Using CFD and ray tracing to estimate the heat losses of a tubular cavity dish receiver for different inclination angles

Craig, K.J.*, Slootweg, M., Le Roux, W.G., Wolff, T.M. and Meyer, J.P.
*Author for correspondence
Department of Mechanical and Aeronautical Engineering,
University of Pretoria, Pretoria 0002 South Africa,
E-mail: ken.craig@up.ac.za

ABSTRACT

The process of obtaining an accurate estimate of the heat losses of a tubular cavity receiver absorbing concentrated solar energy from a parabolic dish at various inclination angles and wind speeds is described. Computational fluid dynamics (CFD) was used to simulate the conjugate heat transfer of the absorbed solar radiation to the heat transfer fluid while considering thermal radiation as well as forced and natural convective heat losses. Validation is performed against an experimental heating test at full-scale using heated air and measured wind conditions. On-sun conditions were modelled using the ray-tracing software, SolTrace, adapted for complex geometry receivers using ANSYS mesher and user coding. A 200 million ray result was found to be ray and mesh independent for a meshed receiver surface containing 30 000 elements. The SolTrace heat flux distribution was implemented as a volumetric source in ANSYS Fluent employing user-defined functions. The losses due to thermal radiation out of the cavity, and due to natural convection (using the buoyancy-driven mechanism afforded by gravity and the ideal gas formulation) and forced convection (due to the atmospheric wind) are presented. For the dish considered, 40-50% of the absorbed solar power was transferred to the heat transfer fluid for dish orientations from -45° to 45°, and wind speeds between 0.5 m/s and 4 m/s. This variation was mainly due to a variation in convective heat losses, with thermal radiative heat losses remaining constant at about 30%. The Nusselt numbers from the CFD simulations are compared against correlations from literature.

1. Introduction

1.1 Background

Parabolic dish concentrated solar power (CSP) plants have the benefit of providing a thermal energy solution for remote areas because of their small footprint and high solar concentration. This CSP application differs from other plants (parabolic trough, linear Fresnel, solar tower) in that the receiver with heat transfer fluid is located at a central point that has to move with the reflector to track the sun. This means that typical heat engines for this type of plant include the Stirling engine and potentially a recuperated Brayton cycle (micro-turbine), as considered in the current study. The state of the art of Stirling dishes are summarised by Mancini et al (2003). The recuperated solar dish Brayton cycle has a potential advantage in terms of cost (Mills, 2004), and can be supplemented with thermal storage (Le Roux and Sciacovelli, 2019), a combustion chamber for continuous operation and cogeneration (Le Roux, 2018). In an effort to minimise costs, off-the-shelf turbochargers have been used as turbomachinery in recuperated Brayton cycles, as was done by Visser et al. (2011). Turbochargers are available at relatively low cost from the vehicle industry. Numerous high-temperature solar receivers have been documented in the literature for operation in solar Brayton cycles; however, these solar receivers are typically not optimised for performance in a recuperated Brayton cycle with turbocharger as micro-turbine where the pressure ratios are between 1.3 and 2.5. According to Wang et al (2015), a coiled tube receiver has an advantage in terms of cost and easy installation. The receiver coil design also assists with the reduction of mechanical stresses due to thermal expansion (Heller et al, 2006). According to Harris and Lenz (1983), overall solar collector efficiencies of 60 to 70% can be achieved with advanced systems using open-cavity receivers operating in the range of 773 to 1 173 K.

Le Roux (2015) therefore applied the method of total entropy generation minimisation (optimisation of the global performance), according to Bejan (2006), on the geometries of a low-cost open-cavity tubular receiver and plate-type recuperator of a solar Brayton cycle. With the use of Flownex and SolTrace, Le Roux and Meyer (2016) also showed how a relatively large tube diameter for the open-cavity tubular receiver could be beneficial to the output power of the solar dish Brayton cycle.

Le Roux et al (2014) performed an optimisation study using ray tracing, thermodynamic relations and network heat transfer models. For a low-cost open-cavity tubular receiver and 4.8-m-diameter solar dish, it was found that efficiencies of between 45% and 70% can be achieved with typical mass flow rates of between 0.06 kg/s and 0.08 kg/s and receiver air inlet temperatures of between 900 K and 1 070 K. With the use of SolTrace, the optimum aperture size and tube diameter were identified for a 4.8 m diameter solar dish with 45° rim angle, 1° tracking error and 10 mrad optical error (a low-cost solar dish and tracking system).

The current study aims to assess more accurately the contributions of the different heat transfer loss mechanisms in an open-cavity tubular receiver of a low-pressure solar Brayton cycle.

1.2 Heat losses from dish cavity receivers

Shuai et al (2008) evaluated the radiation performance of various dish cavity receiver shapes using Monte Carlo ray tracing with a spherical receiver having the best radiation performance in the absence of the consideration of natural and forced convection heat losses. Prakash et al (2009) estimated the convective and radiation heat losses using computational fluid dynamics (CFD) simulations of a cylindrical cavity receiver with a wind skirt and compared the results with an experiment. Radiation heat losses were not explicitly simulated and were not dominant at the temperature range studied (below 100 °C), rather, the focus was on the effect of forced convection due to wind as influenced by cavity inclination. Prakash et al (2012) conducted numerical simulation of open cavities (cubic, hemispherical and spherical) for different inclinations and aperture openings, and derived a Nusselt-Rayleigh number correlation involving the inclination angle.

As far as convective heat losses are concerned, Pavlovic and Penot (1991) and Clausing et al (1989) studied mixed convection in an isothermal open cubic cavity experimentally. Pavlovic and Penot (1991) determined Nusselt-Grashof-Reynolds number correlations for different inclinations for temperatures up to 120 °C, while Clausing et al (1989) determined Nusselt-Richardson number correlations for temperatures up to 350 K for various side-facing apertures to the cavity. For these low temperatures, radiation heat losses were not dominant. In addition, the isothermal assumption of these studies limits the applicability of these analyses to high-temperature receivers with significant variation in cavity surface temperature. A comprehensive review of convection heat loss from dish cavity receivers was written by Wu et al (2010). Numerous correlations for convection heat losses were listed in that publication, with notable contributions by Stine and McDonald (1989) and Lovegrove et al (2003) for cylindrical cavities, as investigated in the current study. According to McDonald (1995), the Koenig and Marvin model for natural convection heat loss from high-temperature receivers is valid up to 900 °C. More recently, Abbasi-Shavazi et al (2020) developed a correlation for convection heat loss for a cavity with a non-isothermal surface temperature distribution. As mentioned by Wu et al (2010), the Boussinesq approximation for density as a function of temperature is not valid for high-temperature (>700 °C) receivers, hence in the current study, the more accurate ideal gas law is used for both the internal heat transfer fluid (HTF) flow and the external wind flow as it interacts with the HTF due to the receiver plume exiting into the atmosphere.

1.3 Complex geometry dish receiver modelling

In the current work, the receiver is not exactly cylindrical and is comprised of a winding tube following an oblong cavity. Earlier work using CFD analysis of the dish receiver by Craig et al (2015a, 2015b) employed finite-volume ray tracing (Craig et al, 2015a) and Monte Carlo ray tracing with a crude approximation of the complex geometry using circular elements (Craig et al, 2015b). Recent work by Slootweg et al (2019) and Slootweg (2019) paved the way for using irregular triangular and quadrilateral elements in SolTrace to describe any complex receiver shape accurately, thereby generating an accurate solar absorption distribution in the receiver as input to conjugate heat transfer modelling using CFD. The Slootweg approach was also followed and is described in this paper.

1.4 Modelling approaches

The optical performance of dish receivers is often determined using Monte Carlo ray tracing (e.g. Shuai et al, 2008). The combination of ray tracing and computational fluid dynamics (CFD) has been used by various researchers to investigate the temperature distributions and performance of dish receivers. For example, Li et al (2011) used ray tracing to determine a heat source profile for a cavity receiver with glass cover, while a constant convection coefficient accounted for convection heat losses at the cover surface as evaluated in a conjugate CFD simulation. Wang et al (2014, 2015) combined ray tracing with CFD in the evaluation and inverse design of an impinging dish receiver. External convection heat losses were not considered. Yuan et al (2015) focused on natural convection heat losses for cavity receivers, both for a cubical central receiver (nominal wall temperature of 750 °C) and a cylindrical cavity dish receiver (nominal wall temperature of 450 °C). CFD simulations using Fluent were compared with experimental results and different turbulence models were evaluated; convective heat loss was generally underpredicted. Heated surfaces were simplified as isothermal or constant heat flux boundary conditions and mesh sizes ranged from less than 1 million to about 1.8 million cells.

To get a true reflection of the thermal performance of a dish receiver, the input power (as determined through ray tracing, accounting for absorbed solar irradiation) needs to be quantified and then applied as heat source in a conjugate CFD simulation. This simulation should then account for radiation heat losses, natural convection heat losses as well as forced convection heat losses due to wind. The last two contributions are highly sensitive to the inclination of the receiver as determined by the orientation of the dish. This is the approach followed and quantified in this paper.

1.5 Layout of paper

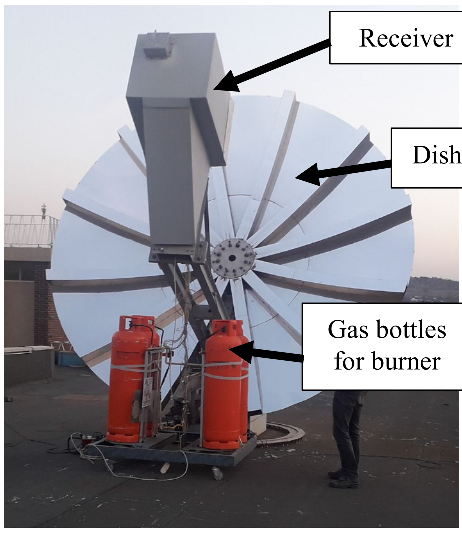
The paper starts by presenting a validation case based on the experimental heating test of a dish receiver using a CFD model that evaluates thermal re-radiation, external forced convection as well as natural convection. After this case, the process of evaluating a solar heat source using ray tracing for complex receiver geometries is described. The resulting volumetric heat source is then implemented into the CFD model, of which results are presented, first for a 0° or upright orientation, and then for a variety of inclination angles to illustrate the variation in heat loss contributions. The results obtained are then compared with correlations from the literature. Finally, conclusions are drawn to complete the paper.

2. Validation case

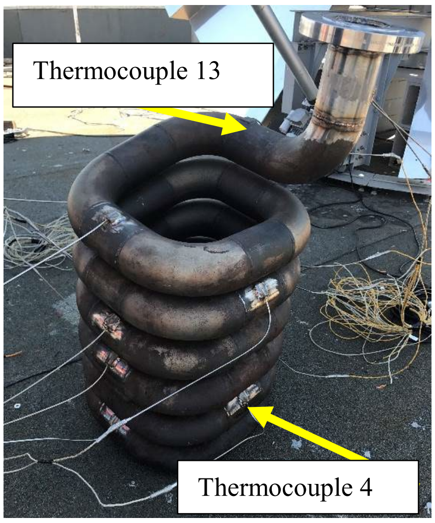
2.1 Experimental set-up and results

An experiment was used for validation of the CFD model (also see a summary of this experiment in Wolff et al, 2018 and Wolff, 2020). In the experiment, the low-cost receiver was heated using a blower and burner at its inlet (see Fig.1d), while temperatures were monitored on the receiver and insulation surfaces as well as the gas inlet and outlet. The experimental set-up is shown in Fig.1 with the locations of thermocouples used to monitor selected surfaces of the receiver tube. Typical results are shown in Fig.2, which include wind speed measured by a local anemometer.

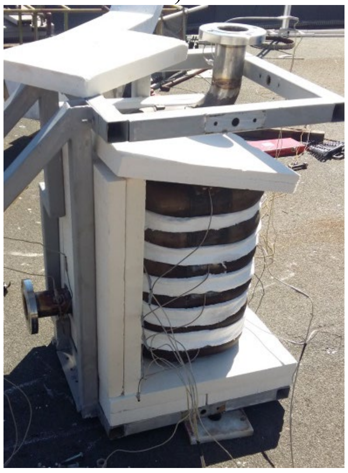
A data set was used as one of the steady-state periods, as indicated in Fig.2. The summary-averaged data for use and comparison in the CFD model is listed in Tab.1 as well as Tab.B.1. Based on an average specific heat of 1 139 J/kg-K, the total heat lost from the hot air during its passage through the receiver was 5 565 W. It was one of the aims of the validation case to quantify the components that made up this total heat loss. Calculations in Wolff et al (2018) using certain assumptions of view factor and surface emissivities provided an estimate of heat lost through thermal re-radiation from the aperture of the receiver, while conduction calculations were used to estimate the heat lost through the insulation material surrounding the tubular receiver, which was transferred to the wind by external forced and natural convection. All these contributors to heat loss were compared with the CFD results to follow.



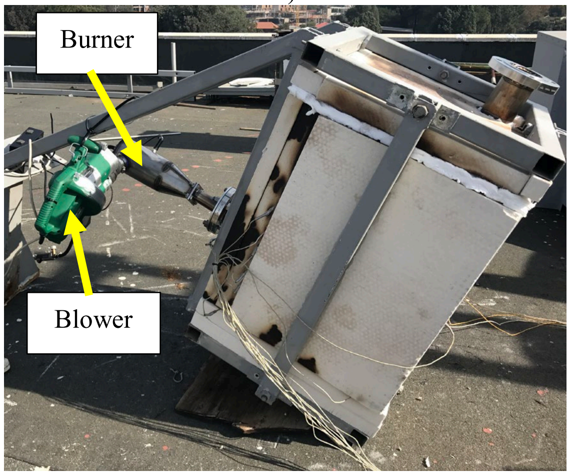
a)



b)



c)



d)

Figure 1 a) Receiver mounted on 4.8 m diameter dish for gas heating test. Note the grey painted cover over the insulation. During the heating test simulated here, the dish faced directly upwards, not as in the photo; b) tubular receiver with selected thermocouple locations; c) tubular receiver while being insulated; d) blower and burner attached to inlet of receiver.

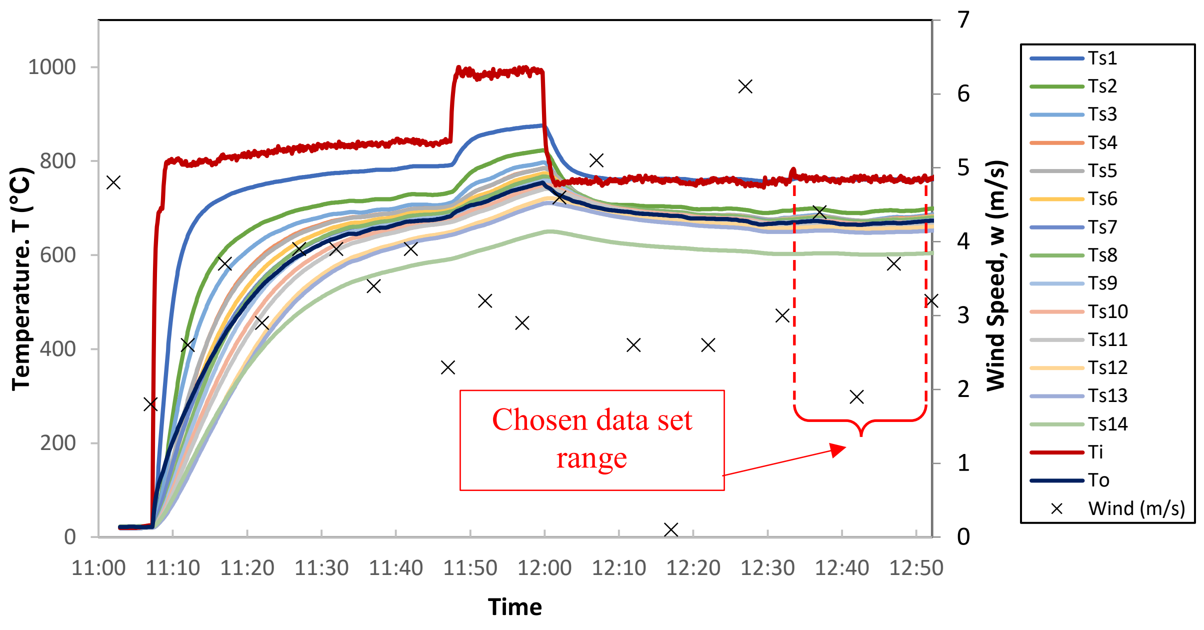


Figure 2 Measured temperatures on the receiver surface (14 thermocouples along the length of the receiver from the inlet to the outlet), inlet and outlet air temperatures (Ti and To), and wind speed on 19 Sept 2017. The chosen steady-state period is also indicated.

Property	**Value**
Receiver tube mass flow rate [kg/s]	0.05323
Inlet temperature (Ts1) [°C]	761.3
Outlet temperature [°C]	669.5
Average wind speed [m/s]	1.9 – 4.4
Environmental temperature [°C]	23

Table 1 Summary of conditions during experiment for upright dish orientation.

2.2 CFD analysis of validation case

From the experimental results (to be discussed in detail below), it was clear that convection heat losses (whether due to the external wind or due to natural convection) accounted for a significant portion of the total heat loss. For this reason, the CFD computational domain needed to be selected with care to account for these mechanisms. In addition, the material models and physics that were resolved had to be adjusted to be able to account for all the convection-related heat transfer mechanisms in conjunction with the radiative and conductive heat transfer.

2.2.1 Computational domain

The computational domain was determined based on the location of the experimental set-up on the roof of the Engineering II building at the University of Pretoria, as shown in Fig.3. Based on measurement data of the SAURAN network (Brooks et al, 2015) station on the roof of the adjacent building, the prevailing wind on the testing date indicated that a side-on wind could be used to simplify the computational domain, as shown in Fig.4. The 5 m upstream fetch corresponded roughly to the start of the roof, which aided in the assumption of a uniform inlet velocity profile. The other boundaries of the domain were somewhat arbitrary in that the roof section was narrow and that no other large obstructions impacted the receiver flow pattern for the studied wind direction. The wake of the dish (simplified for this analysis) was also expected to be thin at this dish orientation.

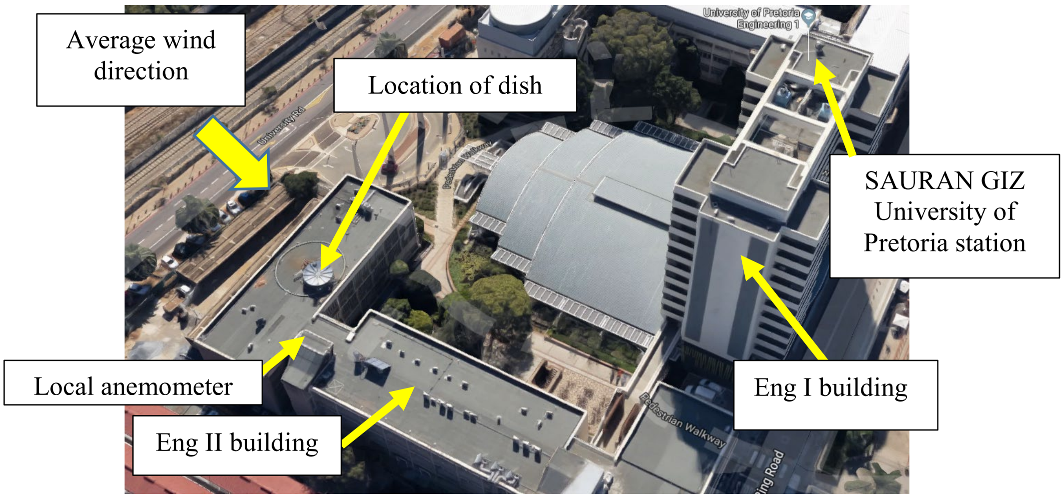


Figure 3 Location of dish on roof of Engineering II building and average wind direction on 19 Sept 2017 (Brooks et al, 2015). Image from maps.google.com

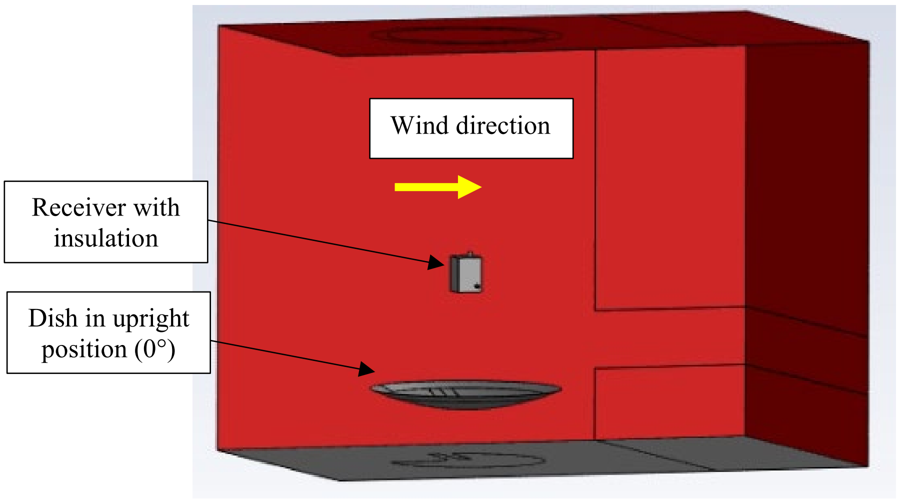
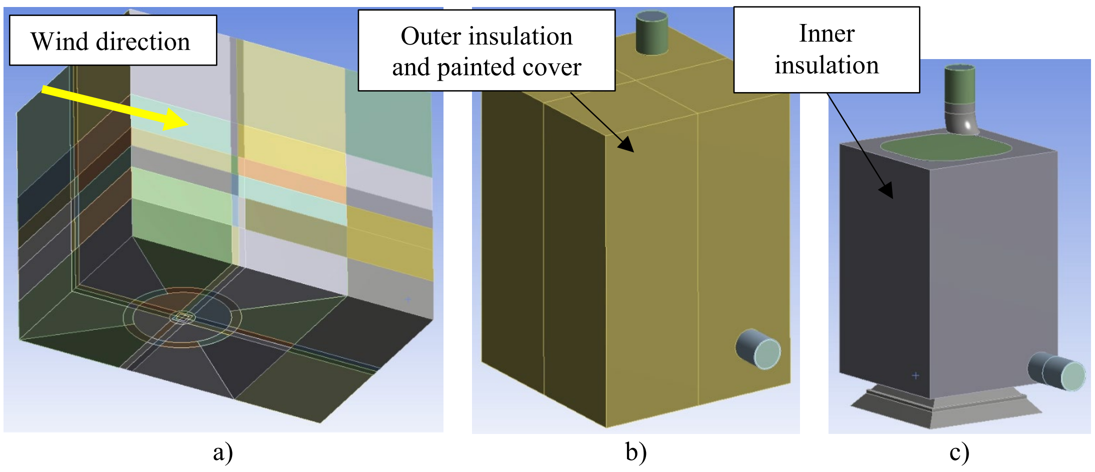
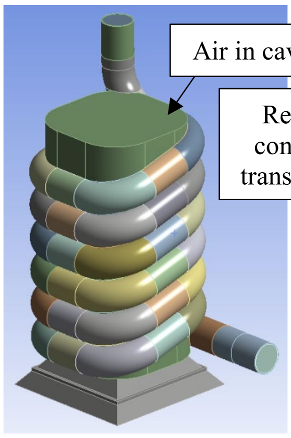


Figure 4 Computational domain (5 m upstream and to sides, 6 m above receiver, 9 m downstream, 1 m clearance below dish) for validation case

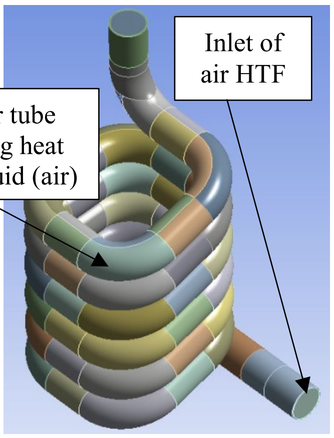
2.2.2 Computational model of receiver

The computational domain was divided into blocks for meshing, as depicted in Fig.5a. The different components of the receiver are displayed in Fig.5b, c (receiver insulation), Fig.5d, e (tube with internal air at HTF and cavity air shown). The tube was segmented to allow for high-quality mapped meshing.

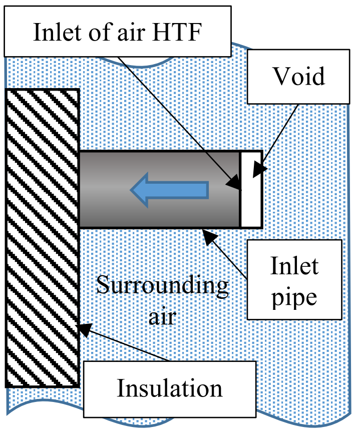




d)



e)



f)

Figure 5 a) Computational geometry of domain; b) receiver outer insulation (fibre board); c) receiver inner insulation (fibre blanket); d) receiver tube with air in cavity; e) receiver tube containing air as heat transfer fluid; f) schematic explanation of inlet boundary condition for HTF with void creating external boundary for inlet condition

2.2.3 Computational mesh

The division of the computational domain allowed for predominantly hexahedral elements for the external domain (Fig.6a) and the tubular sections of the internal fluid (Fig.6b,c). The insulation material and air in the receiver cavity had complex shapes and were meshed using tetrahedral elements. The fluid mesh in the tubular receiver resulted in a maximum y^+ of about four for most of the receiver domain when using a mesh sizing of 2mm for the receiver tubes. The receiver tube mesh density is governed by the overall mesh sizing. A mesh dependency study was performed to investigate the effect of this sizing. For this study, only the tube and the internal heat transfer fluid were considered in the model, with boundary conditions on the inner and outer halves of the pipe approximated by the results obtained for the full model. This involved a mixed boundary condition (radiation and convection) on the inner cavity-facing half of the tube, and a heat flux boundary condition based on the conduction heat loss through the insulation. These settings were based on the values in Tab.1 and a wind speed of 4 m/s. The coarsest mesh considered had a sizing of 4 mm (Fig.6d). The results of this study are shown in Table 2 and Figure 7. The finest three sizes had outlet temperatures that were within a 2% range as shown. Performing a Grid Convergence Index (GCI) study on the 3 mm, 2.5 mm and 2 mm mesh sizes (with a refinement order of about 1.8 based on mesh count) resulted in a GCI asymptotic convergence of 1.0023 for the outlet temperature. From these results, the 2 mm setting was deemed to be fine enough for the subsequent studies. The overall mesh count for the complete CFD model using this size setting was about 42.8 million cells for the validation model. In addition, mesh adaption of this model was performed on the inner boundary of the receiver tube resulting in an overall mesh count of 52 million cells and a maximum y^+ value of 2. This model predicted an outlet temperature and heat loss by radiation that were within 0.1% of that obtained by the 42.8 million cell mesh.

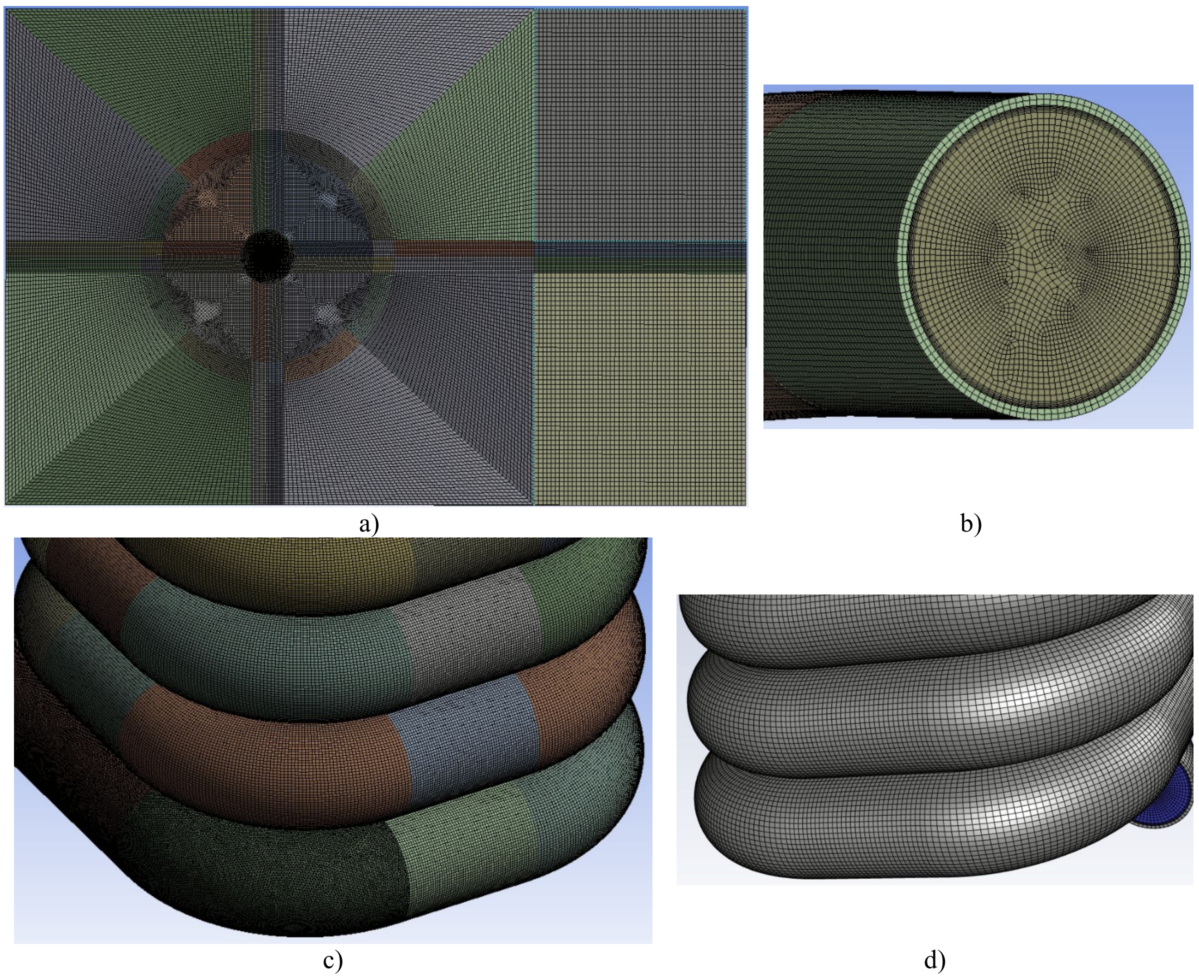


Figure 6 a) Computational mesh of domain (view from bottom); b) internal heat transfer fluid in tube; c) mesh of tubular receiver (2mm size); and d) mesh of tubular receiver (4mm size)

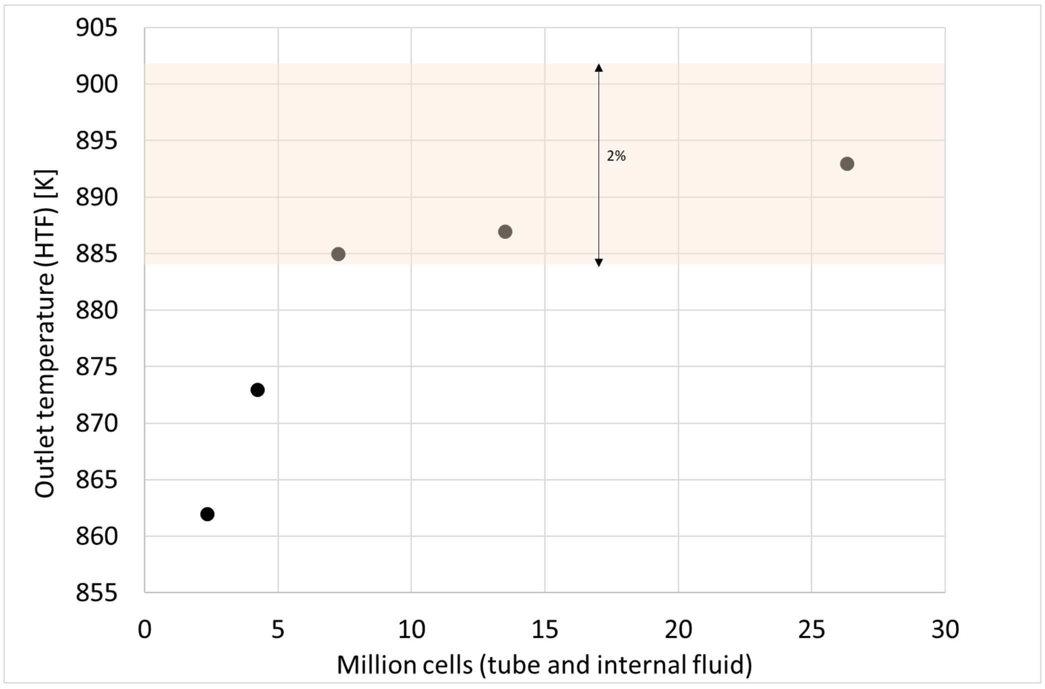


Figure 7 Mesh dependency of receiver outlet temperature [K]. 2% band shown as shaded region around finest mesh result.

Property	Value	Value	Value	Value	Value
Mesh size setting [mm]	1.5	2.0	2.5	3	4
Mesh count [million]	26.3	13.5	7.25	4.22	2.34
HTF outlet temperature [K]	893	887	885	873	862
Pressure drop [Pa]	707	736	768	780	818

Table 2 Results of mesh dependency study. Model of tube and inner fluid only.

2.2.4 Computational settings

The settings are summarised in Tab.3. Turbulent flow was assumed based on both the external and internal conditions (Re = 1E6 based on dish diameter and Re = 2.1E4 based on internal tube conditions). The SST k-ω turbulence model was used for turbulent closure of the steady Reynolds-averaged Navier-Stokes equations, which were solved in double precision using the coupled segregated solver in ANSYS Fluent v2019 R3. Thermal re-radiation was computed using the discrete ordinates (DO) model with 3×3 angular discretisation and 3×3 pixilation, while natural convection was enabled by activating gravity and specifying air as an ideal gas. The influence of the number of angular discretization intervals was determined by varying the phi and theta solid angle increments between two and four in each direction, i.e., 2×2, 3×3 and 4×4, and checking the effect on the HTF outlet temperature and pipe heat losses. From the coarser setting of 2×2 to the finer setting of 4×4, the outlet temperature varied by 0.8% and the pipe heat losses by 2%. The 3×3 setting, used for all the subsequent cases in this paper, was within 0.7% of the pipe heat loss and within 0.04% of the outlet temperature determined by the finer 4×4 setting. Note that these settings for the DO model have profound implications on computational resources required. The 2×2 setting implies that 2×2×8=32 additional equations are solved, while for the 4×4 setting 4×4×8=128 additional equations are solved in addition to the mass conservation, three momentum equations, energy equation and two turbulence transport equations. The memory overhead is also significant.

All the external boundaries except for the ground and inlet wind were specified as outlet boundaries at 0 Pa gauge pressure. The specified operating density was based on an absolute atmospheric pressure of 86.6 kPa and environmental temperature of 23 °C (Brooks et al, 2015), with the reference pressure location for buoyancy-driven flow at the top of the domain. Body-force weighted discretisation was used for the pressure equation. The thermal conductivities of the insulation and stainless steel 316 tube material were specified as functions of temperature.

Finally, the internal air path was connected to the external wind environment through the outlet of the receiver. However, the receiver inlet boundary condition location was enabled by creating a small void between the external domain and the receiver HTF, thereby creating the effect of the blower and burner in the computational domain (see Fig.5f for a schematic explaining this condition). The use of air as medium neglected any combustion products that were present due to the burner gas providing the heat source.

Property	Value
Receiver tube emissivity (highly-oxidised stainless steel at 1000 K) (Çengel and Ghajar, 2015)	0.8
Insulation outer surface emissivity (primer paint at 300 K) (Çengel and Ghajar, 2015)	0.9
Insulation thermal conductivity [W/m.K] (Wolff et al, 2018)]	(0.085, 0.112, 0.145) at (473, 673, 873) K
Operating pressure (abs) [kPa] (Brooks et al, 2015)	86.6
Operating (environmental) temperature [°C] (Brooks et al, 2015)	23
Ground temperature [°C]	45

Table 3 Computational settings, material properties and boundary conditions for validation case

2.3 Results of validation case

The integrated results of the CFD simulation of the validation case are summarised in Tab.4. It shows that the overall heat transfer between the heated air and environment was captured accurately, with a 6 K or 0.9% difference between experiment and CFD outlet temperature at the median wind velocity of

4 m/s. The radiative heat loss was predicted within 7%, while the conduction heat losses through the insulation outer walls compared well. The overall heat loss from the HTF was underpredicted by 7.3% at a simulated wind speed of 4 m/s. The comparison between the experimental thermocouple and simulated temperatures at approximately the same locations is displayed in Fig.8. It shows that the CFD consistently overpredicted the experimental temperatures with an average deviation of 17.2 K (or 1.8% using the average absolute temperature), although a similar trend is shown for most of the receiver tube length. At the inlet, there is a difference in how the insulation was modelled, hence the deviation there. There is a slight effect of wind speed on these temperatures with higher speeds leading to lower temperatures as expected. The root mean square deviation of the experimental measurements during the period considered is small. Possible reasons for the overprediction are the fact that the insulation material did not seal tightly in the experiment and that the temperature on the receiver surface was influenced by the presence of hot air in the gaps (the modelled geometry had no air gaps). Additionally, some heat could have been lost through cracks between the insulation boards, especially at the top of the receiver, thereby aiding natural convection losses and lowering the temperature in the experiment.

Property	**Experiment**	**CFD**		
Wind speed [m/s]	1.9 – 4.4	3	4	5
Mass flow rate [kg/s]	0.05323			
Inlet temperature [°C]	761.3			
Outlet temperature [°C]	670	679	676	673
Average receiver inner-surface temperature [°C]	674[a]	676	673	670
Minimum receiver inner-surface temperature [°C]	602[b]	604	593	590
Maximum receiver inner-surface temperature [°C]	694[c]	698	696	696
Average receiver insulation outer-surface temperature [°C]	72.2*	65.1	61.5	58.9
Radiation heat loss through aperture [W]	2 764[d]	2 598	2 570	2 559
Heat loss from sides of insulation [W]	796	771	771	771
Heat loss from top of insulation [W]	116	150	150	150
Heat loss from bottom of insulation [W]	157	152	151	150
Total conduction heat loss [W]	1069[e]	1 073	1 072	1 071
Heat loss from heat transfer fluid [W]	5 565	5 003	5 158	5 368
Pressure drop of HTF [Pa]	-	971	969	969
External convection heat loss [W]	1732	1 332	1 543	1 738

* side surfaces only [a] Average of experimental thermocouple values [b] Outlet thermocouple (no.14)
[c] Inlet thermocouple (no.2) [d] Based on average thermocouple temperature [e] Appendix B

Table 4 Comparison between experimental and CFD results – Validation case

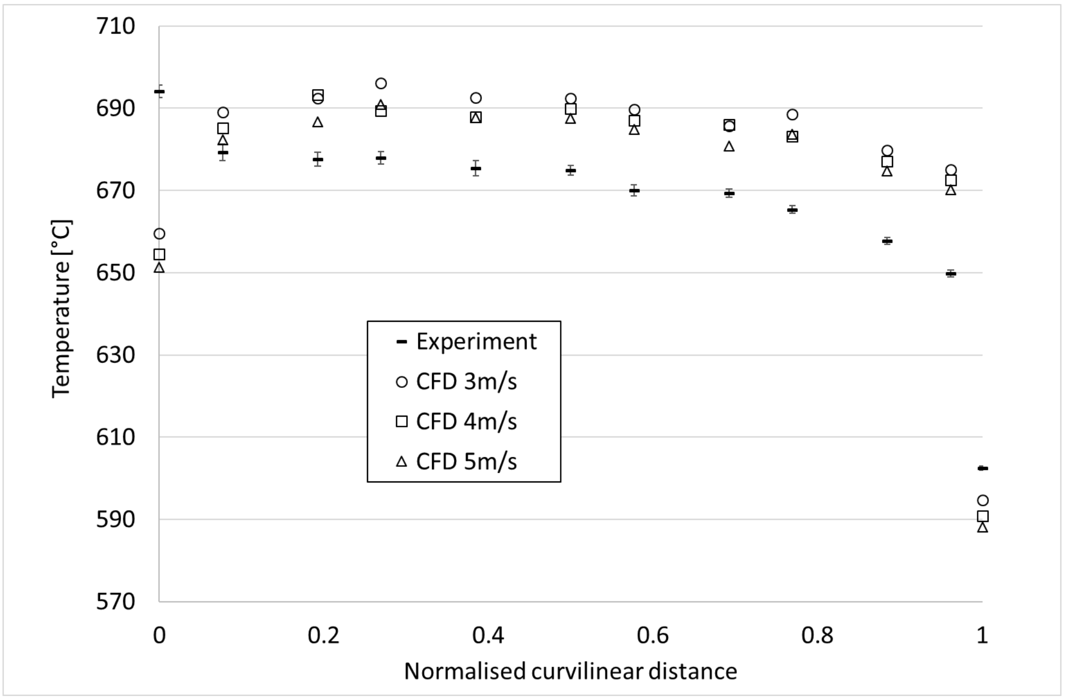
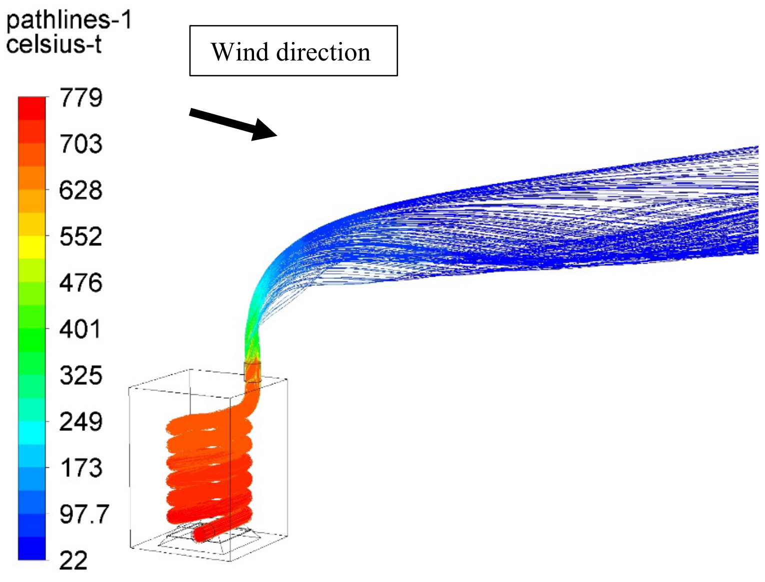


Figure 8 Comparison between measured and simulated temperatures [°C] on receiver surface (for 12 thermocouples) as a function of normalised curvilinear distance along the receiver for different wind speeds in the simulation. The RMS range is shown for the experimental results.

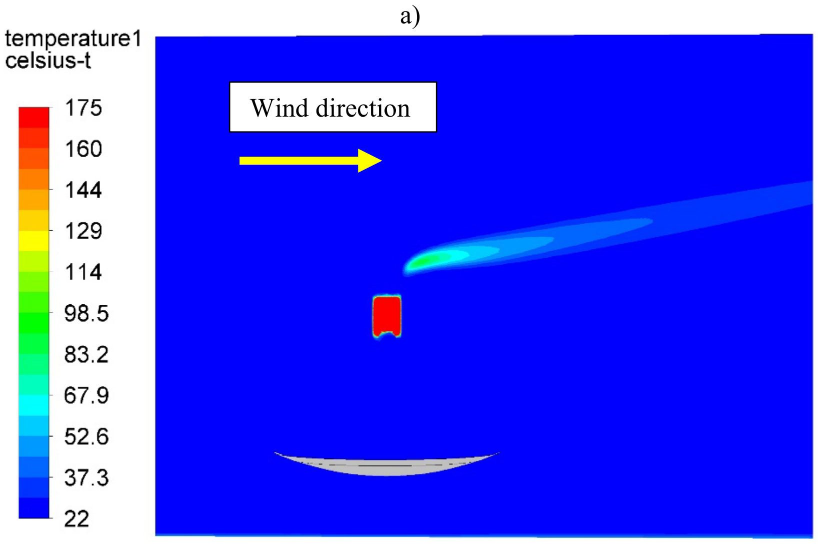
However, on the whole, the CFD validation was successful and it is assumed that the method (which included radiative as well as external forced and natural convection) can be applied to other solar heat source scenarios, as discussed in the next section.

The detailed CFD results are presented next. The interaction between the hot air emanating from the tubular receiver and the external wind caused the plume to be bent and to follow the wind direction, as depicted in Fig.9 in the form of pathlines (Fig.9a) and temperature contours on the central y = 0 plane (Fig.9b).

Figure 10 illustrates the cooling of the hot air due to heat losses, both through conduction into the insulation and through convection and radiation through the receiver aperture. The wind direction (shown in Fig.10a) resulted in a slight asymmetric convection pattern at the mouth or aperture of the receiver, while buoyancy effects resulted in some stratification of the air trapped in the cavity.



a)



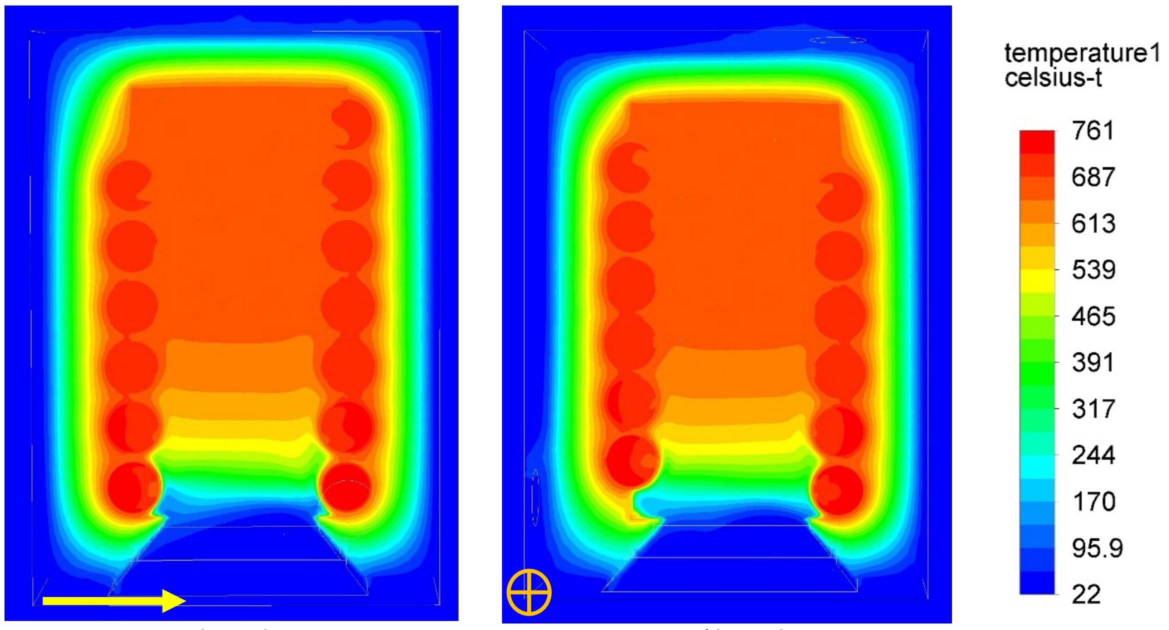
b)

Figure 9 Plume of receiver depicted using a) flow pathlines coloured by temperature [°C] and b) temperature contours (clipped at 175 °C) on y = 0 plane – Validation case, 4 m/s

The variation in temperature on the outer insulation surfaces is shown in Fig.11, with the regions surrounding the inlet and outlet pipes being the hottest, as expected. There was a slight secondary effect of convection cooling due to the wind direction. The wind formed a wake behind the receiver and the influence of the dish was not significant at this orientation.

The convection and radiation heat losses were dominant in the locations of highest surface temperature close to the inlet, as confirmed in the surface temperature contours of Fig.12 and total and radiation surface heat flux contours in Fig.13. The latter is reported as positive values, while the total heat losses are reported as negative in ANSYS Fluent.

As a final CFD result of the validation case, the incident radiation contours as solved by the discrete ordinates method are depicted in Fig.14. This radiation signature is useful in understanding the view factor experienced by the high-temperature sections of the receiver and their role in contributing to high radiation heat losses.



a) y = 0 b) x = 0

Figure 10 Temperature contours [°C] in a) y = 0 and b) x = 0 plane through receiver – Validation case, 4 m/s

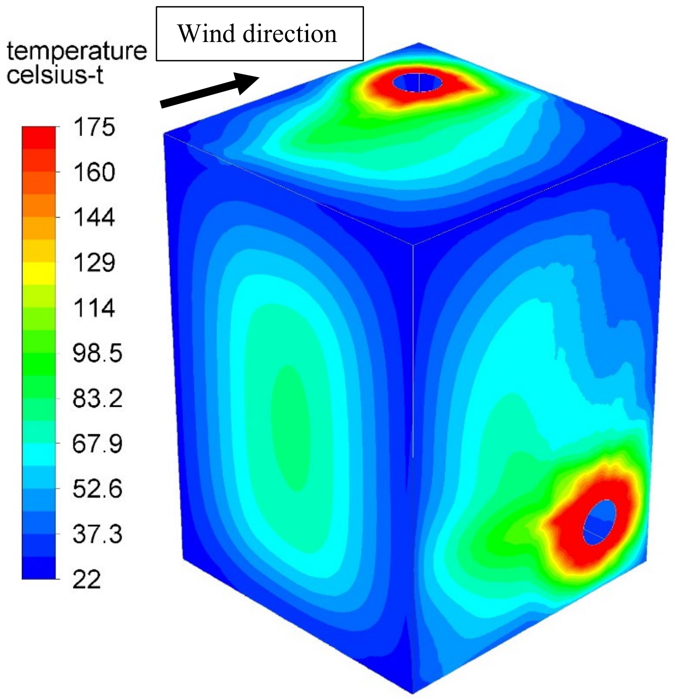


Figure 11 External temperature distribution of receiver insulation outer surface (clipped at 175 °C) – Validation case, 4 m/s

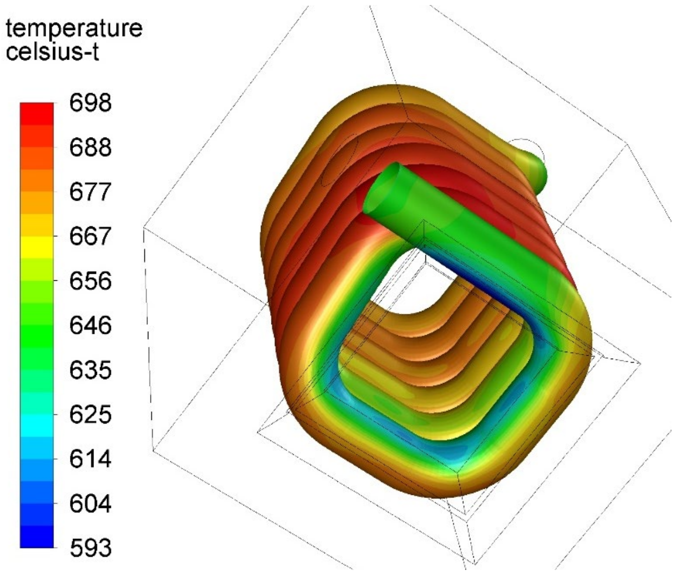


Figure 12 Receiver surface temperature contours [°C] – Validation case, 4 m/s

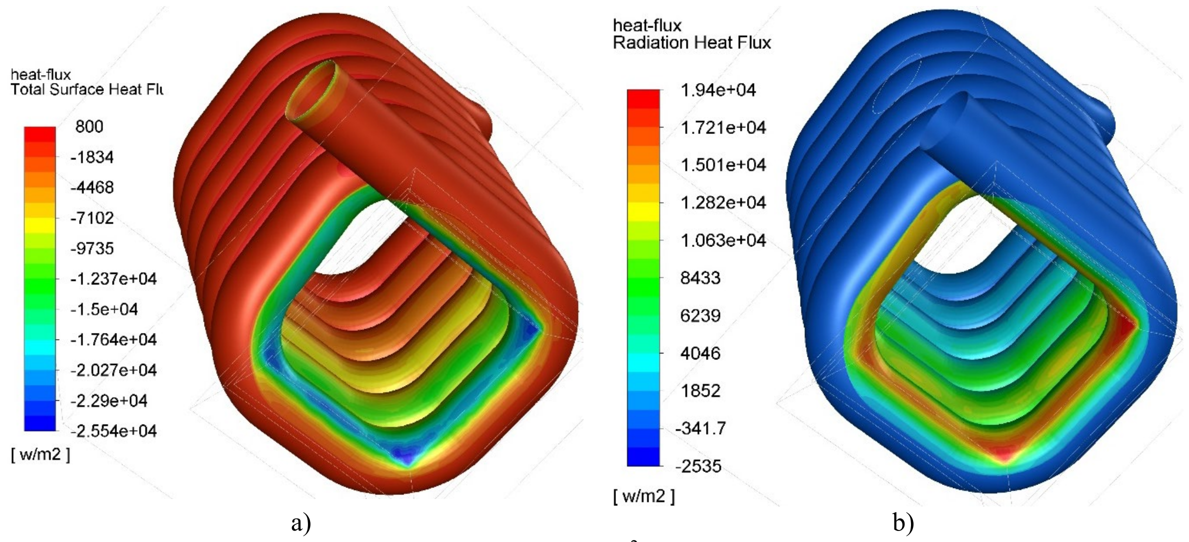


Figure 13 a) Total and b) radiative heat flux [W/m^2] on receiver tube walls – Validation case, 4 m/s

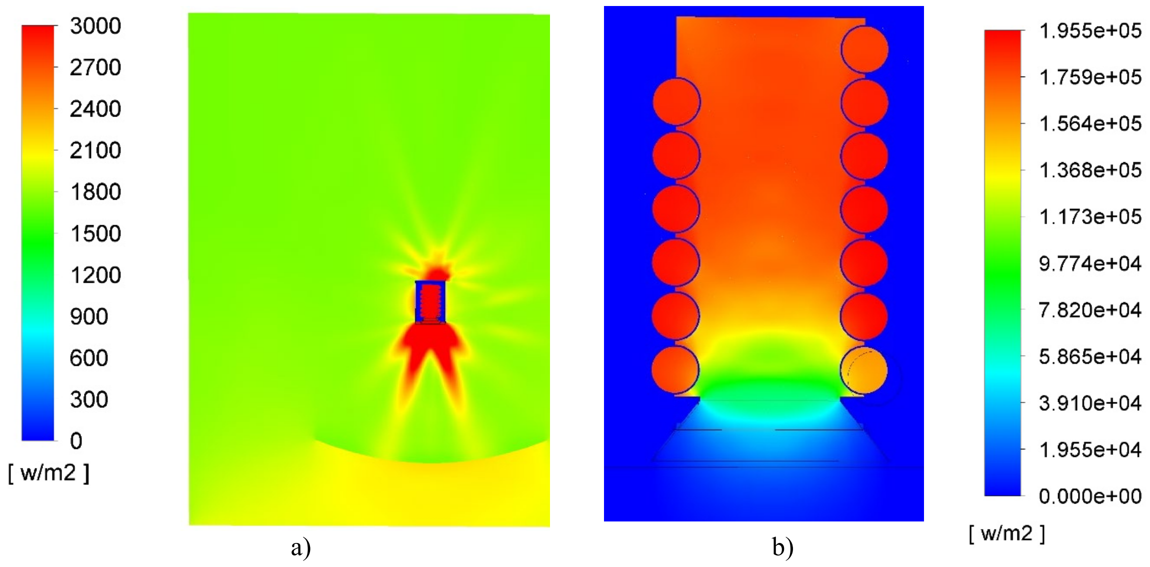


Figure 14 Incident radiation [W/m^2] on y=0 plane a) overall, clipped at 3 kW/m^2, b) local in receiver cavity. Validation case, 4 m/s

3. Solar heat source using ray tracing

3.1 Complex geometry modelling in SolTrace

The method developed by Slootweg et al (2019) and Slootweg (2019) was used here to replace the hot burner inlet with an on-sun condition, i.e. the HTF air entered at a lower temperature and then was heated by the concentrated solar irradiation of the dish focused on the receiver aperture. SolTrace (Wendelin and Dobos, 2013) was used as the ray-tracing software and was based on the Monte Carlo method. The numerical solution approach involved using ANSYS mesher to generate a meshed or tessellated representation of the tubular receiver's surfaces exposed to the concentrated sun. Each of these mesh elements was then converted and implemented as a separate element(s) in SolTrace by using the available irregular triangular elements. A schematic illustrating the method is given in Fig.15 with the equations for the transformation explained in Appendix A. The code used to convert the mesh file into input files for SolTrace is given in Appendix C, and the script to simulate the model in SolTrace is given in Appendix D. More details regarding the method can be found in Slootweg (2019).

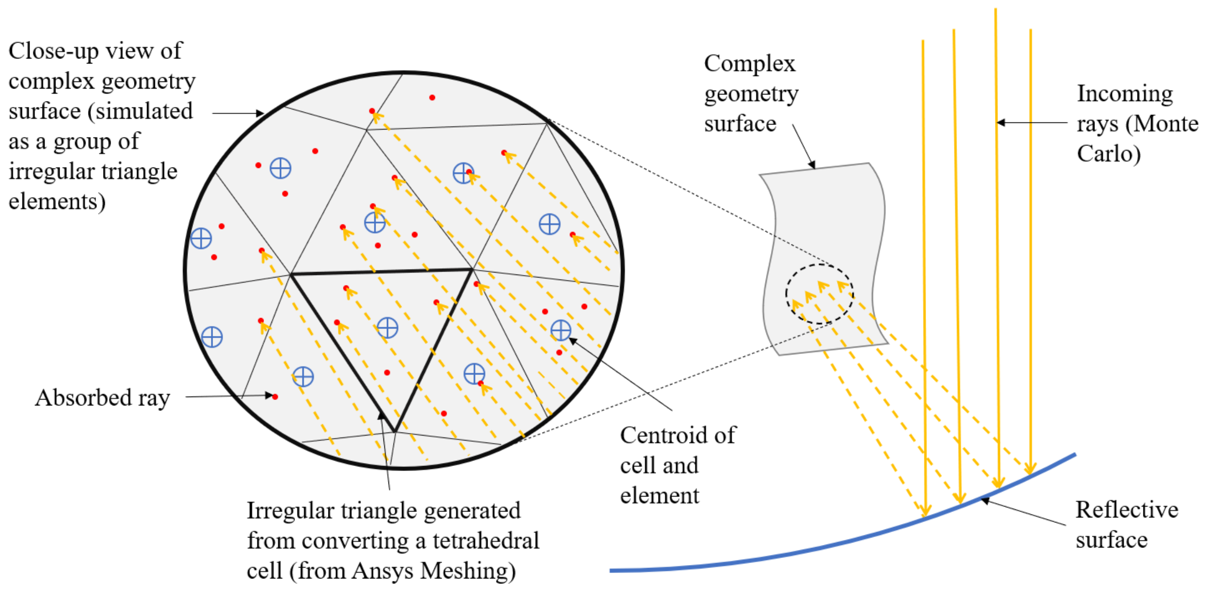

Close-up view of complex geometry surface (simulated as a group of irregular triangle elements)
Complex geometry surface
Incoming rays (Monte Carlo)
Absorbed ray
Centroid of cell and element
Irregular triangle generated from converting a tetrahedral cell (from Ansys Meshing)
Reflective surface

Figure 15 Schematic illustrating the method developed by Slootweg et al (2019) and Slootweg (2019), where a complex geometry surface is approximated by converting a computational mesh to irregular triangular elements to be used in SolTrace. A flux distribution is determined by applying the average absorbed ray of an element and applying the average value to the centroid of the element.

3.2 Ray-tracing model set-up

A side view of the SolTrace model is shown in Fig.16. The mounting arm of the receiver (which would cast a shadow) was omitted from this model. The geometrical parameters are listed in Tab.5 and the optical properties in Tab.6. A maximum sun condition with a direct normal irradiation (DNI) value of 1 000 W/m^2 and a pill-box sunshape with a 9.3 mrad angle were assumed. Typical rays are shown in Fig.17 with the shadow cast by the receiver clearly visible.

Dimension	**Value**
Receiver width [m]	0.614
Receiver height [m]	0.916
Aiming height/focal length [m]	2.897
Dish diameter [m]	4.8

Table 5 Dimensions of dish and receiver in SolTrace

	Reflectivity	**Slope error (mrad)**	**Specularity error (mrad)**
Dish	0.8	3.0	3.0
Receiver tubes	0.2	0.71	100
Inner insulation surfaces	0.3	5	100
Insulation surfaces at the aperture	0.76	1	0.2
External insulation surfaces	0	0.71	0.14

Table 6 Optical properties of dish and receiver in SolTrace

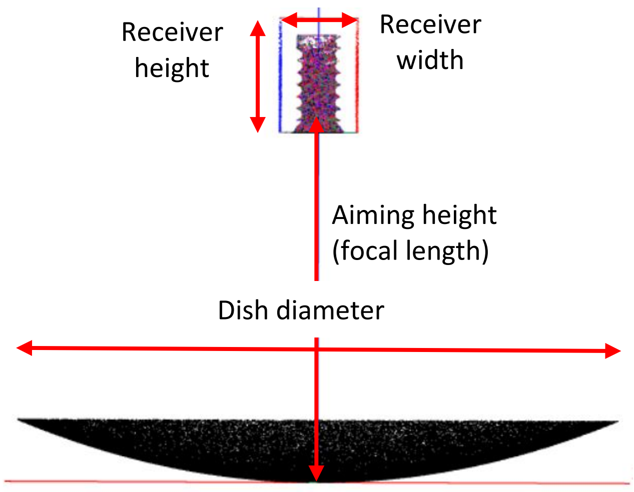


Figure 16 SolTrace model showing ray hits on dish and receiver

3.3 Ray and mesh independence study

The density of the ANSYS mesh had an influence on the number of ray intersections required in SolTrace for a converged solution. A separate investigation was conducted to determine the number of ray intersections required for ray independence. A summary of this study is shown in Figures 18 and 19. In Fig.18, the absorbed radiative flux is displayed as the ray count is increased for a mesh count of 31 456 elements. Of these elements, 20 672 were on the pipes (as shown in Fig.18), while the rest were used for the insulation surfaces between the pipes and at the receiver aperture. The size of the mesh can be seen clearly in Fig.18a as each meshed element is given a different colour corresponding to the number of ray hits times power per ray divided by element area. A clear and converged pattern emerged for ray counts above 100 million (10^8).

This convergence is quantified in Fig.19, first in the total absorbed power value converging at a ray count above 10^8 (shown as a total heat value (Fig.19a), and percentage error (Fig.19b)), and in the form of an error plot for each element (Fig.19c), where the error was calculated per element to highlight the fact that even if the total absorbed flux was determined correctly, it was where the flux was distributed that took more rays to converge, e.g. the last 1% of elements (red and black traces in Fig.19c) still experienced a 100% ($\log_{10}(100) = 2$) error or more for a ray count of 10^7 or 10 million.

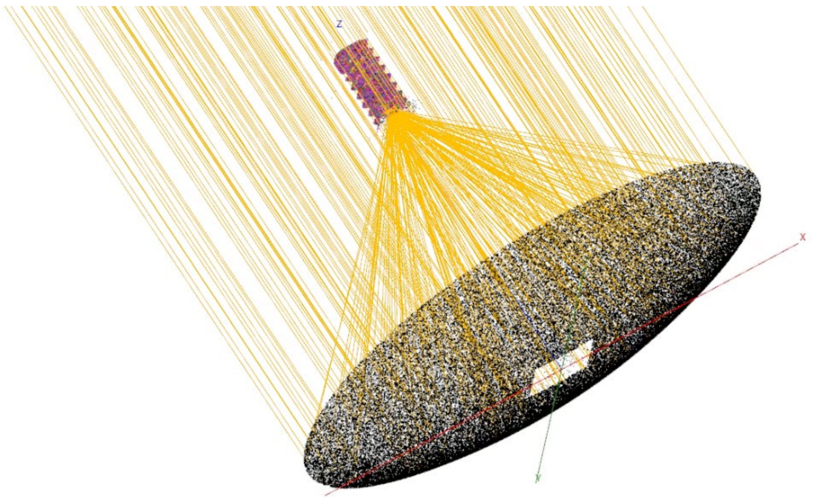


Figure 17 SolTrace sample rays, note shadow cast by receiver insulation (the latter not shown)

To ensure that at least 90% of the elements had a maximum deviation of under 1%, the ray independence study was conducted up to 200 million or $10^{8.3}$ rays for the mesh count of 31 456 elements and was used in the on-sun results to follow.

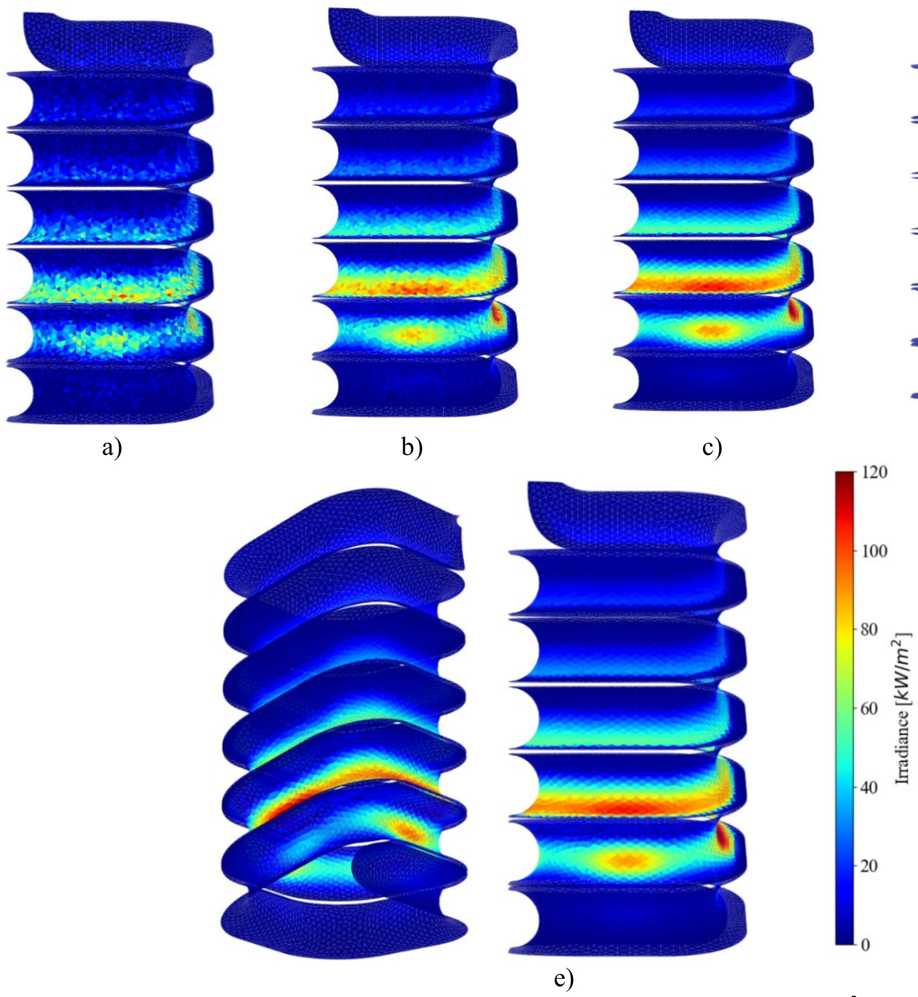
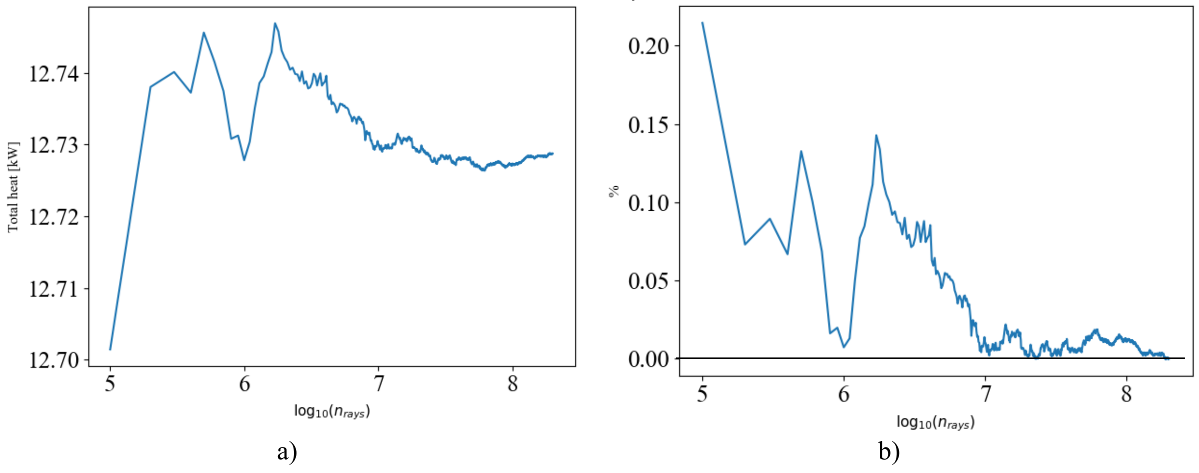
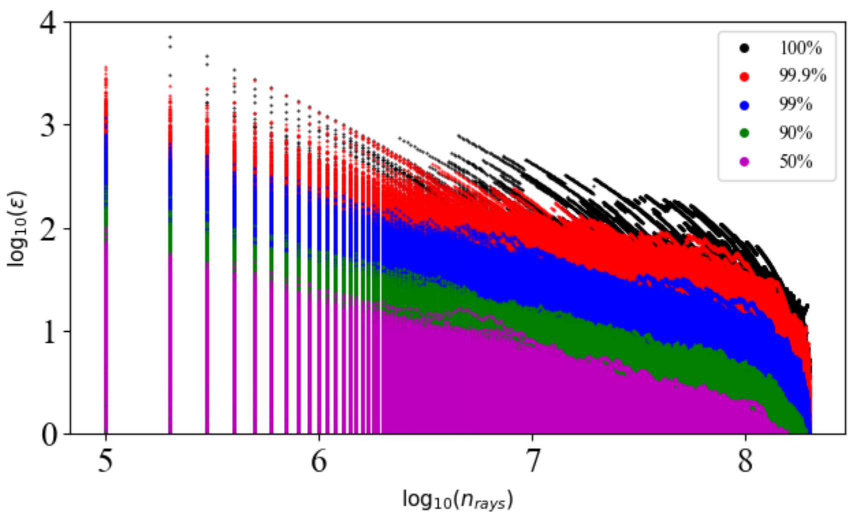


Figure 18 Absorbed radiation on tubular receiver for different ray counts a) 10^5, b) 10^6, c) 10^7, d) 10^8 and e) 2×10^8 rays





c)

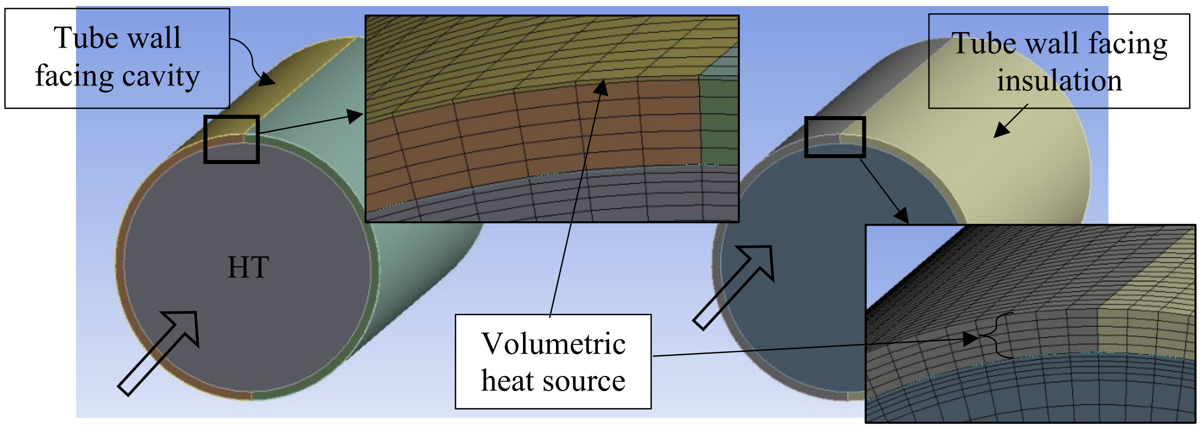
Figure 19 a) Convergence and b) percentage error of total absorbed heat versus ray count for 31 456 mesh elements; c) error progression as a function of ray count for 31 456 mesh elements. The percentage error of a single element relative to the final simulated value is shown, divided into the elements that represent 50%> of the maximum element flux (magenta), 10% to 50% of the maximum element flux (green), 1% to 10% of the maximum element flux (blue), 0.1% to 1% of the maximum element flux (red) and the rest of the data (zero flux elements were excluded).

3.4 Interpolation of heat source into CFD model

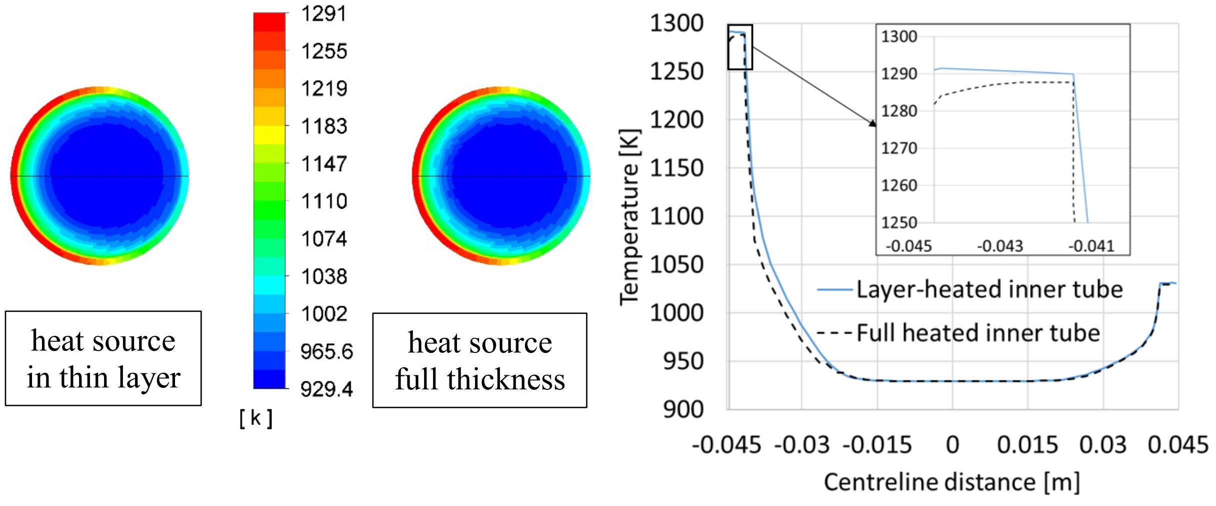
Using the optical properties in Tab.6 with a DNI of 1 000 W/m^2, the total absorbed power from the SolTrace model was 12 730 W, which represents a 70% optical efficiency (88% when excluding the dish reflectivity in the optical efficiency). Note that the total absorbed power depends on the dish reflectivity as well as the reflectivity of the inner insulation surfaces (for a dish reflectivity of 90% and inner insulation surface reflectivity of 70%, the total absorbed power will be 14 828 W).

The SolTrace generated heat flux [W/m^2] was then interpolated into ANSYS Fluent as a volumetric heat source by dividing the flux through the tube thickness and applying it on the whole of the inside half of the tube. Because of the curvature of the pipe, especially around the turns, the inner-surface area of the tubular receiver times the thickness was not exactly equal to the volume of the receiver tube in the CFD model. This error was about 4.2% and hence the heat source was adjusted by this amount so that the ANSYS Fluent total heat value [W] was the same as the total SolTrace value (power per ray times final intersection count) [W].

It can be argued that using the whole thickness of the tube as volumetric heat source is not an accurate reflection of the solar concentrated direct normal irradiation being absorbed in the first micrometers below the surface (Bergman et al, 2011), resulting in a difference in the tube inside and outside wall temperatures because of the poor convection heat transfer capability of air as heat transfer fluid in the current application, and with stainless steel being used with its moderate thermal conductivity (around 23 W/m-K for the current tube temperatures). A more accurate implementation therefore would be to model a separate layer on the heated tube outer circumference, as done by Moghimi et al (2015). To test the difference between these two implementation methods, a test case was simulated with two sections of tube of length 2 m, side by side, the one with the volumetric heat source applied to a thin layer (0.15 mm was used), and one with the heat source applied to the whole 3.05 mm tube thickness. The conditions were based on those given in Tab.7 in the next section with external tube wall boundary conditions based on the convection, radiation and conduction heat losses simulated in the next section. The heat source used is based on the maximum heat flux obtained in Fig.20, namely 112 kW/m^2. The setup is illustrated in Fig.21 with the temperature distribution obtained shown both as cross-sectional temperature contours and as a plot along the centreline of the pipes. The effect of the implementation is to give a difference in inner and outer wall temperatures as expected, but the difference in outer temperature is only 0.8%. This means that the simpler method of using the whole tube thickness is justified in the current application, but such proof should be obtained before using this approach in general.



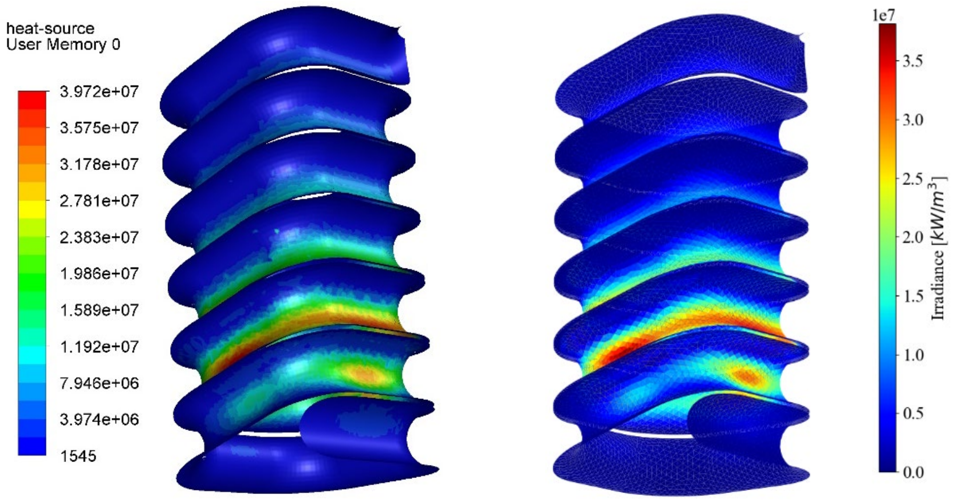
a)



b) c)

Figure 20 Comparison of volumetric heat source implementation method. a) Computational setup; b) Temperature contours [K] in cross section, indicating lines for plot in c); c) Temperature [K] along centreline for tube heated in layer only and in full thickness of tube

The interpolated ANSYS Fluent heat source distribution was implemented using the procedure described in Moghimi et al (2015). The process involved using a user-defined scalar to interpolate the source, then copying this scalar to a user-defined memory and assigning this memory location to a heat source for the relevant solid cell zones. The result is depicted in Fig.21 together with the SolTrace distribution for comparison. The total (integrated) heat source differed by less than 0.12% from the SolTrace value.



a) b)

Figure 21 Volumetric heat source [W/m^3] as a) interpolated into ANSYS Fluent, from the b) SolTrace heat flux divided by tube thickness of 3.05 mm to give [W/m^3], 2×10^8 rays.

4. Upright orientation CFD results

To assess the on-sun performance, the conditions specified in Le Roux and Meyer (2016) were used with the hot air plume still exiting to atmosphere. When used in a recuperated Brayton cycle, this air stream will pass through the downstream components, like the turbine and the recuperator, to generate a net cycle power output of about 1-2 kW. The conditions are listed in Tab.7 with the summary CFD results. As shown in Tab.7, the radiation heat losses accounted for 31% of the total absorbed power (provided as heat source). The convection heat loss from the receiver aperture only accounted for 14% of the heat provided due to the downward orientation of the receiver. For other orientation angles, this convection heat loss was expected to increase. These losses, combined with the heat losses from the external surfaces of the receiver insulation, implied that about 44% of the absorbed solar heat was transferred to the heat transfer fluid.

Property	**CFD Value**
Mass flow rate [kg/s] (Le Roux et al, 2014)	0.06
Inlet temperature [°C] (Le Roux et al, 2014)	657
Environmental temperature [°C]	22
Wind speed [m/s]	4
Outlet temperature [°C]	744
Total (integrated) solar heat source [W]	12 746
Average receiver inner-surface temperature [°C]	780
Average receiver insulation outer-surface temperature [°C]	64.2
Radiation heat loss through aperture [W]	3 944
Heat loss from sides of insulation [W]	867
Heat loss from top of insulation [W]	180
Heat loss from bottom of insulation [W]	150
Heat gain by heat transfer fluid [W]	5 667
Pressure drop of HTF [Pa]	1 113
Convection heat loss from receiver aperture [W]	1 733

Table 7 CFD results for simulation using solar heat source

The detailed CFD results are displayed in the following figures. The focused heating by the dish caused higher temperatures deeper into the tubular cavity receiver, illustrating one of the advantages of cavity receivers. Because these higher temperatures were deeper into the cavity, the radiation and convection heat losses associated with them were less. The temperature contours in Fig.22 confirm that the location of maximum heat source matched the region of higher temperatures on the receiver inner surface. The heat loss profile (not shown) corresponded to these regions of higher temperature.

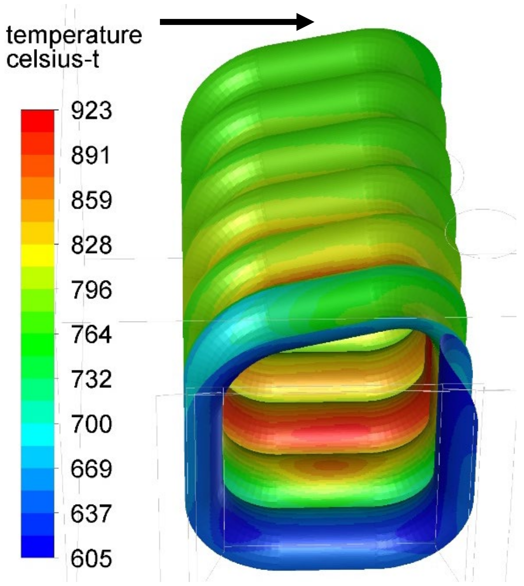


Figure 22 Receiver surface temperature contours [°C] – Solar-heated case 0° (upright)

The receiver insulation temperature profile shows hot regions in several areas. These are due to the HTF pipe inlet and outlet, and due to absorbed radiation and conduction in the aperture region (Fig.23, note that temperatures were clipped at 175 °C). The effect of the wind direction (see arrow in Fig.23) on the insulation temperatures is noticeable, but is not significant in this receiver orientation (facing down).

Finally, the heating effect of the HTF by the absorbed solar irradiation is illustrated in the temperature rise of the HTF pathlines in Fig.24. The local fluid temperature reached a value higher than the outlet temperature in the region of the highest heat source.

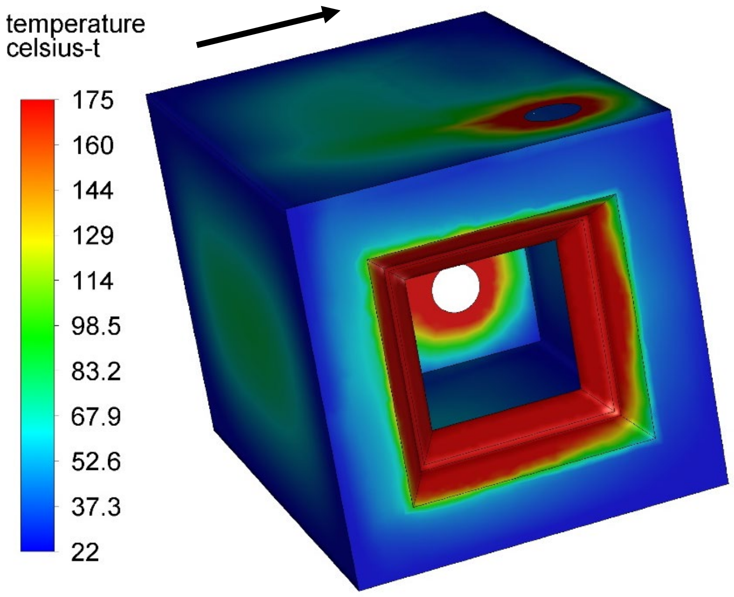


Figure 23 Receiver insulation temperature [°C] contours clipped at 175 °C – Solar-heated case 0° (upright)

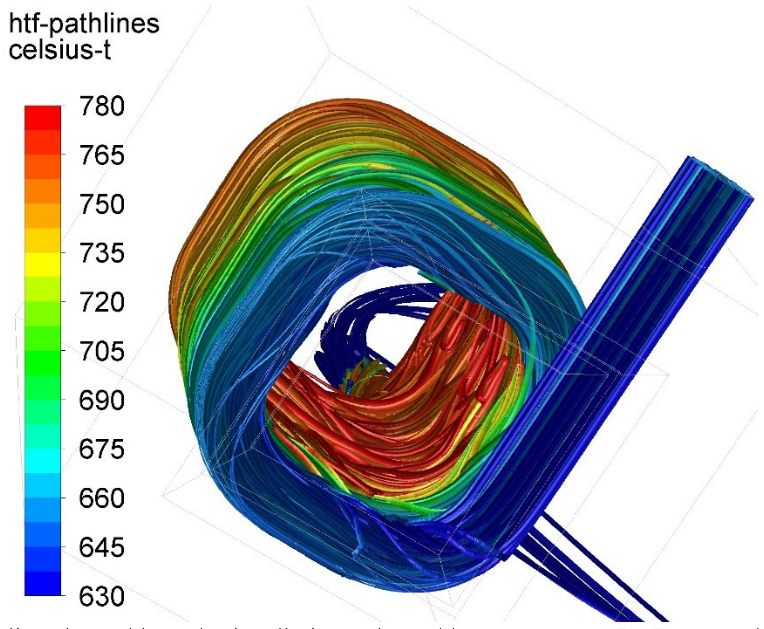


Figure 24 Pathlines heated by solar irradiation coloured by temperature [°C], range chosen to highlight heating of HTF – Solar-heated case 0° (upright)

5. Effect of dish orientation and wind speed on heat transfer

5.1 Inclination of cavity receiver

The orientation of the dish affects the inclination of the cavity receiver as it rotates to remain pointed at the sun. It is expected that an increase or decrease in the inclination angle from the 0° upright position presented above will result in an increase in heat losses because of two convective phenomena. Firstly, buoyancy effects as driven by gravity will result in natural convection heat losses from the cavity as its heat gets "leaked" upwards due to the tilted aperture and density differences. Secondly, a receiver that is exposed to the windward side will "catch" more of the wind, resulting in a washing out of the heat contained in the cavity. Conversely, a cavity that is facing downwind will be exposed to the suction pressure in the separation bubble, which will "suck" out some of the heat from the cavity. Technically, these contributions are termed natural and forced convection, and it is expected that the cavity will operate in the so-called mixed-convection regime.

Given that the temperatures experienced by the cavity surfaces are high (around 780 °C for the upright 0° case at a wind speed of 4 m/s), it is expected that radiative heat losses will dominate over convective heat losses. The results in this section will test this hypothesis.

5.2 Solving strategy for different dish orientations

Five different dish orientations were considered, as shown in Fig.25. The rotation mechanism of the dish was not considered or realised but the various orientations did result in a different distance between it and the ground, as shown in Fig.25. Five different CFD domains were then generated, each with the dish in a different orientation. For the higher dish orientation angles, the attacking wind caused a much larger wake. For these cases, the solution domain was extended both in length and height. An example is shown in Fig.26 for the 22.5° orientation with domain extents indicated.

The steady RANS equations were still solved, neglecting any transient effects due to atmospheric turbulence or vortex shedding (Wolmarans and Craig, 2019). The solving strategy that was followed amounted to keeping the inner mesh that contained the receiver with its internal flow as well as a cylindrical region around it and above the dish as constant for all the dish rotations. This process is illustrated in Fig.26 for the 22.5° orientation. Essentially, two separate meshes were generated, one was the constant cylinder mentioned above, and the other was the outer volumes as shown. The two meshes were then combined in ANSYS Fluent in the following manner:

- Import the inner mesh (in the upright orientation 0°).
- Translate it vertically to the origin.
- Rotate it by the required angle depending on the outer mesh being considered.
- Translate the rotated cylinder to the correct location matching the outer mesh.
- Import the outer mesh.
- Assign mesh interfaces where the regions touch.
- First translate and then rotate the combined mesh to the original upright orientation of the inner cylinder. This is required because the heat source file is written in these original coordinates and needs to be interpolated to the correct location of the receiver.
- Choose operating and boundary conditions and solve for different wind speeds.

The mesh counts for the models ranged from 42.8 million for the 0° orientation, to 64.6 million cells for the extreme orientations (e.g. 45°). The DNI was kept at 1000 W/m^2 for all orientations, implying that the same solar volumetric heat source was used.

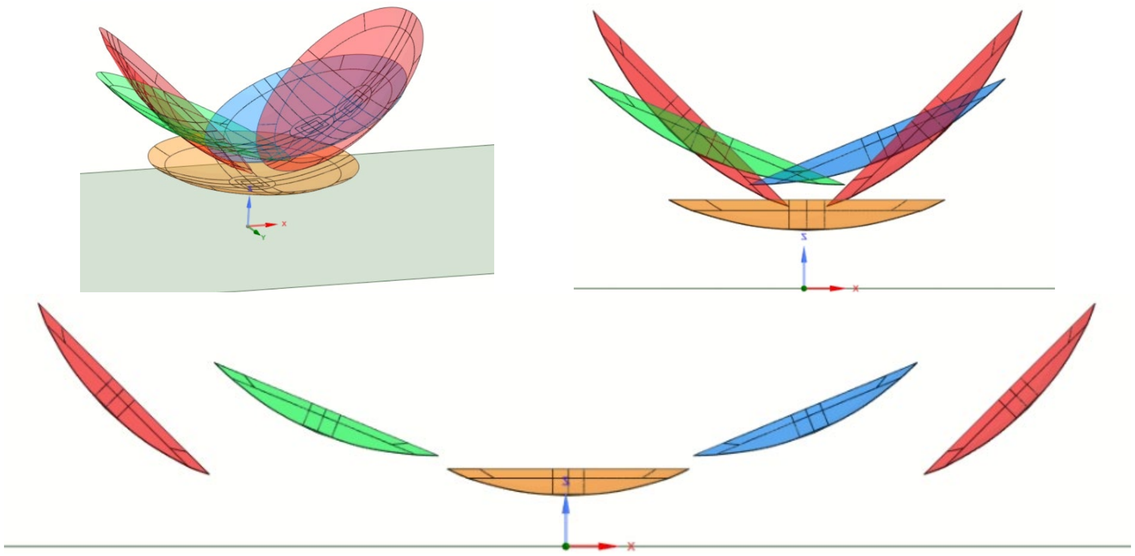


Figure 25 Dish orientation angles and clearance from ground

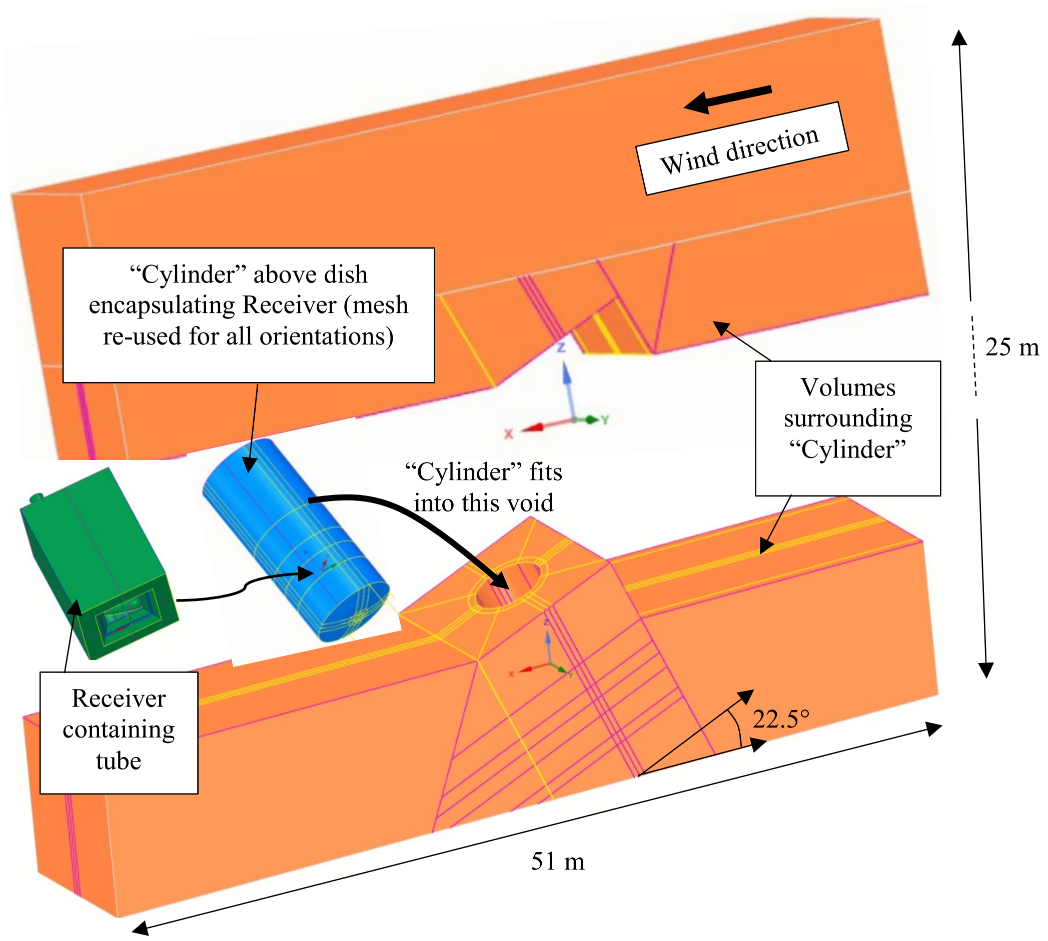

Wind direction
"Cylinder" above dish encapsulating Receiver (mesh re-used for all orientations)
25 m
Volumes surrounding "Cylinder"
"Cylinder" fits into this void
Receiver containing tube
22.5°
51 m

Figure 26 Strategy for accommodating different dish orientations and re-using the mesh of the cylindrical portion above the dish containing the receiver. Shown for 22.5° orientation.

5.3 Results for different dish orientations and wind speeds

The effect of dish orientation is to rotate the cavity receiver causing the wind and flow pattern caused by the dish to influence the convection heat transfer from the cavity. Tab.8 shows the results for a wind speed of 4 m/s. Notable from the table is that the radiation heat loss and insulation exterior temperatures were relatively constant with orientation. The convection heat loss was sensitive to the orientation as expected and was the main contributor to the HTF outlet temperature variation, shown in Fig.27. To assess the balance between forced and natural convection, the wind speed was reduced, first to 2 m/s and then to 0.5 m/s. The shape of the graph stayed relatively constant for lower wind speeds, but at the highest wind speed, there was an asymmetry due to the fact that the aperture facing the wind (negative dish orientations) would be more exposed to forced convection heat losses than positive orientation angles where the dish would shield the receiver to some extent. This is confirmed by a higher temperature being reached by the HTF for +45° compared with -45° at 4 m/s.

For the 4 m/s wind speed, the percentage absorbed power converted to the HTF varied between 40% and 45%. The radiation heat loss percentage stayed relatively constant between 29% and 31% with the main variation due to the convection heat loss (between 16% and 22%). Simulations were also conducted at lower wind speeds (not shown in Tab.8 but discussed later). For 2 m/s and 0.5 m/s, the HTF heat conversion varied between (43% and 50%) and (45% and 50%) respectively, again mainly due to a variation in convective heat losses: (10%-16%) and (10%-15%) for 2 m/s and 0.5 m/s respectively.

Dish orientation [°]	**-45**	**-22.5**	**0**	**22.5**	**45**
Total (integrated) solar heat source [W]	12720	12737	12741	12746	12730
HTF outlet temperature [°C]	732	745	744	736	735
Average receiver inner-surface temperature [°C]	763	781	780	768	765
Average receiver insulation outer-surface temperature [°C]	62.4	65.5	64.2	63.3	63.3
Radiation heat loss through aperture [W]	3765	3950	3944	3750	3918
Heat loss from sides of insulation [W]	849	862	867	854	856
Heat loss from top of insulation [W]	181	191	180	181	176
Heat loss from bottom of insulation [W]	137	146	150	141	154
Heat gain by heat transfer fluid [W]	5058	5935	5867	5328	5260
Convection heat loss from receiver aperture [W]	2730	1653	1733	2492	2366
Pressure drop of HTF [Pa]	1085	1086	1113	1097	1075
Minimum receiver temperature (exposed to sun) [K]	560	597	605	579	589
Maximum receiver temperature (exposed to sun) [K]	912	923	921	911	896
Average cavity air temperature (Fluent) [°C]	542	628	600	542	513

Table 8 CFD results for different dish orientation angles (4 m/s wind)

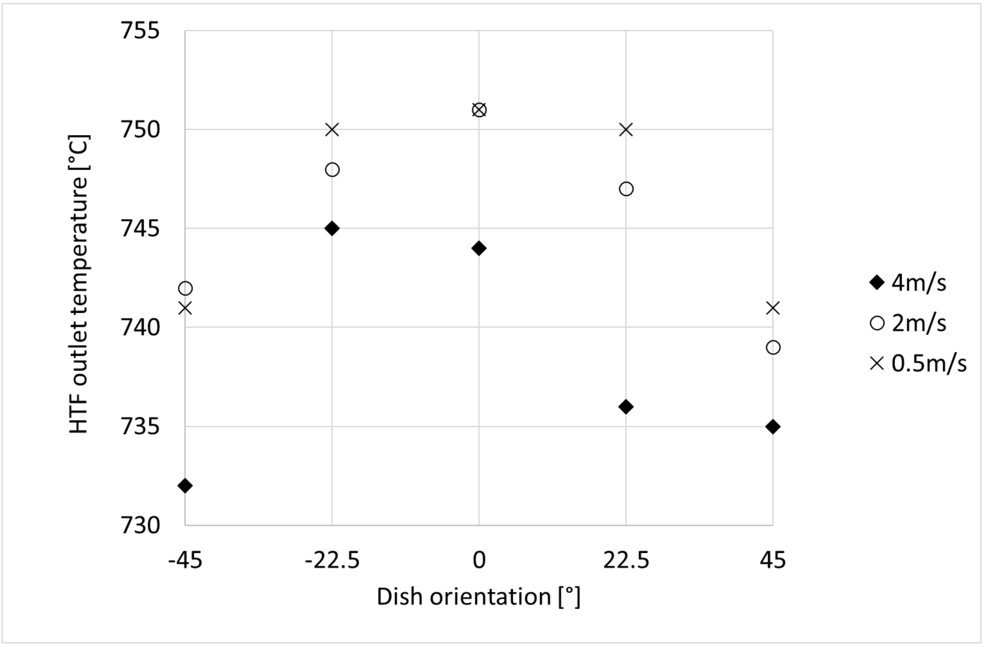


Figure 27 Variation of HTF outlet temperature [°C] with dish orientation for different wind speeds [m/s]

The CFD flow field results are presented as velocity magnitude contours (Fig.28), the temperature distribution of the thermal plume from the receiver (Fig.29), and surface temperature contours on the receiver tube (Fig.30); all for a wind speed of 4 m/s. Although the focus is not on the dish wake in the current work, it is clear that only at the 45° orientation (Fig.28e), is the receiver located in the wake of the dish. Since it is not possible to separate natural convection from forced convection, there seems to be a complex interaction between these mechanisms at all dish orientations. This will be discussed further in the next section.

The thermal plume trajectories shown in Fig.29 predominantly followed the flow patterns of Fig.28 as the relatively high wind speed dominated. The effect of dish orientation on receiver surface temperatures was not considerable, as plotted in Fig.30 using the scale for all figures of the 0° orientation for comparison. Since the cavity was deep and the highest heat source was located about a third of the way in (see Fig.21), the highest temperature zone was relatively shielded from the effect of the wind, hence the fairly constant radiative heat transfer losses in Tab.8.

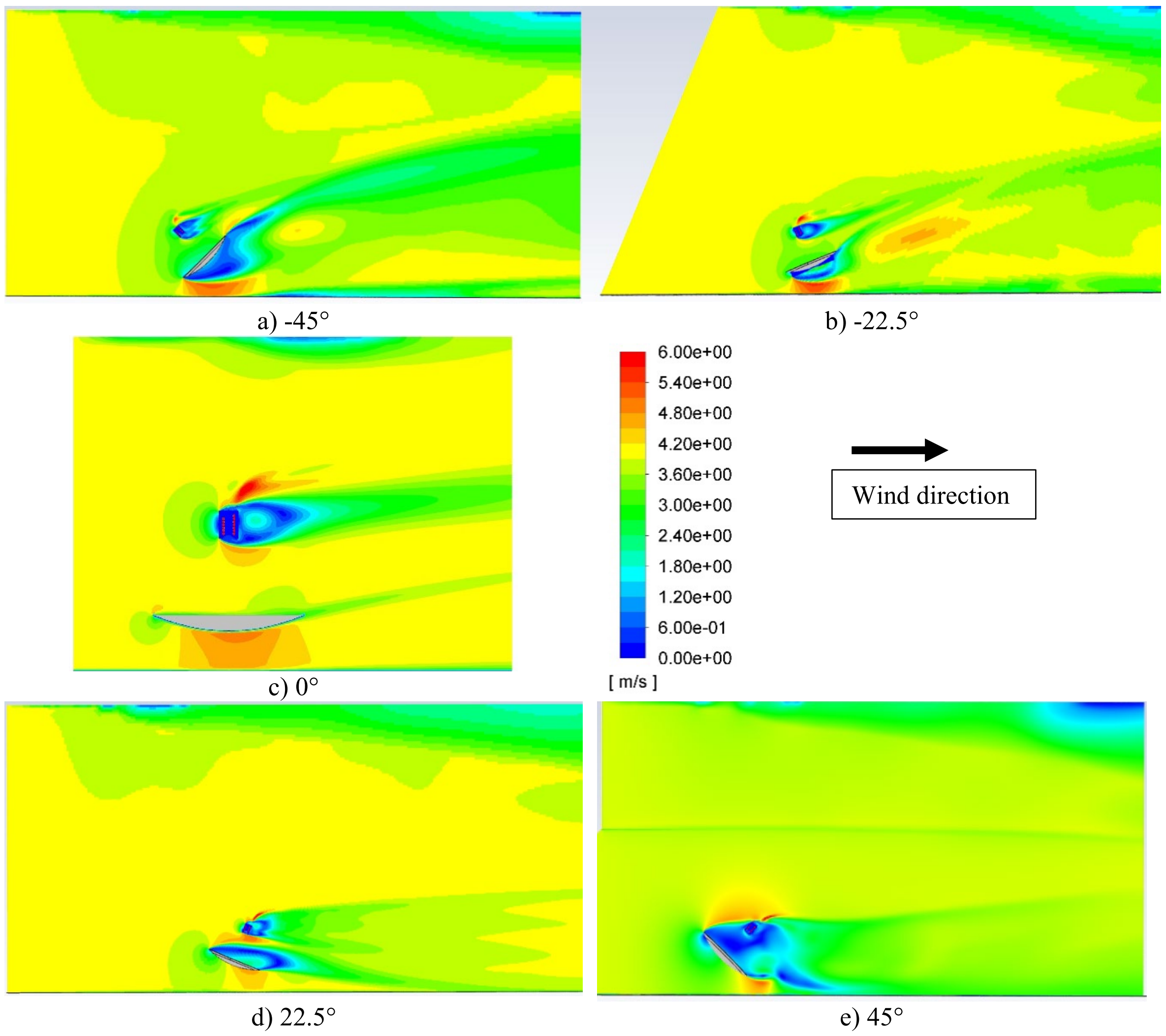


Figure 28 The effect of dish orientation on the flow pattern as shown by velocity magnitude contours on the y = 0 plane in the range 0 – 6 m/s for a wind speed of 4 m/s. a) -45° b) -22.5° c) 0° d) 22.5° e) 45°

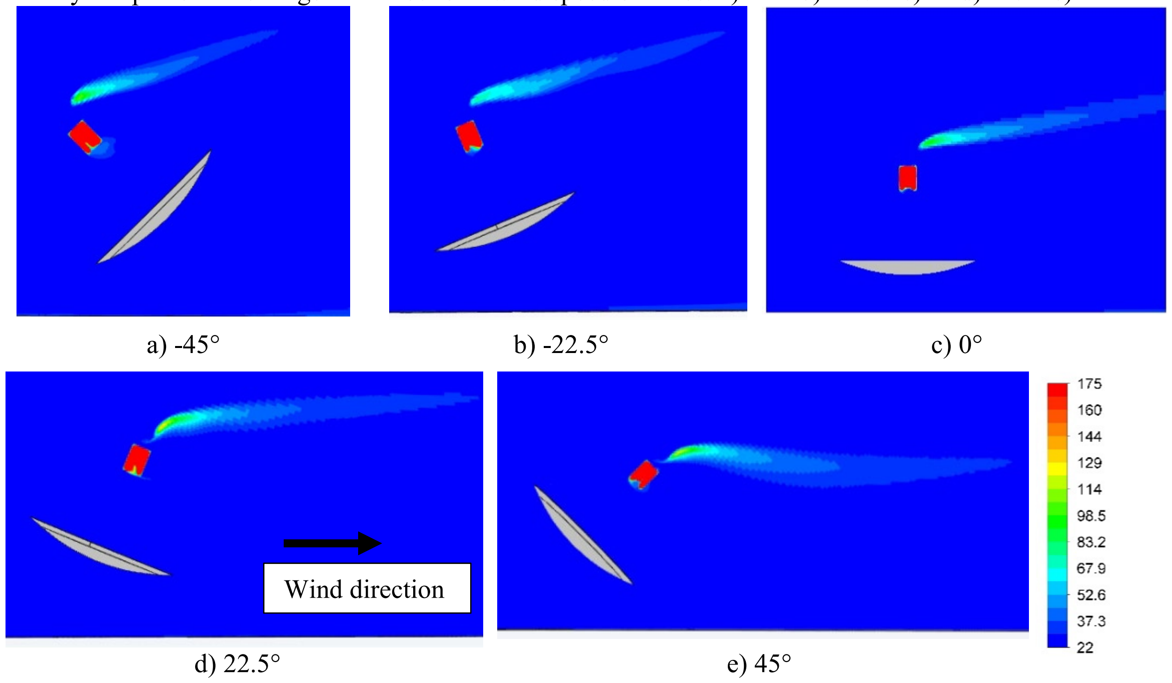


Figure 29 The effect of dish orientation on the plume trajectory shown as temperature contours on the y = 0 plane in the range 22 – 175 °C for a wind speed of 4 m/s. a) -45° b) -22.5° c) 0° d) 22.5° e) 45°

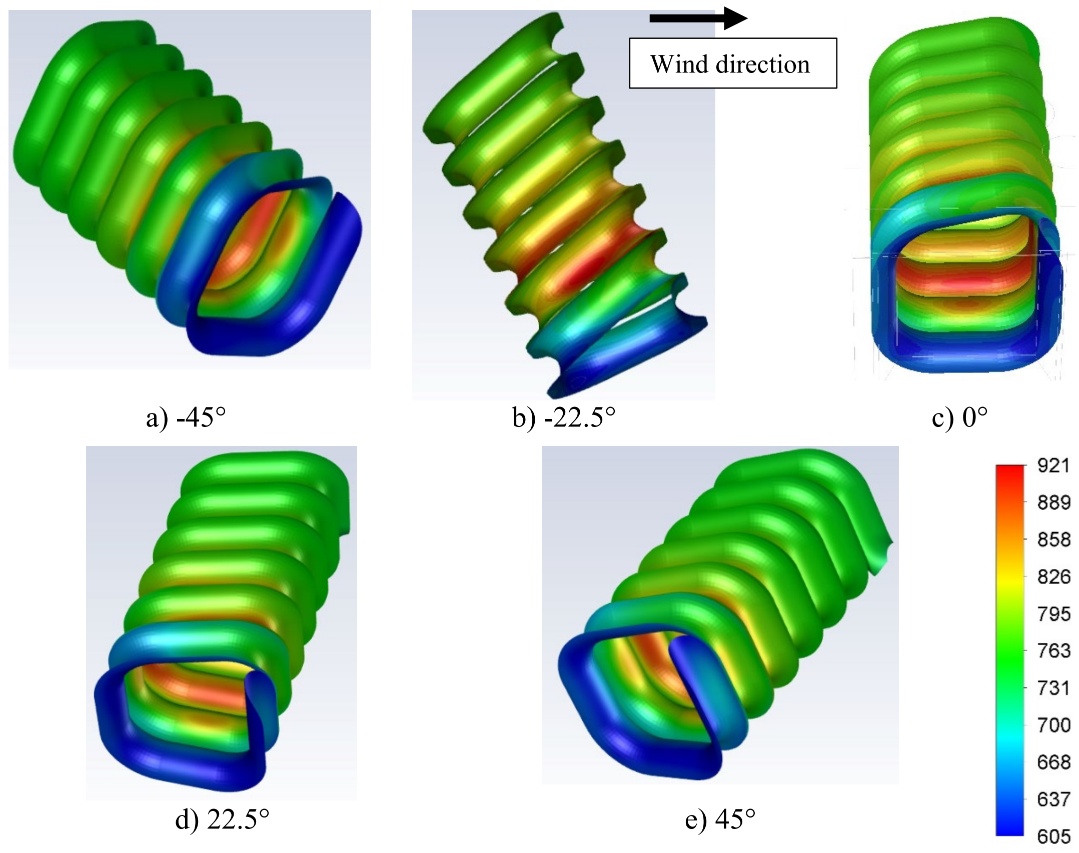


Figure 30 The effect of dish orientation on the temperature contours of the inner receiver surface for a wind speed of 4 m/s. a) -45° b) -22.5° c) 0° d) 22.5° e) 45°

5.4 Comparison with literature correlations

Prakash et al (2012) developed a correlation based on numerical simulations of cubic, hemispherical and spherical cavities (with wall temperatures ranging between 100 °C and 300 °C) with different aperture-to-length ratios at different cavity inclinations:

$$Nu_L = 0.0136 Ra_L^{1/3}(1+cos\theta)^{2.72}\left(\frac{a}{L}\right)^{0.72}, \qquad (1)$$

with θ the inclination angle (0° being sideways facing and 90° being downward facing, corresponding to a dish orientation of 0°), a the aperture size and L the cavity depth or length. This correlation is not directly applicable to the current study because of the lower temperature range and because the current cavity is approximately cylindrical and long (see Fig.31) with an aperture width-to-length ratio of 0.32. Nevertheless, a plot of this correlation for two Rayleigh numbers, defined as

$$Ra_L = GrPr = \rho^2 C_p g \beta (T_w - T_a) L^3/(\mu k)\,, \qquad (2)$$

is shown in Fig.33 for Rayleigh numbers 3.5E8 and 8.5E8 based on the range obtained for the current CFD simulations. T_w and T_a are the average receiver inner-wall surface and ambient (upstream) temperature respectively. C_p is the specific heat at constant pressure, g is the gravitational constant and β is the thermal expansion coefficient assumed to be $1/T_{cav}$, where T_{cav} is the volume-averaged fluid temperature in the cavity as calculated from the CFD solution. The other fluid properties, dynamic viscosity, μ, and thermal conductivity, k, as well as the specific heat, C_p, in Equation 2 are calculated using this bulk fluid temperature, T_{cav}, the former using Sutherland's law.

Stine and McDonald (1989) suggested the following correlation for a natural convection Nusselt number for a cylindrical cavity with different aperture sizes (a):

$$Nu_L = 0.088Gr_L^{1/3}\left(\frac{T_w}{T_a}\right)^{0.18}(cos\theta)^{2.47}\left(\frac{a}{L}\right)^s; s = 1.12 - 0.98\left(\frac{a}{L}\right), \quad (3)$$

with Gr the Grashof number. This correlation was also plotted in Fig.32 using CFD values at wind speeds of (4, 2 and 0.5 m/s) for Gr, T_w and T_a.

Another correlation for a cylindrical cavity is that of Koenig and Marvin (1981) as reported in Wu et al (2010), Harris and Lenz (1985) and McDonald (1995). This correlation is based on higher cavity temperatures (ranging between 550 °C and 900 °C) but also does not take strong wind effects into account. The correlation is

$$Nu_L = 0.52P(\theta)(L_c)^{1.75}(Gr_L Pr)^{\frac{1}{4}};\ P(\theta) = 0.707(cos\theta)^{2.2}\ , \quad (4)$$

with the expression for cavity inclination: $P(\theta)$ valid for the range of inclinations considered here. L_c is 1 for the current cavity since the aperture effective radius remains constant through the cavity length. This correlation was also plotted in Fig.33 using CFD values at wind speeds of (4, 2 and 0.5 m/s) for Gr.

When the cavity surface temperature is not uniform and known (as in the current work), the correlation by Abbasi-Shavazi et al (2020) is applicable:

$$Nu_{L_c} = 0.126\left(Gr_{L_c}\right)^{\frac{1}{3}}(T^*)^{0.11}AR^{-0.52}\left(\frac{A_{cz}}{A_w}\right)^{0.80}, \quad (5)$$

where AR is the aspect ratio of the cavity (*L*/*a* in Fig.31), A_w the cavity wall area and A_{cz} the convective zone area (see Abbasi-Shavazi et al, 2020), which accounts for the stagnant zone in pure convection (being the surface of the volumes shown in Fig.31 for the 22.5° and 45° orientations added to the aperture area). This aperture area is where the inclination angle is incorporated into the correlation. T^* is calculated from:

$$T^* = \frac{T_{max}-T_\infty}{T_{min}-T_\infty}, \quad (6)$$

with T_{max} and T_{min} the maximum and minimum temperatures in the cavity, which implies that these are either measured, or computed as in the current study. This correlation was also plotted in Fig.32 using the current CFD data but using the cavity length as characteristic length for consistency. Note that the correlation was developed for slightly lower Grashof numbers (2.6×10^5 to 1.4×10^7) than in the current work (1.9×10^6 to 6.9×10^7) when using their characteristic length of

$$L_c = D_{ap,\perp} + \frac{D_{ap}}{2}, \quad (6)$$

where D_{ap} is the aperture diameter and $D_{ap,\perp}$its vertical projection. Furthermore, the temperature range considered was between 355 °C and 650 °C. Although this correlation was not developed for wind conditions, it does give non-zero convection heat loss at 90° inclination.

To compare with these correlations, the Nusselt number is calculated from the CFD results using

$$Nu_L = \frac{hL}{k} = \frac{Q_{conv}L}{(T_w-T_a)A_{cav}k(T_{cav})}, \quad (7)$$

where the thermal conductivity is evaluated at the cavity bulk fluid temperature, T_{cav}. A_{cav} is the surface area of the cavity (see Fig.31) as calculated from the CFD model. T_w is the receiver tube inner-surface average temperature. Q_{conv} is the convection heat loss from the CFD defined as

$$Q_{conv} = Q_{abs} - Q_{HTF} - Q_{cond} - Q_{rad} \,, \tag{8}$$

where Q_{abs} is the absorbed solar power, Q_{HTF} the heat transferred to the HTF, Q_{cond} the heat conducted through the walls of the insulation and Q_{rad} the radiation heat loss from the receiver inner surfaces. These values are given in Tab.8 for the 4 m/s wind speed.

The CFD Nusselt numbers are shown in Fig.32 for the different wind speeds evaluated. Note that some of the data points are connected by straight lines to help the reader distinguish between the many correlations being compared. Since inclination angles were simulated on both sides of the vertical, the absolute values of inclination were taken. This neglects the fact that the wind direction has a large influence on the convective heat losses, as confirmed when plotting the Nusselt number versus the orientation angle in the range -45° to +45° in Fig.33. For the lowest wind speed (0.5 m/s) when natural convection dominates, the current results are closest to the correlations of Prakash et al (2012) and Abbasi-Shavazi et al (2020) in Fig.32. The largest deviation between the correlations and the current data is at 90° inclination, which corresponds to the upright or 0° dish orientation. The reason is that with the forcing wind, even this orientation experiences some convective heat losses. The correlations of Stine and McDonald (1989) and Koenig and Marvin (1981) predicted zero convection heat losses at the upright orientation because they only considered natural convection.

What is evident from Fig.33 is that there is not a large difference in the convection heat losses between 2 m/s and 0.5 m/s. The reason for this is that the thermal plume is not significantly affected at close proximity to the receiver by the low wind speeds, as confirmed in Fig.34. The role of natural convection, or gravity-driven buoyant flow, is clear by the upward trajectory of the plume, especially in Fig.34b).

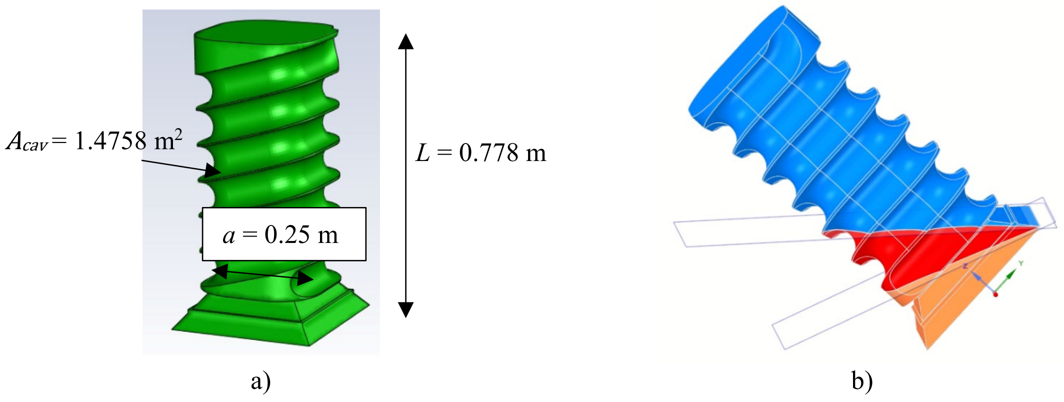


Figure 31 a) Receiver cavity air shape and dimensions; b) convection zone volumes for A_{cz} area calculation (22.5° and 45° orientations)

Pavlovic and Penot (1991) determined that the Nusselt number should scale in the mixed convection regime with $Re\,/\,Gr^{\,0.5}$ based on dimensional analysis. Together with the Gr dependency of Nu for natural convection, they proposed a correlation as

$$Nu_L = aGr^b\left(1 + c\left(\frac{Re}{Gr^{0.5}}\right)^d\right), \tag{6}$$

where the coefficients a, b, c and d are all functions of θ. To investigate to what extent mixed convection is present, Fig.35 plots Nu versus $Re\,/\,Gr^{\,0.5}$ for all the cases simulated. The figure distinguishes between positive and negative inclination with respect to the wind direction.

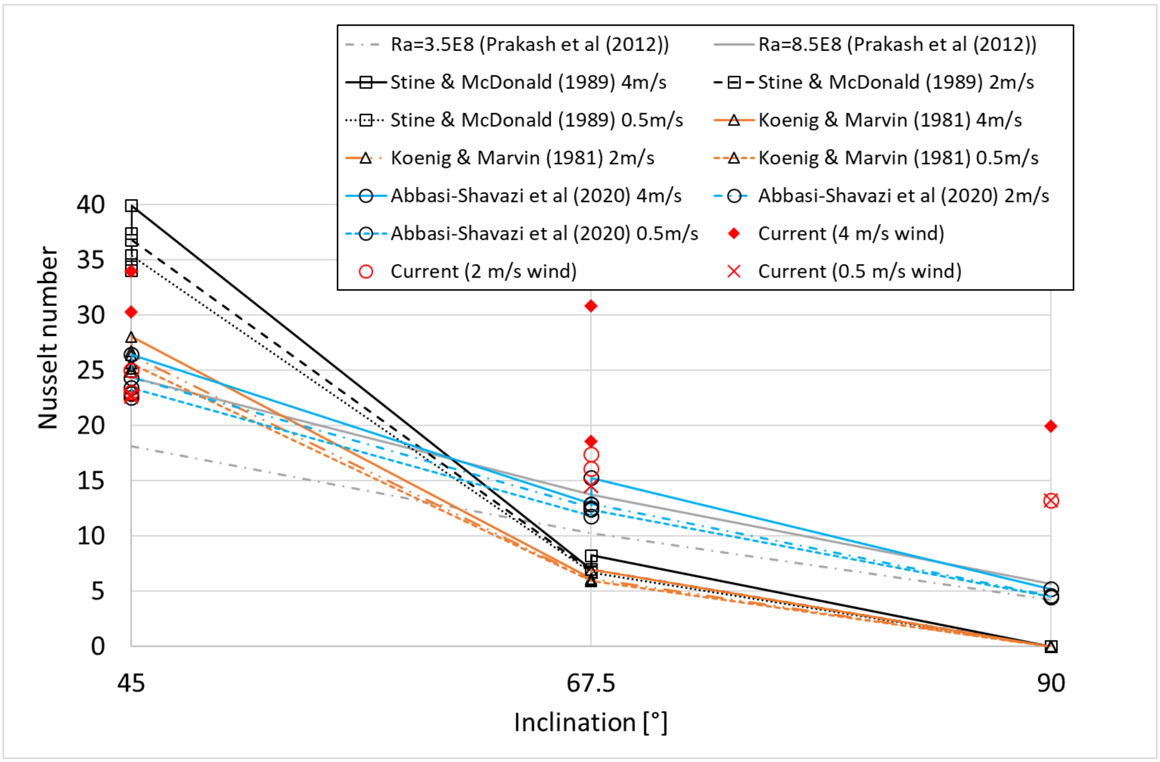


Figure 32 Nusselt number versus cavity receiver inclination angle as compared with correlations of Prakash et al (2012), Stine and McDonald (1989), Koenig and Marvin (1981) and Abbasi-Shavazi et al (2020)

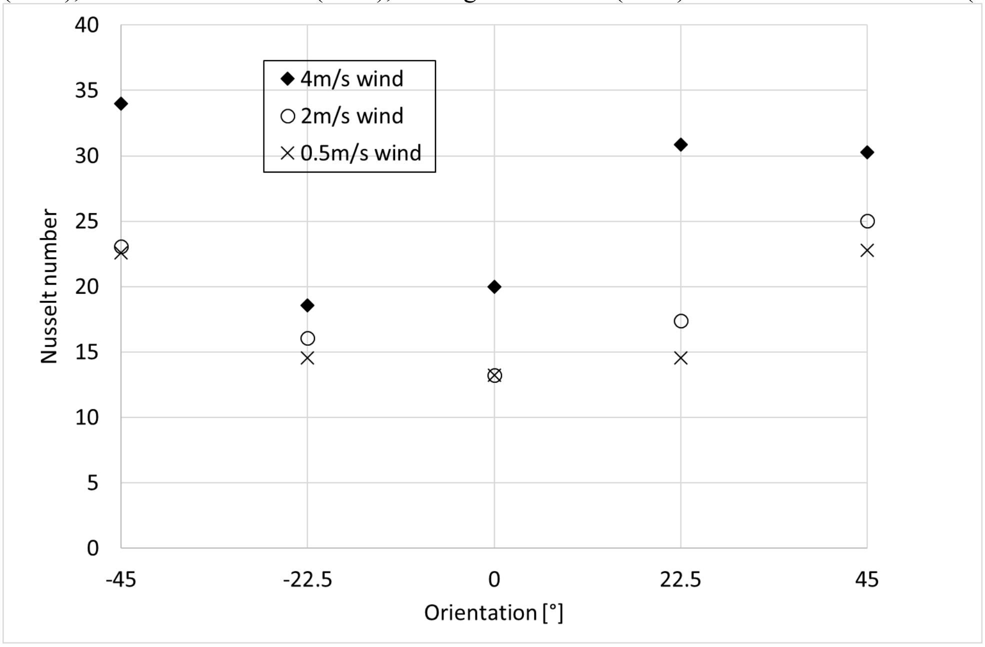


Figure 33 Nusselt number versus cavity receiver inclination angle for all wind speeds considered

Equations 1, 3 and 5 suggest that Nu is a function of $Ra_L^{1/3}$, or $Gr_L^{1/3}$ since Pr is relatively constant in this study. Plotting this relationship in Fig.36 confirms that this is true with a mild influence of inclination angle at higher Rayleigh numbers which correspond to higher orientation angle.

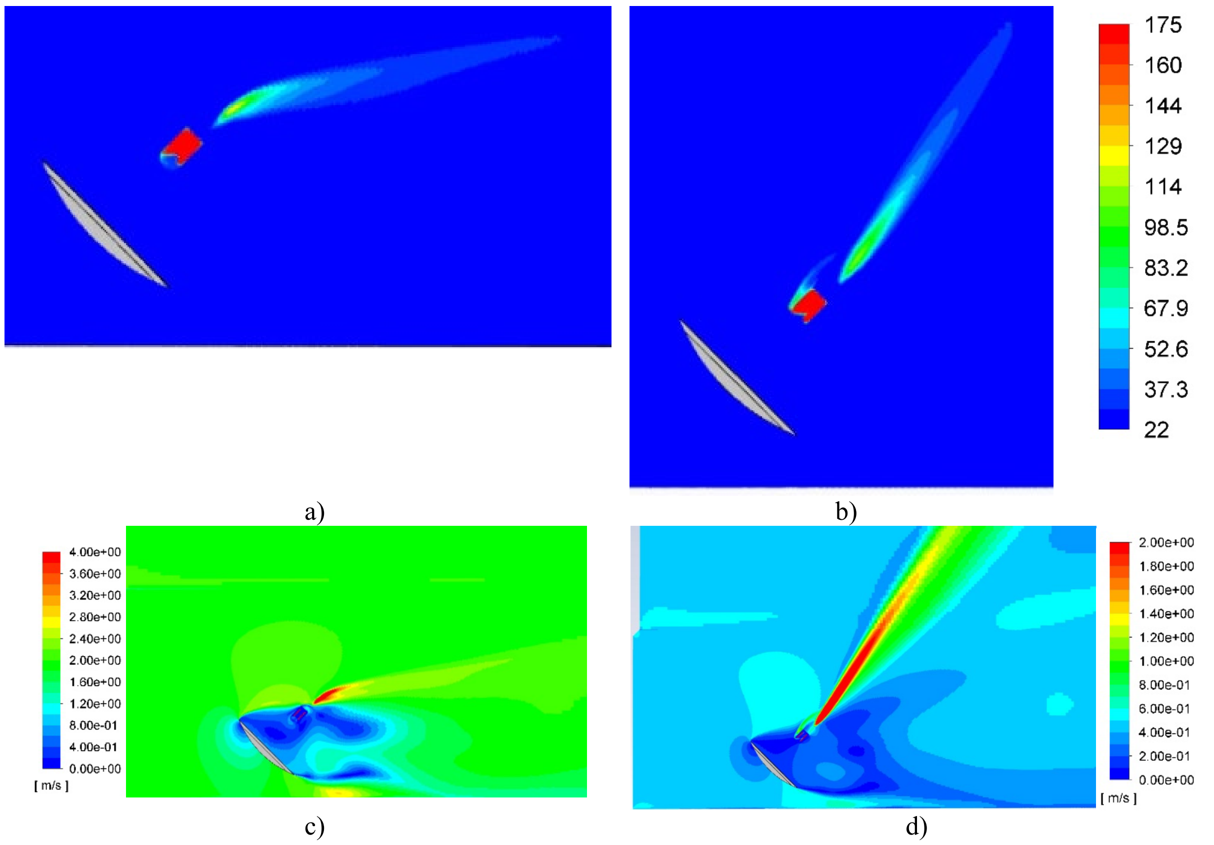


Figure 34 The effect of wind speed on the plume trajectory – temperature contours in the range 22 – 175 °C: a) 2 m/s b) 0.5 m/s; and velocity magnitude contours: c) 2 m/s d) 0.5 m/s for a dish orientation of 45°

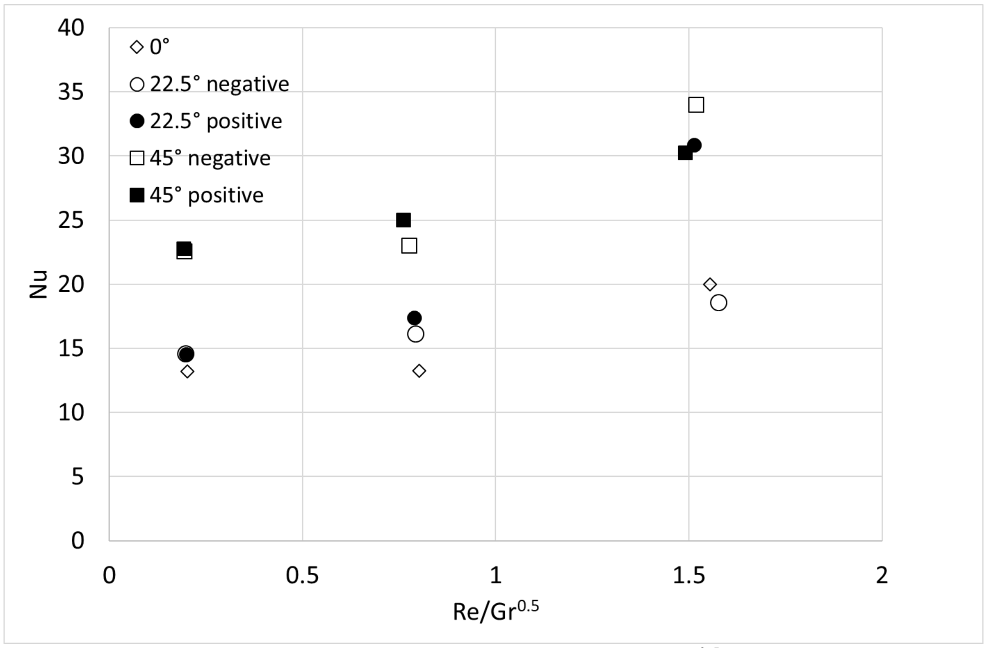


Figure 35 Nusselt number versus Re/Gr$^{0.5}$

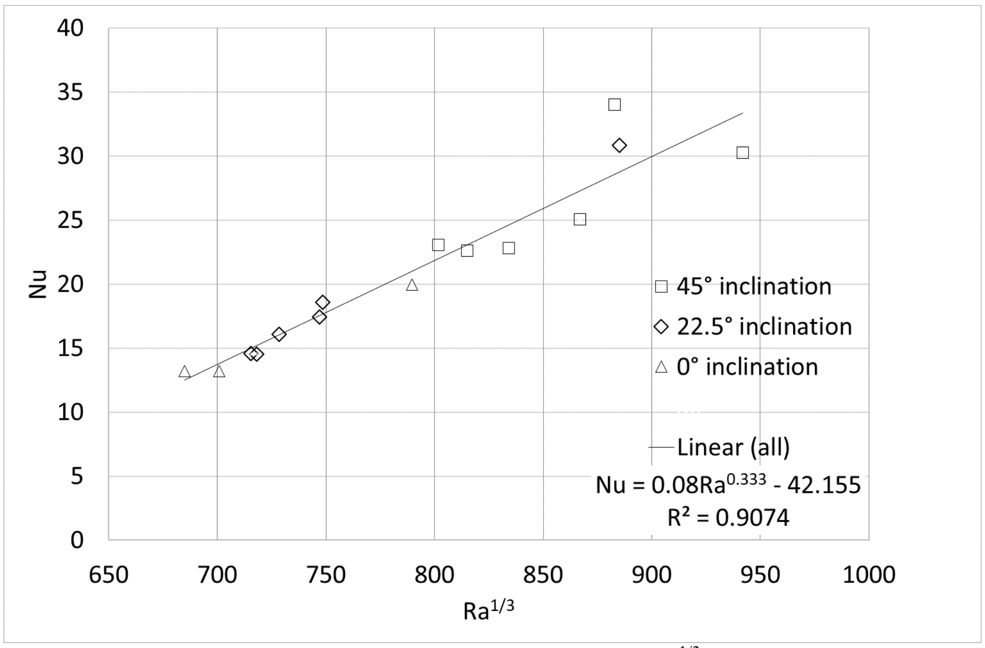


Figure 36 Nusselt number versus $Ra^{1/3}$

6. Conclusions

The paper described the numerical simulation of heat losses from a complex geometry receiver for a solar dish in a low-pressure recuperated Brayton cycle application. The heat loss mechanisms considered included thermal re-radiation, natural convection due to buoyancy and external forced convection due to wind, as well as conduction through the surrounding insulation material from where it was also convected and radiated to the surroundings. To validate the approach, an experimental set-up using heated air was replicated. The outlet temperature of the experiment was predicted to within 0.9%, while the average receiver temperatures were simulated within 1.8%. Using the same approach, a solar heat source was considered next. This heat source was generated using SolTrace and scripting to resolve the receiver surface using meshed elements. The meshed elements were obtained from the same CFD model as used in the subsequent conjugate heat transfer analysis. The heat source was implemented as a volumetric heat source distribution employing user coding. Different dish orientations and different wind speeds were considered.

The following conclusions can be drawn from the study:

- For the solar heat source considered with a receiver inlet air temperature of 657 °C and a mass flow rate of 0.06 kg/s, the optical efficiency was 70% and the efficiency of converting the absorbed solar radiation to the heat transfer fluid varied between 40% and 50% depending on the dish orientation and wind speed.
- The thermal radiation heat loss from the cavity remained relatively constant at about 30% for all conditions considered, driven by the high temperatures of the cavity surface (in the range 730-750 °C).
- The variation in thermal efficiency was therefore due to convection heat losses, which varied between 10% and 16% depending on wind speed and dish orientation.
- Correlations from literature for convection heat losses from open cavities compared favourably with the current results.

Detailed results of temperature and heat transfer distributions as determined in this paper allow for the accurate estimation of optical and thermal efficiencies, and will aid in the development of improved receivers for parabolic dishes.

Acknowledgements

The authors would like to acknowledge the support of the University of Pretoria (South Africa) and the South African Department of Science and Innovation for student bursary support. Additionally, the Centre for High Performance Computing (CHPC) in South Africa is acknowledged for computational resources. This work is based on the research supported by the National Research Foundation (NRF) of South Africa (Grant Number 109311) as well as the Technology Innovation Agency of South Africa (TIA).

References

Abbasi-Shavazi, E., Torres, J.F., Hughes, G., Pye, J., 2020, Experimental correlation of natural convection losses from a scale-model solar cavity receiver with non-isothermal surface temperature distribution, *Solar Energy*, Vol. 198, pp. 355-375.

Bejan, A., 2006. Advanced Engineering Thermodynamics, 3rd ed., New York: John Wiley & Sons, Inc.

Bergman, T.L., Lavine, A.S., Incropera, F.P., Dewitt, D.P., 2011, Fundamentals of Heat and Mass Transfer, seventh ed. John Wiley and Sons.

Brooks, M.J., Du Clou, S., Van Niekerk, J.L., Gauché, P., Leonard, C., Mouzouris, M.J., Meyer, A.J., Van der Westhuizen, N., Van Dyk, E.E., Vorster, F., 2015, SAURAN: A new resource for solar radiometric data in Southern Africa, *Journal of Energy in Southern Africa*, Vol. 26, pp. 2-10, (sauran.ac.za).

Çengel, Y.A., Ghajar, A.J., 2015. Heat and Mass Transfer: Fundamentals & Applications, 5th edition, New York: McGraw-Hill Education, pp. 442-500.

Clausing, A.M., Lister, L.D., Waldvogel, J.M., 1989, Combined convection from isothermal cubical cavities with a variety of side-facing apertures. *International Journal of Heat and Mass Transfer*, Vol. 32 (8), pp. 1561-1566.

Craig, K.J., Le Roux, W.G., Meyer, J.P., 2015a, Computational Fluid Dynamics Analysis of Parabolic Dish Tubular Cavity Receiver, 3rd Southern African Solar Energy Conference (*SASEC 2015*), 11-13 May 2015, Kruger Park, South Africa.

Craig, K.J., Marsberg, J., Meyer, J.P., 2015b, Combining Ray Tracing and CFD in the Thermal Analysis of a Parabolic Dish Tubular Cavity Receiver, *SolarPACES 2015*, 13-16 October 2015, Cape Town, South Africa, AIP Conference Proceedings 1734, 030009 (2016); doi: 10.1063/1.4949061.

Harris, J.A., Lenz, T.G., 1985, Thermal performance of solar concentrator/cavity receiver systems. *Solar Energy*, Vol. 34 (2), pp. 135-142.

Heller, P., Pfänder, M., Denk, T., Tellez, F., Valverde, A., Fernandez, J., et al., 2006, Test and evaluation of a solar powered gas turbine system, *Solar Energy*, Vol. 80, pp. 1225-1230.

Le Roux, W.G., Bello-Ochende, T., Meyer, J.P., 2014, The efficiency of an open-cavity tubular solar receiver for a small-scale solar thermal Brayton cycle, *Energy Conversion and Management*, Vol. 84, pp. 457-470.

Le Roux, W.G., 2015. Thermodynamic optimisation and experimental collector of a dish-mounted small-scale solar thermal Brayton cycle, Thesis: University of Pretoria.

Le Roux, W.G. and Meyer, J.P., 2016, Modeling the small-scale dish-mounted solar thermal Brayton cycle, *SolarPACES 2015*, 13-16 October 2015, Cape Town, South Africa, AIP Conference Proceedings 1734, 060002-1–060002-8; doi: 10.1063/1.4949144.

Le Roux, W.G., 2018, Feasibility study of a hybrid small-scale dish-mounted solar thermal Brayton cycle with cogeneration, *Proceedings of the 16th International Heat Transfer Conference*, IHTC-16, August 10-15, 2018, Beijing, China, IHTC16-24185.

Le Roux, W.G. and Sciacovelli, A., 2019, Recuperated solar-dish Brayton cycle using turbocharger and short-term thermal storage, *Solar Energy*, Vol. 194, pp. 569-580.

Li, Zhigang, Tang, Dawei, Du, Jinglong, Tie Li, 2011, Study on the radiation flux and temperature distributions of the concentrator-receiver system in a solar dish/Stirling power facility, *Applied Thermal Engineering*, Vol. 31, pp. 1780-1789.

Lovegrove, K., Taumoefolau, T., Paitoonsurikarn, S., Siangsukone, P., Burgess, G., Luzzi, A., Johnston, G., Becker, O., Joe, W., Major, G., 2003, Paraboloidal dish solar concentrators for multi-megawatt power generation. In: Proceedings of the International Solar Energy Society (ISES) Solar World Conference, Goteborg, Sweden.

Mancini, T., Heller, P., Butler, B., Osborn, B., Schiel, W., Goldberg, V., Buck, R., Diver, R., Andraka, C., Moreno, J., 2003, Dish-stirling systems: An overview of development and status, *Journal of Solar Energy Engineering*, Vol. 125, pp. 135-151.

McDonald, C.G., 1995, Heat loss from an open cavity. Sandia Laboratory, Report SAND95-2939.

Mills, D., 2004, Advances in solar thermal electricity technology. *Solar Energy*, Vol. 76, pp. 9-31.

Moghimi, M.A., Craig, K.J., Meyer, J.P., 2015, A novel computational approach to combine the optical and thermal modelling of linear Fresnel collectors using the finite volume method, *Solar Energy*, Vol. 116, pp. 407-427.

Pavlovic, M.D., Penot, F., 1991, Experimental in the mixed convection regime in an isothermal open cubic cavity, *Experimental Thermal and Fluid Science*, Vol. 4, pp.648-655.

Prakash, M., Kedare, S.B., Nayak, J.K., 2009, Investigations on heat losses from a solar cavity receiver, *Solar Energy*, Vol. 83, pp. 157-170.

Prakash, M., Kedare, S.B., Nayak, J.K., 2012, Numerical study of natural convection loss from open cavities, *International Journal of Thermal Sciences*, Vol. 51, pp. 23-30.

Shuai, Yong, Xin-Lin Xia, He-Ping Tan, 2008, Radiation performance of dish solar concentrator/cavity receiver systems, *Solar Energy*, Vol. 82, pp. 13-21.

Slootweg, M., Craig, K.J., Meyer, J.P., 2019, A computational approach to simulate the optical and thermal performances of a novel complex-geometry solar tower molten salt cavity receiver, *Solar Energy*, Vol. 187, pp. 13-29.

Slootweg, M., 2019, Numerical performance analysis of novel solar tower receiver, MEng dissertation, University of Pretoria (http://hdl.handle.net/2263/70354)

Stine, W.B., McDonald, C.G., 1989, Cavity receiver convective heat loss. In: Proceedings of the International Solar Energy Society (ISES) Solar World Conference, Kobe, Japan.

Visser, W.P.J., Shakariyants, S.A. and Oostveen, M., 2011. Development of a 3 kW microturbine for CHP applications. Journal of Engineering for Gas Turbines and Power 133, pp. 042301:1-8.

Wang, Wujun, Xu, Haoxin, Laumert, B., Strand, T, 2014, An inverse design method for a cavity receiver used in solar dish Brayton system, *Solar Energy*, Vol. 110, pp. 745-755.

Wang, W., Laumert, B., Xu, H., Strand, T., 2015, Conjugate heat transfer analysis of an impinging receiver design for a dish-Brayton system. *Solar Energy*, Vol. 119, pp. 298-309.

Wendelin, T., Dobos, A., 2013. SolTrace: a ray-tracing code for complex solar optical systems. Technical Report NREL/Tp-5500-59163.

Wolff, T.M., Le Roux, W.G., Meyer, J.P., 2018, Heat loss analysis for an open-cavity tubular solar receiver, *Proceedings of the 16th International Heat Transfer Conference*, IHTC-16, August 10-15, 2018, Beijing, China, IHTC16-24010.

Wolff, T.M., 2020, Initial testing of a collector for a solar-dish Brayton cycle, MEng dissertation, University of Pretoria, to be submitted.

Wolmarans, J.R., Craig, K.J., 2019, One-way fluid-structure interaction of a medium-sized heliostat using scale-resolving CFD simulation, *Solar Energy*, Vol. 191, pp. 84-99.

Wu, Shuang-Ying, Xiao, Lan, Cao, Yiding, Li, You-Rong, 2010, Convection heat loss from cavity receiver in parabolic dish solar thermal power system: A review, *Solar Energy*, Vol. 84, pp. 1342-1355.

Yuan, J.K., Ho, C.K., Christian, J.M., 2015, Numerical simulation of natural convection in solar cavity receivers, *Journal of Solar Energy Engineering*, Vol. 137, 031004-1-10.

Appendix A: Transformation from mesh element to SolTrace primitive geometry

Due to a lack of proper documentation on the transformation of primitive geometries in SolTrace, the following description is provided, with an irregular triangle being used as an example.

The information needed to define a typical flat irregular triangle in SolTrace is the coordinates, aim points, the rotation around the z-axis, as well as the coordinates of each corner of the triangle on an *x*-*y*-plane. The problem with this way of defining the element aperture is that one is unable to clarify how the transformation occurs without documentation. If one assumes that the translation occurs after the rotation around the origin, there are still at least 12 different ways of rotating the element. This is if one assumes that the rotation occurs using either one of the two conventions: the Euler angle or the Tait-Bryan angle rotations (both providing six possible sequences of rotation axes).

This documentation provides the reader with the procedure of transforming the element, where it lies in its position, which one would analyse to the position where one could obtain the information needed as input for SolTrace.

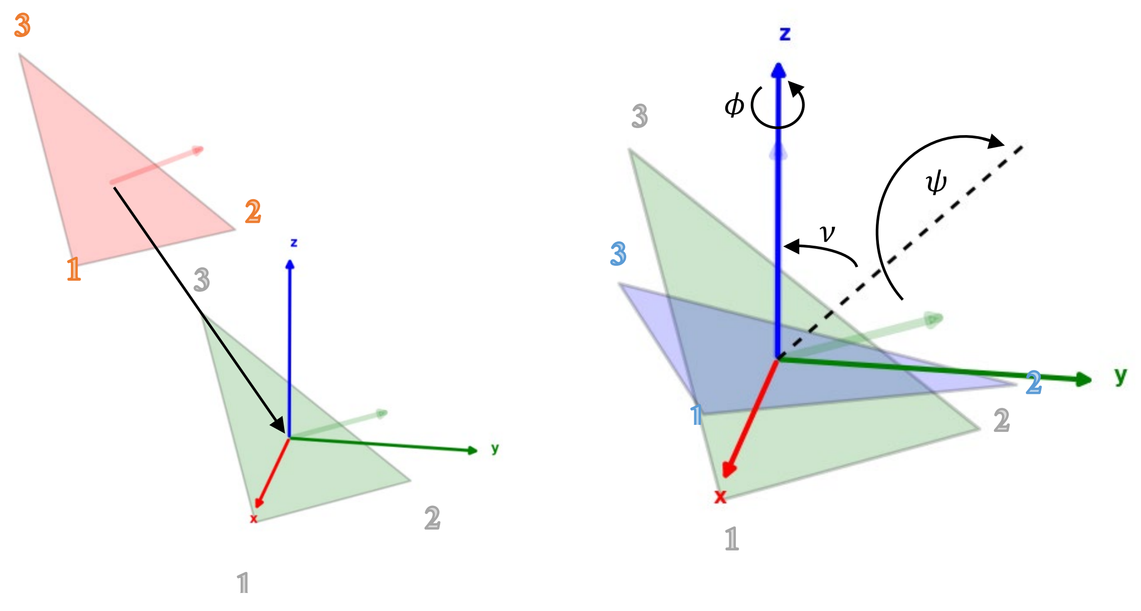


Figure A.1 The transformation of a mesh element to a SolTrace primitive irregular triangle illustrated with the image on the left: (a): displaying the translation to the origin; and the picture on the right: (b): displaying the rotation around the origin.

One needs to define a reference point for the object to start. This reference point around which the object will rotate is needed. For simplicity, the authors decided to use the centroid of the triangle as reference point by taking the average of the coordinates of the three corners of the triangle. The defined reference point will be used as input values for the *x*-, *y*- and *z*-coordinates in the SolTrace System Stage section. The element is then translated to the origin point by subtracting the reference point's coordinates from each corner's coordinates. This is formulated in Equation (A-1) and illustrated in Figure A.1a as follows:

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix}_{\text{translate}} = \begin{bmatrix} x \\ y \\ z \end{bmatrix}_{\text{corner}} - \begin{bmatrix} x \\ y \\ z \end{bmatrix}_{\text{ref}} \qquad \text{(A-1)}$$

The normal vector of the triangle plane can now be calculated by having the cross-product of the two vectors that move from one corner to the other two corners. For simplicity, it is assumed that the corner one used as the point of origin is labelled 1, and the other two corners are labelled 2 and 3. The cross-product, which also represents the normal vector of the plane, is then calculated as follows:

$$\vec{v}_{normal} = \langle x_1 - x_2, y_1 - y_2, z_1 - z_2 \rangle \times \langle x_2 - x_3, y_2 - y_3, z_2 - z_3 \rangle \qquad \text{(A-2)}$$

The aim points that will be used in SolTrace are then calculated as follows:

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix}_{\text{aim}} = \begin{bmatrix} x \\ y \\ z \end{bmatrix}_{\text{ref}} + \begin{bmatrix} v_x \\ v_y \\ v_z \end{bmatrix}_{\text{normal}} \qquad \text{(A-3)}$$

With the coordinates and aim points calculated, the next step is to rotate the geometry. In this instance, rotation is defined by three variables, ϕ, ν and ψ. Consider the illustration given in Figure A.1b:

- ϕ represents the angle of rotation about the *z*-axis, also known as roll;
- ν represents the angle of rotation about the *y*-axis, also known as pitch;

- ψ represents the angle of rotation about the *x*-axis, also known as yaw.

First, it is assumed that there is no rotation about the *z*-axis. Therefore, one can assume that the value corresponding to the rotation around the z-axis (ϕ) is zero. The other two variables are calculated as follows:

The normal vector's (v_{normal}) x and z components are used to form a new vector that represents the vector that would form after the pitch rotation takes place. The new vector, represented as v_v, is constructed as a unit vector as follows:

$$v_v = \frac{\langle v_{normal,x}, 0, v_{normal,z} \rangle}{\sqrt{{v_{normal,x}}^2 + {v_{normal,y}}^2 + {v_{normal,z}}^2}} \tag{A-4}$$

The angle v is then calculated as follows:

$$v = \sin^{-1}\left(\frac{v_{v,x}}{\sqrt{{v_{v,x}}^2 + {v_{v,z}}^2}} \right) \tag{A-5}$$

One could use the following equation to calculate the yaw angle (ψ):

$$\psi = -\cos^{-1}(v_v \cdot v_{normal}) \tag{A-6}$$

With the angles of rotations calculated, one can calculate the rotation matrices for each axis. These are represented with the following three forms of rotation matrices:

$$R_z(\phi) = \begin{bmatrix} \cos\phi & -\sin\phi & 0 \\ \sin\phi & cos\phi & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$R_y(v) = \begin{bmatrix} \cos v & 0 & \sin v \\ 0 & 1 & 0 \\ -\sin v & 0 & \cos v \end{bmatrix} \tag{A-7}$$

$$R_x(\psi) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\psi & -\sin\psi \\ 0 & \sin\psi & \cos\psi \end{bmatrix}$$

With the separate rotation matrices calculated, one can calculate one overall rotation matrix as follows:

$$R(\phi, v, \psi) = R_z(\phi) R_y(v) R_x(\psi) \tag{A-8}$$

The transformed coordinates on the *x*-*y*-plane can now be calculated using the rotation matrix and the coordinates of the transformed triangle. This is done using the following equation:

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix}_{transform} = \begin{bmatrix} & R(\phi, v, \psi) & \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix}_{translate} \tag{A-9}$$

This equation is used for each corner of the triangle. The transformed coordinates are then used as the *x*- and y-coordinates in the aperture editor. One will note that the *z*-coordinate will be zero or close to zero. This is to be expected, since the triangle aperture is meant to be rotated onto the *x*-*y* plane.

Appendix B: Experimental conduction heat loss

The values used for the calculation of the conduction heat losses through the top, bottom and side walls of the receiver are shown in Tab.B.1.

Property	**Side 1**	**Side 2**	**Side 3**	**Side 4**	**Top**	**Bottom**
Average receiver wall temperature (measured) [K]	943.8	944.8	946.7	946.7	926.9	959.8
Average insulation surface temperature (measured) [K]	324.3	351.9	347.5	357.4	346.3*	385.1*
Average heat transfer area [m^2]	0.307	0.307	0.307	0.307	0.185	0.122
Average insulation conductivity (calculated) [W/m.K]	0.106	0.108	0.108	0.109	0.107	0.112
Average insulation thickness [m]	0.1	0.1	0.1	0.1	0.1	0.05
Average conduction heat loss rate (calculated) [W]	202.3	197.4	199.1	197.1	115.7	157.2

Table B.1 Summary of conduction heat losses (*calculated based on average side heat transfer coefficient)

Additional material (to be made available online)

Appendix C: Code for parsing mesh file of complex geometry and generating input files for SolTrace simulation

The following Python code is used to parse the .msh file of the complex geometry surface mesh, which will be simulated in SolTrace, as well as to generate the input files needed for running the SolTrace code given in Appendix D.

```
# -*- coding: utf-8 -*-
"""
This file parses the assigned .msh file, and converts it to the required input files needed for a SolTrace
simulation

@author: M Slootweg
"""

import os
import numpy as np
import pandas

def soltrace_transform(x,y,z,face_section):
    """
    The triangular element defined in space is transformed to values that can be used as input for SolTrace
    :param x: The x-positions of the triangular element
    :param y: The y-positions of the triangular element
    :param z: The z-positions of the triangular element
    :param face_section: face section value is added as part of the output
    :return: A list of values in the order that is consistent with the input needed for SolTrace.
    """

    inflation_factor = 1.0

    x_mid = (x[0]+x[1]+x[2])/3
    y_mid = (y[0]+y[1]+y[2])/3
    z_mid = (z[0]+z[1]+z[2])/3

    z_rotate = 0

    x_new = x - x_mid
    y_new = y - y_mid
    z_new = z - z_mid

    vec_norm = np.cross([x_new[0]-x_new[1],y_new[0]-y_new[1],z_new[0]-z_new[1]],[x_new[0]-
x_new[2],y_new[0]-y_new[2],z_new[0]-z_new[2]])
    vec_norm = vec_norm/np.linalg.norm(vec_norm)

    x_aim = vec_norm[0] + x_mid
    y_aim = vec_norm[1] + y_mid
    z_aim = vec_norm[2] + z_mid

    R = np.linalg.inv(Rot_RPY(vec_norm))
    vector_new = np.array([x_new,y_new,z_new])
    vector_transform = np.mat(R) * np.mat(vector_new)
    vector_transform = vector_transform*inflation_factor

    x_transform = np.array(vector_transform[0,:]).ravel()
    y_transform = np.array(vector_transform[1,:]).ravel()

    x_transform_new = np.array([x_transform])
    y_transform_new = np.array([y_transform])

    area_matrix = np.transpose(np.r_[x_transform_new, y_transform_new, np.ones((1,3))])
    area = np.abs(0.5*np.linalg.linalg.det(area_matrix))


return(np.array([x_mid,y_mid,z_mid,x_aim,y_aim,z_aim,z_rotate,x_transform[0],y_transform[0],x_transform[1],
y_transform[1],x_transform[2],y_transform[2],area,face_section]))



def file_len(fname):
    with open(fname) as f:
        for i, l in enumerate(f):
            pass
    return i + 1

root = os.getcwd()
mesh_file = "example.msh")
file_name = os.path.join(root, mesh_file)

f = open(file_name)
```

```
eof = file_len(file_name)
current_pos = 1
node_num = 0
face_num = 0
sec_num = 0
nodes = []
X = []
Y = []
Z = []
faces = []
BCs = []

new_line = f.readline()

while current_pos <= eof:
    if new_line.startswith("(2"):
        dimensions = float(new_line[3])
        print("Dimensions = " + str(dimensions))
        new_line = f.readline()
        current_pos += 1

    elif new_line.startswith("(10"):
        if float(new_line[5]) == 0:
            nodes_info = []
            in_brac1 = new_line[new_line.find("(")+1:new_line.rfind(")")]
            in_brac2 = in_brac1[in_brac1.find("(")+1:in_brac1.rfind(")")]
            temp_node_info = map(str,in_brac2.split())
            for ind in temp_node_info:
                nodes_info.append(int(ind,16))
            new_line = f.readline()
            current_pos += 1
        elif float(new_line[5]) != 0:
            new_line = f.readline()
            current_pos += 1
            sec_num += 1
            while not new_line.startswith("))"):
                temp_node = list(map(float,new_line.split()))
                node_num+=1
                if dimensions == 2:
                    nodes.append([sec_num, node_num, temp_node[0], temp_node[1]])
                    X.append(temp_node[0])
                    Y.append(temp_node[1])
                elif dimensions == 3:
                    X.append(temp_node[0])
                    Y.append(temp_node[1])
                    Z.append(temp_node[2])
                    nodes.append([sec_num, node_num, temp_node[0], temp_node[1], temp_node[2]])
                new_line = f.readline()
                current_pos += 1

    elif new_line.startswith("(13"):
        print(str(new_line[5]))
        if float(int(new_line[5],16)) == 0:
            face_info = []
            in_brac1 = new_line[new_line.find("(")+1:new_line.rfind(")")]
            in_brac2 = in_brac1[in_brac1.find("(")+1:in_brac1.rfind(")")]
            temp_face_info = map(str,in_brac2.split())
            for ind in temp_face_info:
                face_info.append(int(ind,16))
            new_line = f.readline()
            current_pos += 1
        else:
            face_info2 = []
            in_brac1 = new_line[new_line.find("(")+1:new_line.rfind("(")]
            in_brac2 = in_brac1[in_brac1.find("(")+1:in_brac1.rfind(")")]
            temp_face_info = map(str,in_brac2.split())
            for ind in temp_face_info:
                face_info2.append(int(ind,16))
            sec_num2 = float(face_info2[0])
            face_type = float(face_info2[3])
            element_type = float(face_info2[4])
            new_line = f.readline()
            current_pos += 1
            while not new_line.startswith("))"):
                if element_type == 4:
                    temp_face = map(str,new_line.split())
                    temp_face2 = []
                    for ind in temp_face:
                        temp_face2.append(int(ind,16))
                    if len(temp_face2) != 0:
                        face_num+=1
                        len1 = np.sqrt((X[temp_face2[0]-1]-X[temp_face2[2]-1])**2 + (Y[temp_face2[0]-1]-
Y[temp_face2[2]-1])**2 + (Z[temp_face2[0]-1]-Z[temp_face2[2]-1])**2)
                        len2 = np.sqrt((X[temp_face2[1]-1]-X[temp_face2[3]-1])**2 + (Y[temp_face2[1]-1]-
Y[temp_face2[3]-1])**2 + (Z[temp_face2[1]-1]-Z[temp_face2[3]-1])**2)
                        if len1 > len2:
                            faces.append([sec_num2, face_num, face_type, 3, temp_face2[0], temp_face2[1],
temp_face2[3]])
```

```
                            face_num+=1
                            faces.append([sec_num2, face_num, face_type, 3, temp_face2[1], temp_face2[2],
temp_face2[3]])
                        else:
                            faces.append([sec_num2, face_num, face_type, 3, temp_face2[0], temp_face2[2],
temp_face2[3]])
                            face_num+=1
                            faces.append([sec_num2, face_num, face_type, 3, temp_face2[0], temp_face2[1],
temp_face2[2]])
                    new_line = f.readline()
                    current_pos += 1

                elif element_type == 3:
                    temp_face = map(str,new_line.split())
                    temp_face2 = []
                    for ind in temp_face:
                        temp_face2.append(int(ind,16))
                    if len(temp_face2) != 0:
                        face_num+=1
                        faces.append([sec_num2, face_num, face_type, 3, temp_face2[0], temp_face2[1],
temp_face2[2]])

                    new_line = f.readline()
                    current_pos += 1

                else:
                    new_line = f.readline()
                    current_pos += 1

    elif new_line.startswith("(45"):
        temp_arr = new_line.split()
        if temp_arr[2] == "wall":
            in_bracl = new_line[new_line.find(" ")+2:new_line.find(")")]
            id_zone_name = in_bracl.split()[2]
            id_zone = in_bracl.split()[0]
            BCs.append([id_zone,id_zone_name])
        new_line = f.readline()
        current_pos += 1
    else:
        new_line = f.readline()
        current_pos += 1

nodes = np.array(nodes)
faces = np.array(faces)
BCs = np.array(BCs)

print("")
print("Start boundary selection:")
print("")
k=0
BC_select = []
if BCs.any:
    print("Select boundaries to be simulated:")
    print(" ")
    for k in range(len(BCs)):
        print(k+1 , ". " , BCs[k][1])
else:
    print("There are no boundaries which are classified as walls.")

print(" ")
BC_select = input("Select the boundaries to be simulated in list form[]: ")

BC_select = eval(BC_select)
Boundaries = np.array([[0,"heliostat"],[0,"flat_aperture"]])
for i in BC_select:
    Boundaries = np.append(Boundaries,[BCs[int(i)-1][:]],axis=0)

section_list = [row[0] for row in Boundaries[2:]]
section_list = [float(i) for i in section_list]

print(" ")
print("Which of the selected boundaries is to be used as heat source/absorber?")
Absorber_select = input("Select in list form[]: ")

Absorber_select = eval(Absorber_select)
Absorber_boundaries = []
for i in Absorber_select:
    Absorber_boundaries = np.append(Absorber_boundaries,[BCs[int(i)-1][0]],axis=0)

Absorber_boundaries = [int(i) for i in Absorber_boundaries]

optic_properties = pandas.read_csv('optical_property_set.csv')
optic_properties = optic_properties.values

print(" ")
print("Start assigning optical properties to boundaries:")
print(" ")
k=0
```

```
if optic_properties.any:
    print("Select from the following optical properties list:")
    print(" ")
    for k in range(np.shape(optic_properties)[0]):
        print(k+1 , ". " , optic_properties[k][0])
else:
    print("There are no boundaries which are classified as walls.")

print(" ")
optic_select = []
optics = [np.zeros(np.shape(optic_properties)[1])]
for i in range(np.shape(Boundaries)[0]):
    optic_select = (input(str(Boundaries[i][1])+" : "))
    optics = np.append(optics,np.array(optic_properties)[[int(optic_select)-1]],axis=0)
optics = np.delete(optics, (0), axis=0)

boundary_optics = np.append(Boundaries,optics,axis=1)

full_optics_list = [row[1].lower() for row in boundary_optics[0:]]
INDEX = []
for index in enumerate(full_optics_list):
    if index[1] == "aperture":
        INDEX.append(index[0])

aperture = []
other_boundary_optics = []
i = 0
for row in boundary_optics:
    if i in INDEX:
        aperture.append(row)
    else:
        other_boundary_optics.append(row)
    i += 1

other_boundary_optics_list = [row[0] for row in other_boundary_optics[2:]]
other_boundary_optics_list = [float(i) for i in other_boundary_optics_list]
aperture_list = [row[0] for row in aperture]
aperture_list = [float(i) for i in aperture_list]

X = np.array(X)
Y = np.array(Y)
Z = np.array(Z)

root = os.getcwd()

file_name = os.path.join(root, "boundary_optics.txt")

with open(file_name, "w") as fh:
    for i in range(np.shape(boundary_optics)[0]):
        boundary_stuff = ','.join(map(str, boundary_optics[[i]].ravel()))
        fh.write(str(boundary_stuff))
        fh.write("\n")

file_name = os.path.join(root, "receiver_coordinates.txt")

k = 0
absorber_element_list = []
absorber_element_coordinates = []

with open(file_name, "w") as fh:
    for i in range(0,int(faces[-1,1]),1):
        if faces[i,0] in other_boundary_optics_list:
            x = []
            y = []
            z = []
            for j in range(int(faces[i,3])):
                x.append(X[int(faces[i,4+j])-1])
                y.append(Y[int(faces[i,4+j])-1])
                z.append(Z[int(faces[i,4+j])-1])
            sol_trans = ','.join(map(str, soltrace_transform(x,y,z,faces[i,0])))
            fh.write(str(sol_trans))
            fh.write("\n")
            for l in Absorber_boundaries:
                if faces[i,0] == int(l):
                    absorber_element_list.append(k)
                    absorber_element_coordinates.append([x,y,z])
            k += 1

file_name = os.path.join(root, "absorber_element_coordinates.txt")

with open(file_name, "w") as fh:
    for i in absorber_element_coordinates:
        absorber_element = ','.join(map(str, i))
        fh.write(str(i))
        fh.write("\n")

file_name = os.path.join(root, "absorber_element_list.txt")
```

```python
with open(file_name, "w") as fh:
    for i in absorber_element_list:
        fh.write(str(i))
        fh.write("\n")

file_name = os.path.join(root, "aperture_coordinates.txt")

with open(file_name, "w") as fh:
    for i in range(0,int(faces[-1,1]),1):
        if faces[i,0] in aperture_list:
            x = []
            y = []
            z = []
            for j in range(int(faces[i,3])):
                x.append(X[int(faces[i,4+j])-1])
                y.append(Y[int(faces[i,4+j])-1])
                z.append(Z[int(faces[i,4+j])-1])
            sol_trans = ','.join(map(str, soltrace_transform(x,y,z,faces[i,0])))
            fh.write(str(sol_trans))
            fh.write("\n")

print("")
print("Script finished succesfully")
```

Appendix D: SolTrace script

The code presented here is used to simulate a complex geometry receiver for a parabolic dish. The input files generated using the code of Appendix C are used as input files. This code is written in the LK scripting language, which is the supported language in SolTrace.

```
/*
Script for a parabolic dish and complex geometry receiver
*/

/* ******************************************
     Python Commands
   ****************************************** */

//  Import commands from spreadsheet or text file

//  The simulation_list.txt contains information which is used as summary of the inputs needed. These input
//  values are assigned to variables.

py_file = open('simulation_list.txt', 'r');

xrow = [];
j = 0;
simulation_list = [];
line="";

while ( read_line( py_file, line ) )
{
       xrow = split(line, ',');
       simulation_list[j] = xrow[0];
       j = j + 1;
}
outln(simulation_list);
x = to_real(simulation_list[0]);
y = to_real(simulation_list[1]);
z = to_real(simulation_list[2]);
amount_of_rays = to_real(simulation_list[3]);
dni_value = to_real(simulation_list[4]);
heliofield_file = simulation_list[5];
simulate_dish = to_real(simulation_list[6]);
simulate_receiver = to_real(simulation_list[7]);
project_name = simulation_list[8];
element_hit_file = to_real(simulation_list[9]);
ray_data_file = to_real(simulation_list[10]);
amount_of_seeds = to_real(simulation_list[11]);

/* ******************************************
     SOLTRACE SCRIPT OF PS10 HELIOSTAT FIELD
   ****************************************** */

ProjectName = project_name;

/* ******************************************
     configure Direct Normal Irradiance
   ****************************************** */

I_n = dni_value;
```

```
outln("DNI = "+I_n);

/* ******************************************
     configure Sun Shape
   ****************************************** */

Sun.useldh = false;

Sun.x = s[0];
Sun.y = s[1];
Sun.z = s[2];
Sun.shape = 'p';
Sun.halfwidth = 4.65;
sunopt(Sun);

/* ******************************************
     configure an optical property data set
   ****************************************** */

fileID=cwd() + '\\boundary_optics.txt';
py_file = open(fileID, 'r');

xrow = [];
j = 0;
optical_properties = [];
line="";

while ( read_line( py_file, line ) )
{
       xrow = split(line, ',');
       optical_properties[j][0] = to_real(xrow[0]);
       optical_properties[j][1] = xrow[1];
       optical_properties[j][2] = xrow[2];
       optical_properties[j][3] = to_real(xrow[3]);
       optical_properties[j][4] = to_real(xrow[4]);
       optical_properties[j][5] = to_real(xrow[5]);
       optical_properties[j][6] = to_real(xrow[6]);
       optical_properties[j][7] = to_real(xrow[7]);
       optical_properties[j][8] = to_real(xrow[8]);
       optical_properties[j][9] = to_real(xrow[9]);
       optical_properties[j][10] = to_real(xrow[10]);
       j = j + 1;
}
no_optics = j;

clearoptics(); // remove any optical properties currently defined

for (j=0; j<(no_optics); j++)
{
       k = 1;
       i = 0;
       while (j > i)
       {
              if (optical_properties[j][2] == optical_properties[i][2])
                     k = 0;
              i++;
       }
       if (k == 1)
       {
              addoptic(optical_properties[j][2]); // adds element to current stage
              general_optics_front.refl = optical_properties[j][3];
              general_optics_front.trans = optical_properties[j][4];
              general_optics_front.errslope = optical_properties[j][5];
              general_optics_front.errspec = optical_properties[j][6];
              general_optics_back.refl = optical_properties[j][7];
              general_optics_back.trans = optical_properties[j][8];
              general_optics_back.errslope = optical_properties[j][9];
              general_optics_back.errspec = optical_properties[j][10];

              opticopt( optical_properties[j][2], 1, general_optics_front);
              opticopt( optical_properties[j][2], 2, general_optics_back);
       }
}

/* ******************************************
       Import receiver elements
       ****************************************  */
clearstages(); // clear the system

aiming_height = 2.897;
deviation = 0.14875;
rim_height = 0.09755;
c = 1/(2*aiming_height);

if (simulate_receiver == 1)
{
       fileID=cwd()+ '\\receiver_coordinates.txt';
       py_file = open(fileID, 'r');

       xrow = [];
       j = 0;
       receiver_coordinates = [];
       line="";

       while ( read_line( py_file, line ) )
       {
              xrow = split(line, ',');
              receiver_coordinates[j][0] = to_real(xrow[0]);
              receiver_coordinates[j][1] = to_real(xrow[1]);
              receiver_coordinates[j][2] = to_real(xrow[2]);
              receiver_coordinates[j][3] = to_real(xrow[3]);
              receiver_coordinates[j][4] = to_real(xrow[4]);
              receiver_coordinates[j][5] = to_real(xrow[5]);
              receiver_coordinates[j][6] = to_real(xrow[6]);
              receiver_coordinates[j][7] = to_real(xrow[7]);
              receiver_coordinates[j][8] = to_real(xrow[8]);
              receiver_coordinates[j][9] = to_real(xrow[9]);
              receiver_coordinates[j][10] = to_real(xrow[10]);
              receiver_coordinates[j][11] = to_real(xrow[11]);
              receiver_coordinates[j][12] = to_real(xrow[12]);
              receiver_coordinates[j][13] = to_real(xrow[13]);
              receiver_coordinates[j][14] = to_real(xrow[14]);
              j = j + 1;
       }
       no_receiver_elements = j;
       outln("Number of receiver elements "+no_receiver_elements);

       addstage( 'single_stage' );

       // we need to set the current stage to be active so we can add elements to it
       activestage( 'single_stage' );

       Receiver_details.virtual = false;
       Receiver_details.multihit = true;
       Receiver_details.tracethrough = false;
       Receiver_details.x = 0;
       Receiver_details.y = 0;
       Receiver_details.z = 0;
       Receiver_details.ax = 0;
       Receiver_details.ay = 0;
       Receiver_details.az = 1;
       Receiver_details.zrot = 0;

       stageopt('single_stage',Receiver_details);

       for (j=0; j<(no_receiver_elements); j++)
       {
              addelement(); // adds element to current stage
              receiver_details.en = true;
              receiver_details.x = receiver_coordinates[j][0];
              receiver_details.y = receiver_coordinates[j][1];
              receiver_details.z = receiver_coordinates[j][2] + aiming_height - deviation;
              receiver_details.ax = receiver_coordinates[j][3];
              receiver_details.ay = receiver_coordinates[j][4];
              receiver_details.az = receiver_coordinates[j][5] + aiming_height - deviation;
              receiver_details.zrot = receiver_coordinates[j][6];
              receiver_details.aper =
['i',receiver_coordinates[j][7],receiver_coordinates[j][8],receiver_coordinates[j][9],receiver_coordinates[
j][10],receiver_coordinates[j][11],receiver_coordinates[j][12],0,0];
              receiver_details.surf = ['f',0,0,0,0,0,0,0,0];
              for (k=0; k<(no_optics); k++)
              {
                     if (receiver_coordinates[j][14] == optical_properties[k][0])
                     {
                            receiver_details.optic = optical_properties[k][2];
                     }
              }
              receiver_details.comment = receiver_coordinates[j][13];

              elementopt( j, receiver_details );
       }

       receiver_coordinates = [];

}

/* *****************************************
     add dish and receiver cover
   ***************************************** */

element_count = j;
receiver_height = 0.9162;
receiver_width = 0.614;
receiver_apperture_width = 0.400;
receiver_rim_width = (receiver_width-receiver_apperture_width)/2;
```

```
receiver_rim_pos = receiver_apperture_width/2+(receiver_width-receiver_apperture_width)/4;

if (simulate_dish == 1)
{

       addelement();
       dish_details.en = true;
       dish_details.x = 0;
       dish_details.y = 0;
       dish_details.z = 0;
       dish_details.ax = 0;
       dish_details.ay = 0;
       dish_details.az = 1;
       dish_details.zrot = 0;
       dish_details.aper = ['c',4.8,0,0,0,0,0,0,0];
       dish_details.surf = ['p',c,c,0,0,0,0,0,0];
       dish_details.optic = optical_properties[0][2];
       elementopt( element_count,dish_details);
       element_count = element_count+1;
}

/* *****************************************
    start the trace
   ***************************************** */

for (seed=1; seed<(amount_of_seeds+1); seed++)
{

       traceopt({ 'rays'= amount_of_rays , 'maxrays' = 10*amount_of_rays , 'seed'=seed , 'cpus'=7 ,
'include_sunshape'=true , 'optical_errors'=true} );
       trace();

       T_area = ((sundata(){"xmax"})-(sundata(){"xmin"}))*((sundata(){"ymax"})-(sundata(){"ymin"}));

       /* *****************************************
              Write all Ray Data to File
          ***************************************** */

       Inters=nintersect();

       if (ray_data_file == 1)
       {

              j=0;
              k=0;
              TEMP = "PosX,PosY,PosZ,Element,Stage,RayNumber";
              fileID=cwd() + '\\' + ProjectName + '\\trace_results_seed_'+seed+'_rays_'+amount_of_rays;

              if (file_exists(fileID))
                     {remove_file(fileID);}
              file = open(fileID,'w');
              write_line(file,TEMP);
              while (j < Inters)
                     {
                     temp=raydata(j);
                     if (temp[6]!=0)
                            {
                            TEMP = temp[0]+","+temp[1]+","+temp[2]+","+temp[6]+","+temp[7];
                            write_line(file,TEMP);
                            k = k+1;
                            }
                     j=j+1;
                     }
              close(fileID);
       }

       /* *****************************************
              Write Element Hits in stage to File
          ***************************************** */

       if (element_hit_file == 1)
       {
              T_area = ((sundata(){"xmax"})-(sundata(){"xmin"}))*((sundata(){"ymax"})-(sundata(){"ymin"}));
              DNI = to_real(I_n);
              Power_Per_Ray = DNI*T_area/sundata(){"nrays"};

              py_file = open('absorber_element_list.txt', 'r');

              xrow = [];
              j = 0;
              absorber_list = [];
              line="";

              while ( read_line( py_file, line ) )
              {
                     xrow = split(line, ',');
                     absorber_list[j] = to_real(xrow[0]);
                     j = j + 1;
```

```
		}
		no_absorber_elements = j;
		outln("Number of tested absorber elements "+no_absorber_elements);

		curr_stage = activestage('single_stage');
		elem_num = nelements();
		j=0;
		fileID=cwd() + '\\' + ProjectName + 
'\\Element_Hit_Results_seed_'+seed+'_rays_'+amount_of_rays;
		if (file_exists(fileID))
			{remove_file(fileID);}
		file = open(fileID,'w');
		outln("number of elements = ",elem_num);

		for (k=0; k<(no_absorber_elements); k++)
		{
			j = absorber_list[k];
			area = to_real(elementopt(j){"comment"});
			elem_hits = rayhits(0,j,1);
			Elem_Watts = (Power_Per_Ray*elem_hits)/(area); //[W/m2]
			write_line(file,Elem_Watts);
		}
		close(fileID);

		j=0;
		fileID=cwd() + '\\' + ProjectName + 
'\\Element_Hit_Results_seed_'+seed+'_rays_'+amount_of_rays+'_part_2';
		if (file_exists(fileID))
			{remove_file(fileID);}
		file = open(fileID,'w');

		for (k=(no_absorber_elements-1); k>=0; k--)
		{
			j = absorber_list[k];
			area = to_real(elementopt(j){"comment"});
			elem_hits = rayhits(0,j,1);
			Elem_Watts = (Power_Per_Ray*elem_hits)/(area); //[W/m2]
			write_line(file,Elem_Watts);
		}
		close(fileID);
	}
}
```