application concerned.
Notice that this is a multiregion case, with different physics among regions: a solid domain (walls of tubes and headers) and a fluid domain (liquid sodium inside the receiver) interacting by means of heat exchange. To solve a multiregion case the working steps are as follows:

1. Define multiple meshes, one for each region
2. Create field variables on each mesh
3. Solve governing equation on each mesh
4. Multiregion coupling at the boundary interface between regions
5. Subiterations until a fully-coupled solution is reached

3.2 Mesh quality parameters

Mesh generation consists in dividing the physical domain into a finite number of discrete regions called control volumes or cells in which the solution is sought. This step is crucial, because accuracy of the solution depends greatly on the quality of the mesh. Three parameters that define quality meshes are [27]:

- Non-Orthogonality
- Skewness
- Aspect Ratio

As shown in Figure 3.2, mesh orthogonality is the angular deviation of the vector S (located at the face center f) from the vector d connecting the center of two cells (P and N). When this angle is too big it can affect the accuracy by introducing non-negligible

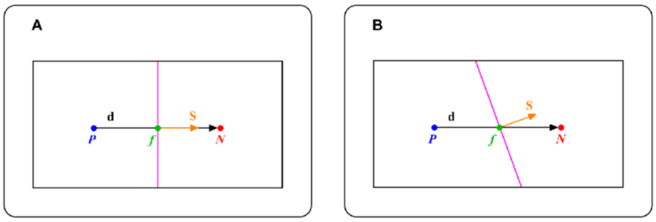


Figure 3.2: *Difference between orthogonal (A) and non-orthogonal (B) cells*

numerical diffusion. Typically, maximum values accepted are about 70 degrees, but of

course the smaller is the angle, the better is the quality.
Another concern that can cause significant numerical diffusion is skewness. According to figure 3.3, skewness arises when there is a discrepancy between the location of the face center, f, and where the center-to-center vector meets the face, f' . In general a common limit value for skewness is 4, but again a lower value is better.
As concern aspect ratio (AR), it's the ratio between the long side Δx and Δy (Figure 3.4). High values of AR results in very elongated cells, which is not necessarily a problem, it depends on the flow, turbulence model and solver. For example they can be accepted if gradients in the largest direction are small.

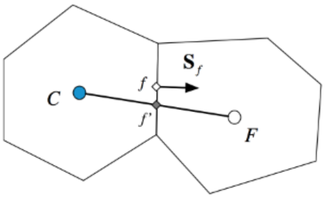


Figure 3.3: *Skewness between neighbor cells*

The last aspect to be taken into consideration about mesh quality is refinement level near the wall. According to the *Law of the wall*, the average velocity of a turbulent flow at a certain point is proportional to the logarithm of the distance from the wall, or the boundary of the fluid region:

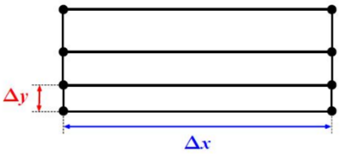


Figure 3.4: *Mesh aspect ratio*

$$u^+ = \frac{1}{k} lny^+ + C^+ \tag{3.1}$$

where u^+ is dimensionless velocity, calculated as ratio between u parallel to the wall and friction velocity u_t:

$$u_t = \left(\frac{\tau_w}{\rho}\right)^{\frac{1}{2}} \tag{3.2}$$

τ_w is wall shear stress and ρ is fluid density.
y^+ is the distance from the wall made dimensionless with friction velocity u_t and kinematic viscosity ν:

$$y^+ = \frac{yu_t}{\nu} \tag{3.3}$$

Effects of velocity and wall distance on the refinement of a mesh are well illustrated in Figure 3.5. Different near-wall treatment are employed in CFD according to the y^+ range.

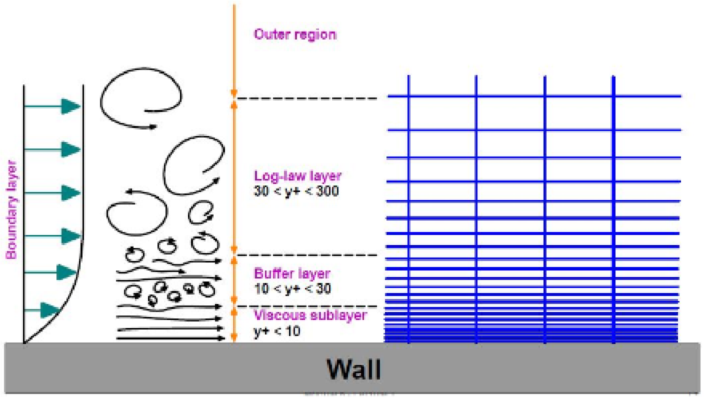


Figure 3.5: *Scheme of wall layers*

- Viscous sublayer ($y^+ \leq 6$): Turbulence models are modified to enable the viscosity-affected region to be resolved with all the mesh down to the wall, including the viscous sublayer. When using a low Reynolds turbulence model to solve the near-wall region, the first cell center must be placed in the viscous sublayer (preferably $y^+ = 1$) leading to the requirement of abundant mesh cells. Thus, substantial computational resources are required

- Log-law layer ($30 \leq y^+ \leq 300$): use of wall functions. They are equations empirically derived and used to satisfy the physics in the near wall region. The first cell center needs to be placed in the log-law region to ensure the accuracy of the results. Wall functions are used to bridge the inner region between the wall and the turbulence fully developed region. When using the wall functions approach, there is no need to resolve the boundary layer causing a significant reduction of the mesh size and the computational domain

3.3 Mesh pre-processing with SALOME

The first step to do to create a mesh is to define the geometry of the case in form of STL files. The receiver has initially been created with a CAD software, then exported as a STEP file in SALOME.
SALOME is an open-source software that provides a generic Pre- and Post-Processing platform for numerical simulation. This pre-processing step is necessary in order to generate a good quality mesh, because exporting surfaces to OpenFOAM directly from CAD could lead to the following problems:

- When the whole model is exported, the volume is closed, but the triangulation is very bad

- When the model is exported in several STL, the connected edges don't share the same points (the resultant surface is not waterproof)

- Possibility to have triangles of infinitesimal area

Once a well triangulated and watertight geometry has been obtained with SALOME, it's possible to define names for single or group of surfaces: this will make a lot easier to assign boundary conditions in OpenFOAM.
Figure 3.2 shows the complete receiver, but later only half receiver will be meshed: in fact, we take as assumption Azimuth angle between Sun and the tower equal to zero, so that the solar flux is completely symmetrical. As a consequence the whole problem can be considered symmetrical with respect to z axis, with an halving of cells number and computational time. In the end, a total of 22 STL file are exported from SALOME (inlet header, outlet header, pipes from 1 to 10). Each pipe is divided into 2 STL file, as can be seen in Figure 3.8. This is done only for an easier definition of boundary conditions: only one half of each tube is hit by solar radiation, while on the back side

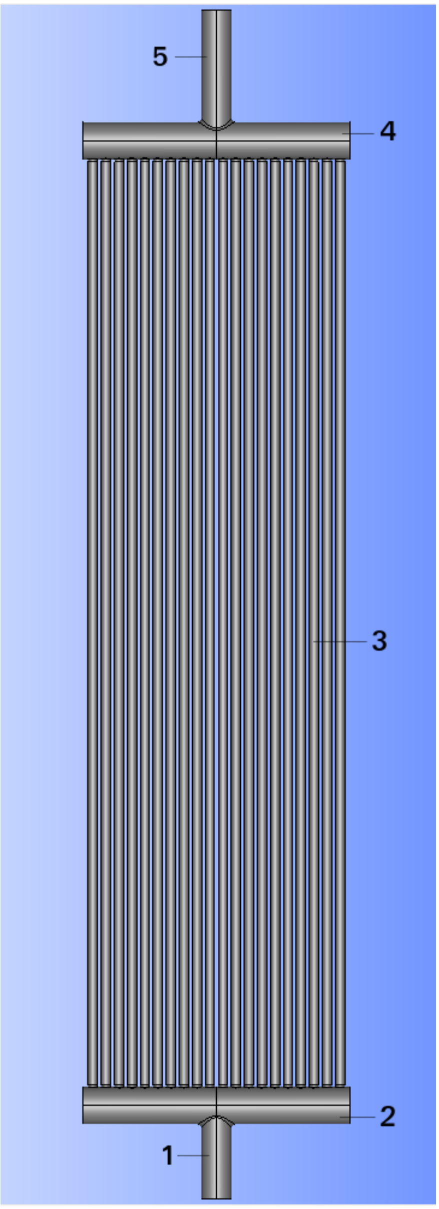


Figure 3.6: *STL model of the receiver. (1) inlet tube, (2) inlet header, (3) absorbing tubes, (4) outlet header, (5) outlet tube*

an adiabatic condition can be applied.

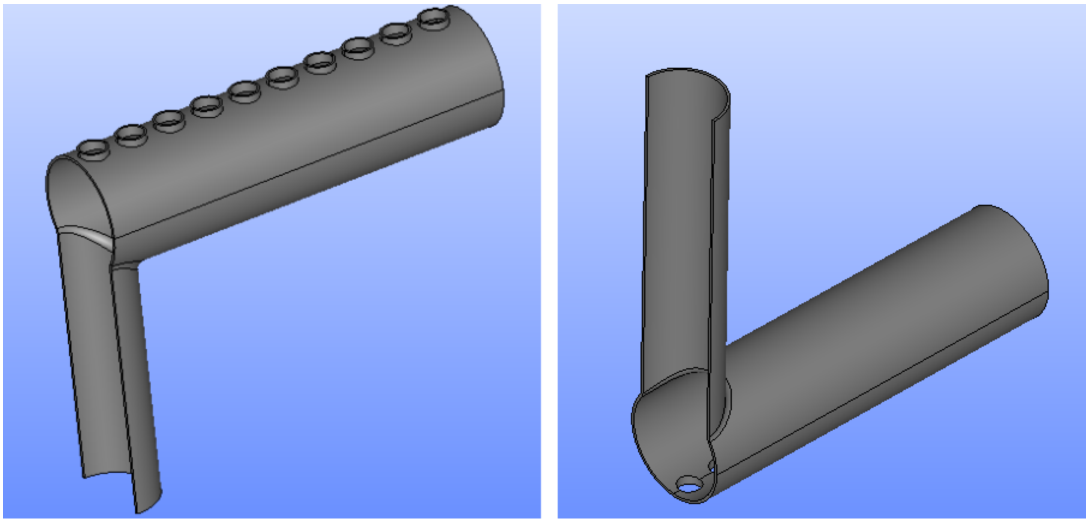


Figure 3.7: *STL geometry of the half headers: inlet headers (left) and outlet (right)*

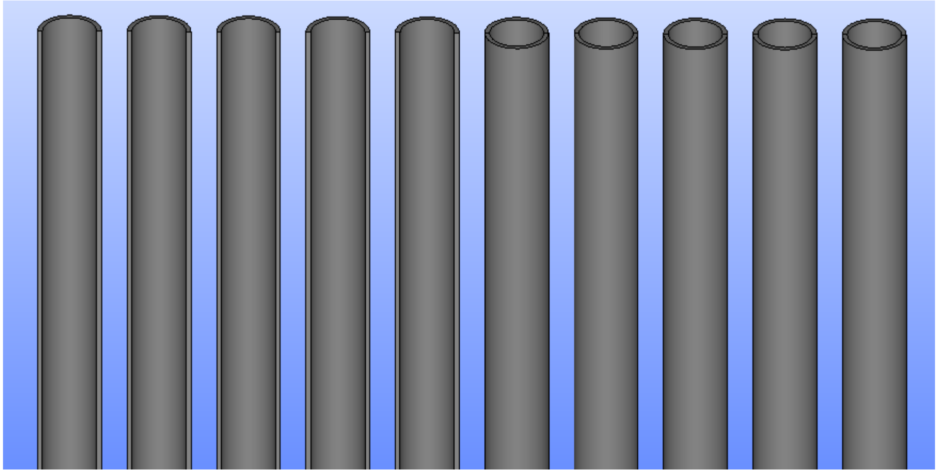


Figure 3.8: *STL geometry of absorbing tubes: in first five only the adiabatic side is shown, while the other are shown in full*

3.4 Meshing with SnappyHexMesh

The mesh generation utility used is snappyHexMesh: it generates 3-dimensional meshes containing hexahedra (hex) and split-hexahedra (split-hex) automatically from triangulated surface geometries, or tri-surfaces, in Stereolithography (STL) or Wavefront Object (OBJ) format. In order to run snappyHexMesh, the user requires the following [39]:

- One or more tri-surface files located in a *constant/triSurface* sub-directory of the case directory

- A background hex mesh which defines the extent of the computational domain and a base level mesh density. This background should contain completely all STL files, as shown in Figure 3.9. It can be created with utility *blockMesh* or exported from SALOME in UNV format

- A snappyHexMeshDict dictionary, with appropriate entries, located in the system sub-directory of the case

In practice, what this tool do is to take an already existing mesh (background mesh) and chisels it into the wanted geometry. This process is divided in three steps: castellation, snapping, layers addiction.

Castellation and refinement: cells are removed from background mesh until intersections with STL surfaces are found; only cells on the border and inside the surface are kept. Cells crossed by a boundary are then splitted into smaller ones according to the refinement level written in the dictionary file, so that a coarse version of the geometry is created

Snapping: this phase attempts to adapt the castellated mesh to conform to the input geometry by displacing vertexes towards the surface. The procedure is iterative, and the process stops when the distance between cells vertexes and surface is lower than a given tolerance

Layer addition: it's an optional stage, since the mesh output from the snapping stage can be suitable for some applications. However, irregular cells along boundary can still be present, so additional cell layers are added to produce a smooth surface.

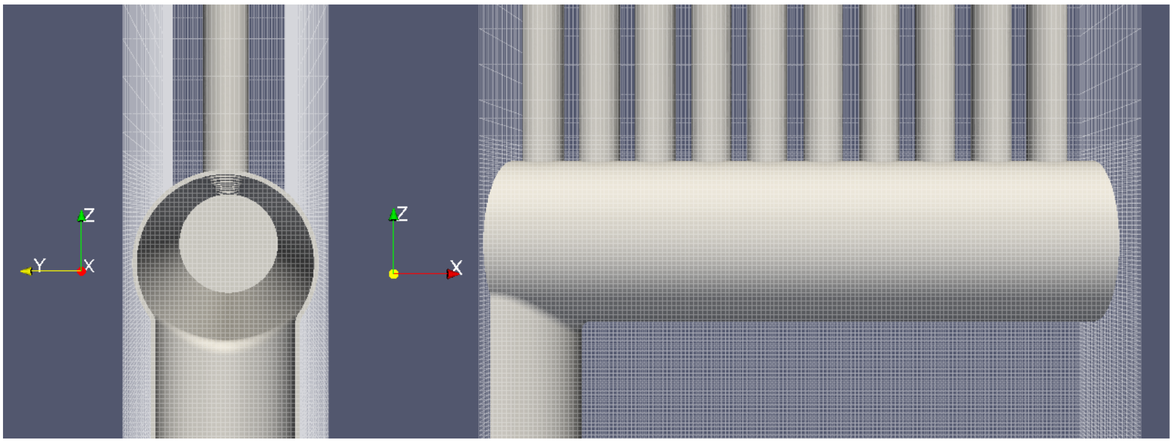
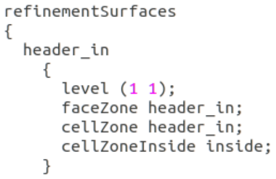


Figure 3.9: *STL geometry contained into the background mesh*

All the previous steps are controlled by the dictionary *snappyHexMeshDict*. Since it's a multiregion case, different FaceZones and cellZones have to be created: by creating a zone, a new solid or fluid domain is defined. This means that all cells contained in a certain zone will be treated by OpenFOAM as a different domain, with its own boundary conditions and governing equations. FaceZones are simply boundaries between different cellZones. Each zone of course must have a different name, for simplicity the same name of the original STL files. Another important entry is refinement level for each zone, i.e. the number of times background cells are split during castellation phase. Clearly high refinement levels generate very fine cells and high quality meshes, but also increase greatly computation time.

Notice that cells of the background mesh are very elongated. In theory, using cubes is recommended in order to obtain high-quality meshes, but the choice has been made taking into consideration the geometry of absorbing tubes, which are very long and thin. Using cubes would result into an enormous number of cells in axial direction, while for our purposes is better to have a finer mesh in radial direction.



```
refinementSurfaces
{
  header_in
    {
      level (1 1);
      faceZone header_in;
      cellZone header_in;
      cellZoneInside inside;
    }
```

Figure 3.10: *Example of a dictionary entry with the definition of CellZones,FaceZones and refinement level for the inlet header*

Of course this choice will increase max and average aspect ratio of the mesh, but it's an acceptable trade-off to limit the total computation time.

Lastly, as previously mentioned, the *addLayers* step is not mandatory, but it's recommended to create a proper boundary layer. In the case of this work it's very important to understand what happens inside the thermal boundary layer, and so a finer mesh close to walls is necessary. To add layers it's necessary to specify the patch on which cells will be created, the total thickness and number of new layers. To reduce number of cells, layers will be added only to boundaries between fluid domain and tube walls, and not at the boundaries with headers. A total of 8 layer will be added.

3.5 Running applications in parallel

Given the dimensions of the receiver and the high number of cells that will be needed to completely mesh it, performing a parallel computation is definitely a good option.

Traditionally, computer software works with *serial computation.* To solve a problem, algorithms are constructed and implemented as a serial stream of instructions. These instructions are executed on a central processing unit on one computer. Only one instruction may execute at a time, after that instruction is finished, the next one is executed.

In *parallel computation* the problem is broken into independent parts so that each physical core of a multi-core system can execute his own part of the algorithm simultaneously with the others. This results in a drastic reduction of simulation time. Running cases in parallel is particularly helpful with workstations, thanks to the high number of physical cores with respect to common computers.

To run operationally in parallel with OpenFOAM it is necessary to follow some specific steps:

1. Decompose the domain: *decomposePar* utility is used and it is based on the *decomposeParDict* dictionary located in the *System* directory

2. Distribute the jobs among the processors or computing nodes: OpenFOAM uses the public domain OpenMPI implementation of the standard Message Passing Interface (MPI). By using MPI, each processor runs a copy of the solver on a separate part of the decomposed domain

3. Reconstruct the domain: *reconstructPar* utility is used to obtain the final result

Both *snappyHexMesh* and the solver will be run in parallel.

3.6 Mesh description

In this section will be shown in details the result of meshing process and will be discussed the quality of the mesh, which is a fundamental issue in a CFD simulation.
After running *snappyHexMesh*, all background cells are splitted and used to create zones accordingly to user specification. Then, to divide each zone into proper independent domains, it's necessary to run the command *splitMeshRegions* with the option *cellZones*. The final result is composed by 23 domains, 1 fluid and 22 solids:

- domain2: fluid region of liquid sodium inside the receiver
- domain0: part of the background mesh which is external to the receiver. This part has no interest for us and will be deleted
- header_in: solid region of the inlet collector
- header_out: solid region of the outlet collector
- tubeadiab1...10: adiabatic halves of the ten tubes
- tubeheat1...10: halves of the ten tubes with an imposed heat flux

Now all domains and boundaries have been created. Moreover,*splitMeshRegions* generates also several sub-folders inside *0*,*constant* and *system* to store information about each domain. As previously said, mesh and boundaries information are stored inside *constant* folder.
In OpenFOAM, each boundary has a base type that define its characteristic. In this work, three types of boundaries will be used:

Patch: the most generic type of patch that contains no geometric or topological information about the mesh. It's typically used for an inlet or outlet

Wall: there are instances where a patch that coincides with a wall needs to be identifiable as such, particularly where specialist modelling is applied at wall boundaries. A good example is wall turbulence modelling where a wall must be specified with a wall patch type, so that the distance from the wall of the cell centers next to the wall are stored as part of the patch

SymmetryPlane: it just define a symmetry condition on a planar geometry, mirroring quantities

Processor: if a code is being run in parallel, on a number of processors, then the mesh must be divided up so that each processor computes on roughly the same number of cells. The boundaries between the different parts of the mesh are called *processor* boundaries

All patch types are *wall*, with three exception: inlet and outlet faces in fluid domain, characterized by the generic type *patch*, and the faces cut by the symmetry plane which have *symmetryPlane* type. Main patches are shown in Figure 3.11.

As can be seen in Figure 3.12 the cells are elongated in the axial direction (z direction) to reduce the total number of cells and computational time. This of course will result in an higher aspect ratio of the mesh, which is not necessarily a negative issue. In

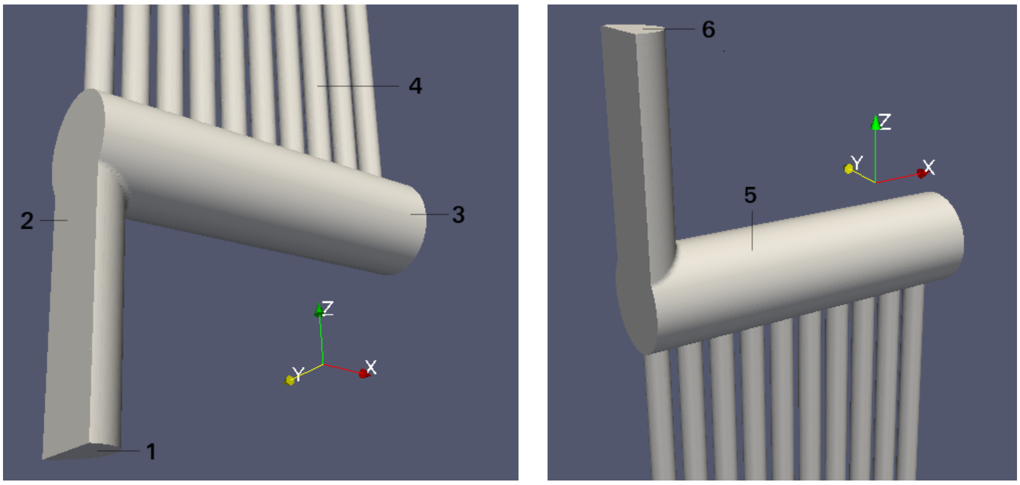


Figure 3.11: *Simulated receiver: (1) inlet patch; (2) symmetry plane; (3) inlet header; (4) absorbing tubes; (5) outlet header; (6) outlet patch*

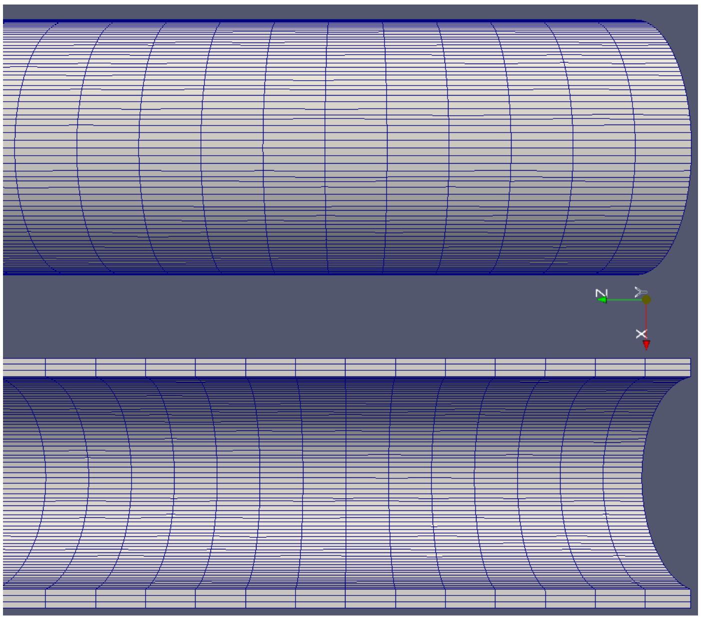


Figure 3.12: *Mesh of two tubes. The top one is a full pipe, while the bottom is only the adiabatic half*

general, if the cells are elongated in the same direction of the flow they will not have a bad impact on results. Moreover, in this work is more interesting to investigate the propagation of heat and momentum in radial direction. That is why radial direction is much more refined, in particular in the near-wall region where very thin cell layers ha been added (Figure 3.13).

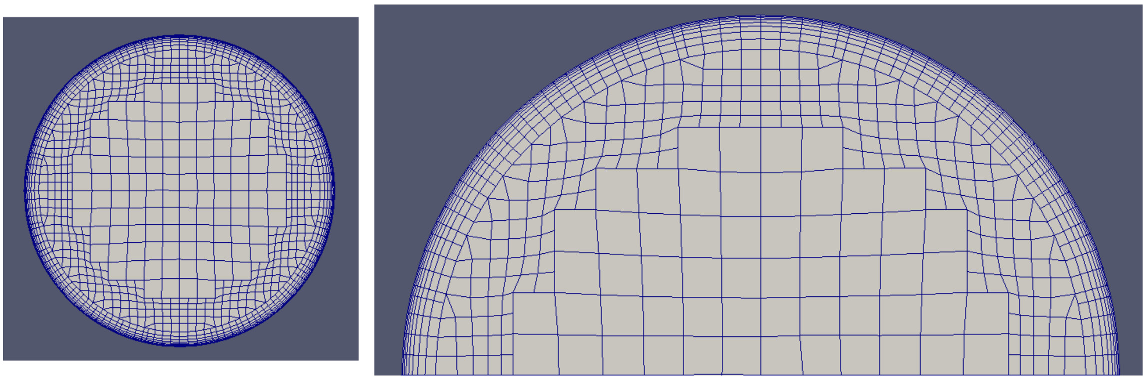


Figure 3.13: *Cross section of fluid domain mesh. On the right it's possible to see more clearly the boundary layer made with added cells*

	domain2	**header_ in**	**header_ out**	**tubeadiab**	**tubeheat**
Number of cells	9261373	201052	217060	91551	91551
max nonOrthogonality	69.599	55.661	56.740	63.340	63.340
max skewness	4.021	3.689	3.776	3.785	3.785
Max aspect ratio	20.781	12.022	9.468	19.994	19.994

Table 3.1: *Mesh details and quality parameters*

Total mesh is made of 11510505 cells.

Chapter 4

OpenFOAM case description

4.1 Boundary conditions

Now that the mesh is ready, it's necessary to define the physics of the case by imposing conditions at each boundary.
After defining the patch type, it's possible to set the actual boundary condition [33]. Here the list of conditions used:

- fixedValue: supplies a fixed value constraint to the desired field
- fixedGradient: supplies a fixed gradient constraint to the desired field
- zeroGradient: applies a zero-gradient condition from the patch internal field onto the patch faces
- fixedFluxPressure: sets the pressure gradient to the provided value such that the flux on the boundary is that specified by the velocity boundary condition
- flowRateInletVelocity: creates a uniform velocity field normal to the patch adjusted to match the specified flow rate. It works with both volumetric and mass flow rate.
- inletOutlet: provides a generic outflow condition, with specified inflow for the case of return flow
- noSlip: fixes the velocity to zero at walls
- turbulentIntensityKineticEnergyInlet: provides a turbulent kinetic energy condition,based on user-supplied turbulence intensity, defined as a fraction of the mean velocity
- alphatJayatillekeWallFunction: provides a thermal wall function for turbulent thermal diffusivity (alphat) based on the Jayatilleke model
- kqRWallFunction: provides a suitable condition for turbulence *k, q, R* fields for the case of high Reynolds number flow using wall functions

- nutkWallFunction: provides a turbulent kinematic viscosity condition when using wall functions, based on turbulence kinetic energy
- omegaWallFunction: provides a wall function constraint on turbulnce specific dissipation, omega
- compressible::turbulentTemperatureCoupledBaffleMixed: mixed boundary condition for temperature, to be used for heat-transfer on back-to-back baffles
- calculated: this boundary condition is not designed to be evaluated; it is assumed that the value is assigned via field assignment

Now let's look more in detail at boundary conditions. They are located in subfolders inside *0* directory, one for each region.

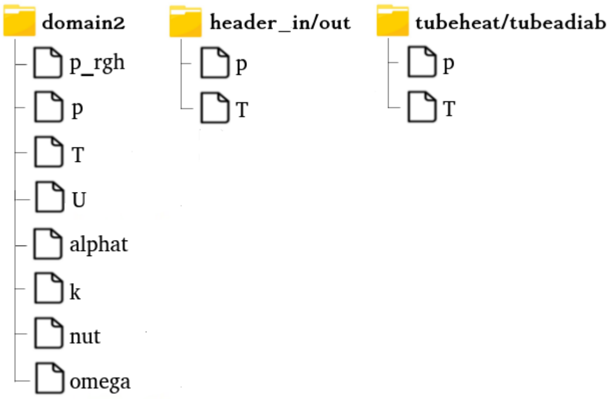


Figure 4.1: *File structure in 0 folder*

VELOCITY AND PRESSURE

Let's start with condition on velocity and pressure. At the *inlet* patch of fluid domain is set a fixed volumetric flow rate of $0.00394 m^3/s$ (corresponding to $3.272 kg/s$); of course, since only half receiver is simulated, the volumetric flow rate will be $0.00197 m^3/s$. Mass flow rate used in real Jeamalong pilot plant is not specified in the available data, so it has been found with a rough estimation based on other available informations: each module is declared to work with $1.2 MW_{th}$, and a nominal ΔT of sodium between inlet and outlet of 290 K. So it's possible to write:

$$\dot{Q} = \dot{m}_{Na} \overline{c_p} \Delta T \tag{4.1}$$

where $\overline{c_p}$ is simply the average between specific heat capacity of sodium at declared temperature of inlet/outlet. Obviously this value will differ from the real one used in the plant, but what's important now is to a reasonable inlet condition for our receiver. In general it's a good choice to not specify a fixed value for both velocity and pressure on the same patch, so at the inlet *fixedFluxPressure* is used for pressure, which is basically a fixed gradient condition.

At the outlet there is an opposite condition: velocity has the generic outflow condition *inletOutlet*, while pressure has to be set at a fixed value. The actual value has no importance, due to incompressibility assumption. All other patches are boundary walls between fluid and solid regions, with again *fixedFluxPressure* for pressure and no-slip condition for velocity. Notice that there are two pressure dictionaries in the folder, p and p_{rgh}. That's because the multiregion solver use p_{rgh}, which is the static pressure minus hydraulic pressure:

$$p_{rgh} = p - \rho g H \tag{4.2}$$

where p is static pressure, ρ density, g gravity acceleration and H is an arbitrary height. p file is simply used to calculate again the static pressure at the end, but it's not necessary to the solver.
Regarding pressure in solid domains, it's clear that such conditions would have no physical meaning. The solver needs anyway a dummy file to work, so a generic *calculated* condition will be applied to all patches.

TEMPERATURE
Liquid sodium enters the system with an initial temperature of 543K, so a *fixedValue* condition at the inlet. At the outlet a simple *inletOutlet* condition. All the boundaries with solid regions exchange heat, so the *wall* patches all have the *compressible::turbulentTemperatureCoupledBaffleMixed* condition. This condition has to be placed also on the corresponding patch on the solid domain. Regarding condition of imposed heat flux on tube external surfaces, it will be explained in details in the following section, since it's the most complex and troublesome condition. Lastly, external patches of *tubeadiab* have a *zeroGradient* condition.

TURBULENCE KINETIC ENERGY (TKE)
Generally, the TKE is defined to be half the sum of the variances (square of standard deviations) of the velocity components:

$$k = \frac{1}{2}\left(\overline{(u')^2} + \overline{(v')^2} + \overline{(w')^2}\right) \tag{4.3}$$

where the turbulent velocity component is the difference between the instantaneous and the average velocity $u' = u(t) - \overline{u}$.
At the inlet we need to specify an initial value and a turbulence intensity. The initial guess is found with the formula:

$$k = \frac{3}{2}(I|\overline{u}|)^2 \tag{4.4}$$

where $\overline{u}$ is the average velocity on *inlet* patch and I is the turbulence intensity, estimated with the expression

$$I = 0.16 Re^{-\frac{1}{8}} \tag{4.5}$$

For turbulent simulation I typically assume a value between 0.03 and 0.05. At the outlet there is a simple *zeroGradient* condition, while at walls with headers and tubes *kqRWallFunction* is used.

TURBULENCE DISSIPATION RATE
The specific turbulence dissipation, ω is the rate at which turbulence kinetic energy is converted into thermal internal energy per unit volume and time. It has been defined using the following formula, using turbulent kinetic energy (k) and length scale (l_h):

$$\omega = \frac{\sqrt{k}}{l_h} \tag{4.6}$$

Length scale is a physical quantity describing the size of the large energy-containing eddies in a turbulent flow. In pipe flows it can be estimated from the hydraulic diameter, with the simple expression: $l_h = 0.09 d_h$ Similarly to TKE, wall functions are used on all wall patches, with *omegaWallFunction*

TURBULENT THERMAL DIFFUSIVITY
This parameter express thermal diffusivity in condition of turbulent flow ($\alpha_t\ m^2/s$). The condition at wall is *alphatJayatillekeWallFunction*, a wall function that estimates α_t from turbulent Prandtl number (Pr_t) and fluid thermal conductivity k.
Pr_t is the ratio between the momentum eddy diffusivity and the heat transfer eddy diffusivity, and can assume values from 0.7 to 0.9 depending on the Prandtl number for the largest part of fluids. However, liquid metals represent an exception, having Pr_t greater than 1. For this simulation, the chosen value is 1.9.
Conditions at inlet and outlet are not important, since no heat transfer occurs in those boundaries, so a generic *calculated* is set, with a random value as a placeholder.

TURBULENT VISCOSITY
Turbulent (or eddy) viscosity is the turbulent transfer of momentum by eddies giving rise to an internal fluid friction, in a manner analogous to the action of molecular viscosity in laminar flow, but taking place on a much larger scale.
Boundary condition are similar to the previous ones, with the wall function *nutkWallFunction* in all the patches but *inlet* and *outlet*.

4.2 Heat flux boundary conditions

Conditions in solid region, in particular on the external side of the tubes, are far more complex with respect to the ones in fluid domain, and that's why they deserve a deeper discussion.
Solid regions require only two dictionaries in each subfolder: one for temperature and the other for pressure (this one is just a dummy file required by the software, it has no real purpose). So the only real boundary condition to set is the one on temperature.
As already mentioned, the heat flux on the receiver tubes is non-uniform, with a sort of bell-shaped distribution, as shown in Figure 4.2
In previous sections is described how to obtain tables with heat flux values as a func-

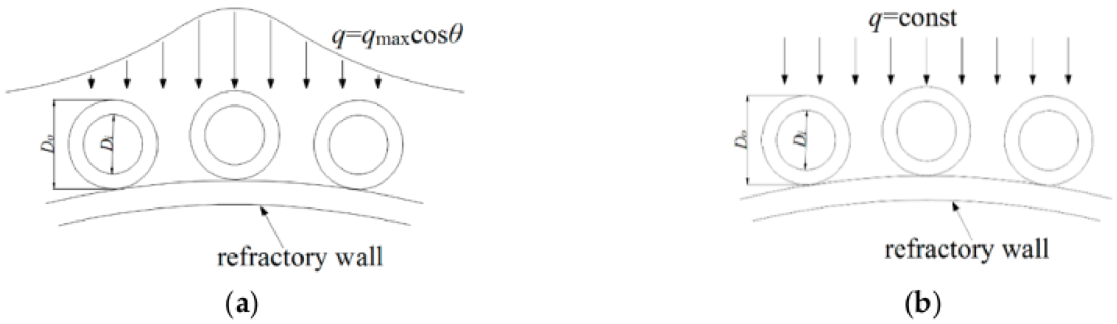


Figure 4.2: *Difference in distribution between non-uniform heat flux (a) and uniform (b)*

tion of spacial coordinates, but the problem is how to read those tables in OpenFOAM. In fact, in the standard code there is no function able to set mapped gradients of temperature on a patch, but it can be done with the help of *swak4Foam* and its *groovyBC* *Swak4Foam* is an additional library for OpenFOAM that allows to use particular boundary conditions, called *groovyBC*, in which it's possible to define new variables and functions that are calculated at every internal iteration.
The groovy function we need is called *lookuptables2D* [35], and it can read a table of values dependent on two variables: in our case it will read the flux intensity dependent on X and Y coordinates. The tables has to be provided in a simple text file with the following syntax:

$(y_1 \; ((x_1 \;\; val_{11})(x_2 \;\; val_{21})...(x_n \;\; val_{n1})))$
$(y_2 \; ((x_1 \;\; val_{12})(x_2 \;\; val_{22})...(x_n \;\; val_{n2})))$
...
$(y_m \; ((x_1 \;\; val_{1m})(x_2 \;\; val_{2m})...(x_n \;\; val_{nm})))$

Where of course val_{nm} is the value of heat flux in the point with generic coordinates (x_n, y_m). OpenFOAM is now able to read the values and locate those points, and they can be used to define a *gradientExpression* inside the *groovyBC*.
The expression we want to write is:

$$\lambda \frac{\partial T}{\partial n} = \alpha q''(x, r, \theta) \tag{4.7}$$

Where α (absorbtivity) is set to 0.95.
However, this not a realistic heat flux condition, because it doesn't take into account thermal losses due convention and radiation.
Convective losses are expressed in the following way:

$$\dot{q}_{conv} = h_{conv}(T_\infty - T_s) \tag{4.8}$$

h_{conv} is convective heat transfer coefficient between the external surface of tubes and environment, assumed at $T_\infty = 300K$. This coefficient strongly depends on wind speed, and clearly it is not easy to estimate with precision. In most of the year, wind speed is below 2 m/s, with peak of 2.4 during hottest months. Using 2 m/s is a good choice.

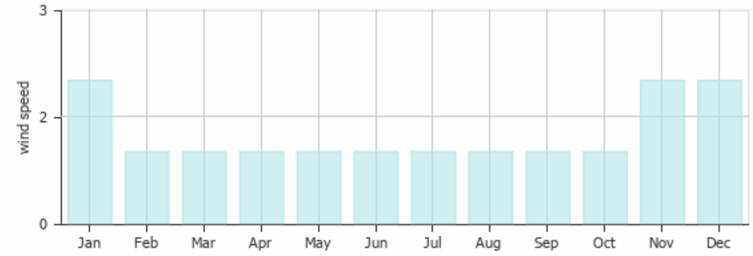


Figure 4.3: *Monthly wind speed (m/s) in Canberra, measured with standard anemometer at 10m from ground level*

Knowing wind speed, it is possible to use empirical correlations to estimate h_{conv}.
The average laminar Nusselt number Nu_{lam} is calculated by equation [25]:

$$Nu_{lam} = 0.664\sqrt{Re}\sqrt[3]{Pr} \tag{4.9}$$

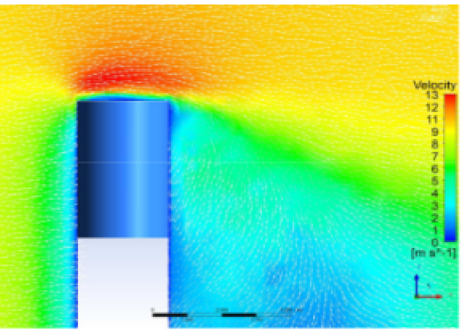
for $Re < 10^5$.
In case of turbulent flow ($10^5 < Re < 10^7$)The average turbulent Nusselt number is calculated by equation:

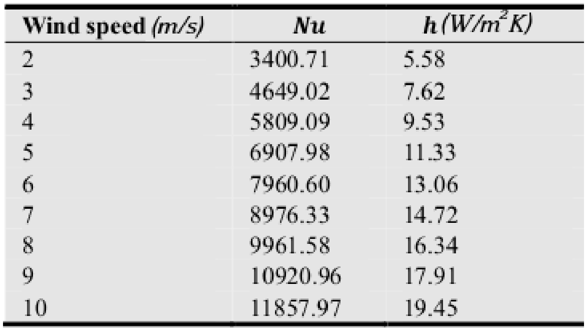
$$Nu_{turb} = (0.037Re^{0.8}Pr)/\left(1 + 2.443Re^{-0.1}\left(Pr^{-\frac{2}{3}} - 1\right)\right) \tag{4.10}$$

The convection heat transfer coefficient can be calculated from Nu by:

$$h_{conv} = \frac{\lambda Nu}{l_c} \tag{4.11}$$

In the table in Figure 4.4 are reported Nusselt numbers and heat transfer coefficients at different wind speed. Being the average wind speed in Canberra slightly lower than $2m/s$, it's been decided to take $5W/m^2K$ as convective heat transfer coefficient.
Regarding radiation losses, they can be expressed as:





Wind speed (m/s)	Nu	h (W/m²K)
2	3400.71	5.58
3	4649.02	7.62
4	5809.09	9.53
5	6907.98	11.33
6	7960.60	13.06
7	8976.33	14.72
8	9961.58	16.34
9	10920.96	17.91
10	11857.97	19.45

Figure 4.4: *Convective heat transfer coefficient as a function of wind speed (cylindrical geometry) [25]*

$$\dot{q}_{rad} = h_{rad}(T_{sky} - T_s) \tag{4.12}$$

h_{rad} is linearized radiation heat transfer coefficient, defined as:

$$h_{rad} = \sigma\epsilon(T_s^{*2} + T_{sky}^2)(T_s^* + T_{sky}) \tag{4.13}$$

σ is Stefan-Boltzmann constant ($5.67\ 10^{-8}Wm^{-2}K^{-4}$), while ϵ is emissivity. A research of 2014 studied the impact of emissivity factor on radiative losses, considering different ϵ values between 0.2 and 0.8 [54]. In order to keep the most conservative approach possible, a value of 0.8 has been used in this simulation.
T_{sky} is temperature of the sky, set at $10°C$, while T_s^* is the external wall temperature of each tube evaluated at previous time step. To evaluate the previous time step temperature will require an additional procedure: in fact, OpenFOAM deletes all values at each time step. Again, *groovyBC* has interesting features to solve this problem: the declaration of a *StoredVariable*.
StoredVariable is a particular type of variable since it's now rewritten, making possible to use values of old time steps in the calculation. In Figure 4.4 it's shown how to set the variable *StoredT*, with a arbitrary initial value. At each time step, the content of *StoredT* is copied in a normal variable, *oldT*, which is then used in the equation. Finally, the stored variable is rewritten with the actual value of temperature to be used

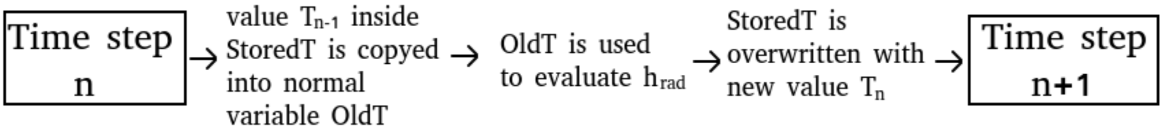

Time step n
value T_{n-1} inside StoredT is copyed into normal variable OldT
OldT is used to evaluate h_{rad}
StoredT is overwritten with new value T_n
Time step n+1

Figure 4.5: *Iterative procedure to evaluate radiative heat transfer coefficient*

in the next iteration.

Now it's possible to write a complete energy balance that takes into account all losses at the tube surface:

$$\lambda\frac{\partial T}{\partial n} = \alpha q''(x,r,\theta) + h_{conv}(T_\infty - T_s) + h_{rad}(T_{sky} - T_s) \tag{4.14}$$

Since *groovyBC* needs an expression for gradient, we divide everything by conductivity and obtain the final condition:

$$\frac{\partial T}{\partial n} = \frac{\alpha}{\lambda}q''(x,r,\theta) + \frac{h_{conv}}{\lambda}(T_\infty - T_s) + \frac{h_{rad}}{\lambda}(T_{sky} - T_s) \tag{4.15}$$

This gradient is the net heat flux entering the system, because the losses are already considered and removed from the balance. Notice also that radiative interactions among tubes are not taken into account.

4.3 Constant folder

As previously said, the *constant* folder contains informations about mesh and physical properties of the case. The first object inside *constant* is the *triSurface* folder, containing STL files from which mesh has been generated. Then, a *regionProperties* file: it simply tells the solver which domain is solid and which is fluid.

All the other folders are created during mesh splitting, each one containing the following files:

- polyMesh: this folder contains informations about the mesh such as faces, patch types, boundaries...
- thermophysicalProperties: dictionary containing models and data about energy, heat and physical properties
- radiationProperties: dictionary with informations on radiation models, view factors and radiative properties of the surfaces. In our case the radiation model is not used, but still this file is required by the solver

Then in folder *domain2* there are two additional dictionaries, necessary to describe a fluid behavior:

- turbolenceProperties: this dictionary is read by any solver that includes turbulence modelling. Within that file is the *simulationType* keyword that controls the type of turbulence modelling to be used. To be included only in fluid domains.

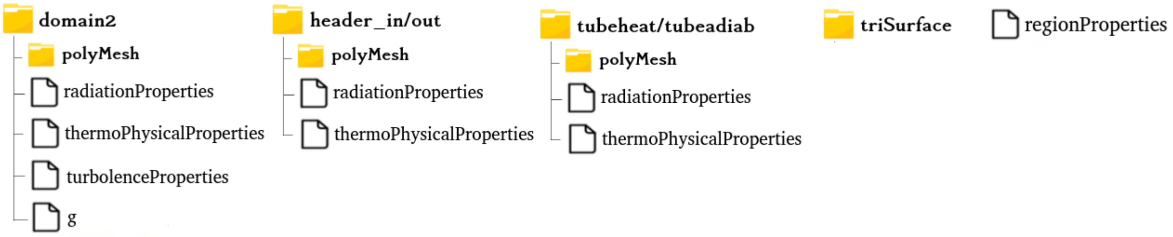


Figure 4.6: *File structure of Constant folder*

- g: dictionary with information on gravity field. Gravity acceleration is set at $-9.81m/s$ along z-axis

Two of the previous dictionaries deserve a deeper discussion: one of them is for sure *thermophysicalProperties*, necessary to both fluid and solid domains. In fluid domain, it's necessary to choose among different combination of properties to completely define the fluid. In the case of this work the *heRhoThermo* model had been set, which requires the entries *specie,transport, equation of state* and *thermodynamics*.
Inside *specie* is simply specified that the fluid is a pure mixture, with molar weight of 22.99 kg/kmol.
Transport contain informations on dynamic viscosity and thermal conductivity of the fluid, *equationOfState* on density, *thermodynamics* on specific heat capacity. To create a realistic model, the assumption of having properties constant with temperature cannot hold: temperature change greatly from inlet to outlet and it's not possible to neglect the impact on thermophysical quantities, so a polynomial model has been chosen.
The general form valid for ρ, k, c_p is the following:

$$\chi = C_0 + C_1T + C_2T^2 + C_3T^3 \tag{4.16}$$

Coefficients are shown in Table 4.1.
Regarding dynamic viscosity, the expression is different from other quantities:

Properties	C0	C1	C2	C3
ρ	950.1	-0.22976	1.46e-05	5.638e-09
c_p	1658.2	-0.8479	4.4541e-04	0
λ	105.78	-5.1767e-02	4.8696e-06	0

Table 4.1: *Coefficients for density (ρ), specific heat capacity (c_p) and thermal conductivity (k)*

Dynamic viscosity it's not polynomial, but will be given with the expression:

$$\mu = C_0 exp\left(\frac{C_1}{T}\right) \tag{4.17}$$

Properties	C0	C1
μ	8.51e-05$(\frac{\rho}{1000})^{0.33}$	1.04ρ

Table 4.2: *Dynamic viscosity coefficients*

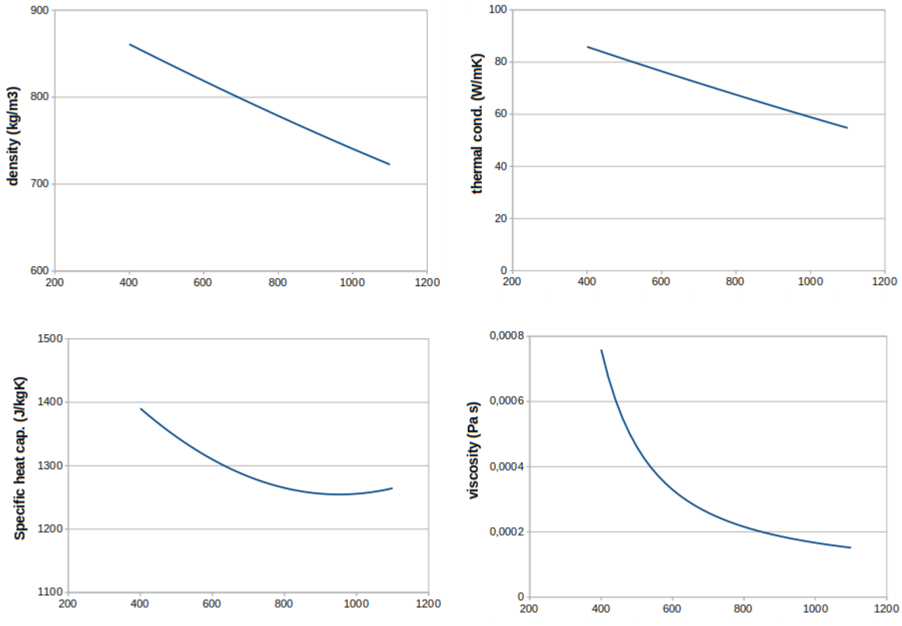


Figure 4.7: *Properties of liquid sodium plotted versus temperature (T in Kelvin)*

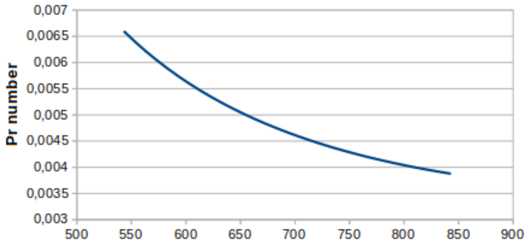


Figure 4.8: *Prandtl number in liquid sodium plotted versus temperature (T in Kelvin)*

In Figure 4.7 and 4.8 all thermophysical quantities are plotted, showing a decreasing trend with temperature. Effect of this trend on heat transfer will be discussed in the next chapter.
Thermophysical properties for solid regions are far more simple thanks to the assumption of constant quantities. All the entries for the model *heSolidThermo* are summarized in Table 4.3. The other dictionary that need explanation is *turbolenceProperties*. This file tells the solver what type of turbulence to use:

- laminar: no turbolence used;
- RAS: use Reynolds-averaged simulation (RAS) modelling;
- LES: uses large-eddy simulation (LES) modelling;

In this work a two equation RAS turbulence model will be used. They are the most common types of turbulence models: models like the *k-epsilon* and the *k-omega* have become standard models and are commonly used for most types of engineering problems. By definition, two equation models include two extra transport equations to represent the turbulent properties of the flow. This allows a two equation model to

Properties		
ρ	7940	kg/m^3
c_p	460	J/kgK
k	35	W/mK
MW	54.46	$kg/kmol$

Table 4.3: *Properties of steel (Incoloy-800) [31]*

account for history effects like convection and diffusion of turbulent energy. Most often one of the transported variables is the turbulent kinetic energy, k. The second transported variable varies depending on what type of two-equation model it is. Common choices are the turbulent dissipation, ϵ, or the specific turbulence dissipation rate, ω. In this work the chosen model is *k-omega SST*, which is a sort of hybrid between the two models: $k - \omega$ formulation makes the model suitable all the way down to the wall through the viscous sub-layer, but it can also shift to $k - \epsilon$ behavior in free-stream.

4.4 System folder

Similarly to previous folders, inside *system* there are several sub-folders, one for each different domain. These sub-folders contain *fvSchemes* and *fvSolution* files, that will be explained better later.

Outside subfolders the most important dictionary to include is *controlDict*, in which are contained the type of solver used for the case, informations about time steps of the simulation and, eventually, any other library and function.

The chosen solver will be *chtMultiRegionSimpleFoam*, a steady-state solver for buoyant, turbulent fluid flow and solid heat conduction with conjugate heat transfer between solid and fluid regions. Notice that for a steady-state simulation *endTime* and *deltaT* hold no physical meaning, they are just used to determine the number of overall iterations done to reach a steady-state; *writeInterval* simply define how often data should be saved in a new time folder.

Lastly, a *function* section inside the file is where additional functionalities can be defined, such as residuals and yPlus monitoring or custom functions written with *swak4Foam* to evaluate heat losses. In steady-state simulations, the solver stops when equation residuals are below a specified tolerance. The residual is one of the most fundamental measures of an iterative solutions convergence, as it directly quantifies the error in the solution of the system of equations. Calculation of residuals for pressure, enthalpy, velocity, turbulence intensity and turbulent dissipation is performed by the function $residuals\_fluid$. Also in the solid regions residuals of energy equations are monitored. For a matrix system:

$$Ax = b \tag{4.18}$$

residuals are defined as:

$$r = b - Ax \tag{4.19}$$

and after the normalization procedure:

$$r = \frac{1}{n} \sum |b - Ax| \tag{4.20}$$

Since the equations are numerically solved, it is necessary to define a residuals tolerance, and the solution is said to be converged when the residuals of all fields are smaller than that tolerance. In general, for steady state code, residuals in the order of e-04 are enough to consider the convergence reached. Another value typically used is e-06, in this case it is considered tight convergence.
The other important parameter to keep under control is *yPlus*. As already discussed speaking of mesh quality, this parameter express near wall grid accuracy with respect to flow velocity. In *K-omegaSST* turbulence models it is better to have $y+ \approx 1$, in order to solve near-wall viscous sublayer. In general, keeping $y+ < 5$ can be enough, as a rule of thumb. Other files contained in this folder are:

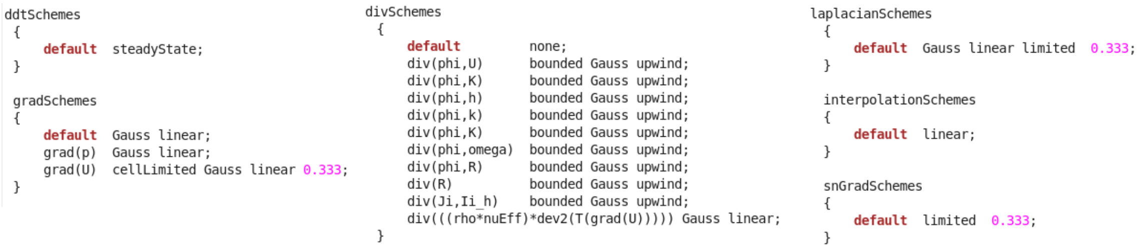
- snappyHexMeshDict: dictionary read during meshing process. (see also previous chapters specifically dedicated to the mesh)

- meshQualityDict: user-defined mesh quality settings. It is an integration to *snappyHexMeshDict*

- surfaceFeatureExtractDict: dictionary file for the *surfaceFeatureExtract* utility, useful to catch sharp edges of the STL geometry or intersections between bodies.

- decomposeParDict: dictionary with specifications on how to split fields and geometry in parallel simulation

At last, let us consider content of domains subfolders: dictionaries *fvSchemes* and *fvSolutions*.The *fvSchemes* dictionary sets the numerical schemes for terms, such as derivatives in equations, that are calculated during a simulation. In Figure 4.9 are shown all the schemes. The first category is *ddtSchemes*, where specify numerical schemes for time derivative: a steady state condition is set. The second sub-dictionary is *gradSchemes*, for gradients terms: *Gauss linear* is a finite volume discretisation of gaussian integration which requires (linear) interpolation of values from cell centers to face centers. Only exception is made for velocity, where a limiting coefficient is imposed to gradient to improve boundedness. Then there is a subdictionary for divergence, which is fundamental since they include both advective and diffusive terms. However, the difference between the two terms is the main problem with *divSchemes*, so that it is convenient to specify manually a scheme for each one of them.



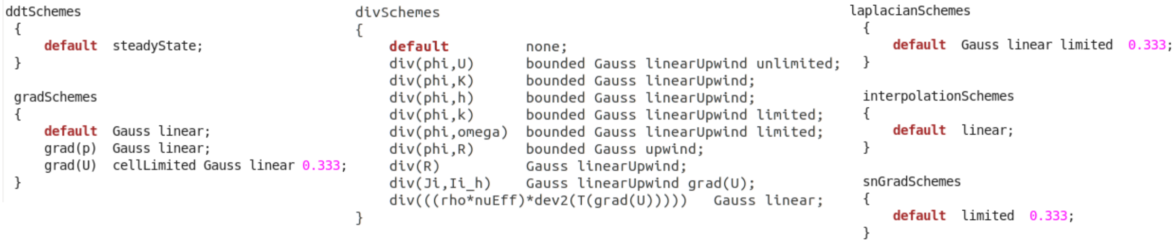
```
ddtSchemes
 {
     default  steadyState;
 }

 gradSchemes
 {
     default  Gauss linear;
     grad(p)  Gauss linear;
     grad(U)  cellLimited Gauss linear 0.333;
 }
```

```
divSchemes
 {
     default          none;
     div(phi,U)       bounded Gauss upwind;
     div(phi,K)       bounded Gauss upwind;
     div(phi,h)       bounded Gauss upwind;
     div(phi,k)       bounded Gauss upwind;
     div(phi,K)       bounded Gauss upwind;
     div(phi,omega)   bounded Gauss upwind;
     div(phi,R)       bounded Gauss upwind;
     div(R)           bounded Gauss upwind;
     div(Ji,Ii_h)     bounded Gauss upwind;
     div(((rho*nuEff)*dev2(T(grad(U))))) Gauss linear;
 }
```

```
laplacianSchemes
 {
     default  Gauss linear limited  0.333;
 }

 interpolationSchemes
 {
     default  linear;
 }

 snGradSchemes
 {
     default  limited  0.333;
 }
```

Figure 4.9: *Example of fvSchemes dictionary used for early iterations, with first order schemes (fluid domain)*

This is not done during first thousands iterations, when an *upwind* scheme is used everywhere. They are first order schemes, in general too inaccurate to catch small



```
ddtSchemes
 {
     default  steadyState;
 }

 gradSchemes
 {
     default  Gauss linear;
     grad(p)  Gauss linear;
     grad(U)  cellLimited Gauss linear 0.333;
 }
```

```
divSchemes
{
    default          none;
    div(phi,U)       bounded Gauss linearUpwind unlimited;
    div(phi,K)       bounded Gauss linearUpwind;
    div(phi,h)       bounded Gauss linearUpwind;
    div(phi,k)       bounded Gauss linearUpwind limited;
    div(phi,omega)   bounded Gauss linearUpwind limited;
    div(phi,R)       bounded Gauss upwind;
    div(R)           Gauss linearUpwind;
    div(Ji,Ii_h)     Gauss linearUpwind grad(U);
    div(((rho*nuEff)*dev2(T(grad(U)))))   Gauss linear;
}
```

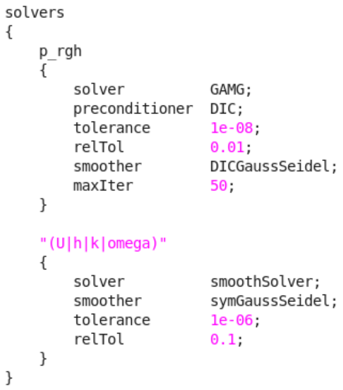
```
laplacianSchemes
 {
     default  Gauss linear limited  0.333;
 }

 interpolationSchemes
 {
     default  linear;
 }

 snGradSchemes
 {
     default  limited  0.333;
 }
```

Figure 4.10: *Example of fvSchemes dictionary with second order schemes (fluid domain)*

scale turbulent eddies, but they are useful to stabilize the solver and help convergence. When residuals will be low enough, it is possible to switch to second order schemes.

Regarding *fvSolution*, this dictionary contains informations about equation solvers, tolerances and algorithms. This file is composed by three sub-dictionaries: *solver, relaxation factors* and *solution algorithms.* In *solvers* are specified the type of algorithm used for each field and the tolerance for residuals.

The entry *tolerance* is the absolute criteria to end the simulation, while *relTol* is the relative tolerance between the initial and the final residual. In the pressure solver tolerances are tighter because typically is the field that require more computational time, so it is better to spend some more iterations especially at the beginning.



```
solvers
{
    p_rgh
    {
        solver          GAMG;
        preconditioner  DIC;
        tolerance       1e-08;
        relTol          0.01;
        smoother        DICGaussSeidel;
        maxIter         50;
    }

    "(U|h|k|omega)"
    {
        solver          smoothSolver;
        smoother        symGaussSeidel;
        tolerance       1e-06;
        relTol          0.1;
    }
}
```

Figure 4.11: *Solvers used in fvSolution dictionary*

Then, it's requested to define solution algorithms, choosing among PISO, PIMPLE and SIMPLE schemes. Obviously they solve the same governing equations, but they differ on how they loop over the equations. In general, SIMPLE algorithm is used for steady state problems.

At last, relaxation factors are defined. Under-relaxation is a technique used for improving stability of a computation, particularly in solving steady-state problems. These factors are included in a range between 0 and 1: values close to 0 increase the stability, but also slow down the computation. An optimum choice is one that is small enough to ensure stable computation but large enough to move forward the iterative process at reasonable speed. In fluid region, similarly to what has been done with the schemes dictionary, a conservative approach will be used for the first thousands of iterations, which means quite low relaxation factor in order to ensure numerical stability. Once a good stability is achieved, it is possible to increase gradually those factors to speed up the convergence.

One last aspect is that in fluid domain is set $nonOrthogonalCorrector = 1$. During each time step, pressure equation is repeated according to number of correctors. This may reduce the influence of non orthogonality in the mesh.

Field	Factor
p_ rgh	0.2
U	0.2
h	0.3
k	0.3
omega	0.3

Table 4.4: *Initial relaxation factors in domain2*

With regards to solid regions, they are simpler to deal with, since only energy equation has to be solved. The aim in those domains is to reach convergence as fast as possible, so second order schemes (*gaussLinear*) are used from the beginning, with higher relaxation factor (starting with 0.5. They can be increased later to even 0.8/0.9.

Chapter 5

Results

At last, the final chapter is dedicated to study the simulation results. First of all will be studied distributions of temperature, velocity and pressure in solid and fluid region then, in the last section, will be performed an estimation of heat losses and thermal efficiency of the receiver.

5.1 Temperature and flow rate distribution

Temperature and mass flow rate of liquid sodium inside the receiver will be analyzed together since they are strongly correlated.
Regarding solid regions, temperature field is the only one considered, since only energy equation is solved. Obviously, temperature distribution follows the circumferential and longitudinal non-uniformity of the heat flux (Figure 5.1). Temperature is different from tube to tube, where the hottest are the ones closer to the receiver symmetry axis. Also, temperature change along z axis, with peak values close to the outlet as a result of sodium heating up while moving towards outlet. From Figure 5.2 it is possible to see temperature distribution on tubes and in particular temperature difference between front and back side of the receiver, which is not exposed to solar radiation. Due to the strong non-uniformity of temperature, it is necessary to define a reference temperature to calculate fluid properties (Prandtl, Nusselt, Reynolds, heat transfer coefficient...). In this work, bulk temperature is defined integrating temperature over cross section area of each tube. Profile of T_{bulk} in each tube is shown in Figure 5.3, with numbering

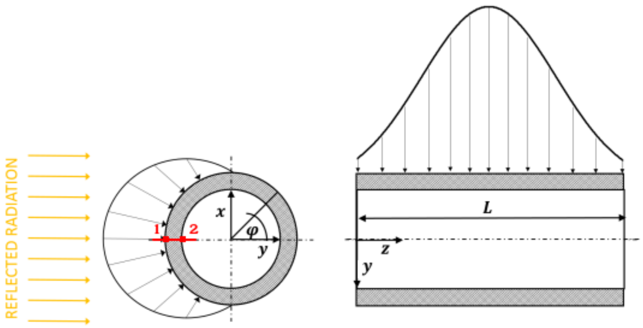


Figure 5.1: *Heat flux variation over the length and radial coordinate of a generic tube*

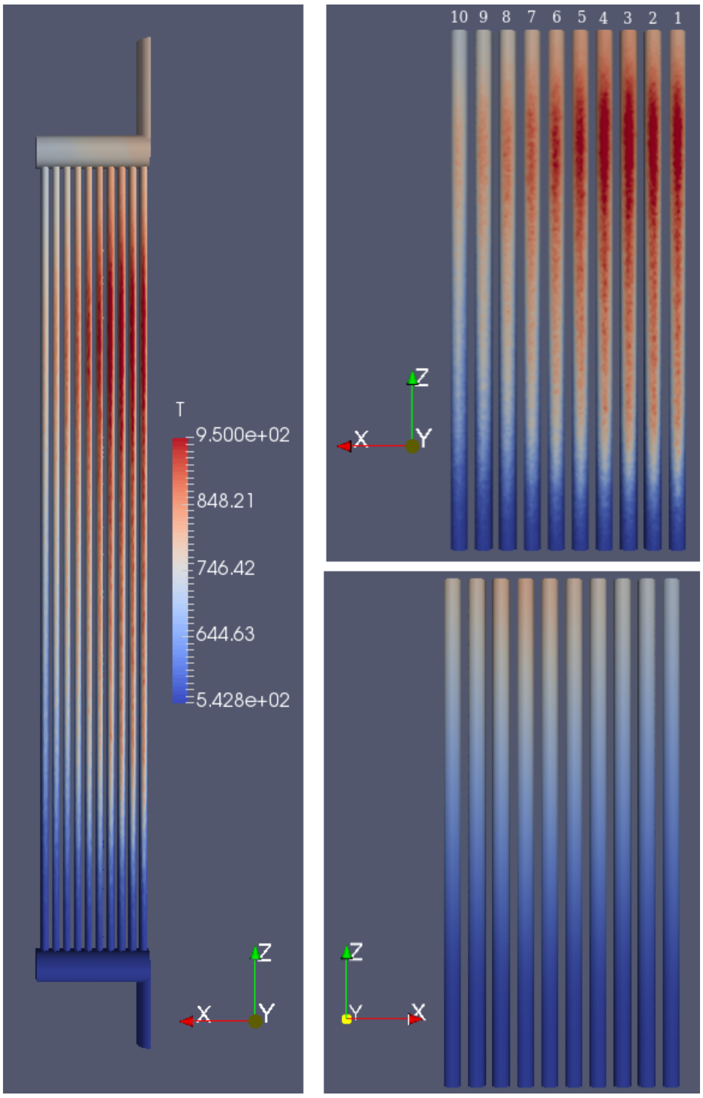


Figure 5.2: *Temperature distribution on the receiver. On the right only absorbing tubes are shown, front view (top) and back view (bottom)*

according to Figure 5.2. Central tubes, hit by the highest heat flux intensity, are also the ones with highest bulk temperature at the outlet. Both bulk temperatures in the fluid domain and temperatures in metal depend greatly on the mass flow rate of sodium crossing each tube: a proper design of the inlet header is crucial to ensure a good distribution of HTF inside the receiver and to reduce peaks of temperature that could be dangerous for materials. In this work, inlet tube is placed in the middle of inlet header in order to increase fluid flow rate and cooling inside hottest tubes (Figure 5.4). In Table 5.1 a resume of temperature and mass flow rate is presented. Mass flow rate decrease progressively moving from symmetry axis (*Tube10* shows a flow rate 40% lower than *Tube1*, with numbering according to Figure 5.2), but even so outlet bulk in lateral tubes is significantly lower. This suggest possible improvements in heliostats aiming strategy, trying to increase heat flux intensity on lateral tubes in order to improve overall performances. After mixing in the outlet header, liquid sodium exit from the outlet pipe with a bulk temperature of $T_{bulk} = 505°C$. Considering an

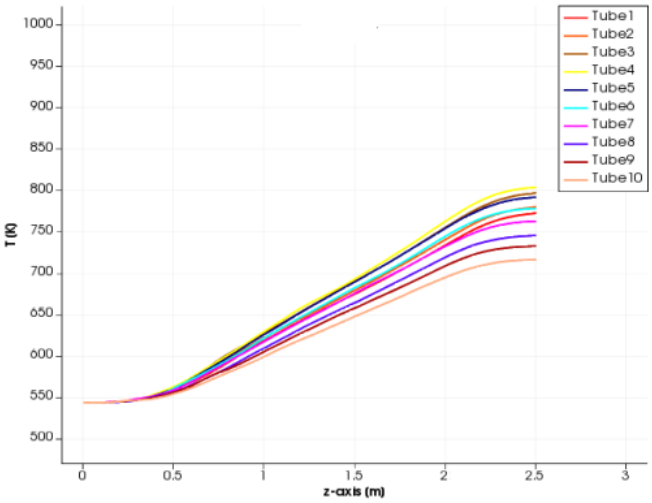


Figure 5.3: *Temperature profile plotted along z axis of each tube*

inlet temperature of 270°C, the result is $\Delta T = 235°C$. It is a good output even though temperature range of sodium is still not fully exploited. The bottleneck of the system is indeed the solar field: to increase heat flux delivered from mirrors would lead to higher temperature and (as shown in following sections) to better thermal efficiencies.

	Peak T$[°C]$	**Bulk T**$[°C]$	**Mass flow rate**$[kg/s]$
Tube1	755	499	0.2071
Tube2	751	507	0.1964
Tube3	752	523	0.1818
Tube4	762	530	0.1690
Tube5	705	518	0.1648
Tube6	694	504	0.1627
Tube7	645	489	0.1511
Tube8	623	474	0.1467
Tube9	581	459	0.1341
Tube10	546	444	0.1226

Table 5.1: *Comparison between peak temperatures measured on metal, bulk temperatures of sodium at the outlet and mass flow rate in each tube; numbering according Figure 5.2*

It can be interesting to study also temperature drop within metal thickness, since peak temperatures can be challenging for the material in some areas. With reference to Figure 5.1, the maximum value of heat flux in radial direction is found at $\varphi = 180°$. Temperature drop along segment 1-2 (red line in Figure 5.1) is plotted versus dimensionless axial coordinate in Figure 5.6. Plots allow to highlights temperature differences in steel: tubes with higher incident solar radiation have also greater drop, with peak temperature drop of 90°C difference between external and internal walls. Lateral tubes, which present sensibly lower temperatures, show a more uniform temperature distribution with similar values on the two surfaces. This aspect can be noticed also in from Table 5.1, where temperature difference between external surface and bulk fluid was

$> 200°C$ in tubes hit by higher heat flux, while lateral tubes have smaller temperature differences. In all tubes, peak values of temperature are found more or less at 80% of tube length.

Temperature drop within walls is influenced by internal heat transfer coefficient (will

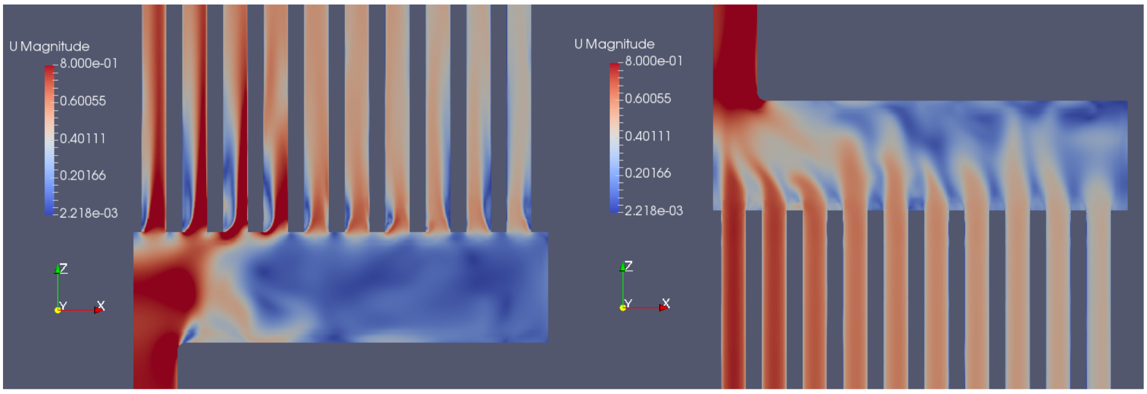


Figure 5.4: *Distribution of liquid sodium inside inlet header (left) and outlet header (right)*

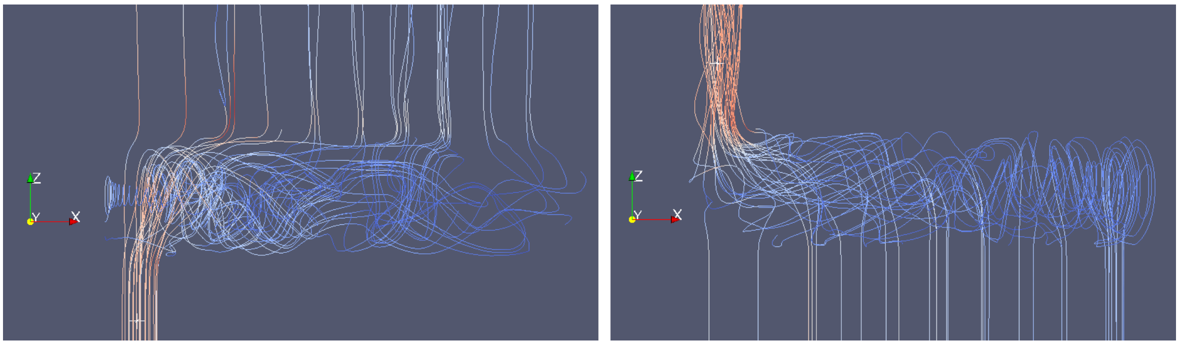


Figure 5.5: *Streamlines showing the path of liquid sodium inside inlet header (left) and outlet header (right)*

be discussed in next sections) and by thermal resistance offered by cylindrical tubes:

$$R_{cyl} = \frac{ln\left(\frac{r_2}{r_1}\right)}{2\pi L\lambda} \tag{5.1}$$

To take full advantage of sodium as HTF and reduce the temperature drop within tube walls would be better to explore the use of higher conductivity steels and, if possible, to reduce wall thickness. Tubes are made in Incoloy 800HT, a steel alloy based on nickel-chromium that can operate at temperatures up to 899°C [31], so it could withstand even the worst condition.

Another important aspect to be considered is turbulence, since it influence greatly heat transfer process. In the first section of each tube the flow is irregular and turbulent, with components of velocity in all directions. This first unsteady region is called *entrance length*: it's the length necessary to the flow in order to reach a complete developed velocity profile. Inside a pipe, a flow is said developed when velocity is no more dependent on axial coordinate, but change only in radial direction. In such cases,

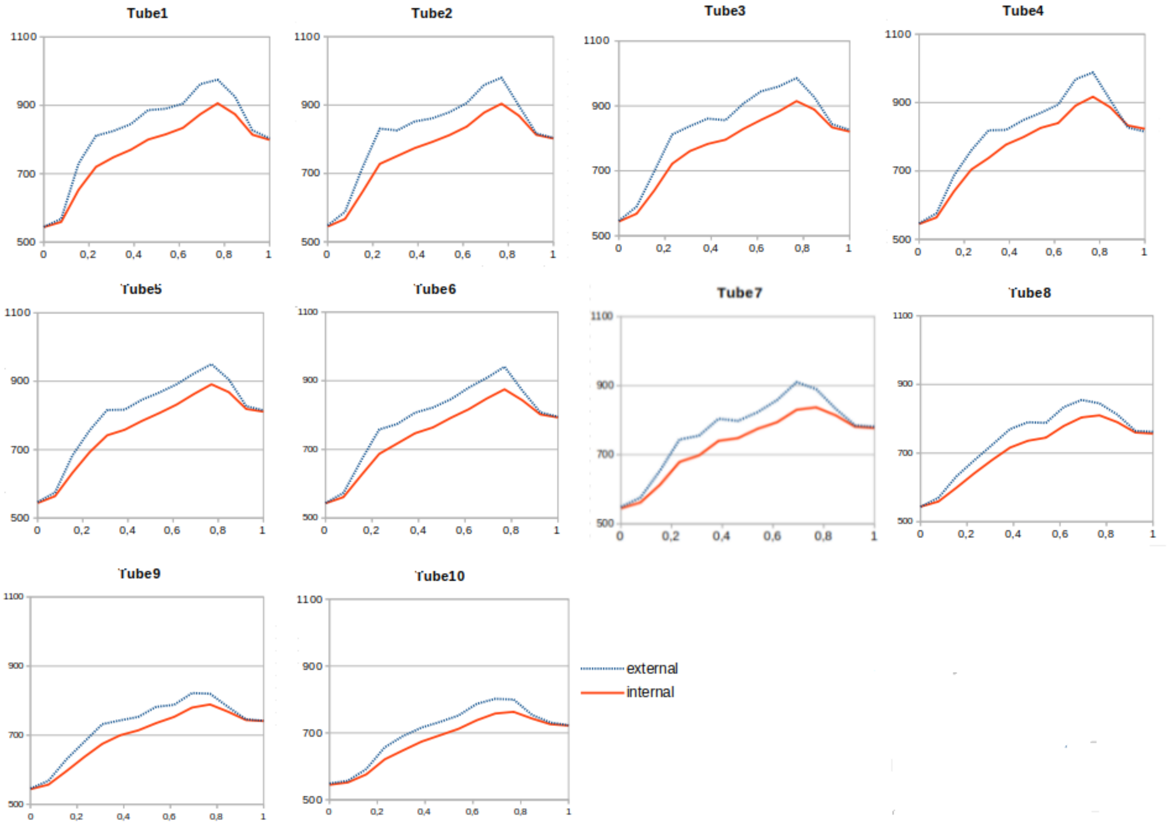


Figure 5.6: *Temperature of external and internal surfaces of each absorbing tube along dimensionless length. y- axis is $T(K)$, x-axis is $z^* = z/L$.*

velocity assumes the parabolic profile typical of flow in pipe. The parabolic shape depends on Re number and on friction factor with walls. In general, turbulent flows show a flatter curve with respect to laminar case, with an almost flat value and a sharp drop near the pipe wall. The thin layer next to the wall is where viscous forces are dominant, called *viscous* or *laminar sublayer*. In Figure 5.8 it is shown the change in parabolic curve inside entry length, with a developed bell shape obtained between $z/L = 0.3 \div 0.4$.

From theory, in internal convection through a pipe, the flow can be considered turbulent if:

$$Re_D = \frac{\rho \langle \omega \rangle D_H}{\mu} > 2300 \tag{5.2}$$

where $D_H = \frac{4A_{cross}}{P_{pipe}}$ is the hydraulic diameter and $\langle \omega \rangle$ is bulk velocity, defined as:

$$\langle \omega \rangle = \frac{1}{A} \int_A \omega dA \tag{5.3}$$

From Figure 5.9 it's clear the turbulent nature of the flow. As expected, the flow is turbulent in all regions with values in the range $19000 < Re_D < 60000$. This also confirm the choice of using a *kOmegaSST*, which is a turbulent model. Tubes with higher mass flow rate shows also higher Re numbers. It is interesting to notice the increasing trend along z axis: mass flow rate is equal within the same tube, but dynamic viscosity decrease with temperature so that Re grows along z coordinate.

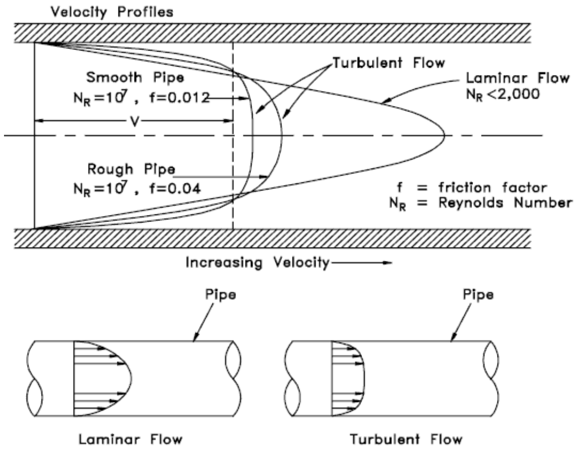


Figure 5.7: *Comparison between parabolic profile in laminar and turbulent flow*

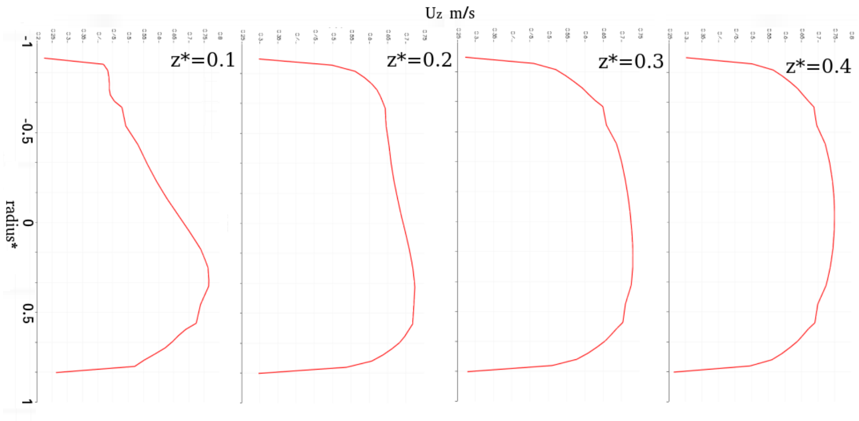


Figure 5.8: *Evolution of velocity profile in the entrance length of tube1. Curves are plotted at different dimensionless length z*=z/L and dimensionless radius, r*=r/R (R=0.0125)*

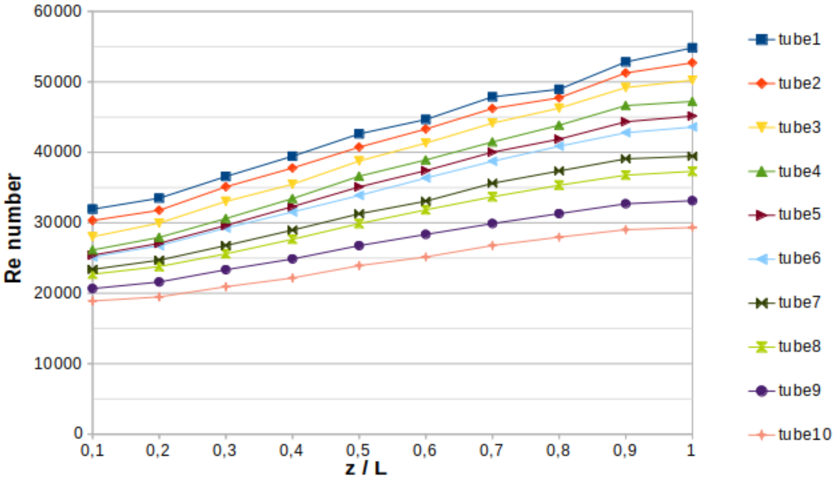


Figure 5.9: *Re number plotted along tube length. Each tube is crossed by a different mass flow rate (see Table 5.2)*

5.2 Pressure drop

Pumping process of HTF inside tubes is not a parameter that influence thermal efficiency of the receiver, that is the purpose of this work. However, it's a crucial parameter to take into account during plant design process and it influences heavily global efficiency and LCOE of produced electricity.
Pressure drops in a solar tower plants aren't localized only in the receiver, because there is a long piping system to connect the absorbing section (placed at the tower top) with cold and hot tanks located on the ground. Neglecting losses due to geometry changes, in general pressure drops along a pipe can be expressed as (Darcy-Weisbach equation) [49]:

$$\Delta P = f\frac{L}{d_i}\frac{u^2}{2g}\rho \tag{5.4}$$

where f is the friction factor, d_i the tube inner diameter, u and ρ velocity and density of the fluid and L is the tube length.
Assuming constant mass flow rate, increasing tube cross section area lead to a smaller pressure drop, since inner diameter grow while fluid velocity decrease. Also, liquid sodium has lower density with respect to common Solar Salts (see Section 1.2 for comparison), which is good for pumping. On the other hand, bigger cross sections influence badly heat transfer coefficient (see Section 5.4), so the optimal choice should be a trade-off between optimization of heat transfer and limiting pumping power.
It's also true that pressure losses inside receiver are generally low and the majority of pumping power is spent in various other piping sections of the plant [48].

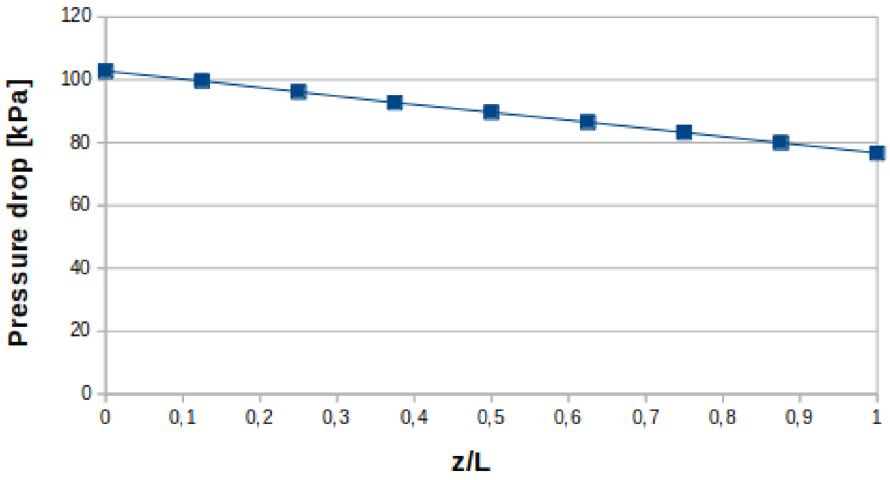


Figure 5.10: *Pressure drop along receiver length*

5.3 Heat transfer characteristics

In this section will be performed a calculation of heat transfer coefficients of fluid inside inside tubes, in order to confirm the expectations from theory. The heat transfer coefficient is defined as:

$$h = \frac{\lambda\, Nu}{L_c} \tag{5.5}$$

where L_c is characteristic length, which can be assumed equal to diameter for flow in pipes. More complex is to define Nusselt number (Nu).

Nusselt number can be seen as the dimensionless temperature gradient close to an heat exchanging surface, provided that temperature is normalized with a characteristic temperature difference [55]. To understand better the meaning of Nu number, starting from (5.5) it's possible to write:

$$Nu = \frac{L_c h}{k} = \frac{h \Delta T}{k \frac{\Delta T}{L_c}} = \frac{q_{conv}}{q_{cond}} \tag{5.6}$$

So the physical meaning of Nusselt number is the ratio of convective to conductive heat transfer across a boundary. Values close to unity mean that conductive heat transfer mechanism is predominant with respect to convection, such as in the case of liquid metals.

The expression of local Nusselt can be written as:

$$Nu_x = \frac{-\left.\frac{\partial T}{\partial x}\right|_{x=wall} L_c}{T_{bulk} - T_{wall}} \tag{5.7}$$

However, internal wall temperature and gradient variate greatly along inner tube circumference, due to the non-uniform distribution of solar radiation. It is convenient to define an average Nu number, by integrating T gradient and T_{wall} over internal surface of each tube:

$$Nu = \frac{-\overline{\frac{\partial T}{\partial x}}\, L_c}{T_{bulk} - \overline{T_{wall}}} \tag{5.8}$$

In Figure 5.11 Nu numbers in each tube are plotted versus Peclet number and dimensionless tube length. Data from numerical simulation are compared to results from empirical correlations. The following models are proposed [17], all valid for $3 \cdot 10^3 \leq Re \leq 1 \cdot 10^6$ and $0.0001 \leq Pr \leq 0.1$: Weigan-Ferguson (5.9), Aoki (5.10) and Kays (5.11).

$$Nu = 5.51 + 0.018 Pe^{0.8275} \tag{5.9}$$

$$Nu = 5.51 + 0.015 Pe^{0.865} \tag{5.10}$$

$$Nu = 5.31 + 0.0221 Pe^{0.8174} \tag{5.11}$$

where Peclet number is defined as $Pe = Re\ Pr$. Notice that those correlations are given with the assumption of uniform heat flux, so it is expected a certain difference with numerical values. In Figure 5.12 differences between numerical data and correlations are provided in percentage, showing a maximum difference contained within a ±10% range. Despite the remarkable difference in starting assumptions, the comparison is still useful as a reference to understand if numerical values are reasonable. Nusselt numbers inside the receiver are comprised within a range of 6 to 8: they are small as expected in a heat transfer process dominated by conduction.

From Figure 5.11, tubes crossed by higher mass flow rate have also higher Peclet number, being Pe dependent on Reynolds. Nu shows a slightly increasing behavior along tubes axial coordinate, even though results are quite irregular and there is no clear trend. In general, Nu number is better in tubes hit by a high heat flux intensity, resulting in a better heat transfer process. This aspect is confirmed looking at Figure 5.13, where heat transfer coefficient inside each tube has been plotted. Due to the particular characteristics of liquid metals, those value are extremely high, in the range between 22000 and $26000W/m^2K$. As expected, central tubes show better performances thanks to higher Nu number. The trend along tube length is decreasing, due to the reduction of thermal conductivity with temperature. Performances are very

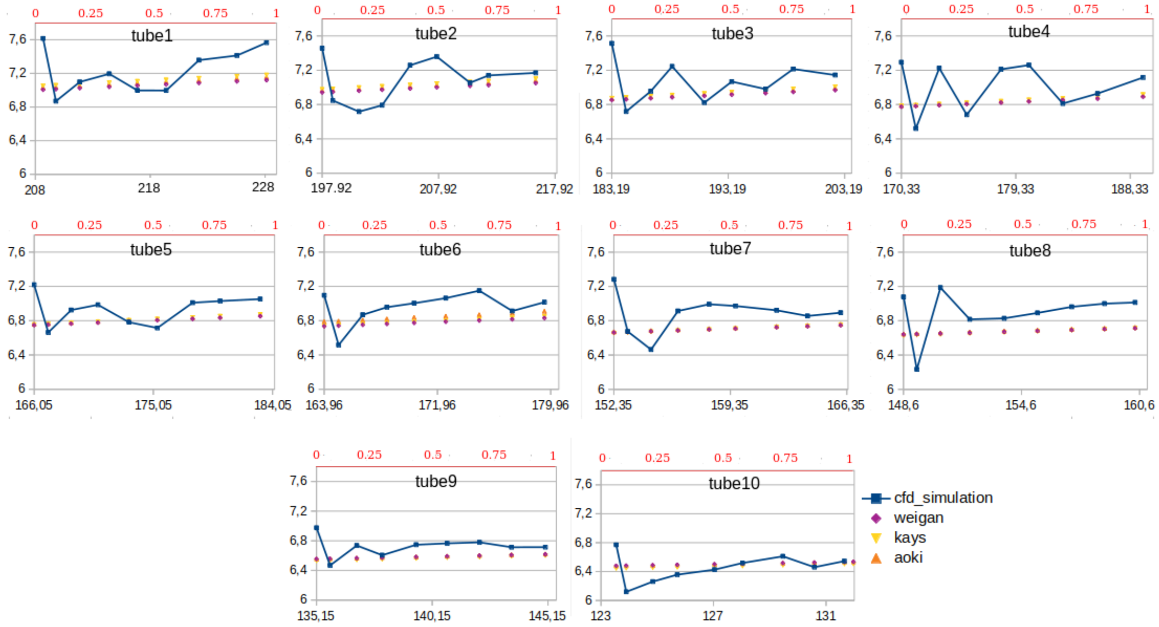


Figure 5.11: *Nusselt numbers in each tube plotted versus Peclet (black x-axis). Red x-axis show dimensionless tube length z/L. Dots show values obtained empirical correlations*

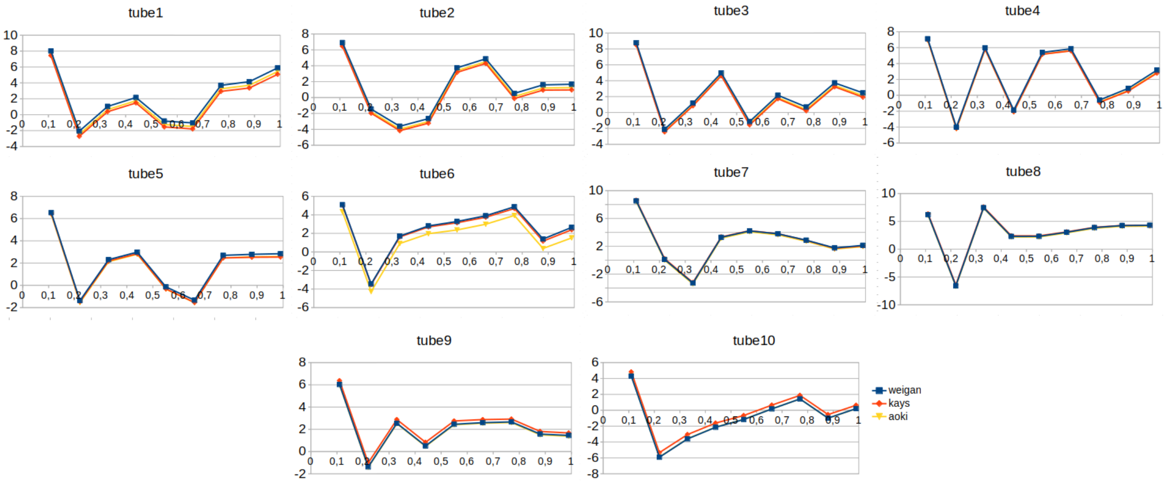


Figure 5.12: *Percentage difference between numerical data and values found with three correlations by Weigan-Ferguson, Kays and Aoki*

positive when compared to conventional salts: to give a better idea of the actual difference with the most common HTF in solar application, in Figure 5.14 a comparison with a test pipe crossed by conventional molten salts is showed. For the same temperature range h can be from 2.5 to 5 times higher than the one in Solar Salts or Hitec.
Thinking of possible improvements, it can be noted how turbulence is positive for the heat transfer process. In order to increase Re numbers, a different design of the receiver could be used, with an higher number of absorbing tubes but with smaller diameter: considering the same mass flow rate, smaller diameters would result in an higher bulk flow velocity and consequently in an higher level of turbulence. At the same time, smaller tubes can handle smaller thermal stresses, so there is a technical limit to diameter reduction. Another way to enhance turbulence inside tubes tube could be to increase mass flow rate. However, also an higher incident flux would be required to not reduce outlet bulk temperature. Indeed, increasing power per m^2 delivered to the receiver would lead to a performances upgrade, with higher thermal power output and efficiency. Of course, the choice of employing a small solar field imposes a limitation to the available heat flux; further increases could be obtained with more refined field layouts and tracking systems in order to improve cosine and intercept factors.

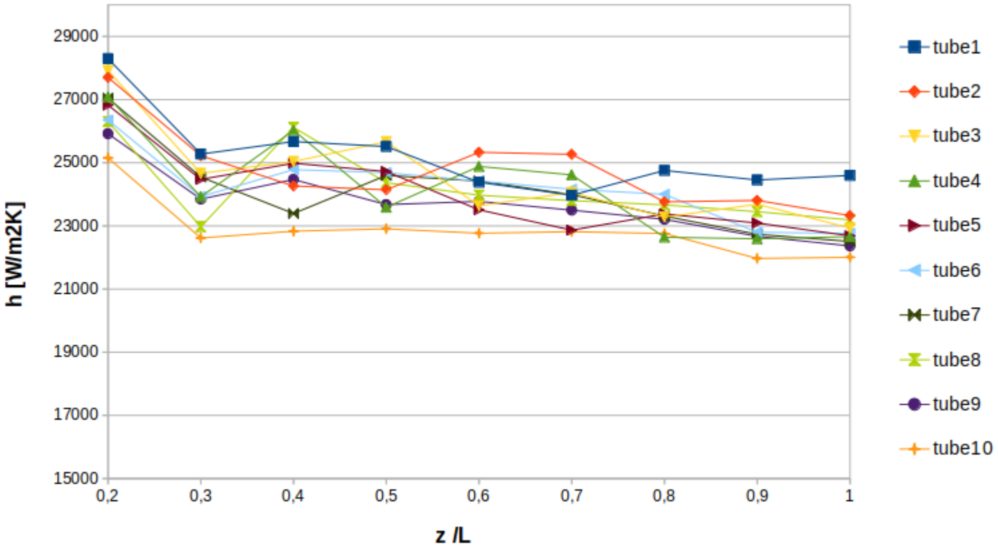


Figure 5.13: *Heat transfer coefficient along dimensionless tube length*

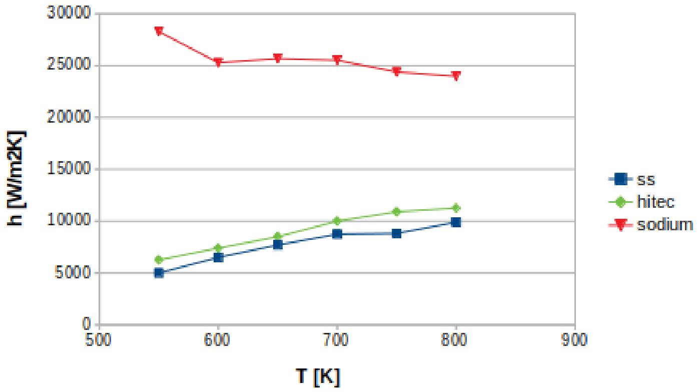


Figure 5.14: *Comparison between heat transfer coefficient in tube1 vs a test pipe [29] with Hitec and Solar Salts ($\dot{m} = 1kg/s$), at same T range*

5.4 Efficiency

At last, let us evaluate heat transfer performances of the receiver, which is basically the very scope of this work.
To evaluate heat fluxes will be used an OpenFOAM function called *WallHeatFlux*: this tool is able to calculate the inlet/outlet heat fluxes of a single mesh region. Using this function three heat fluxes are given as a result:

- a positive (entering) flux through external tube surface, which is the heat entering due to solar radiation
- a negative (exiting) flux through internal tube surface, which is the heat entering liquid sodium
- a negative (exiting) flux through the surface with the adiabatic half of each tube (see Section 3.3 also for clarification)

It is important to notice that *WallHeatFlux* function returns the net heat flux entering the system. In fact, remembering the definition of boundary condition made in Section 4.2, convective and radiative losses have been already subtracted from the gross incident heat flux, and the absorptance factor has been considered as well. So, to know the initial gross power and compute thermal efficiency those losses have to be added to the net power.

To calculate convective and radiative losses a proper function has been defined in *controlDict* thanks to *swakExpression*. It consists in another *swak4Foam* instrument, that give us complete freedom in writing a custom expression and decide on which patch applying it. At each iteration expressions (5.12) and (5.13) are evaluated and saved into a text file in *postProcessing* folder. A more detailed description on temperatures and constants used in those expression can be found in Section 4.2.

$$\dot{Q}_{conv} = A_{ext} h_{conv} (T_\infty - T_s) \tag{5.12}$$

$$\dot{Q}_{rad} = A_{ext} \sigma \epsilon (T_{sky}^4 - T_s^4) \tag{5.13}$$

Moreover, there is one more loss to take into account due to the non-perfect absorptance of surfaces. Steel is coated with a layer of Pyromark, which has a huge absorptance

	Net power [W]	**Gross power [W]**	**Efficiency**
Tube1	53901.3	59131.2	0.9115
Tube2	52618.2	57813.3	0.9101
Tube3	51000.6	56221.6	0.9071
Tube4	48421.2	53527.8	0.9045
Tube5	44260.3	49036.1	0.9026
Tube6	40245.8	44676.8	0.9008
Tube7	36074.7	40169.3	0.8998
Tube8	31262.4	34961.7	0.8941
Tube9	27796.3	31210.8	0.8905
Tube10	23346.2	26406.8	0.8841
Half rec.	408926.8	453155.5	0.9024

Table 5.2: *Heat balance and efficiency on each tube and on the whole receiver*

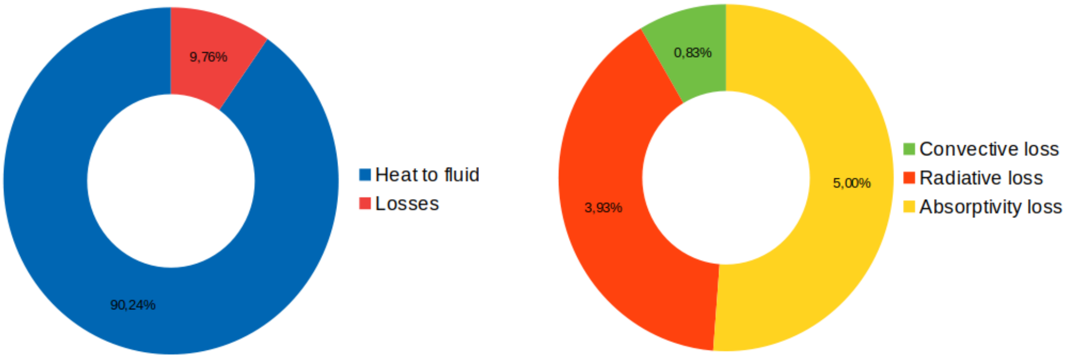


Figure 5.15: *Pie charts graphically showing receiver efficiency (left) and the weight of each loss (right)*

($\alpha = 0.95$ in this case). Also this loss had been already considered in the boundary condition, so that the gross amount of thermal power delivered from solar field should be 5% bigger then the one obtained with *wallHeatFlux*.
Once all heat fluxes are known, thermal efficiency can be defined as:

$$\eta_{th} = \frac{\dot{Q}_{net}}{\dot{Q}_{gross}} = \frac{\alpha \dot{Q}_{gross} - \dot{Q}^{loss}_{conv} - \dot{Q}^{loss}_{rad}}{\dot{Q}_{gross}} \tag{5.14}$$

Considering again the hypothesis of perfect symmetry, the whole system receive 910.7 kW of gross power, delivering 817.8 kW to HTF. It is interesting to notice (Table 5.2) that tubes hit by higher flux density show also the highest efficiencies: this was predictable from heat transfer coefficient analysis in previous section. Also, many studies have demonstrated how performances of central receivers grow with flux intensity [45][47]. From Table 5.3 and Figure 5.16 it is possible to consider the order of magnitude of each loss. Convective losses are almost negligible, accounting for less than 1% of the total heat loss (Figure 5.15). This is expected: considered average wind speed in the area is quite low, and the receiver surface is extremely small. Moreover, in case convection losses would represent a problem, it's always possible to shift to a cavity-type receiver capable to shield the receiver from wind.
Much more important are losses due to radiation. They are strongly dependent on T,

	Conv. losses [W]	Rad. losses [W]	Abs. losses [W]	Total loss [W]
Tube1	387.3	2026.9	2815.8	5229.9
Tube2	389.4	2052.7	2753.0	5195.1
Tube3	396.9	2146.9	2677.2	5221.1
Tube4	397.9	2159.8	2548.9	5106.6
Tube5	389.3	2051.5	2335.0	4775.8
Tube6	378.8	1924.7	2127.5	4431.0
Tube7	369.1	1812.7	1912.8	4094.7
Tube8	356.8	1677.7	1664.8	3699.3
Tube9	347.5	1580.1	1486.2	3414.5
Tube10	336.1	1467.1	1257.5	3060.6
Receiver	3749.12	18900.8	21578.8	44228.7

Table 5.3: *Heat balance on each tube*

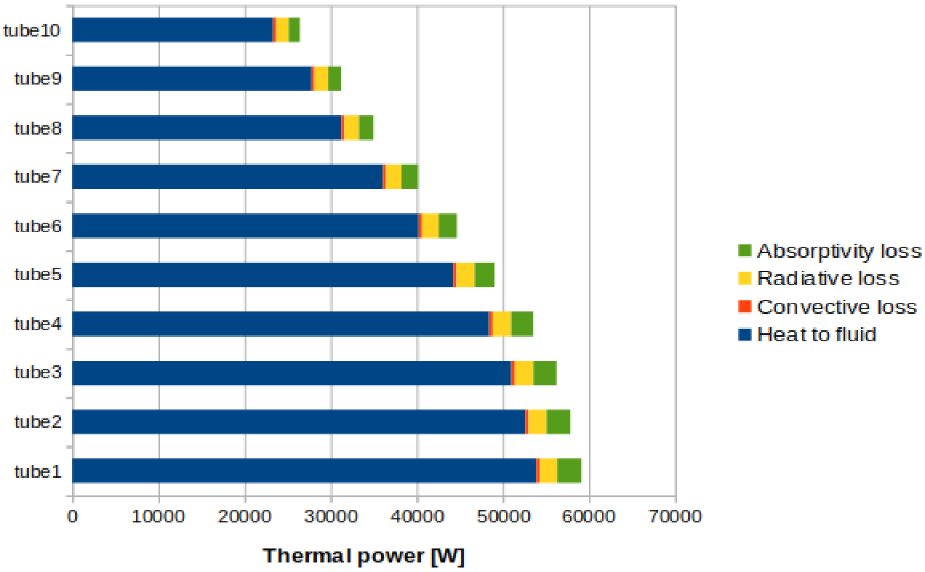


Figure 5.16: *Bars plot showing gross power entering each tube, with net heat entering the system and losses*

since the total radiant heat power emitted from a surface is proportional to the fourth power of its absolute temperature. Accordingly, we can observe in Figure 5.16 how radiation losses have a bigger impact on first four tubes, which are also the hottest, and then decrease in tubes with lower average wall temperature.

Notice that in this work view factors between tubes are not considered, so the radiation loss results slightly overestimated.

The last loss, absorptivity loss, does not depend on thermophysical characteristics of the sodium, but only on the receiver materials. It is clear that keeping a constant loss of 5% on every surface, zones with an higher flux intensity will be more affected, in particular at the receiver middle where it is concentrated the peak of solar radiation. In first four tubes absorptivity and radiative losses are comparable, while the second seem to be the main source of heat losses for tubes at lower temperature.

5.5 Comparison with other studies

To better understand the goodness of this simulation it could be useful to compare it with other works on the same argument. In particular, three studies will be taken into account, containing thermal performances of receiver run both with liquid sodium and conventional salts.
The first cited work [45] is a part of the Australian Solar Thermal Research Initiative (ASTRI) effort to design a 91% efficient receiver for a new high temperature CSP system. The aim was to compare performances liquid sodium within different temperatures range and peak fluxes. In this case, only results of liquid sodium receiver are reported (Figure 5.17).

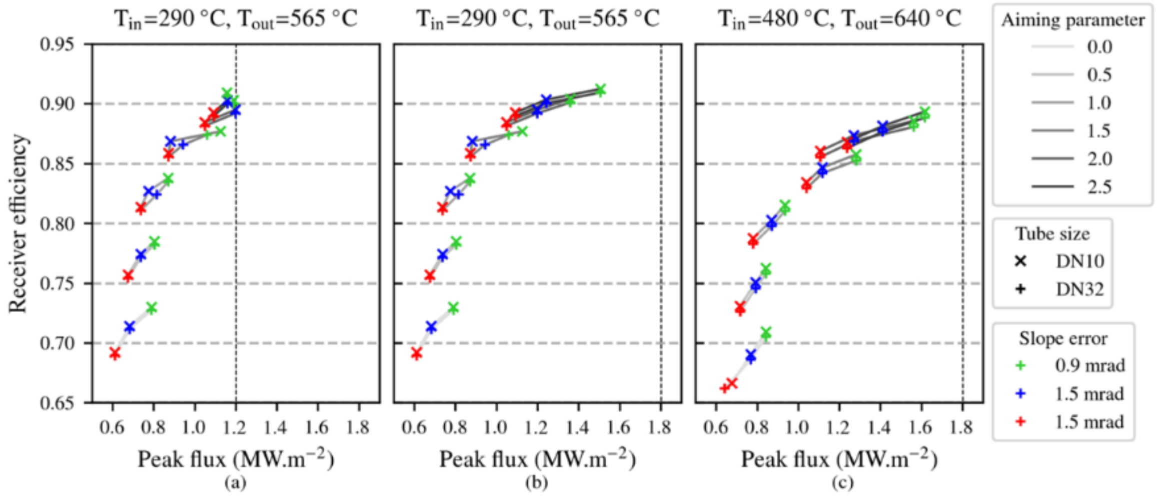
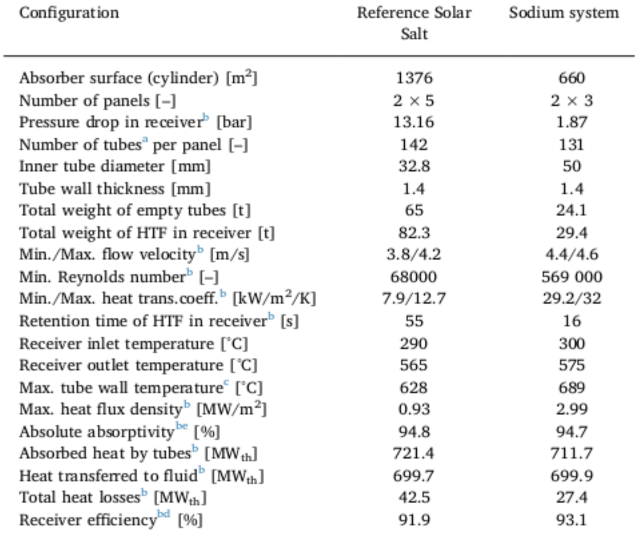


Figure 5.17: *Receiver efficiency for sodium receiver as a function of the receiver aperture peak flux for (a) HTF temperatures between 290° C and 565° C and a flux limit of 1.2 MW/m2 , (b) HTF temperatures between 290° C and 565° C and a flux limit of 1.8 MW.m/2 and (c) HTF temperatures between 480° C and 640° C and a flux limit of 1.8 MW/m2*

The work was more focused on optical analysis, so results are given as a function of slope error and aiming parameter (slope error is the angular deviation between the normal direction of the actual surface and an ideal one; aiming parameter is a multiple of the standard deviation of the heliostat vertical flux distribution on the receiver, and it determines an acceptable vertical aiming region for each heliostat). For each configuration, the best efficiency is taken: (a) 91%, (b) 91%, (c) 89%.
The second work considered [46] is a techno-economical study of a large scale solar plant with sodium as HTF, which is compared with a conventional one run with solar salts. In Figure 5.18 is presented a resume of the two configurations performances. The sodium type is run with the highest peak heat density (2.99 MW/m2) among all configurations taken into account, and show also the highest efficiency, 93%. The third, and last, reference is a LCOE evaluation of a sodium receiver through a thermal and pressure drop analysis. This study is the most significative for us, since it is done on a small scale sodium receiver [47], with incident flux and absorbing area comparable with data used in this work.



Configuration	Reference Solar Salt	Sodium system
Absorber surface (cylinder) [m^2]	1376	660
Number of panels [–]	2 × 5	2 × 3
Pressure drop in receiver[b] [bar]	13.16	1.87
Number of tubes[a] per panel [–]	142	131
Inner tube diameter [mm]	32.8	50
Tube wall thickness [mm]	1.4	1.4
Total weight of empty tubes [t]	65	24.1
Total weight of HTF in receiver [t]	82.3	29.4
Min./Max. flow velocity[b] [m/s]	3.8/4.2	4.4/4.6
Min. Reynolds number[b] [–]	68000	569 000
Min./Max. heat trans.coeff.[b] [kW/m^2/K]	7.9/12.7	29.2/32
Retention time of HTF in receiver[b] [s]	55	16
Receiver inlet temperature [°C]	290	300
Receiver outlet temperature [°C]	565	575
Max. tube wall temperature[c] [°C]	628	689
Max. heat flux density[b] [MW/m^2]	0.93	2.99
Absolute absorptivity[be] [%]	94.8	94.7
Absorbed heat by tubes[b] [MW$_{th}$]	721.4	711.7
Heat transferred to fluid[b] [MW$_{th}$]	699.7	699.9
Total heat losses[b] [MW$_{th}$]	42.5	27.4
Receiver efficiency[bd] [%]	91.9	93.1

Figure 5.18: *Main features and performances of a receiver run with SS and Sodium as HTF [46]*

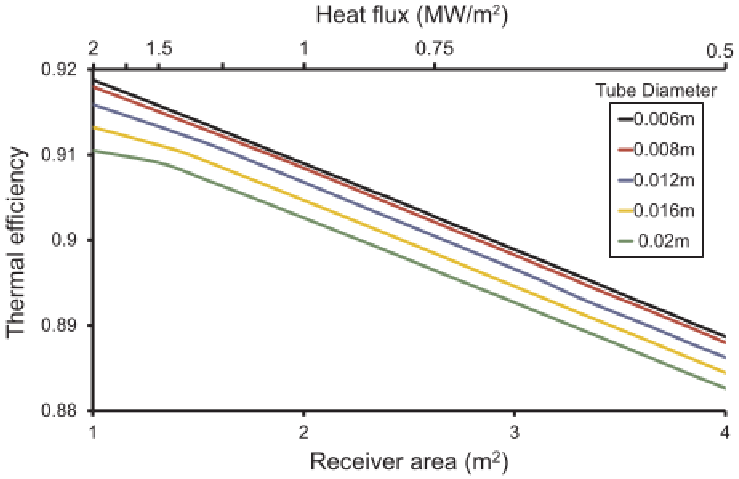


Figure 5.19: *Thermal efficiency for receiver configurations in study [47], varying tube diameter and heat transfer area*

In Figure 5.19 are shown the performances of the receiver changing area and incident heat flux. From the plot it's possible to see the trend of efficiency and how it decreases with area: that's because radiative and convective losses grow proportionally to surface. On the other hand higher flux intensity it's clearly positive, since it allows to exploit better heat transfer qualities of sodium.

The configuration taken as reference is the one with tube diameter of 0.02 m and absorbing area 2 m^2, since it's very similar to the receiver presented in this work. Also results are very similar, with a measured efficiency in this case of 90%. At last, in Figure 5.20 is shown a graphic resume of the previously presented data. Thermal efficiency of the simulated receiver is comparable to the results found in other studies, with values slightly higher than 90%. Only one configuration differs sensibly from the other (the one with 93% efficiency): it is the one with an unusually high heat flux intensity, highlighting once again how high heat flux is crucial to enhance sodium

receivers efficiency. Of course many other parameters can influence thermal efficiency such as wall thickness, material thermal conductivity, absorptivity, emissivity and tube diameter. Regarding the effects of receiver dimension, having a small external area is indeed positive to reduce heat losses, but at the same time it is difficult to ensure a good interception of solar rays with such a little target, with a worse mirrors optical efficiency.
Another important parameter to evaluate the goodness of a solar receiver is the temperature at the outlet. Under this point of view, our receiver shows a $\Delta T = 235°C$, with an outlet T_{bulk} of $505°C$, the lowest among all configurations. From Figure 5.21, all other receivers work in a temperature range of $\Delta T = 270°C$, with the only exception of a single high-temperature configuration working between 480 and 640°C ($\Delta T = 160$°C). Even though the outlet temperature in receiver simulated in this work is the lowest, performances in the small scale plant are similar to other sodium configurations, proving the feasibility of small scale tower plants.

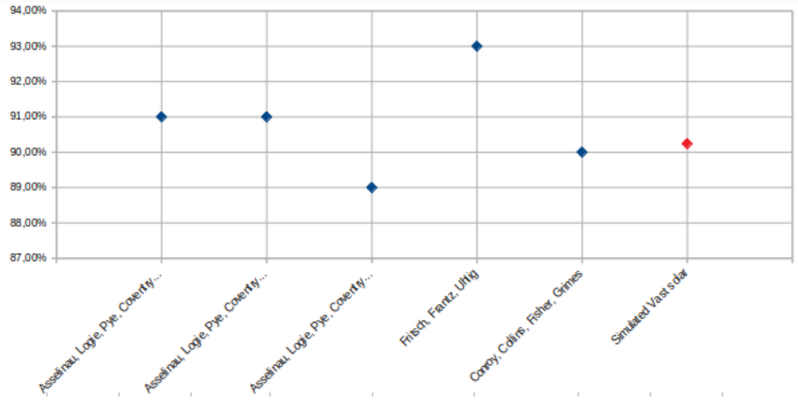


Figure 5.20: *Efficiency of simulated Vast solar plant compared to other configurations presented in this section*

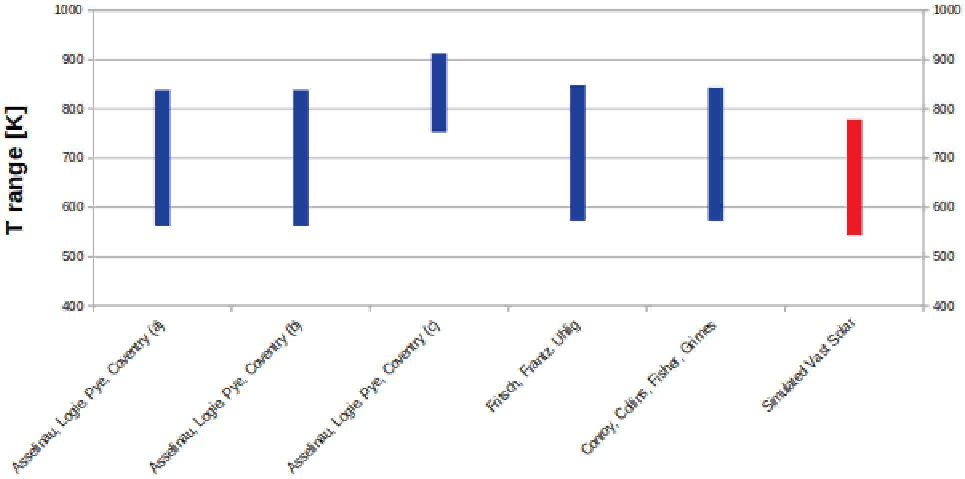


Figure 5.21: *Comparison between temperature range of simulated Vast solar plant and the other configurations presented in this section*

Chapter 6

Conclusions

In this work a CFD simulation of a small scale solar receiver run with liquid sodium has been performed.

In general, heat transfer fluids used in typical applications are Solar Salts or Hitec (Sec.1.2). The aim of this work was to investigate performances of liquid sodium instead of conventional salts: in fact, like all liquid metals sodium is characterized by a very low Prandtl number (in this work the range is $0.0045 < Pr < 0.0065$) and high thermal conductivity. As a consequence, liquid metals show heat transfer coefficients much higher than conventional salts, and that's why they're so interesting in heat transfer application, as explained in Sec.1.3.

In Sec.2.2 is described how the software SolarPilot has been used to generate a solar field made of 700 two-axis tracking heliostats. Each heliostat is very small, with only $3.6m^2$ of reflective area. Also the receiver is compact - especially if compared to conventional central receiving technologies - a total of $2m^2$ absorbing area. In particular, the simulated receiver is made by 20 absorbing tubes $2.5m$ long and with $25mm$ diameter, and placed at $27m$ from the ground. The very limited dimensions of sodium receiver make possible to reach high concentration ratio (1260) even with small reflective area, ensuring a significative reduction of capital and operational costs. The solar field is designed with a nominal DNI of 950 W/m^2 and it generates a peak heat flux of 1.47 MW/m^2, with an average of 0.76 MW/m^2. Then, with the use of SolTrace, a ray-tracing simulation has been done to obtain the distribution of sun rays over absorbing surfaces. In this way it has been possible to generate tables containing spatial coordinates of intersection points between traced rays and tube surfaces.

After that, the proper CFD simulation of the receiver has been run with the open-source software OpenFOAM. In Chapter 3 and 4 is reported a detailed description of meshing process, done with the application *snappyHexMesh*, together with a complete setup of the multiregion case.

All boundary conditions applied to the case are described in details (Sec.4.1), with particular care to heat flux boundary condition on tubes external patches. After setting up the case, the simulation has been run with *chtMultiRegionSimpleFoam*, a steady-state solver.

The CFD simulation shows how liquid sodium flow is heated up from $270°C$ (at the inlet) to an outlet bulk temperature of $505°C$, with a $235°C$ increase. Data about velocity and pressure drops inside the receiver have been reported as well.

The flow is turbulent with Re number in the range between $19000 < Re < 60000$, depending on the flow distribution inside each tube. Nu numbers inside tubes have been estimated, and the results compared to empirical correlations. As expected from theory, Nu is quite low ($6 < Nu < 8$) meaning that heat transfer mechanism is dominated by conduction.

Also, estimation of heat transfer coefficient h show the goodness of sodium thermal performances with values in the range of $22000 < h < 28000\ W/m^2K$. This means an h from two to five times better than conventional salts at the same temperatures.

Thermal net power absorbed by the fluid is $817.8kW$, with a whole receiver efficiency of 90.24%. Low external surface of tubes helps in reducing heat losses, and together with high heat transfer coefficients ensure an high thermal efficiency. So the simulation highlighted how through the employment of liquid sodium it's possible to obtain good performances even with very reduced absorbing area and solar field. In particular, sodium receiver are best suited for high-temperature, high-flux applications, with performances growing with heat flux intensity. Heat flux is precisely the biggest limitation in downsized solar fields: the trouble in obtaining high values of power per m^2 prevents a full exploitation of sodium characteristics, which could ensure even higher outlet temperatures and efficiencies. For these reasons, it could be productive to test sodium as HTF also with bigger solar fields and high-flux configurations.

However, it should be remembered that liquid sodium presents safety issues in case of leakage, in particular at high temperature, so operating and maintenance procedures have to be specifically designed for sodium plants. Moreover it is not an ideal fluid for storage purposes, having a small energy density.

In conclusion, liquid sodium has proven himself as a viable and promising alternative to Solar Salts in solar applications, even though research and investments are still needed to solve mentioned limits.

Appendices

Appendix A

Attached Listings

Listing A.1: Boundary conditions applied to heated side of tubes

```
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //
dimensions      [ 0 0 0 1 0 0 0 ];

internalField   uniform 543;

boundaryField
{
 tubeheat1_to_header_in
  {
   type   compressible::turbulentTemperatureCoupledBaffleMixed;
   value  $internalField;
   Tnbr   T;
   kappaMethod   solidThermo;
  }

  tubeheat1_to_header_out
  {
   type   compressible::turbulentTemperatureCoupledBaffleMixed;
   value  $internalField;
   Tnbr   T;
   kappaMethod   solidThermo;
  }

  tubeheat1_to_domain2
  {
   type   compressible::turbulentTemperatureCoupledBaffleMixed;
   value  $internalField;
   Tnbr   T;
   kappaMethod   solidThermo;
  }

  tubeheat1_to_domain0
  {
    type                groovyBC;
    valueExpression     ”0”;
    gradientExpression  ”gradT”;
    fractionExpression  ”0”;
    value               uniform 0;
   #include ”include/flux”
  }

  tubeheat1_to_tubeadiab1
  {
    type   compressible::turbulentTemperatureCoupledBaffleMixed;
    value  $internalField;
    Tnbr   T;
    kappaMethod   solidThermo;
  }
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //
```

Listing A.2: Included functions for reading tables and defining variables

```
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //
storedVariables
 (
  {
   name storedT;
   initialValue "300";
  }
 );

variables
 (
  "oldT=storedT;"
  "storedT=T;"
  "heatflux=flux(pos().z,pos().x);"
  "k=23.3;"
  "alfa=0.95;"
  "hconv=5;"
  "sigma=5.67e-8;"
  "epsilon=0.8;"
  "Tamb=300;"
  "Tsky=283;"
  "hrad=sigma*epsilon*(pow(oldT,2)+pow(Tsky,2))*(oldT+Tsky);"
  "gradT=1000*alfa/k*(heatflux)+hconv/k*(Tinf-T)+hrad/k*(Tsky-T);"
 );

lookuptables2D
 (
  {
    name flux;
    outOfBounds clamp;
    fileName "$FOAM_CASE/flux1.data";
    file     "$FOAM_CASE/flux1.data";
  }
 );
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //
```

Listing A.3: ControlDict dictionary

```
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //
application       chtMultiRegionSimpleFoam;
startFrom         latestTime;
startTime         0.01;
stopAt            endTime;
endTime           16000;
deltaT            1;
writeControl      adjustableRunTime;
writeInterval     500;
purgeWrite        0;
writeFormat       ascii;
writePrecision    7;
writeCompression  off;
timeFormat        general;
timePrecision     6;
runTimeModifiable true;
adjustTimeStep    yes;

libs ("libOpenFOAM.so"
       "libsimpleSwakFunctionObjects.so"
       "libswakFunctionObjects.so"
       "libgroovyBC.so"
       "libsimpleFunctionObjects.so"
       "libgroovyStandardBCs.so"
       "libswakThermoTurbFunctionPlugin.so"
      ) ;

functions
{
 residuals
 {
  #includeEtc "caseDicts/postProcessing/numerical/residuals.cfg"
  fields        (p_rgh U h k omega);
  region             domain2;
 }

 yPlusPipe1
 {
  #includeEtc "caseDicts/postProcessing/fields/yPlus"
  region   domain2;
 }

 conv_loss
 {
  type swakExpression;
  valueType patch;
  region tubeheat1;
  patchName tubeheat1_to_domain0;
  outputControlMode timeStep;
  outputInterval 1;
  verbose true;
  variables  (
              "hconv=5;"
              "Tinf=300;"
              "hl_conv=hconv*(T-Tinf)*area();"
             );
 expression  "hl_conv";
 accumulations (sum);
 writeStartTime no;
 }

 rad_loss
 {
  type swakExpression;
  valueType patch;
  region tubeheat1;
  patchName tubeheat1_to_domain0;
  outputControlMode timeStep;
  outputInterval 1;
  verbose true;
  variables  (
              "eps=0.8;"
              "sig=5.67e-08;"
              "Tsky=283;"
              "hl_rad=area()*eps*sig*(pow(T,4)-pow(Tsky,4));"
                        );
 expression  "hl_rad";
 accumulations (sum);
 writeStartTime no;
 }
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * *  //
```

List of Figures

List of Tables

Bibliography

[1] (2019) ”BP Statistical Review of World Energy”, pp.8-11

[2] O.Behar, A.Khellaf, K.Mohammedi (2013) ”A review of studies on central receiver solar thermal power plants”, pp.12-39

[3] K.Lovegrove, W.Stein (2012) ”Introduction to concentrating solar power (CSP) technology”, pp.11-13

[4] W.B.Stine, R.W.Harrigan (1986) ”Solar energy systems design”, chapters 10.1-10.2

[5] K.W.Battleson (1981) ”Solar power tower design guide: solar thermal central receiver power systems, a source of electricity and/or process heat”, pp.21-62

[6] S.Kuravi, J.Trahan, D.Y.Goswami, M.M.Rahman, E.K.Stefanakos (2012) ”Thermal energy storage technologies and systems for concentrating solar power plants”, pp.4-11

[7] M.J.Bignon (1980) ”The influence of the heat transfer fluid on the receiver design”, pp. 99-109

[8] J.Pacio, A.Fritsch, C.Singer, R.Uhlig (2013) ”Liquid metals as efficient coolant for high-intensity point focus receivers”, pp.4.1-4.3

[9] X.Zhuang, X.Xu, W.Liu, W.Xu (2019) ”LCOE analysis of tower concentrating solar power plants using different molten-salts for thermal energy storage in China”, pp.10-12

[10] N.Boerema, G.Morrison, R.Taylor, G.Rosengarten (2012) ”Liquid sodium versus Hitec as a heat transfer fluid in solar thermal central receiver systems”, pp.2295-2299

[11] J.K.Fink, L.Leibowitz (1979) ”Thermophysical properties of sodium”, pp. 7-124

[12] V.Sobolev (2011) ”Database of thermophysical properties of liquid metal coolants for GEN-IV”

[13] V.P.Luster, K.F.Freudenstein (1996) ”Feedback from Practical Experience with Large sodium Fire Accidents”, pp.233-237

[14] J.C.Malet (1996) ”Ignition and combustion of sodium-fire consequences-extinguishment and prevention”, pp.13-14

[15] P.K.Falcone (1986) ”A handbook for solar central receiver design”, pp.2.1/2-2.17/7, pp.2.3/15-2.3/26

[16] G.Cammi (2015) ”Numerical analysis of a solar tower receiver tube operated with liquid metals”, pp.12-14

[17] D.Taler (2016) ”Heat transfer in turbulent tube flow of liquid metals” pp.152-156

[18] J.Coventry, C.Andraka, J.Pye, M.Blanco, J.Fisher (2015) ”A review of sodium receiver technologies for central receiver solar power plants”, pp.749-754

[19] J.Flesch, K.Niedermeier, A.Fritsch, D.Musaeva, L.Marocco, R.Uhlig, E.Baake, R.Buck, T.Wetzel (2012) ”Liquid metals for solar power systems”, pp.2-4

[20] Vast Solar (2016) ”Vast Solar Jemalong hybrid solar park: 50MW photovoltaic (PV) plant”, pp.5-6

[21] ARENA (2014) ”CSP pilot plant” https://arena.gov.au/projects/csp-pilot-plant/

[22] NREL (2017) ”Jemalong solar thermal station” https://solarpaces.nrel.gov/jemalong-solar-thermal-station/

[23] C.Wood, k.Drewes (2019) ”Vast Solar: improving and reducing costs and risks using hogh temperature modular arrays and sodium heat transfer fluid”, pp. 1-5

[24] J.Coventry, P.Burge (2016) ”Optical properties of Pyromark 2500 coatings of variable thicknesses on a range of materials for concentrating solar thermal applications”, pp.3-4

[25] R.Abdiwe, M.Haider (2015) ”Investigtions on heat loss in Solar Tower Receiver with wind speed variation” , pp.163

[26] G.J.Kolb (2013) ”An evaluation of possible next-generation high-temperature molten-salt power towers”, pp.11-14

[27] B.Fabritius, G.Tabor (2015) ”Improving the quality of finite volume meshes through genetic optimisation”, pp. 5-8

[28] J.M.McDonough (2007) ”Introductory lectures on turbolence”, pp.54-57, pp.59-62

[29] J.Liu, Y.He, X.Lei (2019) ”Heat-transfer characteristics of liquid sodium in a solar receiver tube with a nonuniform heat flux”, pp.6-14

[30] C.K.Ho, B.D.Iverson (2012) ”Review of high-temperature central receiver designs for concentrating solar power”, pp. 839-840

[31] Special Metals Corporation (2004) ”INCOLOY alloy 800” https://www.specialmetals.com/assets/smc/documents/alloys/incoloy/incoloy-alloy-800.pdf

[32] L.Marocco, G.Cammi, J.Flesh, T.Wetzel (2015) ”Numerical analysis of a solar tower receiver tube operated with liquid metals”, pp.24-28 (cit. on pp.)

[33] M.B.Hermosilla (2016) ”Study of heat and mass transfer applications in the field of engineering by using OpenFOAM”, pp.83-99

[34] M.Ciabotti (2016) ”Numerical analysis of heavy liquid metal heat transfer experiment in a 19-rod bundle with grid spacers”, pp.19-28

[35] B.F.W.Gschaider (2013) ”The incomplete Swak4Foam reference”, pp.23-24

[36] Bureau of Resources and Energy Economics (2012) ”Australia Energy Resource Assessment”, Chapter 10, pp.271-278

[37] M.J.Wagner, T.Wendelin (2018) ”SolarPILOT: A power tower solar field layout and characterization tool”, pp.3-10

[38] T.Wendelin, A.Dobos, A.Lewandowsky (2013) ”SolTrace: a ray-tracing code for complex solar optical systems”, pp.4-7

[39] A.Jackson (2012) ”A comprehensive tour of SnappyHexMesh”

[40] C.J.Greenshields (2019) ”OpenFOAM user guide, version 7”

[41] D.L.Siebers, J.S.Kraabel (1984) ”Estimating convective energy losses from solar central receivers”, pp.

[42] R.Stieglitz, (2007) ”Handbook on Lead-bismth Eutectic Alloy and Lead Properties,Material Compatibility,Thermal-hydraulics adn Technologies”, Chapter 10, pp.648-650, 665-670

[43] J.A.Duffie, W.A.Beckman (2013) ”Solar engineering of thermal processes” Fourth ed. , pp.12-14

[44] C.K.Ho, B.D.Iverson (2012) ”Review of high-temperature central receiver designs for concentrating solar power”, pp. 835-846

[45] G.Manzini, F.Parozzi (2012) ”Sodium safety issues in in nuclear power plants”, pp.2-4

[46] C.A.Asselinau, W.Logie, J.Pye, J.Coventry ”Limits of the cylindrical absorber design for a sodium receiver”, AIP Conference Proceedings 2033, 040006 (2018); pp. 4-8

[47] A.Fritsch, C.Frantz, R.Uhlig (2019) ”Techno-economic analysis of solar thermal power plants using liquid sodium as heat transfer fluid”, pp. 159-161

[48] T.Conroy, M.N.Collins, J.Fisher, R.Grimes (2000) ”Levelized cost of electricity evaluation of liquid sodium receiver designs through a thermal performance, mechanical reliability, and pressure drop analysis”, pp. 479-481

[49] E.Marchi, A.Rubatta (1999) ”Meccanica dei fluidi, principi e applicazioni idrauliche”, pp. 418-421

[50] T.L.Liu, W.R.Liu, X.H.Xu (2017) "Properties and heat transfer coefficients of four molten-salt high temperature heat transfer fluid candidates for concentrating solar power plants" IOP Conf. Ser.: Earth Environ. Sci. 93 012023, pp.4-6

[51] N.V.Khartchenko, V.M.Kharchenko (2013) "Advanced Energy Systems" Second Edition, pp. 437

[52] N.Bartos, J.Fischer (2015) "Experiences From Using Molten Sodium Metal As A Heat Transfer Fluid In Concentrating Solar Thermal Power Systems", pp.2-4

[53] A.O'Donovan (2013) "Performance Characteristics of a Novel Modular Air-Cooled Condenser", ACC User Group 5th annual meeting

[54] J.Fang, N.Tu, J.Wei (2014) "Effects of Absorber Emissivity on Thermal Performance of a Solar Cavity Receiver", pp.7-9

[55] J.H.Lienhard IV, J.H. Lienhard V (2019) "A Heat Transfer Textbook" 5th Edition, pp.349-357

[56] K.Lovegrove, W.Stein (2012) "Concentrating Solar Power Technology: Principles, Developments and Applications", pp.19-21