Domestic Concentrated Solar Thermal Co-generation Plants for Pakistan PHY 491/492 Senior Project Thesis

Abdullah Khalid
2012-10-0168
Hassan Bukhari
2012-10-0152
Department of Physics
Department of Electrical Engineering
School of Science and Engineering
Lahore University of Management Sciences

Saturday 26$^{\text{th}}$ May, 2012

Abstract

An extensive optical and thermal model of a linear parabolic solar collector has been developed and coded in MATLAB. Ray tracing is used for the optical model which accounts for collector geometry, tracking errors, location of the setup, construction errors, and receiver misalignment to calculate the intercept factor (optical efficiency). The thermal analysis is done by solving heat balance equations in the receiver which in our case is a cheap alternative to a vacuum tube, a metal tube painted black covered with a glass sleeve. Errors, even small ones have extensively been taken into account, motivated by the economics and production capabilities of developing countries. Using this complete model, we can gauge the optimum parameters of the system, most importantly collector size and focal length, and receiver tube diameter. Currently we are testing the turbine and setting up a test bed to validate our collector optical and thermal model. The code will be open sourced once complete and can be used by others working on solar thermal technology on a similar scale.

Contents

List of Figures

List of Tables

1 Introduction

The country is facing an electricity crises, the culprits both burgeoning demand and the inability of the government and private producers to setup large scale plants due to lack of investment and political issues. This has had drastic effects on both the quality of life and the economic activity in the country.

Second, there is an over reliance on fossil fuel based production strategies. In the face of the shortage, and the consequent increase in the prices, of fossil fuels this does not seem to be sustainable strategy.

Third, and perhaps most importantly, there is very little thought put into the environmental impact of the production strategies used in the country. On one hand fossil fuels have contributed mindlessly to increased levels of pollution. On the other hand, hydal power projects, though more environmentally friendly and sustainable, have their effects on the natural ecosystems.

Fourth, renewable energy has been an active area of research across the world for the past few decades and economically feasible solutions have been developed using various such technologies. Even though these technologies and solutions can sometimes be imported, there will always be a need to adapt and optimize them taking local conditions into account.There is however a dearth of renewable energy research in the country.

Fifth, academia and industry collaboration in the country is not at praiseworthy levels. There is a severe need to encourage academically driven solutions to industrial problems.

Taking all of these reasons into account, we decided to look into the field of concentrated solar thermal power as a electricity generation mechanism at the extremely small scale level of 1 kW. The dream was and is a roof top concentrated solar thermal co-generation plant that can provide electricity to the the masses of Pakistan reliably, cheaply and in an environmentally friendly way.

On the face of it, concentrated solar power is nothing arcane as say Quantum Field Theory. The principles are simple and immediately clear to a someone with a clear understanding of physics. The generation of electricity from concentrated solar has been demonstrated at the large scale in the form of solar thermal power stations in several countries [11].

There were three essential ways that this study was an extension of what has already been done in this way. They hinge upon the context within this study was done, illustrated somewhat by the opening paragraphs of this

report. The crucial point to notice is that Pakistan is a developing country. This provides three constraints for an application such as this.

First, access to a hefty investments necessary to develop a solar power plants do not seem to be available. Even though, some solar power plants have been operational for several years [11], this is still a sidelined method of power generation. This was an active area of research during the $70s$ [6], especially during the period when oil prices were on the increase. However, once these prices stabilized, research and funding dried up. This technology has only recently re-emerged as an active field of development in the face of rising fuel prices and dried up fuel wells. For instance, India is not looking solidly into generating power from solar thermal [32], and willing to pay the price to do it. Pakistan on the other hand cannot afford to setup power plants which will not only be expensive in capital terms but will also provide electricity more expensive than other methods. Pakistan is not yet able to pay the price for a environmentally better future.

This brings us to our second point. Any application or product that is developed within Pakistan or similar developing country must provide electricity at cheaper than the grid. As already stated above, existing implementations in the form of large power plants do not do so. Adoption in Pakistan can only happen if the efficiency is increased while bringing down costs so that the end user sees an advantage in switching to an environmentally friendly method of electricity generation.

The third constraint makes our task even harder. Developing countries often do not have access to either high quality production environments, nor assembly line like setups. Similarly, specialized materials such as say low emissivity paints which are used in applications such as solar thermal, are not available and have to be imported. So any product to be used at the commercial level must be able to be made from locally available materials and should not have such a complex design that available commercial production lines are not able to produce it.

These three constraints for us mean that existing development in this area must be extended in three direction. The first constraint implies that we must explore small scale systems, such as at the level of 1 kW, that are cheap enough to be financially viable for deployment in the country. The second constraint directly implies that the final product must generate electricity at cheaper than the grid. The third constraint implies that local materials be used and the design of the final product simple.

We will present our work in the following way. We begin by presenting a literature review in section 2 summarizing some of what has already been

done in the field. Then using this understanding, we will argue in section 3 why we choose to explore a certain design of a concentrated solar power unit, particularly the parabolic trough and the Tesla turbine. Once, the schematics have been laid out, in section 4 we will discuss the theory behind our model. We have developed a model and a simulation of the unit we propose to construct. Since, these are inherently linked, the model computational in nature, we will discuss these together in section 5. The optimization of the construction parameters using this simulation and some of the important results will be discussed in section 6. To verify our model, we have done some experimental work. This has been presented in section 7. We end with a summary and conclusion along with future directions for this work. An extensive bibliography will help a new student in navigating the field.

2 Literature Review

Work on solar power is distributed into two sections divided by time. Interest grew in the 1970s during the Gulf oil crisis, which sent oil prices skyrocketing hence making finding a new fuel of paramount importance. Great work was done both in the form of privte and government funded research and several textbooks were even written which are still in great use today. Once the price of oil went down again this field dried up and only recently have people become involved in research. We will look through work done in both these periods.

Some of the earliest work was funded by the U.S Department of Energy, Bendt et al did fundamental work regarding linear collector design. Collector parameters such as optimum concentration ratios and choice of rim angle are thoroughly discussed in their paper. [14]. As early as the 80s it was thought that solar power could be used to power not only the developed world but the developing world also using collectors manufactured from locally. Funded by USAID H. M. Guven undertook a large research project in India. His focus was that manufacturing and material errors play a significant role when constructing collectors in the developing world. Previously, errors were implemented as a multiplicative constant onto the total efficiency, however he proposed to introduce them as they appeared in the system, so that design decisions could be made based on them [5, 7, 6]. Surprisingly we could not find many implementations of this methodology. One unpublished paper from South Africa directly applied this work with mixed results.

During this time a few texts were written that still remain foundations of work in solar thermal power [8, 24]. These texts were studied extensively and are strongly recommender to any who wish to pursue study in this field. And near the end of this period of development the ASHRAE published a 45 page report on testing the thermal performance of solar collectors. These standards are widely used in testing troughs[2].

The US government picked up solar thermal energy near the year 2000 for in further development. The National Renewable Energy Laboratory, a U.S. Department of Energy Laboratory, again ventured into trough development, publishing several technical reports. R. Forristall as part of this project provides a detailed heat balance model for the receiver which they solve in EES [16].

Development of trough technology is not a solved problem even in the developed world. The EUROTrough is being developed by partner countries all over the Eurozone with the objectives are notably 'secure a European

developed and owned parabolic trough design', reduction of solar collector costs below 200 US/m along with several design side objectives [12].

We also saw several design innovations in a wide variety of papers. These innovations mostly talk about more efficient receivers design with a great many talking about compound parabolic troughs and comparing them to normal 1 axis tracking parabolas trough .Researchers developed an interesting way to bend the reflective material into the parabolic shape using compliant bands [13]. Multiple receiver pipes were suggested in to meet the problem of all the concentrated beam not landing on the receiver due to diffuse reflective surfaces [31]. Such work is directly applicable to our problem statement. Another interesting multiple chamber trough design with different absorber is suggested and analyzed using detailed ray tracing [18].

Existing papers on simulating concentrated power systems are widely available. They often use specific commercial software to bear the brunt of their calculations. A bulk of the literature review was based on going through such papers

Perhaps as steady work was not done in this field for the past twenty thirty years it opened up the possibility of student thesis. [23] used Monte Carlo Ray trace techniques to optimize different and novel linear collector geometries. Whereas models the various components of the SEGS VI solar power plant using a mix of TRNSYS and EES [27]. Models and constructs a complete solar thermal system including a parabolic dish with electronic tracker, impulse turbine and associated heat exchangers and pumps, very similar to the original goals of this thesis [25]. takes this development to Lesetho, where again a parabolic trough system is modeled and manufactured [15].

Work has emerged from Pakistan as well, however the quality has been mixed and development has been slow but it is picking up pace. For example researchers locally developed a trough that succumbed to warping of the reflective sheet. They are still involved in its development.The Renewable Energy and Energy Efficiency Lab at the College of EME NUST installed $400m^2$ of evacuated tube collectors at the Kohinoor drying plant to heat 30,000 liters of water / day from 20C to 60C [29]. Researchers at UET are involved in indigenously developing Stirling engines and have conducted tests on miniature prototypes [20] as well as working on Organic Rankine Cycles and have fabricated a complete system and tested it using n-Pentane [4]. Pakistani researchers based in Hyderabad have even created parabolic dishes by using mirror handicraft i.e. by sticking small pieces of glass onto a satellite dish. Furthermore along with this thesis, there is another bachelor thesis this year at the Lahore University of Management Sciences about generating

electricity using a Stirling engine and a parabolic dish. Indian researchers also tested there hand at making troughs from reinforced fiberglass at the cost of Dollar 48 and tested them according to ASHRAE standards [3].

Other players have been involved in installing projects in developing countries. The Global Environment Facility has installed four solar power stations in India, Mexico, Egypt and Morocco. Lastly, review papers were a good way to gain an introduction and to consolidate the different ideas in the this fast developing field.

3 Solution

In view of goals, motivations and constraints of this study and what has already been done, let us discuss the design of our proposed system. Consider the schematic given in Fig. 3. There are two interconnected cycles. Consider the one labeled as the turbine cycle. This is the standard power-from-heat generation cycle. Without going into the detailed thermodynamics, which will be discussed later, heat is extracted from the high temperature reservoir, some of it converted to useful work in the turbine, and the rest rejected to the low temperature reservoir. The heat moves around the cycle in the 'working fluid' of the turbine cycle. The turbine can be used to run an alternator. This converts mechanical work into electricity, the form of power that we would prefer.

Concentrated solar technology comes into play, by being a source of heat to the high temperature reservoir. The collector present in the collector cycle concentrates sunlight into a 'heat transfer fluid'. This fluid gives up it's heat to the working fluid in the heat exchanger between the two cycles. The pumps shown keep the fluids flowing around in their cycles and establish the required pressure. The condenser is needed to setup the low temperature reservoir by coupling the turbine cycle to the environment.

The above discussion should give us an idea of how the different parts of the system fit together. However, there are a lot of choices for each of the parts. Below we will discuss some of these various possibilities and pick the ones most suitable.

3.1 The Collector

The job of the solar collector is to collect heat energy from the sun and transfer it to our fluid. To achieve this goal, a wide variety of designs have been invented. These range from a simple network of pipes to complex Fresnel lens concentrators.

There are two competing goals when these designs were invented. The first is the goal of temperature attainment, whereby for we want a certain temperature for our application. The second is the goal of economic and mechanical efficiency, whereby we want a design that is both simple and cheap to construct.

We would like to quantify both these parameters, so we can decide between different collector designs easily. We can quantify the first by defining the concentration ratio. Though it leaves out various important factors such as

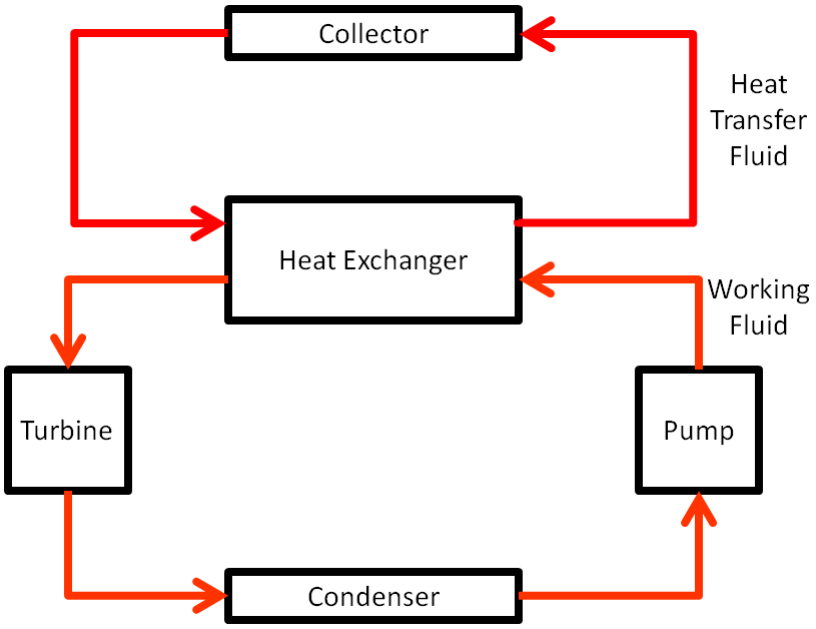


Figure 1: Overview of a concentrated solar thermal power plant.

reflection and absorption coefficients and a variety of other construction dependent factors, it is still decently correlated with the temperature achieved by the collector.

This is defined to be [17]

$$C = \frac{\text{Intensity incident on receiver}}{\text{Intensity incident on collector cross-section}}.$$

Though, often the simpler expression, dependent only on the geometry of the collector is used. [17]

$$C = \frac{\text{Area of receiver}}{\text{Area of collector}}.$$

The goal of construction efficiency is harder to quantify. It varies from situation to situation, and it is up to everyone to determine whether a particular design approved by the first goal is constructionally feasible. However, we can ask one important question. Is tracking required? The quantified parameter here is the the acceptance angle, which dictates how much light has to deviate from a particular axis before optical efficiency falls below an acceptable limit. We will see how this works when we look at various designs.

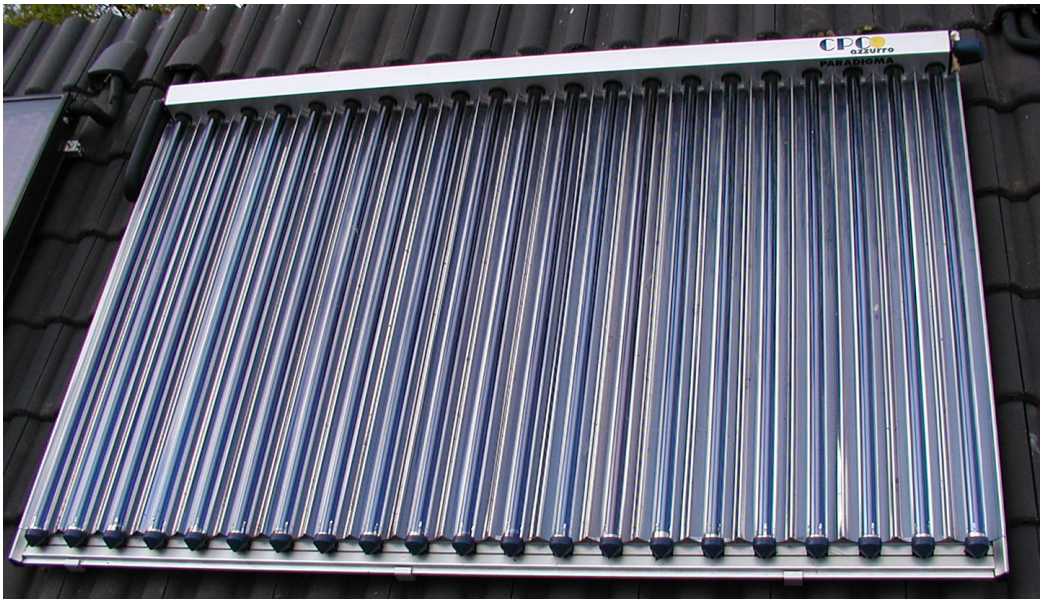


Figure 2: A flat plate collector.

We briefly discuss the various possible designs and by using the concentration ratio and construction issues, explain why we pick the parabolic trough as the ideal concentrator for our application.

We would idealy like high temperatures, since if you recall the efficiency of a heat engine goes up with the temperature difference between the two reserviors. However, from a perspective of construction and because this device is targeted towards a domestic product, we would perhaps not want temperatures so high to be potentially very unsafe.

3.1.1 Flat plate collectors

A picture of a flat plate collector is shown in Fig. 2. This is essentially a network of closely spaced pipes spread over a flat surface, through which a liquid can be pumped. No tracking is needed because the flat collector can absorb all sunlight, whatever the position of the sun. In essence, a full 90 deg acceptance angle. Below the pipes, usually a reflective sheet is spread out to reflect the light that misses the pipes back onto them.

With a configuration like this, the maximum possible ratio is obviously 1, the area of pipes is approximately the same as the area of the collector given closely spaced pipes and reflective sheets, while it can't be more. However,

Figure 3: A Fresnel lens collector being used for solar cooking.

it's relatively simple to construct a flat plate collector. On the other hand, it is found that these collectors cannot achieve temperatures more than 100 deg. This is why solar hot water systems, which typically use these collectors, are in much use today compared to other applications which require much higher temperatures.

3.1.2 Fresnel Lens Collectors

The second type of collector which we discuss are the fresnel lens type collectors. These use an assembly of lens and similar optical devices to direct light onto the required region. Pictures of some of these collectors are given in Fig. 3. As can be seen, some of the designs in this scheme are extremely complicated, not to mention using expensive lens material. However, these collectors have very large concentration ratios, and often also very large acceptance angles. This means high temperatures can be attained without tracking.

3.1.3 Parabolic Concentrators

The third type of collector we discuss are the parabolic type. The idea is that any light rays parallel to the central axis hitting any points reflects off to hit the focus of the parabola. Based on this idea, we can make two types of parabolic concentrators, either circular or linear.

A circular parabolic concentrator, or a parabolic dish, is shown in Fig. 4, while a linear parabolic concentrator or a parabolic trough is shown in Fig. 5. For both of these we can actually compute the theoretical maximum concentration ratio using the geometrical formula.

It comes out to be [28],

$$C_{\text{ideal,3D}} = \frac{1}{\sin^2 \phi}$$

and

$$C_{\text{ideal,2D}} = \frac{1}{\sin \phi}.$$

Here, ϕ is the half angle that any point on earth makes with the sun, which is about 16 minutes. $C_{\text{ideal,3D}}$ comes out to be $40,000$, while $C_{\text{ideal,2D}}$ comes out to be 216.

This seems to put things overwhelmingly in the favor of the parabolic dish, but let us consider the other factors. First, the dish is much harder to construct than the trough. The alignments of the mirrors is much harder to do, and there are other structural problems to be grappled with. Secondly, the dish requires two axis tracking compared to the trough's single axis tracking. This means that the dish has to be rotated around two different axis to effectively track the sun, while the trough requires only a single one, which is much simpler to implement.

Furthermore, temperatures that we need for our application can be attained by both these concentrators. So, the concentration ratio is not such an important factor. In comparison with other types of concentrators, we need to realize that parabolic concentrators have very small acceptance angles, and a high degree of tracking is needed at all times.

3.1.4 Compound Parabolic Concentrators

Compound parabolic concentrators (CPC) try to get away the requirement of tracking from parabolic concentrators. They is done by putting together

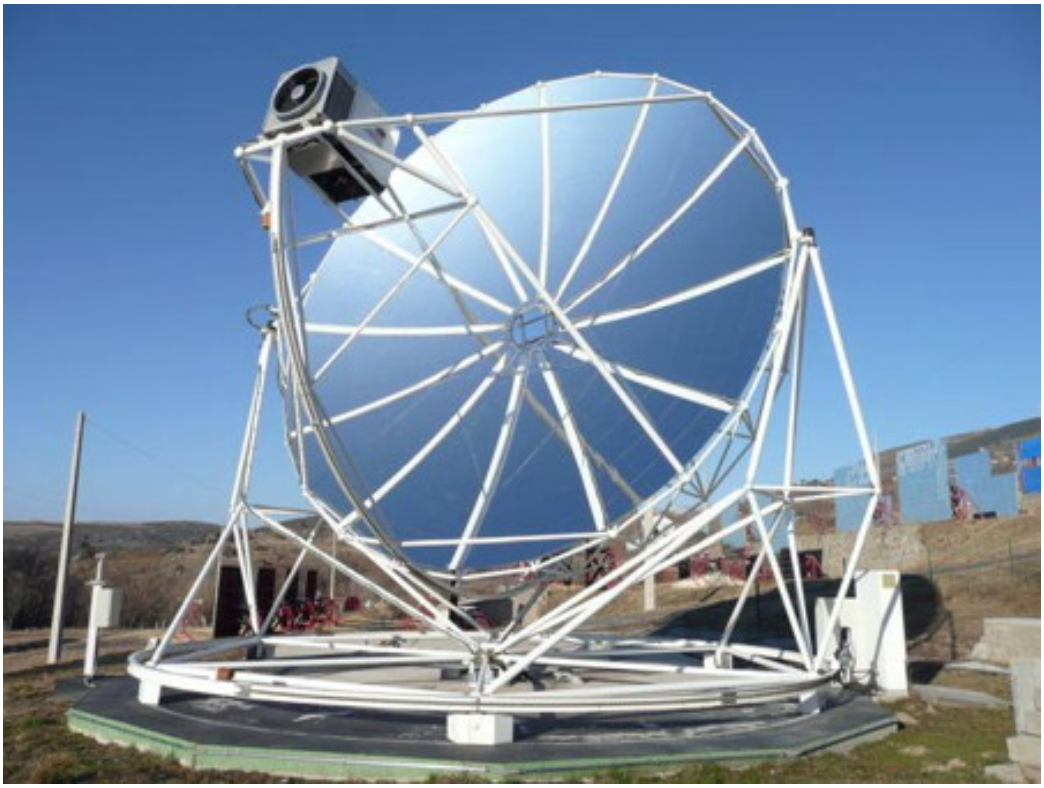


Figure 4: A parabolic dish system, known as the EuroDish.

multiple parabolas so that through multiple reflections ends up at the receiver. This is illustrated in Fig. 6 taken from [17].

However, compound parabolic concentrators are typically hard to make, and there is still some tracking that needs to be done. Their concentration ratio is given by

$$C_{CPC} = \frac{1}{\sin(\theta_{acceptance})}$$

There is an inverse relationship between the concentration ratio and the acceptance angle. In the limit of an acceptance angle of 0 they are parabolic concentrators.

3.1.5 Which collector?

A summary of the above discussion on collector types is given in Table 1

Figure 5: A parabolic trough system.

We concluded the following. The flat collector does not attain the right temperatures. The CPC and Fresnel type concentrators are too complicated to construct and do not seem to be low cost. Among the two parabolic concentrators, the dish is much harder to construct than the trough. Since, both give us our required temperature, we decide to go with the parabolic trough.

3.2 Turbine

The more commonly used engines are bladed turbines and piston engines. Both these engines are extremely complicated. Explanation of steam and piston engines and their problems goes here.

Stirling engines are possible to build on a small scale but have the problem of being too expensive. One reason for this is that they require expensive heat exchangers as they require quick cooling. For example a 1KW two cylinder Stirling engine is sold by Genoastirling is for 12,000 Euros or whereas Whispergen sells gas fired 1KW 4 cyliner sterlings for . Another manufacturer sells x for y (the german ones that came).

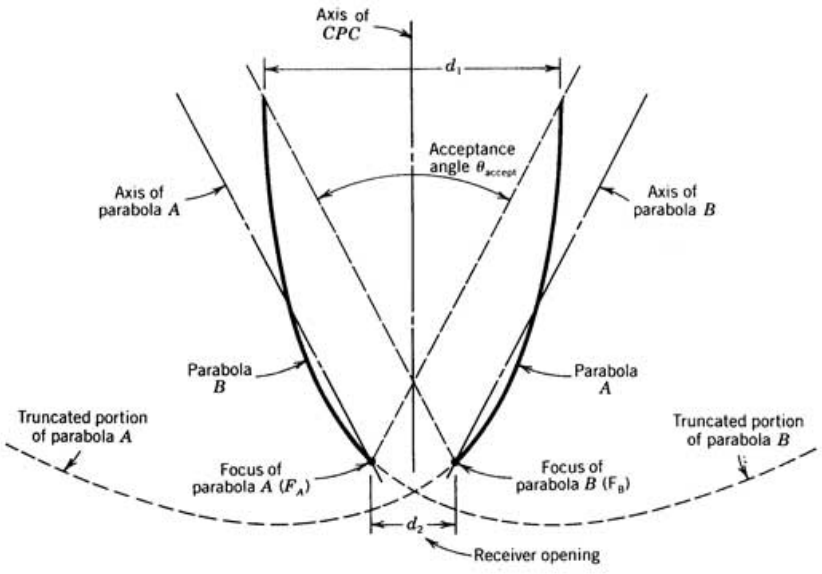


Figure 6: An illustration of how light travels in a compound parabolic concentrator.

Due to this inherent problem which each of the existing solution for this application we choose to explore a promising new avenue of engine type. Nikola Tesla developed his less well known bladeless turbine over a hundred years ago. And while one sometimes hears that it is the future of power it has very little presence in the actual workplace. The Tesla turbine has surprisingly simple way of working or as Tesla put it "All one needs is some disks mounted on a shaft, spaced a little distance apart and cased so that the fluid can enter at one point and go out at another" The core of the turbine is these disks. High speed fluid enters through a nozzle and into the disk as shown in Figure x. The molecules right next to the blades stick to the disks due to adhesion and stop. This stopping reduces the speed of nearby molecules which in turn slow down molecules further away. However, due to viscosity the remaining faster molecules do not want to become separated from these slower molecules and pull on them in turn pulling the disk to which the first ones have stuck onto. Thereby using the fundamental properties of liquids one can create torque. As the fluid does not collide with blades it can transfer more of its energy for rotation.

Hence working on a relatively new technology as the Tesla turbine was exciting and a very practical choice, selecting it was more of a practical consideration. A small Lahore based telecom company was also interested in

Collectors	Concentration	Construction	Pros and Cons
Flat plate	1	Simple	Cheap, low temperatures
Frenel Lens	High	Hard	High temperatures, expensive
Parabolic trough	216	Medium	High temperatures, tracking needed.
Parabolic dish	High	Hard	High temperatures, heavy tracking needed
CPC	High	Hard	High temperatures, expensive, light tracking

Table 1: A summary of collector types

developing a concentrated solar thermal power prototype at a similar scale. They wanted to work with us but interestingly their condition was that the engine should be a Tesla turbine to which we happily agreed.

3.3 Conclusion

So we have decided that our collector will be a parabolic trough, and our turbine will be a Tesla turbine, whose designs we know of. These are the two main design choices, according to which the rest of the system should be. We haven't said anything about the size of the parabolic trough, and as we will see later on there are a wide variety of options available in the design of a trough. Not only is the geometry of the trough, but also it's material are within our control, subject of course to what is available in the local market.

The Tesla turbine however is fixed because we are implementing an already existing design. This puts a fair amount of constraint on our system. However, turbine outputs are subject to the type, flow rate and temperature of the fluid used and this gives us some flexibility.

Let us now present an updated schematic, that reflects our design choices, in Fig. 3.3. The theory of the various parts, we will discuss in subsequent sections.

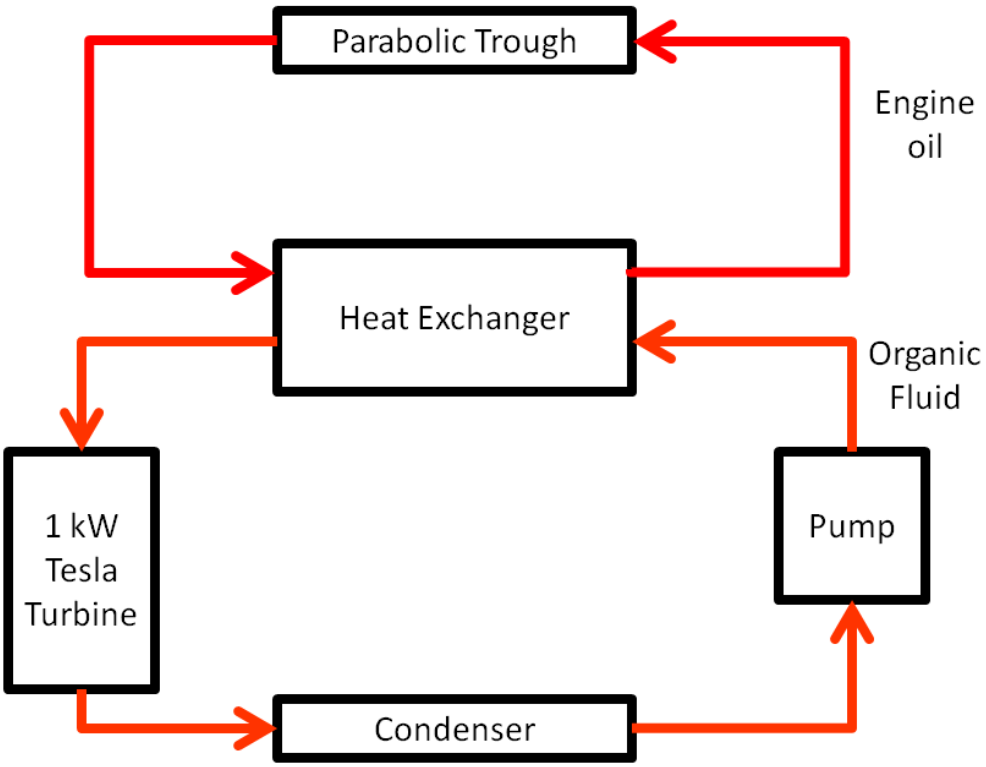


Figure 7: Our design choices incorporated into the overview of the concentrated solar thermal power plant.

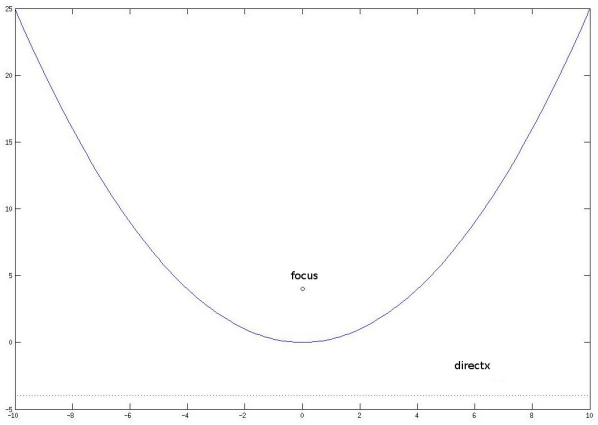


Figure 8: A parabola.

4 Theory

We present here the physics that governs the various modules of the system.

The above discussion is by no means comprehensive. Our aim was to highlight the various parts that can be present in our system. In this section, as already mentioned before, we discuss the theory that governs these various parts. We begin with the collector, which can itself be divided into two parts: the reflector and the receiver. We then move on to the parts of the turbine cycle: the heat exchanger, the turbine itself, and the pump. We finish off with a discussion of tracking.

4.1 Reflector Model

The parabola is very simple quadric function given by

$$y = \frac{x^2}{4f}$$

where f, called the focal length is the parameter of interest. A point, called the focus, and a line, called the directx, are shown. The crucial property that defines a parabola is that every point on the curve is equidistant from the focus and the nearest point on the line.

Why the parabola is useful is because of it's reflection property. This states that for for rays parallel to the the central axis of the parabola, they will reflect off the parabola and hit the focus.

The sun ray's, because the sun is so far away, tend to be effectively parallel by the time they reach the earth's surface. So, if we now rotate our parabola so that it faces the sun, all the sun's rays will concentrate on the focus. To collect lots of sunlight, we can make a long cylindrical parabola, whose focus will then be a line at a distance f from the vertex line, the bottom of the trough.

The central optimization parameter in the geometrical design of a concentrator is the intercept factor. This is defined to be the ratio of the energy incident on the receiver to the energy incident on the collector aperture.

This can be mathematically stated to be,[28].

$$\gamma = \frac{\int_{\text{receiver}} I(x)dx}{\int_{\text{reflector}} I(x)dx}.$$

Here $I(x)$ is the intensity of light at various points. This is shown to be one dimensional for notational simplicity. Being an efficiency parameter, it's value ranges from 0 to 1, the closer to the later the better.

We can define a parameter that captures even more information, the optical efficiency. This is given by

$$\eta = \gamma\alpha.$$

Essentially, the intercept factor into the absorbance of the receiver, α. The absorbance measures how much of the incident light is absorbed by the material, the rest either being reflected or passing through the material. The optical efficiency essentially tells us how much of the incident energy gets absorbed by the receiver, not just what is incident on it. However, since the optical efficiency decouples cleanly into these two parameters, we can consider only the intercept factor in the geometrical design of the collector, leaving the absorbance for when we consider the materials for our construction.

We now discuss various factors that determine the the intercept factor. The reflection properties of the material used to make the reflector is one such property. We will also discuss how to quantify various construction errors. Then we will devote a whole section to the sun.

4.1.1 Reflection Coefficients

This is perhaps the most important factor that determines the intercept factor. The reflector is made of a reflective sheet spread out on a parabolic

trough. The reflection property is quantified by the reflection coefficient, which can be derived from the Fresnel coefficients [10].

For s-polarized light we have,

$$R_s = \left| \frac{n_1 \cos\theta_i - n_2 \cos\theta_t}{n_1 \cos\theta_i + n_2 \cos\theta_t} \right|^2 .$$

Similarly for p-polarized light we have

$$R_p = \left| \frac{n_1 \cos\theta_t - n_2 \cos\theta_i}{n_1 \cos\theta_t + n_2 \cos\theta_i} \right|^2 .$$

These coefficients range from $0-1$, and determine how much of the incident intensity is reflected. By the conservation of energy the rest is reflected i.e. $T = 1 - R$. Since, sunlight does not have a preferred polarization the reflection coefficient we ultimately use is

$$R = \frac{R_s + R_p}{2}.$$

Within these formula, we have four parameters, two refractive indices and the incident and the transmission angle. However, remember that the two angles are related by Snell's law, which if you recall is

$$n_1 \sin\theta_i = n_2 \sin\theta_t.$$

We may mention here that n_1 is the refractive index of the medium from which the light is incident, while n_2 is the transmission medium.

Qualitatively, we see that the reflection coefficients are essentially dependent on the angle of incidence, and peak at normal incidence, when the light ray is perpendicular to the surface. This effect is important, since when we turn our parabola towards the sun, the light rays will all be parallel to the axis of the parabola, but will strike the surface at different angles. Hence, different points on the surface will reflect different intensities of light.

Secondly, the refractive indices in the reflection coefficient are not strictly a material property. They depend on the wavelength of the incoming light. Sunlight has a broad spectrum (Fig 9, and consequently for fully developed model, this distribution must be taken into account. However, this has not been taken into account in any model we have come across and we will correspondingly take an average value for the refractive index.

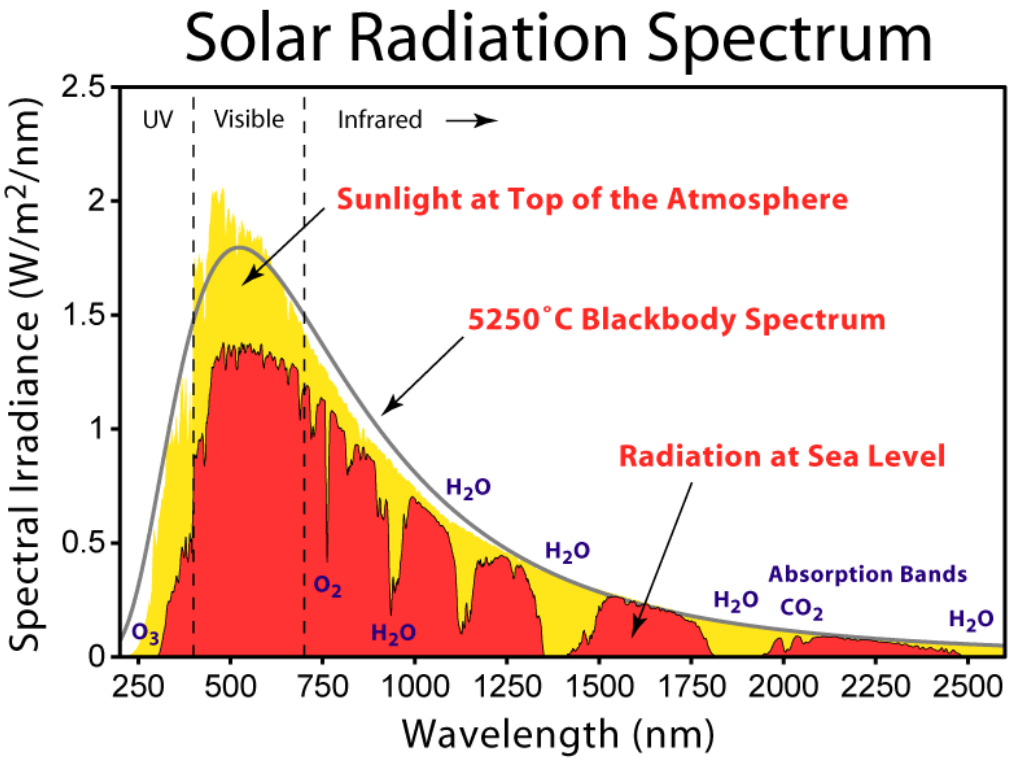


Figure 9: The spectrum of sunlight.

Thirdly, what this average formula misses out on is the specularity of surface. In the simplest model of light we treat light as rays that reflect off at the same angle and in the same plane. This however is not an accurate model. Light's interaction with any medium is a complex phenomena. When light is incident upon a surface it reflects off. This reflected light is not in the form of a ray, but rather diffused. In fact, this diffusion can be modeled as a two dimensional normal distribution, centered around the conventional angle of reflection. How diffuse the light is depends on the surface in question. For highly mirrored surfaces, the standard deviation is pretty low, while for dull surfaces the distribution is well spread out. This effect is important for us because reflected rays going off will not be received by the receiver.

4.1.2 Construction errors

Construction errors can be divided into two types: systematic and random. This is the same distinction that is practiced in physics labs. Dividing them

up in this way allows us to quantify them in different ways.

Systematics errors for us could be distortions of the parabola's profile, mislocations of the receiver from the focus, etc. Sagging is an especially common phenomena which occurs over time. With cheap computational power available, we can compute the affects of these errors using simulations.

Random errors for us will be the small bumps and scratches on the surface of the reflector. Guven [6] in his landmark report treats these errors as being normally distributed. That is every point on the parabola is considered to be rotated from it's ideal position by an angle Δ. These angles are what are distributed randomly. We illustrate this in Fig. **??**.

4.1.3 Sun

There are two features of the Sun that affect the intercept factor. The first of these is the Sun cone, the second the geometric relations between the Sun and the Earth.

The Sun is at an average distance of 150 million kilometers from Earth. Though, we have said previously that at such a large distance, the sun's rays are approximately parallel, this was only a working assumption. The sun subtends an angle of 32 mrad. This means that every point on the earth's surface receives a cone of light from the sun whose angle is 32 mrad. This is shown in Fig. 10.

We would now like to come up with the motion of the Sun in the trough's coordinate system. This means on any day of the year, at any time of the day, we should be able to get a vector for the Sun's position. This is important for tracking purposes, as the vector will tell us how to orient our trough to capture the maximum amount of sunlight. We borrow heavily here from an analysis done by Padilla in [26]. However, our final coordinate system is significantly different from his, and perhaps more suitable.

First we must understand the position of the Sun in relation to the Earth. We need to describe three angles. The first angle captures the motion of the sun across the year. It is called the solar declination angle, δ_s. It is illustrated in Fig. 11.

$$\delta_s = 23.45^o \sin\left(\frac{360(284+n)}{365}\right),$$

where n is the number of the day, with January 1 being $n = 1$. The second angle is the latitude, L. This is the standard geographical latitude angle that

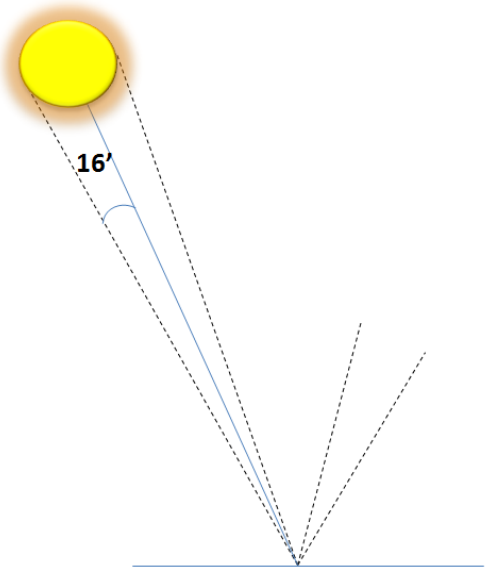


Figure 10: The Sun cone subtends an angle of 16 minutes.

we use to record the positions of places.

The third angle is the the hour angle, h. This describes the motion of the sun across the day. There are three formulas used to define this angle.

$$h = 15^o(t_s - 12),$$

where t_s is the solar time in hours. This can be calculated from the local time by using the following expression

$$t_s = t + EOT + (l_{st} - l_{local})4\text{min/degree},$$

where t is the local time in hours, EOT is the equation of time given below, $l_{st} - l_{local}$ is the difference between the standard time meridian and the local one. The 4 min/degree arises because the sun moves one degree across the sky in 4 minutes. EOT is given by

$$EOT = 0.258\cos x - 7.416\sin x - 3.648\cos 2x - 9.228\sin 2x,$$

where

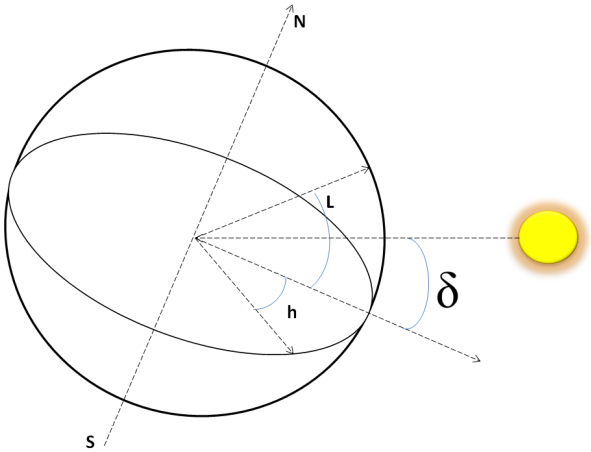


Figure 11: The two angles α and β specify the position of the Sun.

$$x = \frac{360(n-1)}{365.242}.$$

Now when we look at the Sun from a point on the ground, we only need two angles to specify the position of the Sun. These are shown in Fig. **??**.

Using these angles, we can come up with a vector for the Sun's position.

$$V_s = \begin{bmatrix} \cos\beta \sin\alpha \\ \cos\beta \cos\alpha \\ \sin\beta \cos\alpha \end{bmatrix}.$$

Now, how do we relate it to three angles we have before. From the Earth's center, the Sun is given by the following vector

$$V_s' = \begin{bmatrix} \cos\delta_s \sin h \\ \sin\delta_s \\ -cos\delta_s \cos h \end{bmatrix}.$$

This can now be shifted to the location we are on the Earth's surface. This will involve both a rotation and a translation. However, the Earth's radius is negligible compared to the distance between the Earth and the Sun. Let us therefore only consider the rotation. This is given by

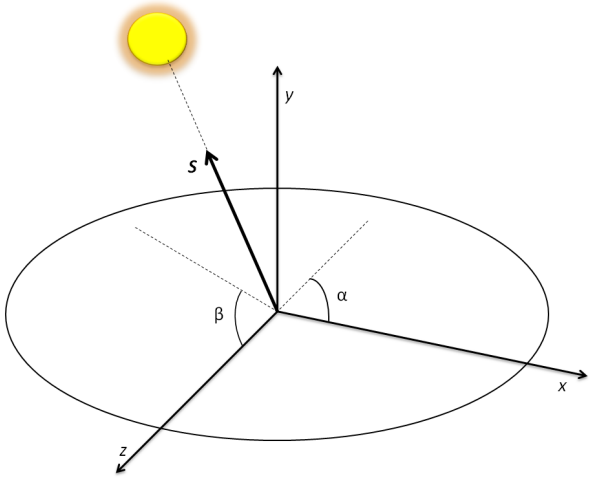


Figure 12: The two angles α and β specify the position of the Sun.

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & \sin L & -\cos L \\ 0 & \cos L & \sin L \end{bmatrix}$$

We can multiply V_s' by this rotation matrix and equate to the V_s above. By dividing the various components of these vectors, we get the following relations

$$\tan\alpha = \frac{\cos\delta_s \sin h}{\sin L \sin\delta_s + \cos L \cos\delta_s \cos h}$$
$$\tan\beta = \frac{\cos L \sin\delta_s - \sin L \cos\delta_s \cos h}{\sin L \sin\delta_s + \cos L \cos\delta_s \cos h}$$

There is only one constraint that we missed out. If the hour angle, h is negative, then should be α. The above equation automatically takes care of the positive case.

$$\alpha = -\left|\alpha\right|, \quad h < 0$$

If we want to know if the Sun is above the horizon, we compute the angle

$$\cos\theta_h = -\tan L \tan\delta_s.$$

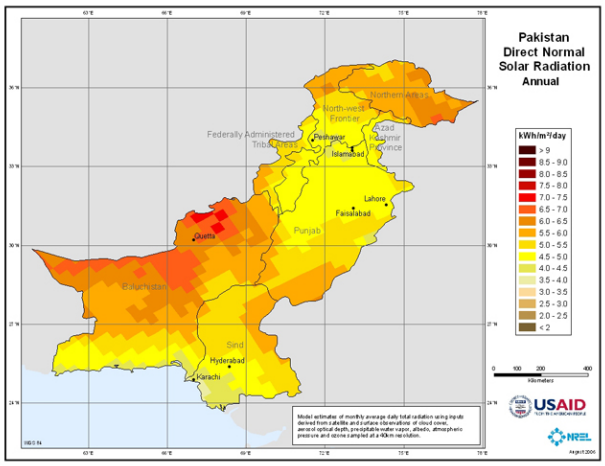


Figure 13: Normal average insolation in Pakistan. As with other natural resources Baluchistan receives the lions share of sunlight as well.

The Sun is above the horizon if $|h| > |\theta_h|$.

Knowing the values of α and β we can compute the position of the Sun in the local coordinate system. We will now need to orient our trough so that it's aperture is facing the Sun.

Done with the positioning of the Sun, we can talk about the solar insolation. This varies from location to location. There are various models, some highly empirical, that fit the actual data pretty good. However, especially in the cities, smog and pollution might affect this data. Moreover, clouds and rain might affect the value of the insolation. For instance, in Pakistan in Punjab, during the monsoon season, the measured insolation values face a sharp dip [22]. Therefore, we recommend that measured values be used instead of those computed from models. The spatial distribution of insolation averaged over the whole year for Pakistan is given in Fig. 13.

4.2 Thermal Model

In this section we will look at the physics that governs the heat transfer. Conductive and radiative heat transfer is simple and well understood however the current theory of convective heat transfer incorporates a wide range of semi empirical parameters and relations. While they are fairly accurate they present us with a fairly accurate description if we were to look at this

problem in a general way we will have to discuss a wide assortment of scenarios diverting us from our problem statement. In our discussion we will look at the base cases and simple additions to it. For example, the metal receiver tube can be either be bare, covered with a simple glass or plastic tube or covered with an evacuated glass tube. Clearly, a pressurized evacuated glass tube is both prohibitively expensive and hard to manufacture in a developing country so we will leave out discussion of this last case and work thoroughly with previous two, as it is those that we wish to optimize, even though the last case is used extensively throughout the world.

4.2.1 Conduction

Experimental observations show us that the rate of conduction doubles when the temperature difference or the area normal to the heat transfer is doubled and halves when the wall thickness is doubled. Using these proportionality we get

$$\dot{Q}_{cond} = -kA\frac{T_2 - T_1}{\Delta x}$$

And in the limiting case (when $\Delta x \to 0$) we get Fourier's law of heat conduction.

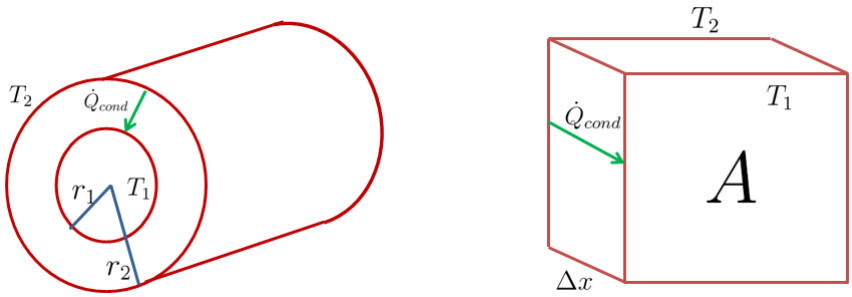
$$\dot{Q}_{cond} = -kA\frac{dT}{dx}$$

We are more interested in how heat transfer takes place radially (Figure 14(a))through hollow cylinders as they compose most of our geometry. We express Fourier's law of heat transfer in terms of r as the cylinder has that symmetry. We now note that heat transfer depended on the area normal to the direction of heat transfer and in this case it is different at for different radii. Therefore we must integrate the area over the radius to get Equation 1 which gives us the radial heat transfer through conduction in this case.

$$\dot{Q}_{cond} = -kA\frac{dT}{dr} = -k(2\pi rL)\frac{dT}{dr}$$

$$\dot{Q}_{cond}\int_{r_1}^{r_2}\frac{1}{r}dr = -k2\pi L\int_{T_1}^{T_2}dT$$

$$\dot{Q}_{cond} = -2\pi kL\frac{T_1 - T_2}{\ln(r_2/r_1)} \tag{1}$$



(a) Hollow cylinder (b) Rectangular block

Figure 14: Heat Transfer via Conduction in different Geometries

4.2.2 Radiation

The energy transfer due to radiations was initially experimentally determined by the work of Joseph Setefan, whose name features in its constant of proportionality.

$$\dot{Q}_{rad} = \sigma_B T^4 \tag{2}$$

However this result was later also obtained by integrating Planck's distribution law over all wavelengths.

What we are again interested in is radiative transfer in cylindrical geometries. For the radiative heat loss from a heated cylinder we have to simply multiply by the area $(2\pi rL)$ and add a term for the emissivity as it will not be a perfect black body.

$$\dot{Q}_{rad} = 2\pi rL\epsilon\sigma_B T^4$$

To see how radiation heat transfer takes place when one cylinder is enclosed in another we would first like to introduce two tools. The first is the analogy between electrical circuits and heat transfer and the second is the view factor.

4.2.2.1 Heat Transfer as Electric circuits In electric circuits we have a current I that is driven from point 1 to point 2 across a certain resistance ($\frac{\rho L}{A}$ for wires) with the help of the difference in electric potential that exists between these two points.

$$I = \frac{V_2 - V_1}{R_{12}}$$

We have a very similar case with heat transfer. Heat is transfer is caused by the presence of temperature difference between two points which causes heat to flow across what we term as thermal resistance, which comes out to be different for different geometries and modes (conduction, convection and radiation). For example if heat is being transferred via conduction across a solid block of material of width a, sides of area A and temperate T_1 and T_2our heat transfer analogy becomes

$$I_{heat} = \frac{T_2 - T_1}{R_{12}}$$

we know from our previous discussion that heat transfer due to conduction is

$$\dot{Q}_{cond} = \frac{kA}{\Delta x}(T_2 - T_1)$$

hence we can say that the conduction resistance in this case is $R_{12} = \frac{\Delta x}{kA}$. What we now want to know is a form for radiation resistance. This example is further illustrated in Figure 7.1.

(a) Electric Circuit (b) Thermal Circuit

Figure 15: Analogy between electric and thermal circuits.

The net radiation q_i by a surface is the difference between the radiosity J_i and irradiation G_i.

$$Q_i = A_i(J_i - G_i) \tag{3}$$

Irradiation is the total amount of incoming radiation incident onto the object. Radiosity is the total radiation intensity leaving a surface and so is the sum of the radiation directly emitted by the surface E_i and the reflected radiation $\rho_i G_i$, a fraction of the incoming that is not absorbed hence $\rho_i = 1 - \alpha_i$ where α_i is the absorbance.

$$J_i = E_i + \rho_i G_i$$

$$J_i = E_i + (1 - \alpha_i)G_i$$

which for an opaque, diffuse, gray surface becomes

$$J_i = \epsilon_i E_i + (1 - \epsilon_i) G_i$$

$$G_i = \frac{J_i - \epsilon_i E_i}{1 - \epsilon_i}$$

substituting this into 3 we get

$$\begin{aligned} Q_i &= A_i (J_i - \frac{J_i - \epsilon_i E_i}{1 - \epsilon_i}) \\ &= \frac{E_i - J_i}{(1 - \epsilon_i)/(\epsilon_i A_i)} \end{aligned}$$

here we make our connection to the electrical analogy. The driving potential is then $E_i - J_i$ and the radiation resistance is $\frac{1-\epsilon_i}{\epsilon_i A_i}$.

4.2.2.2 View Factor If we have a surface i it will be radiating energy in all directions. What we want to know is what fraction of this is intercepted by another surface j. The view factor is then defined as fraction of the radiation leaving surface i that is intercepted by surface j [19]. This is demonstrated in Figure 4.2.2.2 which shows a sphere inside another. The inside sphere has a view factor of 1 with regard to the outside sphere, whereas the outside sphere has a view factor of less than one for the inside sphere. We now proceed to calculate this factor.

Our surface i is radiating in all directions. We want to know what amount of radiation crosses the area of j present at some distance. To do so we see the radiation that reaches surface j from a point on surface i and integrate over both surfaces.

$$dq_{i \to j} = I_i (dA_i \cos \theta_i) d\omega_{j-i}$$

$dA_i \cos theta_i$ shows the project of the area of i normal to the radiation direction and $d\omega_{j-i} = \frac{\cos \theta_j dA_j}{R^2}$ is the solid angle extended by the radiation that intercepts surface j.

$$dq_{i \to j} = I_i \frac{\cos \theta_i \cos \theta_j}{R^2} dA_i dA_j$$

I_i is actually a function of (θ, ϕ) and tells us the directional dependence of the emitted radiation. Iif the surface is a diffuse reflector and emitter it is independent of θandϕ and so we can integrate over all space to get $I_i = \frac{J_i}{\pi}$

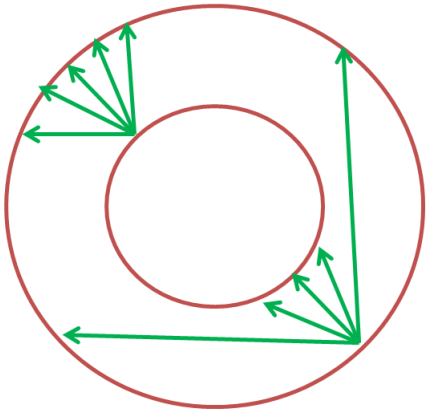


Figure 16: The fraction of light going from surface one to surface 2 is the view factor F_{12}. The arrows show the paths of emitted radiation.

$$q_{i\to j} = J_i \int_{A_i}\int_{A_j} \frac{\cos\theta_i \cos\theta_j}{\pi R^2} dA_i dA_j$$

recalling the definition of the view factor as the fraction of radiation that leaves surface i that is intercepted by surface j we have

$$F_{ij} = \frac{q_{i\to j}}{A_i J_i}$$

Lastly from the above integrals we see we can switch i and j to get the reciprocity relation

$$A_i F_{ij} = A_j F_{ji}$$

Now in an enclosure the irradiation (G_i) is the sum of all the radiation reaching the surface from the other N surfaces in the enclosure.

$$A_i G_i = \sum_{j=1}^{N} F_{ji} A_j J_j$$

We use the reciprocity relation to cancel the area term

$$G_i = \sum_{j=1}^{N} F_{ij} J_j$$

and now we can substitute G_i from Equation 3

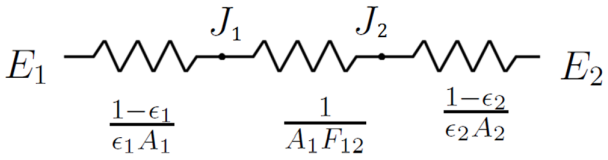


Figure 17: Circuit representation of heat transfer via radiation from inner cylinder to covering cylinder

$$q_i = A_i(J_i - \sum_{j=1}^{N} F_{ij} J_j)$$

For an enclosure with N surfaces $\sum_{j=1}^{N} F_{ij} = 1$ and so we can use this to simplify our result

$$q_i = A_i(\sum_{j=1}^{N} F_{ij} J_i - \sum_{j=1}^{N} F_{ij} J_j)$$

$$q_i = \sum_{j=1}^{N} A_i F_{ij}(J_i - J_j) = \sum_{j=1}^{N} q_{ij}$$

$$q_{ij} = A_i F_{ij}(J_i - J_j)$$

Hence we have now arrived at the circuit analogy for heat transfer through radiation between two surfaces. Here $J_i - J_j$ is the driving potential and $\frac{1}{A_i F_{ij}}$ is the geometrical resistance.

4.2.2.3 Final Answer We can now figure out the radiative heat transfer between two cocentric cylinders. As we only have two surfaces the net rate of emission from surface 1 must equal to the net rate of absorption at surface 2. Our electric circuit is one in series and is shown in Figure **??**. First we have resistance R_1 during emission at surface 1, then a resistance R_{12} for the geometry between the two surfaces and lastly a resistance R_2 for the net radiation absorption at surface 2. Due to the series configuration we simply add the resistances to get the heat transfer which is given in Equation 4. The circuit representation is given in Figure 4.2.2.3.

$$\dot{q}_{12} = \frac{E_1 - E_2}{\frac{1-\epsilon_1}{\epsilon_1 A_1} + \frac{1}{A_1 F_{12}} + \frac{1-\epsilon_2}{\epsilon_2 A_2}} \tag{4}$$

We know the relation of E_1 on temperature by Equation 2 and for our co-centric cylinders we know that $\frac{A_1}{A_2} = \frac{r_1}{r_2}$. Lastly the view factor from the inner cylinder to the outer cylinder is 1. By plugging in these variables and simplifying we get the net radiation going from the inner pipe to the outer cover is

$$\dot{Q}_{rad} = \frac{\sigma 2\pi r_a (T_1^4 - T_2^4)}{1/\epsilon_1 + (1-\epsilon_2)/\epsilon_2 (r_1/r_2)}$$

4.2.3 Convection

And finally we would like to talk about perhaps the most complex heat transfer mechanism, convection. We meet convection in two places in our setup. Firstly heat transfer between the receivers inside wall and the working fluid happens via convection (internal flow). And secondly the pipe and the glass sleeve loose heat to the surroundings via convection.

If a fluid flows over an arbitrary surface and the temperatures of the two surfaces are not the same we can assure ourselves that heat transfer will take place between the two materials. We can write this in a general form

$$\dot{Q}_{conv} = hA(T_1 - T_2)$$

Where h is the convection coefficient which must be determined for different scenarios. We know that at the fluid solid boundary the fluid is stationary due to adhesive forces, hence all heat transfer is through conduction. Hence we equate the heat transfer due to convection to the Fourier's Law for heat transfer at the boundary.

$$\dot{Q}_{conv} = \dot{Q}_{cond}$$

$$hA(T - T_{amb}) = kA\frac{\partial T}{\partial x}|_{x=0}$$

hence h becomes

$$h = \frac{k\frac{\partial T}{\partial x}|_{x=0}}{T - T_{amb}}$$

From this we infer that h is the rate of heat transfer between a solid surface and a fluid per unit surface area per unit temperature difference. However, we would further like to know a way in which we can measure h. We find the ratio of the convective heat transfer $kA\frac{\Delta T}{\Delta x}$ to the conductive heat transfer $hA\Delta T$ and call it the Nusselt number.

$$Nu = \frac{h\Delta x}{k}$$

Various empirical relationships have been found for this dimensionless constant, by first measuring the heat loss by conduction in presence of a stagnant fluid, then measuring the heat loss due to convection by making the fluid flow and then taking the ratio of the two. The convective heat loss coefficient can then be calculated from Nu.

The empirical Nusselt number is expressed in a host of non-dimensional numbers that are used regularly in fluid dynamics. Let us take a minute to first understand the ones we will need to use.

4.2.3.1 Non-dimensional numbers in Fluid Mechanics

Here we will discuss the dimensionless numbers in terms of basic measurable quantities such as density, heat capacity, conductivity etc. It must be kept in mind that these quantities also depend on the temperature of the material.

The **Prandtl Number** is the ratio of the kinematic viscosity to the thermal diffusivity.

$$Pr = \frac{\text{kinematic viscosity}}{\text{thermal diffusivity}} = \frac{\nu}{\alpha} = \frac{c_p \mu}{k}$$

In turn thermal diffusivity α is given by

$$\alpha = \frac{k}{\rho c_p}$$

where k is the thermal conductivity, ρ is the density and c_p is the heat capacity at constant pressure. It tells us how rapidly a substance can conduct heat through itself. The kinematic viscosity is simply the normal viscosity divided by the density

$$\nu = \frac{\mu}{\rho}$$

The **Grashof** number is the ratio of the buoyancy to viscous forces acting on a fluid and comes into play mostly when studying natural convection. The buoyant forces want the liquid to rise whereas the viscous forces may want to prevent this. For pipes it turns out to be

$$Gr = \frac{\text{buoyancy forces}}{\text{viscous forces}} = \frac{g\beta(T_s - T_{bulk})x^3}{\nu^2}$$

where T_s is the temperature of the surface (here the pipe) and T_{bulk} is the bulk temperature of the fluid and D is the characteristic length of the geometry. For pipes it is the diameter of the pipes and the length of vertical flat plates.

The **Rayleigh** number is simply the product of the Grashof and Prandtl numbers. It can be given as

$$Ra = Gr_x Pr$$

which for a pipe in open air becomes

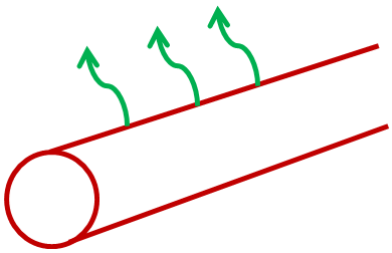
$$Ra = \frac{g\beta(T_s - T_{fluid})D^3}{\nu^2}\frac{c_p\mu}{k} = \frac{g\beta}{\alpha\nu}(T_s - T_{fluid})D^3$$

The **Reynolds** number is the ratio of the intertial forces to the viscous forces. It is given by

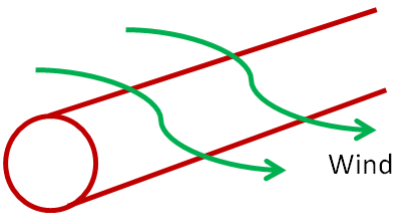
$$Re = \frac{\rho v L}{\mu}$$

where$L$ is the characteristic linear dimension and v is the velocity.

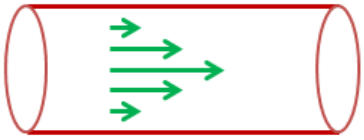
4.2.3.2 Nusselt numbers for various geometries In this section we will list the various Nusselt numbers found empirically by different experimenters. These relations have been taken from [**?**]. We illustrate the various geometries and conditions in Figure .



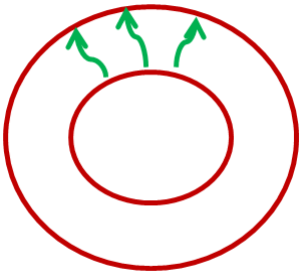
(a) Horizontal cylinder in still air



(b) Horizontal Cylinder in wind



(c) Fluid flow within a cylindrical pipe



(d) Still air in annulus between cocentric cylinders

Figure 18: Different geometries result in different Nusselt Numbers

1. Horizontal Cylinders in still air (Natural convection)

$$Nu = 0.6 + \frac{0.387Ra^{1/6}}{[(1 + (0.559/Pr)^{9/16}]^{8/27}}$$

In this case the parameters (μ,α,etc) within the Ra and Pr numbers are evaluated at the film temperature which is given by

$$T_{film} = \frac{T_s + T_{bulk}}{2}$$

2. Horizontal Cylinder in wind (forced convection)

$$Nu = CRe^m Pr_{amb}^n \left(\frac{Pr_{amb}}{Pr_{surface}}\right)^{1/4}$$

3. Fluid flow within a cylindrical pipe (forced convection) For laminar Flow the Nusselt number is constant at 4.36.

For turbulent flow

$$Nu = \frac{f/8(Re - 1000)Pr_{fluid}}{1 + 12.7\sqrt{\frac{f}{8}}(Pr_{fluid}^{2/3} - 1)} \left(\frac{Pr_{fluid}}{Pr_{pipe}}\right)^{0.11}$$

where the subscripts show the temperature at which the numbers must be calculated at.

4. Annulus between two cocentric cylinders filled with air (natural convection) In this case we do not have a relation for the Nusselt number but a direct relation for the heat transfer from the inner surface a to the outer surface b and the Prandtl number is calculated at the average of the two surfaces.

$$\dot{q}_{ba} = \frac{2.425k_{43}(T_a - T_b)}{(1 + (\frac{D_a}{D_b})^{\frac{3}{5}})^{\frac{5}{4}}}(\frac{Pr\ Ra_{D3}}{0.861 + Pr})^{\frac{1}{4}}$$

4.3 Turbine Cycle

The turbine cycle of the steam type is the standard power generation cycle, consisting of a heat source, turbine, condensor and a pump. This is illustrated in Fig. **??**.

We discuss here the theory of heat exchangers. The theory of the other components and the whole cycle is much more complicated is outside the scope of this project.

4.3.1 Heat Exchangers

Heat exchangers are devices used to transfer heat from one fluid to another. They are of a wide variety of types and sizes, as per the specific needs for which they are needed. They are often classified by the flow arrangement.

Parallel flow The two fluids enter the heat exchanger from one end, flow together and leave from the other end.

Counter flow The two fluids enter the heat exchanger from opposite ends and flow across each other.

Cross flow The two fluids flow perpendicular to each other.

Multi pass The two fluids pass against each other multiple times.

Heat exchangers are designed according to the fluids states, which may be liquid or gas. So they might be liquid-liquid, liquid-gas or gas-gas. Other heat exchangers are meant for boiling off a liquid to a gas or vice versa. Despite this wide variety we present two methods of heat exchanger analysis. These are not what one uses when designing a heat exchanger, but they can serve as useful ballpark measures.

The central parameter in heat exchangers is the overall heat transfer coefficient, U. It's units are W/m^2K. It measures the heat transfer per unit area and per unit difference in temperature between the two fluids. We will use this parameter in the following models, which have been borrowed from [19] and [9].

4.3.1.1 Log Mean Temperature Difference We make the following assumptions.

1. The heat exchanger is insulated from the surroundings.
2. There is no axial conduction.
3. There are not potential or kinetic energy changes.
4. The fluid specific heats are constant.
5. The overall

This allows us to write the heat balance equations between the two fluids,

$$q = \dot{m}_h c_h (T_{h,i} - T_{h,o})$$
$$q = \dot{m}_c c_c (T_{c,o} - T_{c,i}).$$

Here, $\dot{m}$ is the flow rate, c the heat capacity and T the temperature. The subscripts h and c refer to the hot and cold liquids respectively, while i and o refer to the temperature at the input and output of the heat exchanger. These equations will hold whenever we have heat transfer between two liquids. For the heat exchanger we can have a relation from the overall heat transfer coefficient.

$$q = UA\Delta T_m$$

.

Here A is the area of contact between the two fluids and ΔT_m is the mean temperature between the two fluids. This comes out to be

$$\Delta T_m = \frac{\Delta T_2 - \Delta T_1}{\ln(\Delta T_2 / \Delta T_1)}$$

for the parallel and counter flow heat exchangers. For other types of exchangers, there is a factor of F with this mean temperature difference.

We have three equations, with the unknowns q and the two output temperatures. We can use numerical techniques to compute the solutions to these equations.

4.3.1.2 The Effectiveness-NTU Method To avoid the iterative nature of the previous method we introduce another one. This is much simpler and involves two parameters, the effectiveness ϵ and the NTU.

We first determine the maximum possible heat transfer between the two fluids. Irrespective of the values of the heat capacities, flow rates, and input and output temperatures this comes out to be

$$q_{max} = \dot{m} c_{min} (T_{h,i} - T_{c,i}).$$

Here c_{min} is the minimum of c_h and c_c. The other notation is as before. We now say that the heat exchanger possesses a certain effectiveness compared to the maximum i.e. the actual heat transfer in the heat exchanger is

$$q = \epsilon q_{max}.$$

The effectiveness depends on NTU and the heat capacities and flow rates.

$$NTU = \frac{UA}{C_{min}}.$$

$$C_r = \frac{C_{min}}{C_{max}}.$$

It can be determined for various heat exchanger. For instance for the parallel flow and counter flow it is given by

$$\begin{aligned}
\epsilon_{\text{parallel flow}} &= \frac{1 - \exp(-NTU(1 + C_r))}{1 + C_r} \\
\epsilon_{\text{counter flow}} &= \frac{1 - \exp(-NTU(1 - C_r))}{1 - C_r \exp(-NTU(1 - C_r))}.
\end{aligned}$$

This method is fairly simpler. We compute the various quantities and it gives us the heat transfer. Then we can couple them with the heat transfer equations, from above to get the temperature output.

5 Models and Simulation

A model of the solar collector was developed, partly based on the theory developed previously. This in itself consisted of two submodels, one of the reflector and one of the receiver. The reflector model outputs, among other things, the flux landing on the receiver. The receiver model can then use this value to compute how the temperature flowing in the receiver will change. A simulation was written in the popular computational modeling software MATLAB [1]. This allowed us to make predictions of the working of a parabolic trough collector. Though, there are some constraints on and limits on what this computational model can predict, it is perhaps one of the more mature and well developed ones available freely. We plan to further develop this model and open source the code, so that others working on solar collectors can benefit. What code has already been written can be found in the appendix, and a copy can be gotten by emailing the authors.

A very preliminary model of the rest of the system was also developed, but heat exchangers, turbines and pumps have a whole field developed around them and studying the complications within them were outside the scope of this project.

The code developed is extremely modular. Some one might choose to use only the reflector simulator, without any reference to the receiver model and vice versa. Looking into the future, we see that the community would enormously benefit if alternate models of these components and models of the components we were unable to construct are written and coded, each as a separate modules. Then developers and industry may pick and choose which modules to join together, as per their own requirements.

5.1 Input Parameters

We will be discuss the model and implementation by first going over the various input parameters that the model takes in. Some of these are shared by different submodels, so this is the right place to discuss them.

Extensive use of MATLAB data class 'structs' was used to improve modularity and structure of the code. These structs allow us to define objects, such as the 'receiver' which then holds all the properties of the receiver used in our system.

To increase the modularity of the code, the properties of a variety of materials was coded up as these structs. Then the material of the various components

could be specified in one go. This allows the user to quickly swap materials and see the results.

Similarly, the simulation allows one to store meteorological data for different locations. Though, currently it considers monthly averages, the change to daily values is trivial. The simulation currently considers off noon solar intensity values in a rough way. This can also be improved to input the exact value at different times to get better estimates of the performance of the collector.

Maximum effort has been expended to correspond to the SI system of units. However, because a huge chunk of previous work on solar collectors has been done in the US and the the units used by the engineering sector in the US are not SI, we have been forced to make some reference to their insanely arcane and non decimal using units.

Below you will find a series of tables that will describe the various parameters. * parameters have not been properly implemented.

5.1.1 Simulation

There are some important simulation parameters that we must declare. They are shown in Table. 2.

Property	Description
Size*	The 2D of the size of the simulation. Only objects inside this area are considered.
Grain Length	The length of each small section. Making this small increases the accuracy of the simulation.

Table 2: Parameters used to define the simulation.

5.1.2 Reflector

The corresponding struct in our code is called 'trough' since the code is specialized for a parabolic trough. They are shown in Table. 3.

5.1.3 Receiver

The receiver consists of an inner metallic absorber and an outer glass sleeve. Consequently, there are four radii available. They are shown in Table. 4.

Property	Description
Focal length	The focal length of the trough.
Width	The full width of the trough.
Length	The length of the trough.
Refractive index	The refractive index of the reflective sheet.
Surface standard deviation	The standard deviation of the distribution of random bumps and scratches on the surface of the trough in radians.
Specularity	The specularity of the surface. This means what percentage of the reflected light intensity is contained at the conventional reflection angle. The rest of the intensity goes off in the diffuse component of the reflection.
Specularity standard deviation	The standard deviation of the angular spread of the diffuse component of the reflected light. Measured in radians.
Half quantization	The diffuse component of light is divided into a number of rays at some random angle. This parameter determines how many such rays are there. The actual number of rays are twice this number.
Orientation angle*	The orientation angle of the trough with respect to the North, with the positive direction of rotation towards East. Not implemented at all. The trough is assumed to be oriented North to South.
Tracking error	The tracking error of the trough.

Table 3: Parameters used to define the reflector.

5.1.4 Collector Cycle

The collector cycle describes the liquid flow cycle within which the collector is defined. They are shown in Table. 5.

5.1.5 Sun

The characteristics of the sun. There is an inherent assumption that the solar intensity is uniform across the sun disk. However, improvements can be made to consider the actual non-uniform solar intensity. They are shown in Table. 6.

Property	Description
Absorber inner radius	The inner radius of the absorber tube.
Absorber outer radius	The outer radius of the absorber tube.
Glass sleeve inner radius	The inner radius of the glass sleeve.
Glass sleeve outer radius	The outer radius of the glass sleeve.
Absorber material	The material of the absorber.
Emissivity	The emissivity of the receiver. This is redundant since the material specification should cover this property.
Gas	The gas between the absorber and glass sleeve eg. air.
Extra length	If the receiver is longer than the trough, the extra length on each side.

Table 4: Parameters used to define the receiver.

Property	Description
Fluid	The heat transfer fluid.
Flow rate	The flow rate of the heat transfer fluid.

Table 5: Parameters used to define the collector cycle.

Property	Description
Half quantization	The number of rays in each sun cone is twice this number plus one

Table 6: Parameters used to define the Sun.

5.1.6 Location

The location parameters allows us to evaluate the model for different places, times of the year and time of the day. They are shown in Table. 7.

5.1.7 Atmosphere

The properties of the atmosphere. They are shown in Table. 8.

Property	Description
Location	The location, where the collector is placed eg. Lahore. The coordinates and meteorological data must be available.
Date	The date of the year for which the model should be evaluated, in the form of [month day of month].
Time	The time of the day for which the model should be evaluated, in the form of [24h hour minutes].

Table 7: Parameters used to define the Location.

Property	Description
Refractive index	The refractive index of the atmosphere.
Gravity	The value of gravity.

Table 8: Parameters used to define the atmosphere.

5.2 Reflector Model

The reflector model is essentially a ray trace model. To make this work we need to describe the model of the sun and the geometry and material properties of both the trough and the receiver. Some of the properties, such as diffuse reflection, are statistical in nature, and appropriate use of random number generation is done to implement these.

Below we describe what a ray trace model is, and then describe the reflector model as a process. The output of the model is the distribution of light on the receiver.

5.2.1 Ray Trace Model

Ray trace models are useful way to simulate the interaction of waves or particles with a system. Models in this class are not only used for computational physics. They are, for instance, used by graphics artists to make visually realistic illumination and shadowing of scenes.

The basic idea behind a ray trace model [30] is that the waves or particles can be modeled as a set of rays. Furthermore, locally i.e. over short distances, the rays can be made to move in a straight line. Local derivatives can then be used to modify the direction of each ray for the next segment of it's journey. If they are obstacles, such as a reflecting material, the path of the

ray may be changed accordingly on collision. Along the path of the ray properties such as intensity may be changed as per it's interactions with the system. Typically, a large number of rays are sent through the system to get statistical averages. However, Monte-Carlo type ray trace models have been used in solar concentrator applications [23].

Ray trace models are essentially processes. In the simplest type of ray trace model, a large number of light rays are generated for every light source, the statistics corresponding to those of the actual source. These are sent through the system, until they reach some final destination. So for instance, if are modeling how a light from a torch is focused by a converging lens, we might start rays at the torch, let them pass through the lens (or miss the lens), and then follow them for a certain distance. If they have converged the torch is at the focus.

However, since generating a large number of rays is computationally expensive, a more efficient technique is to pick the most important subsystem [21]. Then generate incident rays for that subsystem, and track them both backwards and forward through the system. For instance, in our lens example, we should pick light rays incident on the lens, then trace them back to the torch and forward to the other side of the lens. Such a technique will be used in our model.

5.2.2 MATLAB Optimizations

MATLAB implements very fast matrix multiplications because of the usefulness of this in scientific computations. In comparison, implementing the same procedures using loops in MATLAB turns out to be much slower. In additions, logic operations can also be implemented in MATLAB using logical matrices instead of looped if-else commands. The matrix version again turns out to be much faster.

For these reasons, matrices multiplications has been the preferred methods of implementing procedures compared to loops. Only for those procedures which cannot be implemented using matrices has recourse been made to loops. Till the writing of this report, only one such case has arisen.

5.2.3 Implemented Model

The parameters that have been specified above are inputed by the user. Let's see how we can use those inputs to compute the intensity distribution on the receiver.

The simulation is a process: we start with the inputs, compute one quantity after another and in the end we have the distribution of sunlight on the receiver. Below we describe this process, each heading signifying a new important step in this process. We state again that recourse to the code may be made in the appendix to better understand these sections.

The only important point we must state here is that most of the computation is done in 2D. At the end, an approximation is made, which allows us to infer a fairly accurate 3D result from the 2D result.

5.2.3.1 The Sun We have discussed previously in section 4.1.3 how the position of the sun is computed for locations, dates and times, in terms of the coordinate system suitable for the trough. These can be straightforwardly implemented.

The tracking error of the trough is available. This can be coupled with the position of the sun to compute the rotation of the trough.

5.2.3.2 Computing Trough Geometries We are aware of the length, width, focal length, and the orientation of the trough. We use these to compute the position of each point on the trough. The simulation grain length is the important parameter here since the simulation is only accurate down to this length. Essentially, for different x positions we can compute the corresponding y positions. This may involve

1. Using the equation of the parabola $y = \frac{x^2}{4f}$ to compute the relations between y and x.

2. Moving the parabola around so it's in the right position. For instance it makes more sense physically to have the focus, rather than the vertex, at the origin.

3. Rotate the coordinates to account for the orientation of the trough. This involves simply multiplying existing coordinates by the rotation matrix.
$$\begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix}$$
One must however be careful in the ordering of these steps since rotation and translation don't commute. This matrix is for rotation around the origin.

We implement rotation after moving the focus to the origin since the tracking of the physical trough involves it rotating around the focus.

5.2.3.3 Surface Imperfections However, this is the only average positions of various points of the trough. We are also interested in the local gradient at each of these points. This isn't simply the simple gradient of a curve. The points we have stored only represent the mean position within the accuracy of each grain length. These mean positions do not give information on how the other points within each grain length are distributed.

The above means that we can implement a useful physical characteristic of the trough. There are bound to be surface imperfections on any real trough, in the form of bumps and scratches. We do not mean here systematic distortions in the macroscopic shape of the trough, such as bends or sagging. We mean surface errors that are randomly distributed.[1] These small distortions on the surface can be considered, to an approximation, as small rotations of the surface of the trough. Fig. 19 makes this clear. Each of these rotations is a deviation from the local gradient of the perfect trough. Now by the mean value theorem of statistics, these random distortions, or rotations, may be approximated by a normal distribution. This should have a mean of 0 and some standard deviation. This standard deviation has been inputed by the user in the parameters. We are not yet aware of how this standard deviation can be measured or calculated by the user. However, we do have some indication of the range of values these standard deviations can take [6]. These values were used while we ran our simulation. We can implement it this way.

1. Compute the local gradients using a numerical approximation. MATLAB implements this in it's native library.
2. Compute the gradient of the normal of each point which is just $m_n - 1/m$.
3. Turn these gradients into angles using $\tan^{-1} 1/m_n$.
4. Now we add random perturbations. Using a random number function that outputs a normal distribution, we can add angles with the right standard deviation.
5. Compute the new normal gradients, by $1/\tan\theta$.
6. Compute the new local gradients.

These final local gradients will later be used when we start tracing rays and reflections off the trough are calculated.

[1] These statements are not trivial. It's not simple to claim which construction imperfection is random and which is systematic. A statistician would need to crunch some numbers to answer this question - with some controversy.

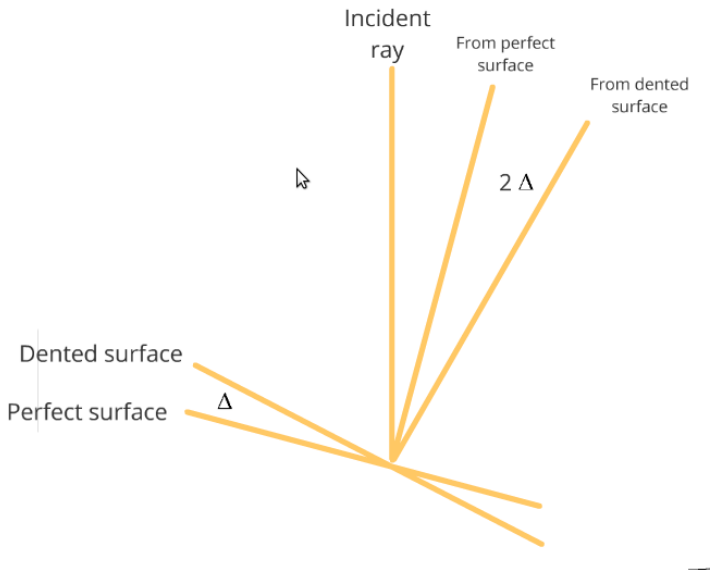


Figure 19: The effect of surface errors on reflection.

5.2.3.4 Receiver Geometry The receiver is much simpler to treat than the trough. In our simulation, we considered it to be a circular pipe, which it typically is physically is. The equation of the circle $x^2 + y^2 = r^2$ can be used to easily compute the coordinates of the receiver. Since, the receiver is the end point of our ray trace model, surface imperfections need not be calculated - reflections off this surface will not be considered.

5.2.3.5 Sun Rays on the reflector With this step we begin our ray tracing procedure. The reflector is the most important optical subsystem we have. So, we generate rays in the following way. For each point of the trough, we figure out where the rays from the sun must come from. Then we reflect those rays.

As we have already discussed in section 4.1.3 the sun subtends an small cone on every point on earth. The cone can be described by two parameters. The angle that it's axis makes with the vertical, and the half angle of the cone. We have both. The vertical angle can be calculated when we computed the position of the sun in the trough's coordinates. The half angle was inputed by the user, and is a constant for Earth.

We begin by computing the angle that the normal at each point of the trough makes with the the sun cone. This can be done by the following formula.

$$\theta_i = \cos^{-1}\left(\left|\frac{s_y - s_x m_i}{1 + m_i^2}\right|\right)$$

Here, i represents the indexing of the points on the trough. m_i is the gradient at each point of the trough. The absolute value ensures that angles stay within the range -90 to 90.

The θ_i are just the angles corresponding to the center of the sun cone. Depending on the half quantization for the sun in the trough properties, we come up with new rays for each point that are just rotations of the cone's central ray. These are symmetric around the cone, and evenly distributed across the angular width of the cone.

Similar to the above process, we also compute the incidence vectors at each point. Since we have the vector for the sun, we can form a cone of vectors. This is done simply by getting all the quantized angles in the sun cone, making rotation matrices and rotating the central vector with each. This can be applied to every point to get all the vectors in all the sun cones.

We also need to take care of the intensities. Our model says that the intensity in each ray in the sun cone is equal. Using this, we can simply divide the total intensity by the number of rays and get the intensity in each ray.

5.2.3.6 Reflections off the trough We need the transmission angles for each ray. Since we have already computed the incidence angles for each possible ray, this is a simple application of Snell's law.

The reflected rays and intensities can be computed in the following way. We have the incidence vectors. We compute the vectors for the normals using the following standard formula.

$$\mathbf{n} = \frac{1}{\sqrt{1 + m_i}} \begin{bmatrix} -m_i \\ 1 \end{bmatrix}.$$

Then we can just compute the reflected vectors by the difference between the incident and twice the normal vectors, taking care that magnitudes of each vector are correct.

Specularity can be implemented by simply expanding out the incident vectors as many times as specified in the constants. We generate a normal distribution with the right characteristics. Each of the reflected vectors gets spread out as per the angle outputted by the random distribution.

The intensity calculations are slightly more complicated. We first compute the intensities for the reflected rays using the Fresnel's coefficients. The central ray in the specular cone gets a certain amount of intensity as specified by the material properties. The rest of the intensity is divided among the other rays for each point.

5.2.3.7 Intersections with the receiver Till now we have the reflected rays, accounting for the sun cone, surface errors and the specularity of the material. We also have the intensity that each of these rays carries.

The receiver is a circle, the rays straight line. Therefore, we can use geometry to compute which rays intersect with the receiver and if they do, at which point. Reflection gradients are simply given by

$$m_i^r = \frac{r_y}{r_x}.$$

We compute the line and circle intersections. This will result in two points of intersection. We can compute the distance between the trough point and each of these intersections. The nearer point is the right intersection.

These intersection do not correspond to the quantization we have done of the receiver. So we iterate over each intersection point, and using a nearest neighbor approach we figure out which receiver point corresponds to these intersection points.

The nearest neighbor approach simply computes the distance between the intersection point and every point on the receiver. The shortest distance wins.

We now need to just account for intensities. For every point on the receiver, we sum the intensities due to every reflected ray that hits it.

5.2.3.8 Going 3D We have the distribution of the flux on the 2D receiver. What happens when we have a 3D system. The difference comes because of the angle of the sun. The arc that the sun makes with the trough will typically not be perpendicular to it's central axis. This means that some of the rays that were going to hit the receiver, won't because they will go beyond the length of the trough. One simple solution to the above is to have the receiver extend beyond the edge of the trough.

We have the parameters for this in the input constant. Taking this into account, we first figure out what length of the trough reflects rays onto the

receiver. We know from the 2D model how much intensity each strip of the trough contributes to the receiver. We multiply by the number of strips and we have our total 3D intensity.

We can also compute the total intensity falling on the trough by multiplying the collector area by the sun's intensity, weighed by the tracking error.

5.2.3.9 The intercept factor The intercept factor can then be computed to be the ratio of the above two quantities.

$$\text{Intercept factor}, \gamma = \frac{\text{Power on receiver}}{\text{Power on trough}}.$$

5.3 Thermal Model

5.3.1 Basic Heat Transfer Model

The output of the optical model is the average heat flux hitting the receiver pipe per unit length. We can use a simple heat transfer model to extract an order of magnitude estimate of the thermal efficiency of the collector.

We take a small length (ds) of fluid that is just entering into the collector area from the side. The energy it will gain will be the heat flux given by the concentrator minus the losses to the environment. As losses depend on the temperature of the fluid we calculate the increase in temperature of the fluid in a small time dt and then recalculate the losses based on the new temperature.

$$\dot{q}_{tr}ds + \dot{q}_{loss,vec} + \dot{q}_{loss,rad} = mc\Delta T$$

Inserting the radiative and convective losses from the discussion in theory.

$$\Delta T = \frac{\dot{q}_{tr}ds - 2\pi Rdsh(T - T_{amb}) - 2\pi Rds\sigma(T^4 - T_{sky}^4}{\rho\pi R^2 dsc}\Delta t$$

This expression is numerically iterated to find the new temperature. We iterate till the time that the fluid stays in the length of the concentrator. If the fluid is moving at speed v and collector length is L then this comes out to be $\frac{L}{vdt}$ steps.

5.3.2 Detailed Heat Balance Model

The previous model has the fault that it does not take into account that heat transfer to the liquid in the pipe is via convection. As we know that the intercept factor depends on the radius of the collector pipe and so we would like to increase our radius. However, the previous model would not show a reduction in heat transfer. To write a more accurate model we will now consider every single mode of heat transfer and extend this model to include one where we include a glass cover to prevent heat loss.

First let us define some variables to aid our discussion. First let us give each surface a number with which we can identify it. For example we will denote the outer surface of metal pipe as surface 2, its temperature will be T_2, its radius R_2 and any related constant will also have 2 in its subscript. We give the surface labels in Table 9. We also summarize the possible modes of heat transfer between these surfaces using the discussion in Section and using the subscripts we have just defined in Table 10.

Table 9: Surfaces under Consideration

Denoted by	Surface
1	Working Fluid
2	Receiver inner surface
3	Receiver outer surface
4	Glass cover inner surface
5	Glass cover outer surface
6	Surroundings
7	Sun (via collector)
8	Sky

Case 1.
First we study the case of when a simple metal pipe (e.g. copper pipe painted black) is used as the receiver. In this case radiation coming in from the trough is absorbed by the pipe outer surface based on its emissivity. Some of this is lost from this surface to the environment by radiation and convection. The rest of the heat travels to the inside surface of the pipe by conduction. It then transfers to the working fluid by convection. These processes are displayed in Figure 20.We then write the heat balance equations for each surface, i.e. equate all the heat coming into the surface with all the heat which is going out so that we are consistent with the conservation of energy.

On surface 2

$$\dot{q}_{23,con} - \dot{q}_{12,vec} = 0$$

Table 10: Modes of heat Transfer

Between Surface	Symbol	Formula
Convection from 2 to 1	$\dot{q}_{12,vec}$	$h_1 2\pi R_2(T_2 - T_1)$
Conduction from 3 to 2	$\dot{q}_{23,con}$	$2\pi k_{23}\frac{T_3-T_2}{\ln(R_3/R_2)}$
Specific for Case 1		
Radiation from 7 to 3	$\dot{q}_{37,rad}$	$\dot{q}_{tr}\alpha_{rec}$
Radiation from 3 to 8	$\dot{q}_{83,rad}$	$2\pi R_3\sigma\epsilon_5(T_3^4 - T_{sky}^4)$
Convection from 3 to 6	$\dot{q}_{63,vec}$	$2\pi R_3 h_{65}(T_3 - T_6)$
Specific for Case 2		
Radiation from 7 to 5	$\dot{q}_{57,rad}$	$\dot{q}_{tr}\alpha_{env}$
Radiation from 3 to 4	$\dot{q}_{43,rad}$	$\frac{2\pi R_3\sigma(T_3^4-T_4^4)}{1/\epsilon_3+(1-\epsilon_4)/\epsilon_4(D_3/D_4)}$
Conduction from 4 to 5	$\dot{q}_{54,con}$	$2\pi k_{54}\frac{T_4-T_5}{\ln(R_4/R_5)}$
Radiation from 5 to 8	$\dot{q}_{85,rad}$	$2\pi R_5\sigma\epsilon_5(T_5^4 - T_{sky}^4$
Convection from 5 to 6	$\dot{q}_{65,vec}$	$2\pi R_2 h_{65}(T_5 - T_{amb})$

$$2\pi k_{23}\frac{T_3 - T_2}{\ln(R_3/R_2)} - h_1 2\pi R_2(T_2 - T_1) = 0$$

On surface 3

$$\dot{q}_{37,rad} - \dot{q}_{23,con} - \dot{q}_{83,rad} - \dot{q}_{63,vec} = 0$$

$$\dot{q}_{tr}\alpha_{rec} - 2\pi k_{23}\frac{T_3 - T_2}{\ln(R_3/R_2)} - 2\pi R_3\sigma\epsilon_5(T_3^4 - T_{sky}^4) - 2\pi R_3 h_{65}(T_3 - T_6)$$

In a pipe where cold fluid is entering at one end and exiting as heated fluid from the other, we have a temperature gradient. However we assume that in each cross section in the steady stay the surfaces achieve a stable temperature. Again as in the basic model before we focus our attention on a small length of the liquid and solve these equations for that small length over which we assume uniform temperature at each surface.

When a fluid in a pipe is exposed to a constant flux it has a linear rise in temperature within it [19]. Initially we know the input temperature of the fluid is close to the ambient temperature. If T_{out} is the temperature at the output of the first small segment we can approximately say that $T_1 = \frac{T_{out}+T_{amb}}{2}$.

We know that the increase in the temperature of the fluid is given by,

$$\dot{q}_{12,vec} - mc(T_{out} - T_{amb}) = 0$$

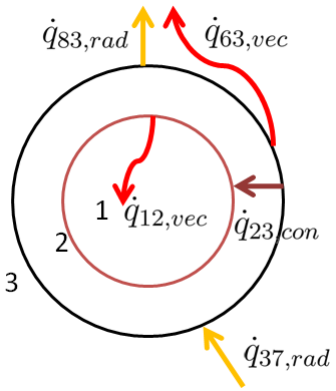


Figure 20: Modes of heat transfer in a simple pipe filled with heat transfer fluid.

and we also substitute T_1 in the first equation. Now we have 5 equations and have to solve for 3 unknowns namely T_2, T_3, and T_{out} for each of the small segments. We can easily iteratively solve for these temperatures using FSOLVE, a built in function in MATLAB, that uses quasi-Newton method to solve such systems of non-linear equations.

One must remember that the temperatures it outputs are the steady state temperatures even though it is iterated for each time step. That means if water at ambient temperature is supplied at one end and exits the other, the simulation will give the steady state result of this setup. If we want to make a closed loop and make the fluid re enter the trough for further heating it will give the steady state result of such a system and miss out on the transient behavior.

Case 2.

Many parabolic trough collectors use evacuated tubes. In our application we would atleast want to incorporate a thin clear glass or plastic insulation cover around the pipe to be able to reach much higher temperatures. We can easily expand this method to model that case. The schematic for this is shown in Figure 21. We will simply again write the heat balance equations on each surface:

On surface 2

$$\dot{q}_{23,con} - \dot{q}_{12,vec} = 0$$

$$2\pi k_{23} \frac{T_3 - T_2}{\ln(R_3/R_2)} - h_1 2\pi R_2 (T_2 - T_1) = 0$$

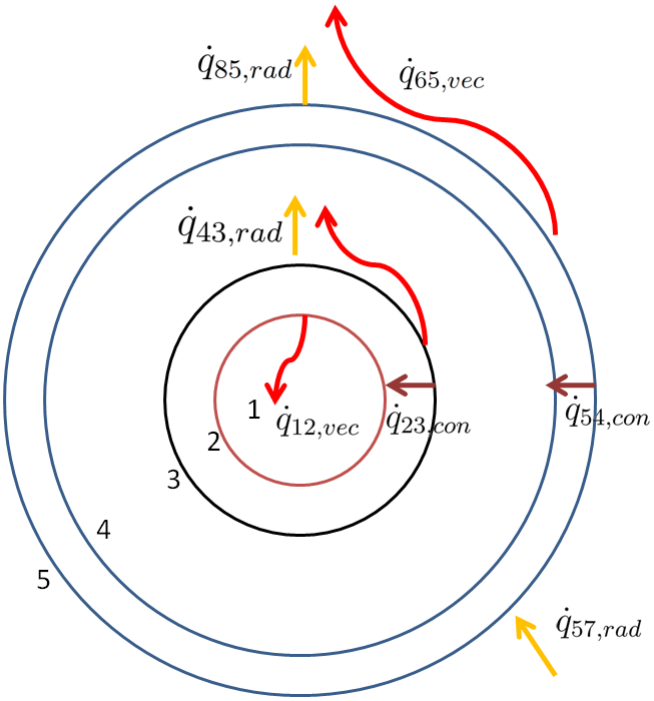


Figure 21: Modes of heat transfer in a simple pipe filled with heat transfer fluid surrounded by a cylindrical plastic of glass sleeve to prevent heat loss.

On surface 3

$$\dot{q}_{37,rad} - \dot{q}_{23,con} - \dot{q}_{43,rad} - \dot{q}_{43,vec} = 0$$

$$\dot{q}_{tr}\alpha_{rec}\tau_{env} - 2\pi k_{23}\frac{T_3 - T_2}{\ln(R_3/R_2)} - \frac{2\pi R_3\sigma(T_3^4 - T_4^4)}{1/\epsilon_3 + (1-\epsilon_4)/\epsilon_4(D_3/D_4)} - \frac{2.425k_{43}(T_3 - T_4)}{(1 + (\frac{R_3}{R_4})^{\frac{3}{5}})^{\frac{5}{4}}}(\frac{Pr\ Ra_{R3}}{0.861 + Pr_{43}})^{\frac{1}{4}} = 0$$

On surface 4

$$\dot{q}_{43,rad} + \dot{q}_{43,vec} - \dot{q}_{54,con} = 0$$

$$\frac{2\pi R_3\sigma(T_3^4 - T_4^4)}{1/\epsilon_3 + (1-\epsilon_4)/\epsilon_4(D_3/D_4)} + \frac{2.425k_{43}(T_3 - T_4)}{(1 + (\frac{R_3}{R_4})^{\frac{3}{5}})^{\frac{5}{4}}}(\frac{Pr\ Ra_{R3}}{0.861 + Pr_{43}})^{\frac{1}{4}} - 2\pi k_{54}\frac{T_4 - T_5}{\ln(R_4/R_5)} = 0$$

On surface 5

$$\dot{q}_{57,rad} + \dot{q}_{54,con} - \dot{q}_{85,rad} - \dot{q}_{65,vec} = 0$$

$$\dot{q}_{tr}\alpha_{env} + 2\pi k_{54}\frac{T_4 - T_5}{\ln(R_4/R_5)} - 2\pi R_5\sigma\epsilon_5(T_5^4 - T_{sky}^4) - 2\pi R_5 h_{65}(T_5 - T_{amb}) = 0$$

We then write the model for three dimensions and replace T_1 and calculated the increase in fluid temperature by Equation X. This time we have 5 equations and have to solve for 5 unknowns namely T_2, T_3, T_4, T_5 and $T_o ut$ which we can again solve using FSOLVE.

5.3.3 Calculating the Parameters of Heat Transfer

As we discussed in the theory the Nusselt, Raleigh, Pratt numbers, coefficient of conduction etc all also depend on temperature, some explicitly and some implicitly. In this section we will discuss their temperature dependence and at which temperature we need to calculate these for various geometries, or see if we need to extract their values from tables.

Most thermal science textbooks have extensive tables in their appendix of the temperature dependence of material properties of most common materials [19]. We input relevant parameters for iron, copper, glass, Pyrex, air, water and engine oil for the expected temperature ranges and then our code would linearly interpolate these values to find the value for the current required temperature.

5.4 Stitching Models Together

We have two models, one for the reflector and one for the receiver. The output of the reflector model is the intensity distribution on the receiver. We sum the intensity distribution to compute the total intensity on the receiver. This total intensity goes into the receiver model. The receiver model can compute the temperature rise of liquid flowing inside it, as it makes one pass through the receiver. The important factor is the temperature of the incoming liquid.

This single pass temperature rise is interesting in itself. For what input temperature it goes down to zero is the maximum possible temperature that our collector can take a liquid with a given flow rate if the liquid is cycled again and again through the collector. This maximum will not be achieved in practice because in a complete system, some heat would be extracted from the liquid by the power generation cycle, through the heat exchanger.

Once we have a model for the power generation cycle, the minimum requirement from it that it tell us the heat extracted from the liquid every cycle, we can join these models together and do more interesting things.

6 Simulation Optimization

Using the model developed in the previous section, we can analyze the behavior of the collector. Our goal here is to come up with the construction parameters of the collector that provide the best performance.

Before we do so, let us look back and see what we have achieved. There is a group of parameters that affect the optical efficiency, while a partially overlapping set of parameters affect the thermal efficiency. These parameters are often at odds with each other, and we will have to weigh them off against each other to find the global optimization.

6.1 Optical Model

Let us begin by looking at the output of the optical model. Our dependent variable will always be the intercept factor, which measures what percentage of the incident light reaches the receiver. We will plot a variety of factors and see how the intercept factor varies with them.

Let's start with the focal length and radius, in Fig. 22. For large radii, there is a broad peak in the range of $0.2 - 0.3$ m. Morever, intercept factor falls sharply with decreasing radius.

We can now add tracking errors into the picture. In Fig. 23, 24 and 25. we see how important having good tracking is. The intercept factor falls down pretty sharply as tracking errors go up. For a 5^o error, it's only for the the large radii, > 0.05 m, that there is an appreciable tracking error. Also, and perhaps more importantly, large focal lengths are more susceptible to tracking errors for all radii.

Let us look at this more explicitly on the plots in Fig. 26. The intercept factor goes up with increasing radius. We again see that the intercept factor is highly susceptible to tracking errors. To guard against this susceptibility the optical model says that we should have a large radius.

Now let us look at another interesting parameter, mislocation of the receiver from it's location. We want to see how we guard against this particular form of construction error. The plots are shown in 27. We see that small radii are extremely susceptible to receiver mislocation. It's only for large radius of about 0.03 m that mislocations stops becoming a source of error.

We found another interesting and unexpected relation. It turns out that width of the trough is an important factor for the intercept factor, as shown in Fig. 28. Increasing the width decreases the intercept factor. Therefore,

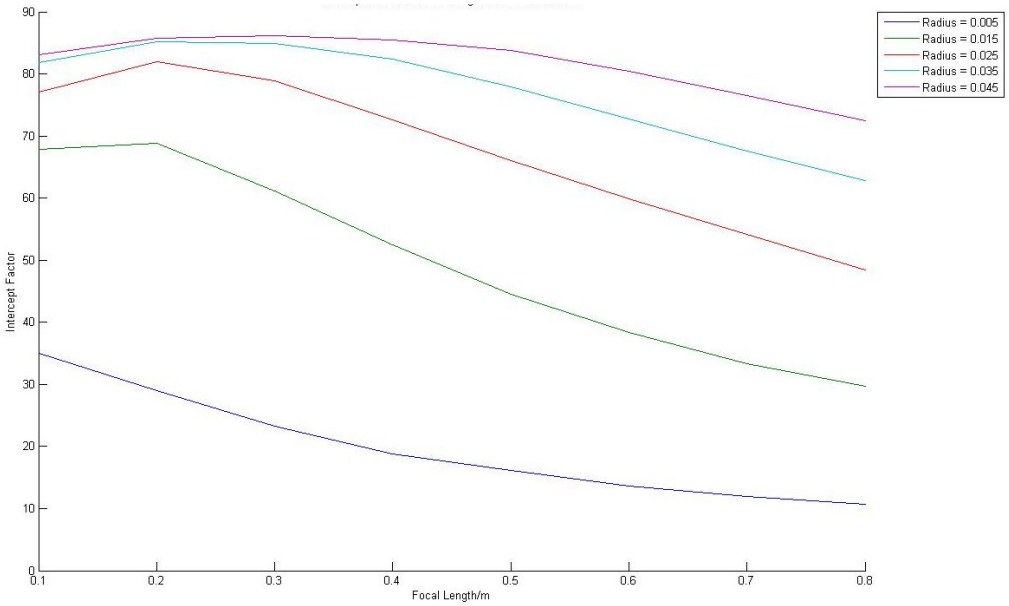


Figure 22: Intercept factor vs the focal length with radius as a parameter.

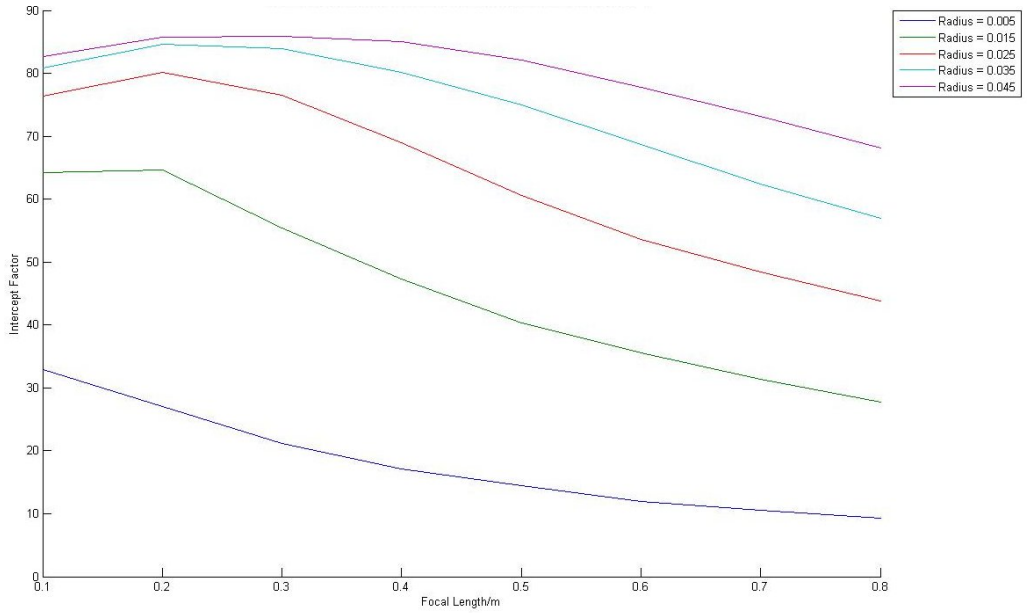


Figure 23: Intercept factor vs the focal length with radius as a parameter with a tracking error of 1^o.

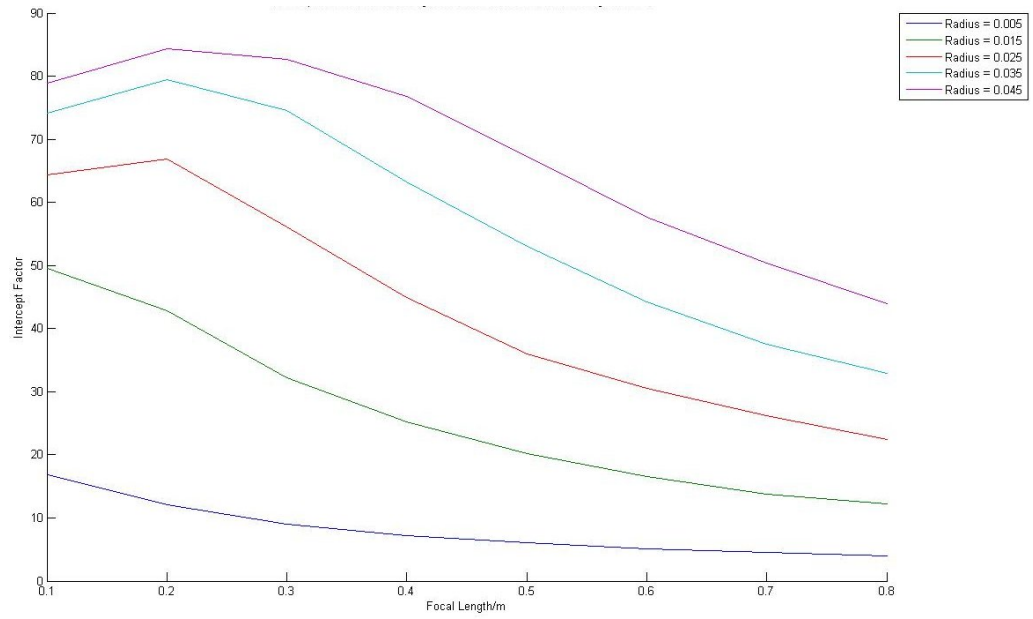


Figure 24: Intercept factor vs the focal length with radius as a parameter with a tracking error of 3^o.

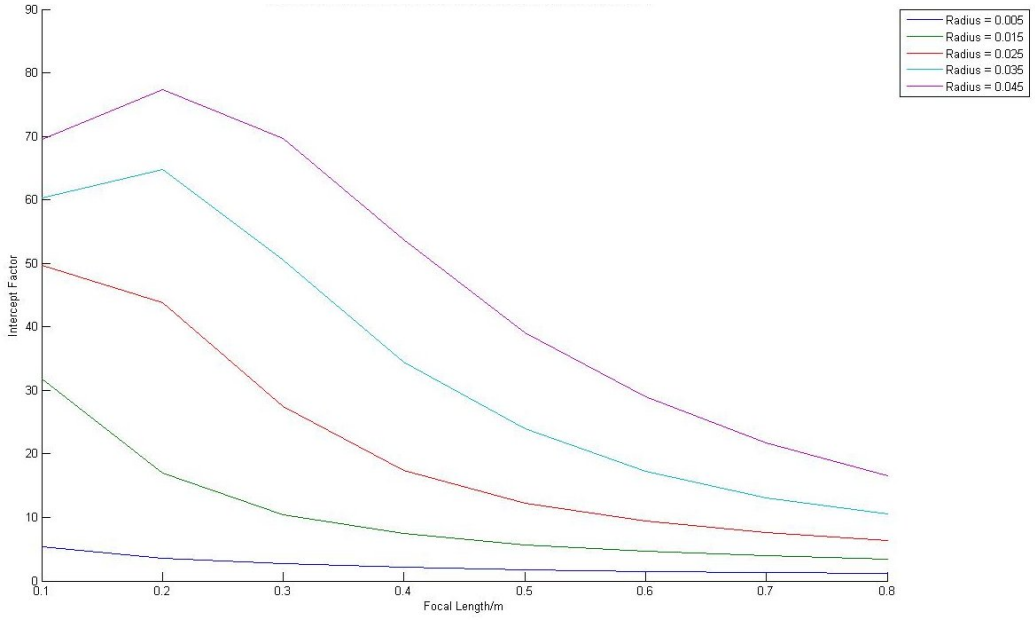


Figure 25: Intercept factor vs the focal length with radius as a parameter with a tracking error of 5^o.

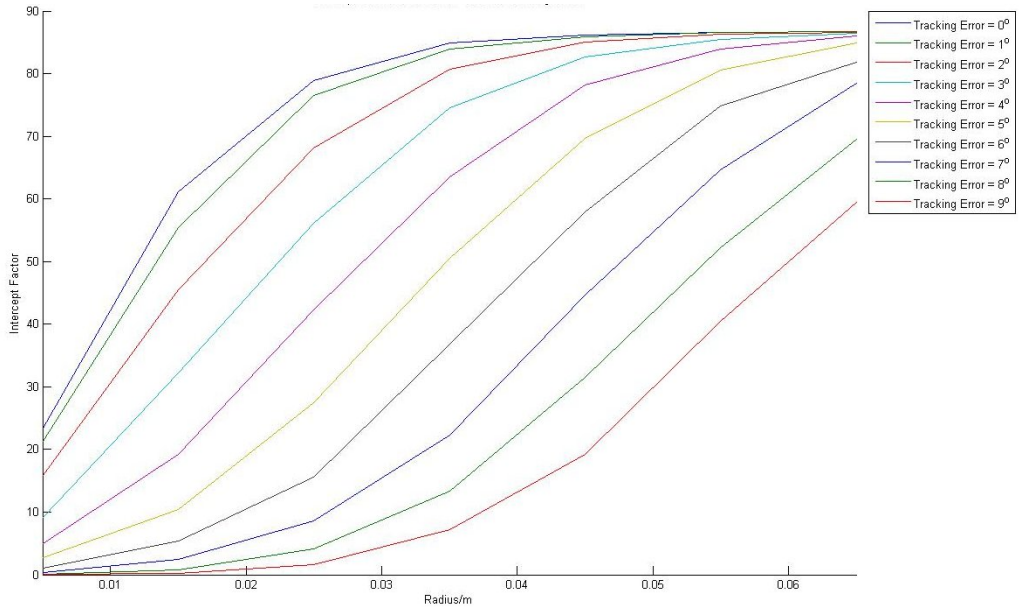


Figure 26: Intercept factor vs radius with tracking as a parameter.

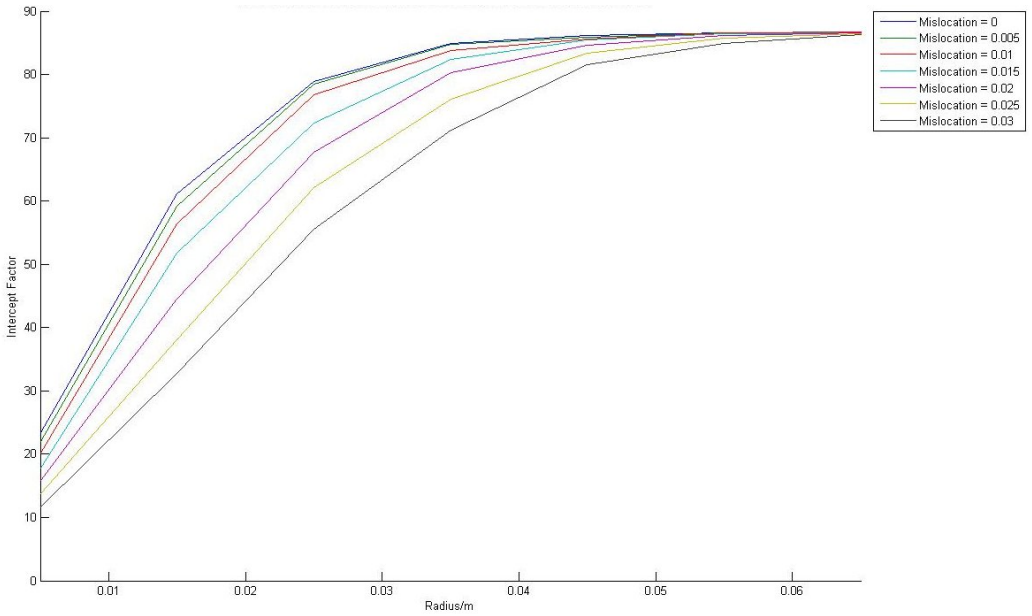


Figure 27: Intercept factor vs radius with receiver mislocation as a parameter.

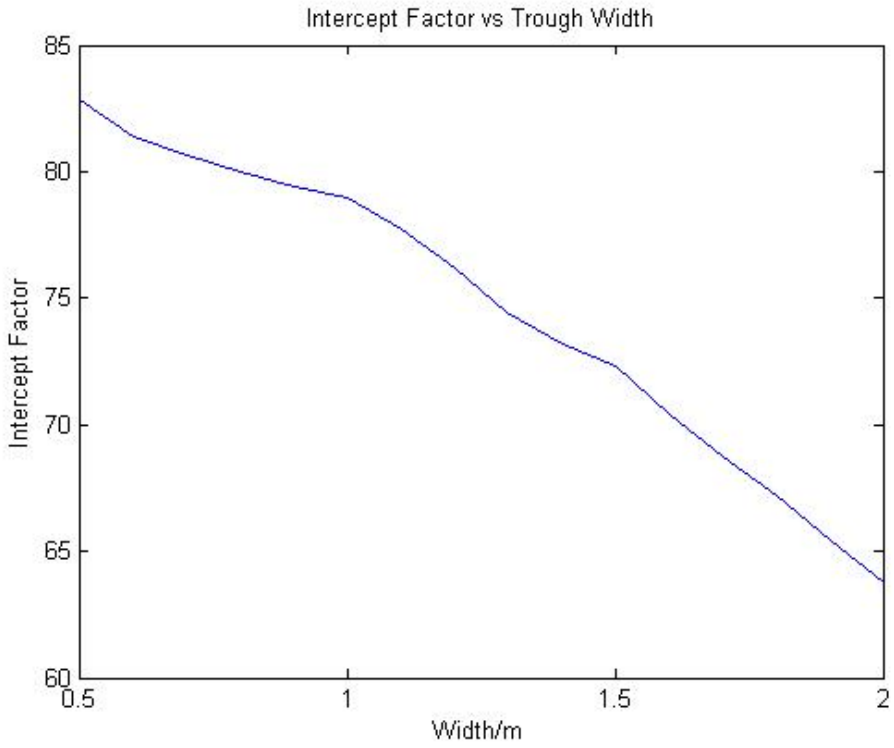


Figure 28: Intercept factor vs the width

the optical model says that the width of the trough should be kept as small as possible. On the other hand the total solar intensity captured will go up as the width goes up. But it also does by increasing the length. So length is the preferred method of changing the area of the collector and not the width.

The above graphs give us some indications of how different construction parameters affect the intercept factor. These results can be summarized as follows,

- The focal length should be small.
- The radius should be large.
- Increasing the width decreases the intercept factor
- Tracking errors cause large drops in the intercept factor.

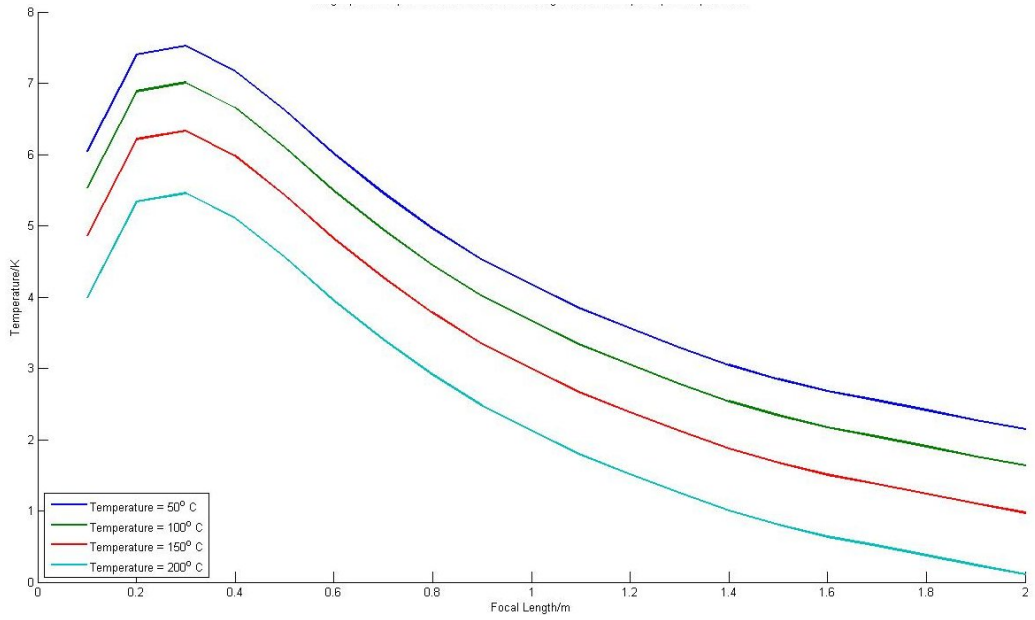


Figure 29: The single pass temperature rise vs the focal length.

6.2 Optical and Thermal System

However, the optical efficiency is not our end goal. It's the temperature increase of our heat transfer fluid. We therefore plug together our two models into a complete collector model which will output the temperature rise of the fluid. We would like to find out the values of the parameters that will increase the temperature rise of the fluid.

Similar to the previous section we will plot the temperature rise against various construction parameters. The input temperature of the liquid is important as the temperature rise depends on it. We plot most quantities for four input temperatures, in 50^o increments from 50^o to $200^o s$.

When we plot the temperature rise vs the focal length we get a nice peak as shown in Fig. **??**. This is near the peak we obtained for the optical model, and since the focal length is not used as a parameter in the thermal model this result is solely because of the optical model. The peak is at around 0.3 m.

We also get a broad peak when we plot against the radius. This is shown in Fig. 30. However, clearly small radii are preferred from how the high temperature graph fall off sharply.

We saw the interesting relationship between width and the intercept factor.

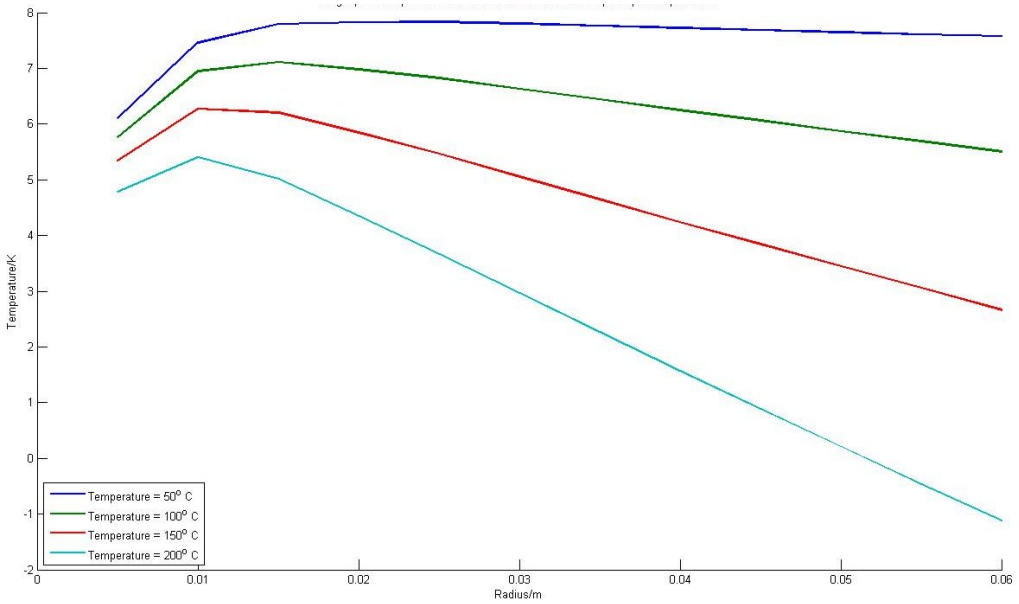


Figure 30: The single pass temperature rise vs radius.

How does this carry over into the thermal model? We kept the total area of our reflective sheet constant. The focal length was then determined from the width. This meant that the aperture area changes with the width. This type of analysis is important because often reflectors are made from sheets that are available in a specific size. The results are shown in Fig. **??** and **??**. The first is for a radius of 1 cm while the second for a radius of 4 cm. There is a sharp peak around a width of about 1.1 m. This is the point at which the increased gains from the increasing width are offset by optical efficiency.

Our final result is the relationship between the length of the trough and the temperature rise. This is shown in Fig. 33. Increasing the lenght of the trough monotonically increases the temperature rise. However as higher temperatures are attained this temperature rise falls off. This fall off depends on the radius as can be done by comparing this figure and Fig. 34.

In conclusion, what have we learned from our complete model. Here is a short summary

- The focal length does peak.
- The radius has a pretty broad speak, but smaller radii are preferred.
- The width offers a sharp peak as well.

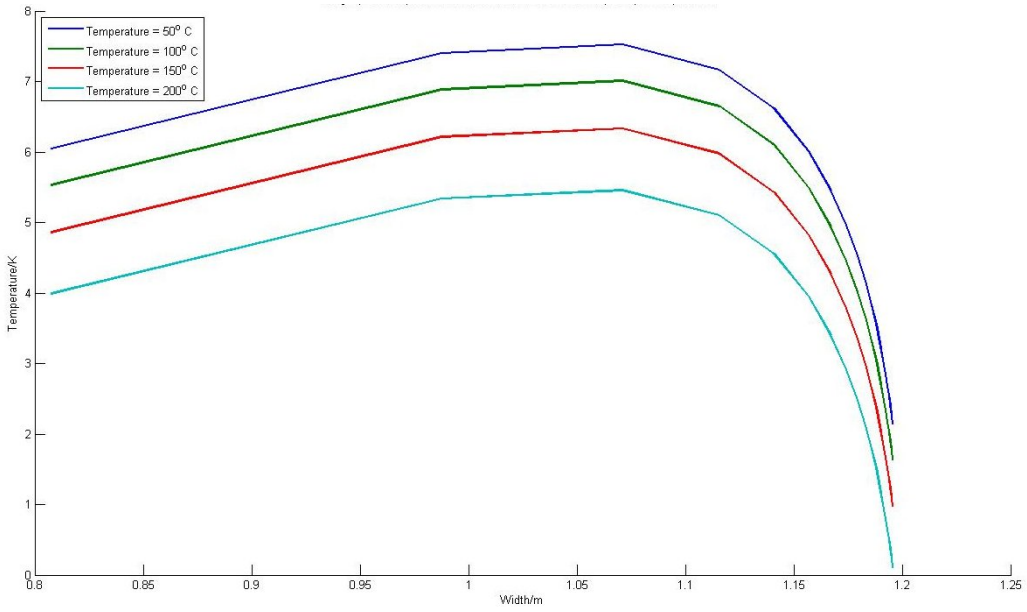


Figure 31: The single pass temperature rise vs the width with radius = 0.01 m.

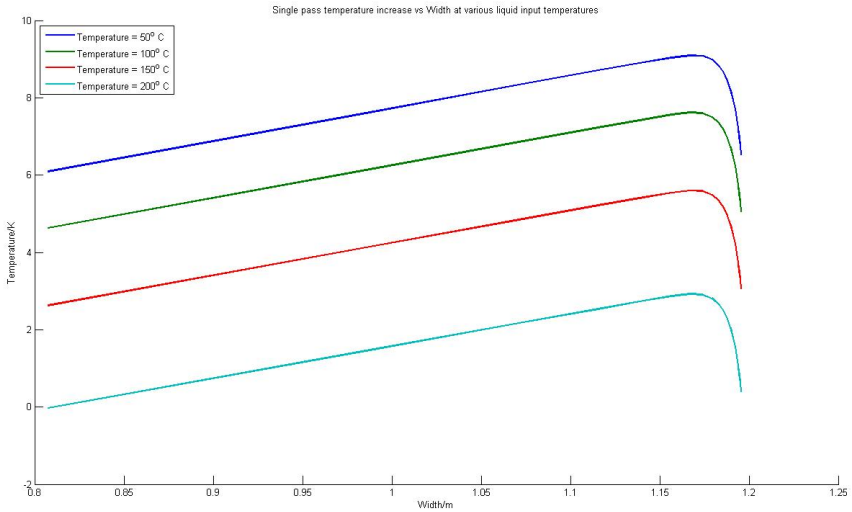


Figure 32: The single pass temperature rise vs the width with radius = 0.04 m.

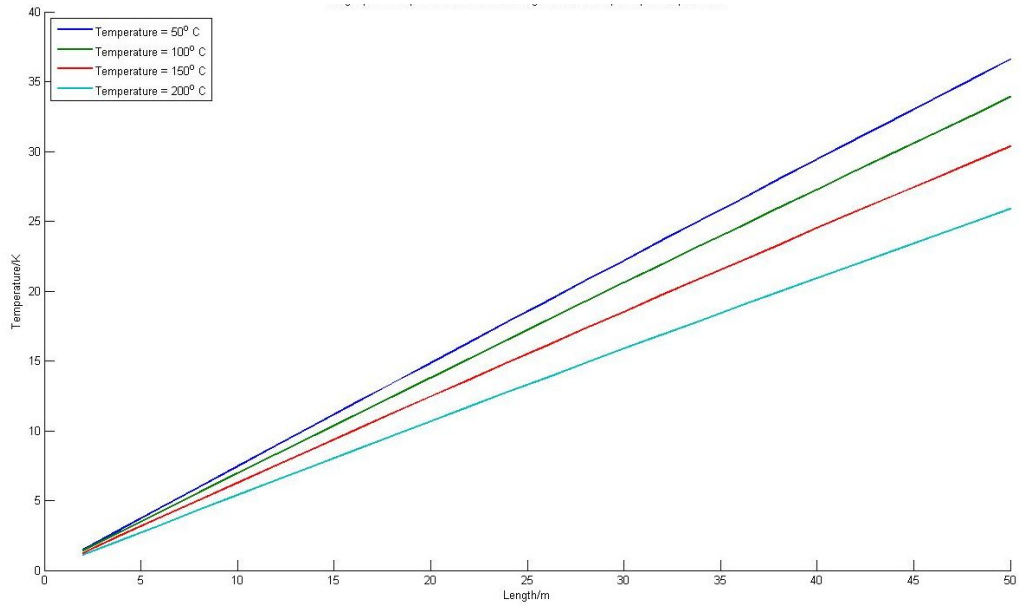


Figure 33: The single pass temperature rise vs the length with radius = 0.04 m.

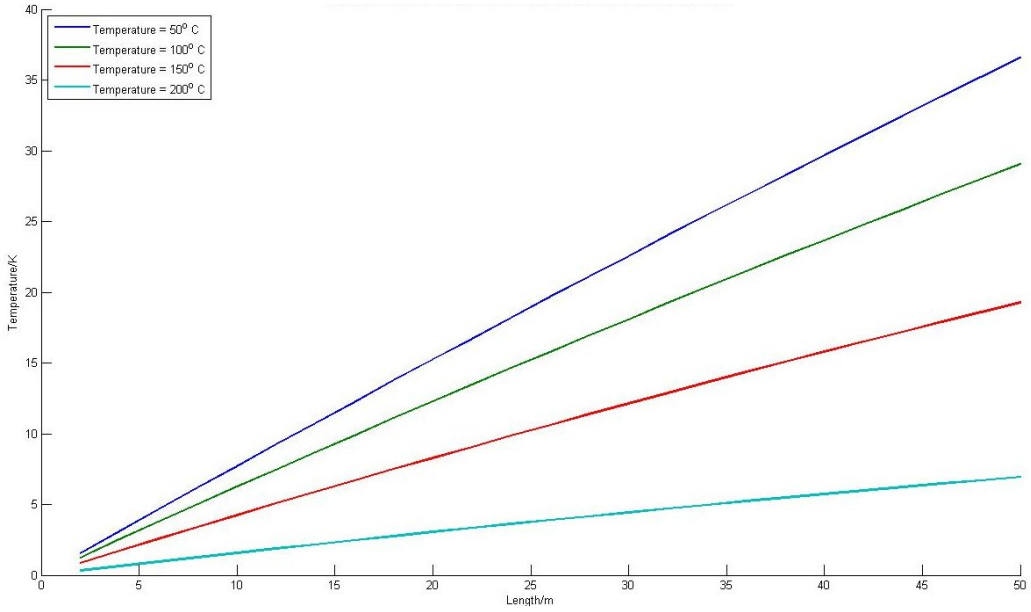


Figure 34: The single pass temperature rise vs the length with radius = 0.04 m.

- Increasing the length does increase temperature, but there is a drop off point.

6.3 Conclusion

The above have only been comparisons between 2 or sometimes 3 variables. There are a wide variety of parameters that all need to be explored. A better optimization will try to look at a global maximum by considering variations in all the parameters.

One axes of optimization that our model ignores is the cost one. Construction costs vary with these parameters, and these costs are typically a strong constraint, especially for developing countries. By also taking these into account, we can create a better optimization.

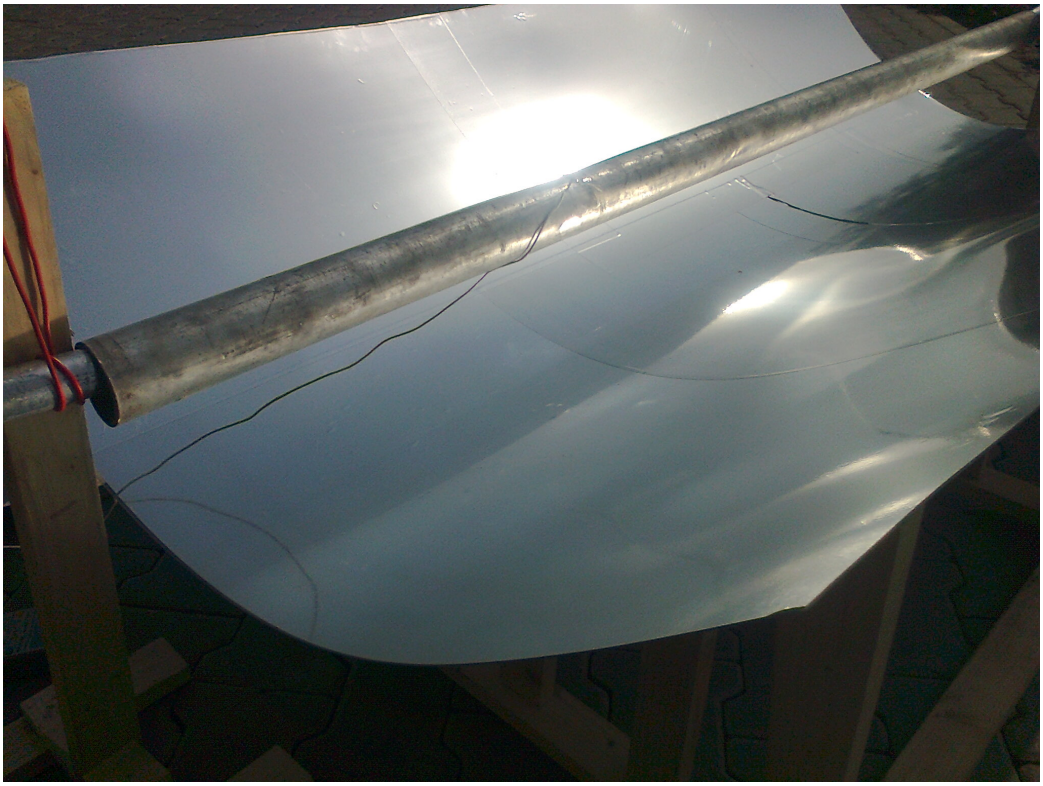


Figure 35: Rough trough constructed to verify simulations.

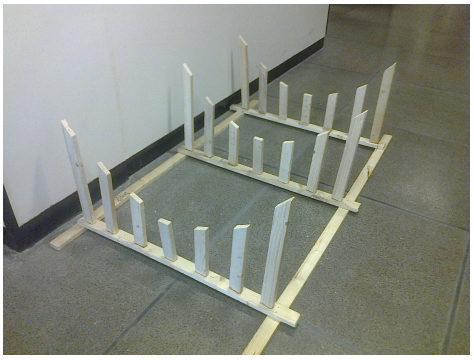
7 Experimental Verification

7.1 Collector Construction

The motivation behind constructing the trough was not to construct the best prototype but rather test the mathematical model and gain some insights in practical trough design by getting our hands dirty and constructing one.

The current design was adapted from (greenology.com). To get a parabolic shape on which to base our design on a graph of a parabola was plotted and printed on a 48x36 inch poster from the Computer Science Department. Wood for the structure was bought from Bhatta Chowk. The wood type was the cheapest available 'partal' priced at Rs. 16/foot for 1 inch by 2 inch, available in 10 foot long pieces. The pieces were cut using a hand saw and hammered and glued into a 4 foot long base of the same material as shown in Figure 36(a).

(a) A handsaw in the hands of an amateur still gives enough accuracy.



(b) l Inside structure of the trough. Has medium robustness and has to be handled with some care.

Figure 36: Constructing trough to test model parameters.

3 such structures were attached to two 6 foot long pieces. No power tool or technical equipment was used in the process.

For the reflective surface, 4x6 feet clear acrylic with a thickness of 1mm was obtained from Brandt Road. Onto this reflective sticker sheets measuring 1.5x2.5 feet were stuck. This surface was then placed onto the structure to obtain the final product shown in Figure 36(b).

7.1.1 Errors in Construction

We are aware of several significant errors at the time of construction. Firstly, because the vertical wood supports had to be hammered in from the bottom it led to slight misplacement of the final support. Secondly, the adhesive sheets had very strong glue and also a strong tendency to stick incorrectly removing the chance of correction once stuck. There are several placed where there bumps in the reflective sheet. Furthermore, as the acrylic is soft it does not maintain a straight shape between supports. We will try to attach thin horizontal support to prevent this. Lastly, although earlier we had disregarded the effects of wind in a mostly wind less place like Lahore. However, during our first test the whole setup was blown away. The wind also deforms the soft acrylic. We will seriously consider wind effects while designing the structure to the final prototype.

7.2 Performance Analysis of the Collector

To track during this study we used simple manual tracking using shadows. The vertical wood used to make the trough all act as a normal to the aperture area. A pipe was place on to the ground in a direction that its shadow was vertically below it. The vertical wood supports were aligned with this pipe thereby making the aperture of the trough normal to the incident sunlight. Water had to be pumped through different radii pipes with different tracking errors and temperature rise recorded. This work is under progress at the time of writing of this report.

7.3 Tesla Turbine Construction and Assembly

The Tesla turbine was constructed based on designs taken from Phoenix Navigations. These detailed CAD drawings were given to a factory in Gujranwala for machining. Included along with the CAD drawings are detailed DIY instructions for making the 7 inch turbine, written by Ken Rieli a Tesla Turbine enthusiast.

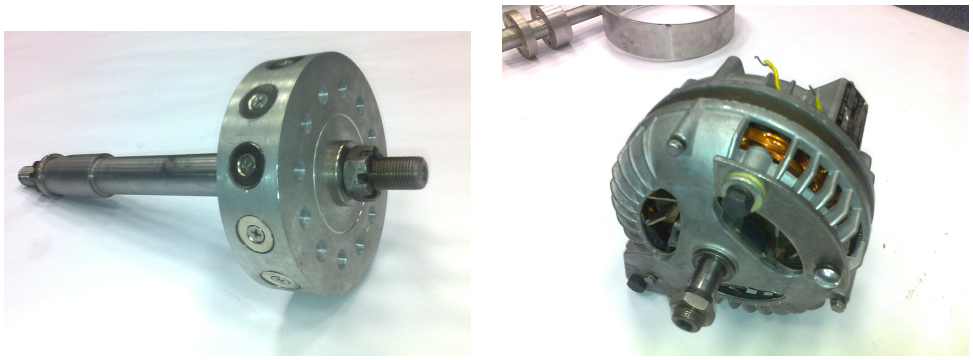
Let us first look at the core component of the Tesla turbine, the turbine disks. It was the deformation of these disks due to poor availability of material that prevented the Tesla turbine from penetrating the market when it was invented. We use XX stainless steel. The disk go into the covering at which one point is open for the nozzle inlet. Nozzle construction is a delicate process and must be very accurate, a good nozzle determines the efficiency of turning gas pressure into shaft horsepower.

(a) Stainless steel Turbine disks. (b) Aluminum disk cover with nozzle.

Figure 37: Basic turbine parts constructed at a factory in Gujranwala.

Attached to the disks and at the heart of the construction is the shaft. On the

other one end it has the rotor. The rotor contains 12 permanent neodymium magnets. After constructing the rotor according to design specifications we found that there was no locally available stator that could fit the required inner diameter. After consultation with Ken we ordered a stator from the US. The permanent neodymium magnets were also ordered from the US, interestingly their other and more prominent use was as door stoppers.



(a) Shaft with permanent magnet rotor. (b) Alternator still in motor casing.

Figure 38: Basic turbine parts constructed at a factory in Gujranwala.

The turbine parts have only recently been finished. This was a long process as it was being constructed for the first time. The parts fit together with ease and we will assemble it soon in the Physics Lab. As the collector construction is still underway, testing will be done with compressed air and steam from a boiler. According to designer specifications this turbine needs 5g of steam at 200 psi per second to produce 1kW of power.

7.4 Costs

We believe that the costs incurred during experimentation and construction are one time and mass produced costs will be much lower. For example, turbine construction was done part by part after studying schematics in detail for the first time and so was very time consuming for the factory. They assured us bulk orders could be made faster and cheaper. The magnets and stator were imported from the US. These parts can be easily acquired from China at much lower rates. Lastly, the reflective sheet used was of low quality (it exhibited diffuse reflection as well as not optimum reflection coefficient) and this must be replaced by reflective mylar.

The parts we were unable to cover include the heat exchange, pump, condenser and tracking mechanism which will add significantly to the costs. All

Part Name	Specifications	Source	Cost
Turbine Machining	Machined on Lathe	Sun Engineers	Rs. 20,000
Stator			Rs. 8000
Neodymium magnets	0.75" dia by 1.25" height	K & J magnetics	Rs. 100/magnet
Wood	2" by 1" Partal	Bhatta Chowk	Rs. 16/foot
Acrylic Sheet	6 feet by 4 feet by 1mm	Brandt Road	Rs. 1850/sheet
Adhesive reflective sheets	1.5 feet by 2.5 feet	Urdu bazaar	Rs. 40/sheet

Table 11: Costs of various parts of construction.

in all we estimate that the complete setup can easily be constructed within the original target amount of Rs. 100,000.

8 Conclusion and Future Directions

We started off with the goal of creating a rooftop solar thermal electricity generation unit. The first step in this direction was the modeling of the solar collector. We presented such a model consisting of submodels with dealt with the optics and thermals of the system. Though our thermal model was largely borrowed, our optical model is superior to those already available in terms of the errors it takes into account. With our model almost complete, we proceeded with figuring out the optimum construction parameters of a collector.

Our optimization results show peaks for various geometry settings where we have varied one, two or three parameters to visually show the working of the code. Our next step is to run the code with N nested for loops, where N are the total number of our variable parameters to find the global optimum of this setup. There will however, be two different optimums, firstly the maximum temperature output and secondly the maximum absorbed energy by the heat transfer fluid at the output. We know these are different as high temperature result in low thermal efficiency. We are interested in both of these results separately.

However, choosing which design to construct will depend on the rest of the setup, i.e the working fluid and heat exchangers etc. We will then package our simulations into a user friendly interface. What we would like is that the user can input their restrictions and desired outputs. For example, if the user has a manual tracking mechanism which can track the sun to 2 degrees and wants to raise the temperature of water from 40 Celsius to 80 Celsius. We will also want to add a cost factor and so for example the user should be able to put in costs of available pipes of different diameters as well as the cost of reflective sheets and trough construction. The software should be then able to weigh the two options of increasing pipe diameter to reduce optical errors vs. increasing the total trough width and length. We believe our code can handle this functionality and just needs a more friendly input interface.

As for our own construction of the prototype, the raw construction of parts of the Tesla turbine is complete and it is waiting assembly. We will construct troughs out of fiberglass based on the optimum parameters with the exception that our fluid flow rate will be restricted by available pumps. Despite serious attempts we have failed at practically understanding heat exchangers and so will attempt to get someone who has experience in this field on board to finish the final assembly of our prototype.

For those who want to build upon our model, the direction to proceed in is

towards this modeling of heat exchangers, turbines and pumps. By having models for all the parts, we can find the optimum parameters and simulate the performance of the complete system. Others may choose to compare our results with work that has already been done in the field. Another direction to choose is to experimentally verify our predictions.

The field of solar thermal power has recently boomed again in the face of rising fuel prices. Research and development in this area is crucial to grapple with the energy needs of the future. Developing countries can especially benefit from such renewable technologies not only as a means to power their economies but also to grow scientifically, technologically and economically.

A Source Code

Please email abdullahkhalids@gmail.com for a copy of the code.

```
%%%%%%%%%%constants.m
%Constants
dataValues;

%Sim area
simulation = struct();
simulation.size = [2 2];
simulation.grainLength = 1e-3;

%Trough Characteristics
trough = struct();
trough.focalLength = 0.2;
trough.focusCoordinates = [0 0];
trough.orientationAngle = 0;
trough.width = 1.;
trough.length = 10;
trough.refractiveIndex = aluminium.refractiveIndex;
trough.surfaceStdDev = 50e-3; %rad
trough.specularity = 0.60;
trough.specularityStdDev = 5e-3;
trough.halfQuantization = 1;
trough.trackingError = deg2rad(0);

%Receiver Characteristics
receiver = struct();
%receiver.radius = 0.03;
receiver.extraLength = 1;
receiver.surfaceStdDev = 0.5e-3;
receiver.emissivity = copper.emissivity;
receiver.absorber = copper;
receiver.gas = air;
receiver.innerDiameterAbsorber = 0.0508; %m
receiver.outerDiameterAbsorber = receiver.
   innerDiameterAbsorber + 0.002; %m
receiver.innerDiameterGlassSleeve = receiver.
   outerDiameterAbsorber + 0.02; %m
```

```
receiver.outerDiameterGlassSleeve = receiver.
    innerDiameterGlassSleeve + 0.007; %m

%Collector cycle
collectorCycle = struct();
collectorCycle.fluid = oil;
collectorCycle.flowRate = 0.5; %kg/s

%Turbine cycle
turbineCycle = struct();
turbineCycle.fluid = butane;
turbineCycle.flowRate = 0.005; %kg/s 0.005
turbineCycle.quality = 0;

%Sun Characteristics
sun = struct();
sun.halfQuantization = 1;

%Location
location = lahore;
location.date = [21  6];
location.time = [12  10];

%Atmosphere
atmosphere = air;
atmosphere.refractiveIndex = 1;
atmosphere.gravity = 9.81;

%Heat Exchangers
heatExchanger = struct();
heatExchanger.UA = 200;

%Turbine
turbine.efficiency = 0.5;

%%%%%%%%%%Calculations.m
%set up some simulation factors
simulation.grainVolume = (simulation.grainLength)^2;

%sun
sun.halfAngle = deg2rad(min2deg(16));
```

```
[sun.widthAngle sun.lengthAngle sun.daytime] =
    SunAngles(location.time,location.date,location.
    latitude,location.longitude,location.
    timezoneLongitude);
sun.positionVector = [sind(sun.widthAngle); cosd(sun.
    widthAngle)]; %check
sun.fullQuantization = 2*sun.halfQuantization + 1;
sun.irradiance = SolarIntensity(location); %W/m^2
%sun.intensityDistribution = PillBox(sun,simulation);

%atmosphere
atmosphere.temperature = AmbientTemperature(location);
atmosphere.windSpeed = WindSpeed(location);
atmosphere.pressure = atm2bar(1);

%compute trough coordinates
trough.rotAngle = deg2rad(sun.widthAngle) + trough.
    trackingError;
trough.height = TroughHeight(trough.focalLength,trough.
    width);
trough.rimAngle = TroughRimAngle(trough.focalLength,
    trough.width);
trough.coordinates = TroughCoordinates(trough,
    simulation);
trough.gradients = TroughGradient(trough.coordinates,
    trough.surfaceStdDev);
trough.coordinates = trough.coordinates(:,2:end-1); %
    throw away the coordinates no longer needed
trough.fullQuantization = 2*trough.halfQuantization +
    1;
if trough.halfQuantization == 0; trough.specularity =
    1; end;

%compute receiver stuff
receiver.radius = receiver.outerDiameterGlassSleeve;
receiver.length = trough.length + receiver.extraLength;
receiver.troughLength = trough.length;
receiver.position = trough.focusCoordinates;
receiver.coordinates = RecieverCoordinates(receiver,
    simulation);
receiver.gradients = ReceiverGradient(receiver);
```

```
%Collector Cycle
collectorCycle.inletTemperature = atmosphere.
    temperature;
collectorCycle.outletTemperature = atmosphere.
    temperature;
collectorCycle.speed = fluidSpeed(collectorCycle.
    flowRate,receiver.radius,collectorCycle.fluid.
    density);
collectorCycle.quality = 0;

%Turbine Cycle
turbineCycle.turbineInletTemperature = atmosphere.
    temperature;
turbineCycle.turbineOutletTemperature = atmosphere.
    temperature;
turbineCycle.inletTemperature = atmosphere.temperature;
turbineCycle.turbineInletPressure = atmosphere.pressure
    ;
turbineCycle.turbineOutletPressure = atmosphere.
    pressure;
turbineCycle.speed = fluidSpeed(turbineCycle.flowRate,
    receiver.radius,turbineCycle.fluid.density);
turbineCycle.quality = 1;


%%%%%%%%% SunAngles.m
function [alpha,beta,daytime] = SunAngles(time,date,
    latitude,longitude,timezoneLongitude)
%Computes the position of the sun given the time,date,
    and location
%time is in the 24 hour clock, in the form [hour min
    seconds] or just
%hours.
%date is in the format [day month], or just day of the
    year
%location is in latitude and longitude, with each
    either a single number of
%an array with each elemement given the degrees,min,
    seconds etc.
```

```
%Get the decimal time
if length(time)>1
    time = deg2dec(time);
end

%Get the day of the year
if length(date)>1
    date = date2day(date(1),date(2));
end

%Get the location
if length(latitude)>1
    latitude = deg2dec(latitude);
end

if length(longitude)>1
    longitude = deg2dec(longitude);
end

%Compute the solar declination
solarDeclination = 23.45*sind(360/365*(284+date));

%Get decimal date
dateDec = date + time/24;

%Calculate the hour angle
x = 360/365.242*(dateDec - 1);

%Corrections to solar time
EOT = 0.258*cosd(x) - 7.146*sind(x) - 3.648*cosd(2*x) -
    9.228*sind(2*x);
%Longitude correction
LongitudeCorrection = 4*(timezoneLongitude - longitude)
    ;
%Correction in hours
Correction = (EOT + LongitudeCorrection)/60;

solarTime = time + Correction;

hourAngle = 15*(solarTime - 12);
```

```
%Compute sun angles
alpha = atand((cosd(solarDeclination)*sind(hourAngle))
    /(sind(latitude)*sind(solarDeclination) + cosd(
    latitude)*cosd(solarDeclination)*cosd(hourAngle)));
beta = atand((cosd(latitude)*sind(solarDeclination) -
    sind(latitude)*cosd(solarDeclination)*cosd(hourAngle
    ))/(sind(latitude)*sind(solarDeclination) + cosd(
    latitude)*cosd(solarDeclination)*cosd(hourAngle)));


if hourAngle<0
    alpha = -abs(alpha);
end

% %Compute the vector for the sun
% S = [cosd(beta)*sind(alpha); cosd(beta)*cosd(alpha);
    sind(beta)*cosd(alpha)]/sqrt(cosd(beta)^2 + sind(
    beta)^2*cosd(alpha)^2);

%Compute Sunrise and sunset angles
sunInSkyAngle = acosd(-tand(latitude)*tand(
    solarDeclination));

if abs(hourAngle) > abs(sunInSkyAngle)
    daytime = 0;
else
    daytime = 1;
end


end

%%%%%%%%%% TroughCoordinates.m

function movRotCoord = TroughCoordinates(trough,
    simulation)
%Computes the coordinates of all the points on the
    trough

%rotation matrix
```

```
rotMat = [cos(trough.rotAngle) sin(trough.rotAngle);
    ...
     -sin(trough.rotAngle) cos(trough.rotAngle)];

%Get all possible x
coord = -trough.width/2-simulation.grainLength:
    simulation.grainLength:trough.width/2+simulation.
    grainLength;

%Compute corresponding y for horizontal parabola
coord(2,:) = coord.^2/(4*trough.focalLength);

% figure;
% hold on;
% axis equal
% axis([-simulation.size(1) simulation.size(1) -
    simulation.size(2) simulation.size(2)]);
% plot(coord(1,:),coord(2,:));

%move focus to origin
coord(2,:) = coord(2,:) - trough.focalLength;

% plot(coord(1,:),coord(2,:));

%rotate the parabola around focus
rotCoord = rotMat*coord;

% plot(rotCoord(1,:),rotCoord(2,:));


%move it
movRotCoord = rotCoord + repmat(trough.focusCoordinates
    ',1,size(rotCoord,2));

% figure;
% hold on;
% axis equal
% axis([-simulation.size(1) simulation.size(1) -
    simulation.size(2) simulation.size(2)]);
% plot(movRotCoord(1,:),movRotCoord(2,:));
```

```
%plot(trough.focusCoordinates(1),trough.
    focusCoordinates(2),'ko');


end

%%%%%%%%%% TroughGradients.m

function gradientsNew = TroughGradient(coord,sigma)
%Computes the surface of the trough

gradientsNew = SurfaceErrors(coord,sigma);


end

%%%%%%%%%% SurfaceErrors.m

function gradientsNew = SurfaceErrors(coord,sigma)
%Introduces surface deformities into a surface

%first compute the gradient of the surface
gradients = Gradient(coord);

%compute gradients of the normal
gradientsNormal = -1./gradients;

%compute angles with the vertical of the normal
    gradients
anglesNormal = grad2angle(gradientsNormal);

%randomly perturbate angles
anglesNormal = anglesNormal + random('normal',0,sigma
    ,1,length(anglesNormal));

%compute new normal gradients
gradientsNormalNew = angle2grad(anglesNormal);

%compute new gradients
gradientsNew = -1./gradientsNormalNew;
```

```
end

%%%%%%%% ReceiverCoordinates.m
function receiverCoordinates = RecieverCoordinates(
    receiver, simulation)
%Computes all the coordinates of the receiver

x = 0:simulation.grainLength:receiver.radius;
y = sqrt(receiver.radius^2 - x.^2);
coord = [x fliplr(x(1:end-1)) -x(2:end) -fliplr(x(1:end
    -1)); y -fliplr(y(1:end-1)) -y(2:end) fliplr(y(1:end
    -1))];

%move to the correct position
receiverCoordinates = coord + repmat(receiver.position
    ',1,size(coord,2));

end


%%%%%%%% ReceiverGradient.m
function gradients = ReceiverGradient(receiver)
%Computes the gradients of the receiver making sure it'
    s a closed surface.
%There are errors on the edges

gradients = SurfaceErrors([receiver.coordinates(:,end)
    receiver.coordinates receiver.coordinates(:,1)],
    receiver.surfaceStdDev);

end

%%%%%%%% fluidSpeed.m
function v = fluidSpeed(flowRate,radius,density)
%Given a fluid with a density flowing through a
    cylindrical pipe with a
%radius at a flowRate, computes the fluid speed.

v = flowRate/(pi*radius^2*density);

end
```

```
%%%%%%%%% ReceiverIntensityDistribution.m
function distribution = ReceiverIntensityDistribution(
    simulation,trough,receiver,sun,atmosphere)
%Computes the intensity distribution over the receiver.

%Compute the incidence angles,vectors and intensities
[anglesIncidence, vectorsIncidence,
    intensitiesIncidence] = SunRays(sun,trough.gradients
    ,simulation);

%Compute transmission angles
anglesTransmission = Transmissions(anglesIncidence,
    atmosphere,trough);

%Compute gradients
[vectorsReflection, intensitiesReflection] =
    Reflections(atmosphere, trough, anglesIncidence,
    vectorsIncidence,intensitiesIncidence,
    anglesTransmission);
gradientsReflection = vectorsReflection(2,:)./
    vectorsReflection(1,:);

%Compute where each reflected ray ends up on the
    receiver
indexes = ReceiverIntersections(gradientsReflection,
    trough,receiver,sun);

%add up all the intensities at the receiver for each
    point
distribution = zeros(1,length(intensitiesReflection));
for i=1:length(intensitiesReflection)
    if indexes(i)~=0
        distribution(indexes(i)) = distribution(indexes
            (i)) + intensitiesReflection(i);
    end
end


end
```

```
%%%%%%%%%% SunRays.m
function [anglesIncidence, vectorsIncidence,
    intensitiesIncidence] = SunRays(sun,gradients,
    simulation)
%Given a sun computes the incidence angles and incident
     vectors on a
%surface given the gradient at each point as well as
    the corresponding
%intensities

%The angle that the center of the sun cone makes with
    each point
anglesConeCenters = acos(abs((sun.positionVector(2) -
    sun.positionVector(1)*gradients)./sqrt(1+gradients
    .^2)));

%Expand them out to get rays for the whole cone
anglesIncidence = ExpandAngles(anglesConeCenters,sun.
    halfQuantization,sun.halfAngle);

%the incidence vectors at each point
vectorsIncidence = RotateVectors(repmat(sun.
    positionVector,1,length(gradients)),sun.
    halfQuantization,sun.halfAngle);

%intensities
IntensityPerCone = repmat(sun.irradiance*simulation.
    grainVolume,1,size(anglesConeCenters,2));
base = ones(1,sun.fullQuantization)/sun.
    fullQuantization;
intensitiesIncidence = ExpandIntensities(
    IntensityPerCone,base);

end

%%%%%%%% Transmissions.m
function anglesTransmission = Transmissions(
    anglesIncidence,medium1,medium2)
%Given incident angles computes the transmission angles
```

```
anglesTransmission = snellLaw(anglesIncidence,medium1.
    refractiveIndex,medium2.refractiveIndex);


end

%%%%%%%%% Reflections.m

function [vectorsReflection, intensitiesReflection] =
    Reflections(material1, material2, anglesIncidence,
    vectorsIncidence,intensitiesIncidence,
    anglesTransmission)
%Computes the reflected angles, vectors, and intensites
     given the incident
%vectors, intensities and the gradient of the
    reflection surface

m = size(vectorsIncidence,2)/size(material2.gradients
    ,2);

%compute the normal vector at each point
normals = repmat(1./sqrt(1 + material2.gradients.^2)
    ,2,1).*[-material2.gradients; ones(1,length(
    material2.gradients))];

%expand these out to the correct size
normalsExpanded = expand(normals,@(x)repmat(x,1,m));

%compute reflected vectors
vectorsReflected = vectorsIncidence - repmat(2*sum(
    vectorsIncidence.*normalsExpanded,1),2,1).*
    normalsExpanded;

%Account for specularity
vectorsReflection = RotateVectorsMat(vectorsReflected,
    material2.halfQuantization,abs(random('normal',0,
    material2.specularityStdDev,1,length(
    vectorsReflected))));
```

```
%reflected intensities are reduced by the frenel
    coefficients
intensitiesReflection = intensitiesIncidence*
    frenelReflectionCoefficient(anglesIncidence,
    anglesTransmission,material1.refractiveIndex,
    material2.refractiveIndex,'mixed');

%then split into specular and non-specular rays
base = (1-material2.specularity)/(2*material2.
    halfQuantization) * ones(1,material2.
    fullQuantization);
base(material2.halfQuantization+1) = material2.
    specularity;
intensitiesReflection = ExpandIntensities(
    intensitiesReflection,base);

end

%%%%%%%%%% ReceiverIntersections.m

function indexes = ReceiverIntersections(
    gradientsReflected,trough,receiver,sun)
%Computes the indexes of the points on the receiver
    where the rays from the
%trough are incident

%expand out coordinates to account for
coordinates = expand(trough.coordinates,@(x)repmat(x,1,
    sun.fullQuantization*trough.fullQuantization));

%Computes intersection with the receiver
pointsIntersections = LineCircleIntersection(
    coordinates,gradientsReflected,receiver.radius,
    receiver.position);

%compute the intersection coordinates using a nearest
    neighbour approach
l = length(pointsIntersections);
indexes = zeros(1,l);

for i=1:l
```

```
    if pointsIntersections(1,i) ~= 0 &&
       pointsIntersections(2,i) ~= 0 && ~isnan(
       pointsIntersections(1,i)) && ~isnan(
       pointsIntersections(2,i)) && ~isinf(
       pointsIntersections(1,i)) && ~isinf(
       pointsIntersections(2,i))
        indexes(i) = NearestNeighbour(receiver.
           coordinates, pointsIntersections(:,i));
    end
end
end

%%%%%%%%%% LineCircleIntersections.m
function pointsIntersections = LineCircleIntersection(
   coordinates,gradients,radius,center)
%Computes the first interesection between a set of
   lines and a circle

%compute the determinent of the quadratic equation of
   intersection.
c = coordinates(2,:) - gradients.*coordinates(1,:);
det = (gradients.*(c - center(2)) - center(1)).^2 - (1
   + gradients.^2).*((center(2) - c).^2 + (center(1)^2
   - radius^2));

%check which lines intersect
isIntersection = det >=0;

%compute both points of intersection
intersections1 = isIntersection.*(-(gradients.*(c-
   center(2)) - center(1)) + sqrt(det)./(1 + gradients
   .^2));
intersections2 = isIntersection.*(-(gradients.*(c-
   center(2)) - center(1)) - sqrt(det)./(1 + gradients
   .^2));
intersections1(2,:) = gradients.*intersections1 + c;
intersections2(2,:) = gradients.*intersections2 + c;

%compute the squared distance between trough point and
   receiver point
dis1 = SquareDistance(intersections1,coordinates);
```

```
dis2 = SquareDistance(intersections2,coordinates);

%figure out the closer one.
pointsIntersections = intersections1;
shouldSwitch = dis2 < dis1;
shouldSwitch = repmat(shouldSwitch,2,1);
pointsIntersections(shouldSwitch) = intersections2(
    shouldSwitch);

end


%%%%%%%%%% Flux.m
function [PowerReceiver, PowerTrough,InterceptFactor] =
     Flux3D(receiverDistribution,trough,receiver,sun,
    simulation)
%2DFLUX3D Given the 2D distribution of flux on a grain
    length width element
% of the receiver, computes the 3D total flux etc

%Set up coordinate system
te = trough.length/2;
ts = -te;
re = receiver.length/2;


%The total flux per strip of the receiver
powerPerStrip = sum(receiverDistribution);

%Length of trough that does not contribute to flux on
    the receiver directly
%above the trough
angleFactor = trough.focalLength*tand(abs(sun.
    lengthAngle));

if angleFactor + ts < re %if the sun is not too
    inclined
     len = angleFactor - (re-te);
     if len < 0
         lengthCorrection = 0;
     else
```

```
        lengthCorrection = abs(len);
    end
else
    lengthCorrection = trough.length;
end

%Compute total power on receiver
PowerReceiver = powerPerStrip*(trough.length -
   lengthCorrection)/simulation.grainLength;

%Compute the total power that the trough received
PowerTrough = trough.width*trough.length*sun.irradiance
   *cosd(sun.widthAngle - rad2deg(trough.rotAngle));

%Compute efficiency
InterceptFactor = PowerReceiver/PowerTrough;


end

%%%%%%%%%% ReceiverTemperature.m
function [Tout,T] = ReceiverTemperature(receiver, flux,
    collectorCycle, atmosphere)
%Computes temperature of the output

segmentLength = 0.1;
m = ceil(receiver.troughLength/segmentLength);

fluxLengthIntensity = flux/receiver.length;
Tin = collectorCycle.inletTemperature;
T = zeros(m,5);
Tguess = atmosphere.temperature*ones(1,5);


for i = 1:m+1

    %Given the input temp and a guess, compute the
       temperatures of this
    %segment
    options=optimset('Display','off');
```

```
        h = @(T) heatBalanceEquations(T,Tin,
            fluxLengthIntensity,receiver,collectorCycle,
            atmosphere,segmentLength);
        T(i,:) = fsolve(h,Tguess,options);

        %Compute input temperature of next segment
        Tin = 2*T(i,1) - Tin;

        %Take current temperatures as guesses for the next
            segment
        Tguess = T(i,:);
    end

    %At the end the output temperature is just given by the
        output of the last
    %segment
    Tout = Tin;

    end

    %%
    function F = heatBalanceEquations(T,Tin,
        fluxLengthIntensity,receiver,collectorCycle,
        atmosphere,segmentLength)

    %1:fluid
    %2:absorber inner
    %3:absorber outer
    %4:glass inner
    %5:glass outer
    %6:air
    %7:sky

    g = atmosphere.gravity;
    sigma = 5.67e-8;
    D2 = receiver.innerDiameterAbsorber;
    D3 = receiver.outerDiameterAbsorber;
    D4 = receiver.innerDiameterGlassSleeve;
    D5 = receiver.outerDiameterGlassSleeve;
```

```
%e2 = 0.88;
e3 = 0.88;
e4 = 0.86;
e5 = 0.86;

absorptanceGlass = 0.15;
absorptanceAbsorber = 0.9;

q5sun = fluxLengthIntensity*absorptanceGlass;
q3sun = fluxLengthIntensity*(1-absorptanceGlass)*
    absorptanceAbsorber;


%Convection from absorber to the fluid
Nu1 = 4.36;
%Nu1 changes when inner tube
k1 = materialProperty(collectorCycle.fluid.
    thermalConductivityTable,T(1));
h1 = Nu1*k1/D2;

q12conv = h1*D2*pi*(T(2)-T(1));

%Conduction in absorber from outer to inner
k23 = receiver.absorber.thermalConductivity;
q23cond = 2*pi*k23*(T(3) - T(2))/log(D3/D2);


%Convection from aborber outer to glass inner
T34 = average(T(3),T(4));
beta34 = 1/T34;
alpha34 = materialProperty(receiver.gas.
    thermalDiffusivityTable,T34);
viscosity34 = materialProperty(receiver.gas.
    viscosityTable,T34);
Pr34 = materialProperty(receiver.gas.prandtlNumberTable
    ,T34);
k34 = materialProperty(receiver.gas.
    thermalConductivityTable,T34);

RaD3 = g*(T(3) - T(4))*D3^3*beta34/(alpha34*viscosity34
    );
```

```
q43conv = 2.425*k34*(T(3) - T(4))*(Pr34*RaD3/(0.861 +
    Pr34))^(1/4)/(1 + (D3/D4)^(3/5))^(5/4);


%Radiation from absorber outer to glass inner
q43rad = sigma*pi*D3*(T(3)^4 - T(4)^4)/(1/e3 + (1 - e4)
    *D3/(e4*D4));


%Conduction within glass from inner to outer
k45 = 1.1; %pyrex
q54cond = 2*pi*k45*(T(4) - T(5))/log(D5/D4);


%Convection from glass to air

vis6 = materialProperty(atmosphere.viscosityTable,
    atmosphere.temperature);
Pr5 = materialProperty(atmosphere.prandtlNumberTable,T
    (5));
Pr6 = materialProperty(atmosphere.prandtlNumberTable,
    atmosphere.temperature);
v = atmosphere.windSpeed;
Re = v*D5/vis6;
if Re<=40
     C = 0.75;
     m = 0.4;
elseif Re<=1000
     C = 0.51;
     m = 0.5;
elseif Re<=200e3
     C = 0.26;
     m = 0.6;
else
     C = 0.076;
     m = 0.7;
end

if Pr5<=10
     n = 0.37;
else
```

```
    n = 0.36;
end

Nu56Wind = C*Re^m*Pr6^n*(Pr6/Pr5)^(1/4);

T56 = average(T(5),atmosphere.temperature);
k56 = materialProperty(atmosphere.
   thermalConductivityTable,T56);
h56 = Nu56Wind*k56/D5;

q65conv = h56*pi*D5*(T(5) - atmosphere.temperature);

%Radiation from glass to sky
Tsky = atmosphere.temperature - 8;
q75rad = sigma*D5*pi*e5*(T(5)^4 - Tsky^4);

%% Equations
F(1) = q12conv - q23cond;
F(2) = q3sun - q23cond - q43conv - q43rad;
F(3) = q43rad + q43conv - q54cond;
F(4) = q5sun + q54cond - q75rad - q65conv;
F(5) = (q3sun - q43rad - q43conv)*segmentLength/(
   collectorCycle.flowRate*collectorCycle.fluid.
   heatCapacity) + 2*(Tin - T(1));

F = real(F);

end

%%%%%%%%% expand.m
function A = expand(mat,fun)
%Expands a matrix by expanding by applying an operation
    given by fun on
%each element

A = cell2mat(arrayfun(fun,mat,'UniformOutput', false));

end

%%%%%%%%% ExpandAngles.m
```

```
function anglesOut = ExpandAngles(anglesIn,n,
    maxRotation)
%Given a set of angles, finds a new set of angles by
    adding and subtracting
%from them

%find the incremental angle
incAngle = maxRotation/n;

if n == 0
     incAngle = 0;
end

%set up a base matrix using which to expand the matrix
base = (-n:n)*incAngle;

%set up a function that expands a single angle
fun = @(x)x+base;

%expand out all angles
anglesOut = expand(anglesIn,fun);

end

%%%%%%%% ExpandIntensities.m
function intensitiesOut = ExpandIntensities(
    intensitiesIn,base)
%Given a set of intensites, expands them out using the
    percentages given in
%base.

n = length(base);
fun = @(x)repmat(x,1,n).*base;
intensitiesOut = expand(intensitiesIn,fun);

end

%%%%%%%% expandmat.m
function A = expandmat(mat,fun)
%Expand a matrix by applying operations to each column
    of the matrix
```

```
[r c] = size(mat);

cel = mat2cell(mat,r,ones(1,c));

A = cell2mat(cellfun(fun,cel,'UniformOutput', false));

end

%%%%%%%%% frenelReflectionCoefficients.m
function R= frenelReflectionCoefficient(incidenceAngles
    ,transmissionAngles,incidentIndex,transmissionIndex,
    polarization)
%Computes the frenel reflection coefficient given the
    input parameters. The
%value of polarization can either be 's' or 'p' or '
    mixed'.

R = 0;

if strcmp(polarization,'mixed')
    R = (s(incidenceAngles,transmissionAngles,
        incidentIndex,transmissionIndex) + p(
        incidenceAngles,transmissionAngles,incidentIndex
        ,transmissionIndex))/2;
elseif strcmp(polarization,'s')
    R = s(incidenceAngles,transmissionAngles,
        incidentIndex,transmissionIndex);
elseif strcmp(polarization,'p')
    R = p(incidenceAngles,transmissionAngles,
        incidentIndex,transmissionIndex);
end


end

function Rs = s(incidenceAngles,transmissionAngles,
    incidentIndex,transmissionIndex)
rs = (incidentIndex.*cos(incidenceAngles) -
    transmissionIndex.*cos(transmissionAngles))./(
    incidentIndex.*cos(incidenceAngles) +
```

```
    transmissionIndex.*cos(transmissionAngles));
Rs = rs.*conj(rs);
end

function Rp = p(incidenceAngles,transmissionAngles,
    incidentIndex,transmissionIndex)
rp = ((incidentIndex.*cos(transmissionAngles) -
    transmissionIndex.*cos(incidenceAngles))./(
    incidentIndex.*cos(transmissionAngles) +
    transmissionIndex.*cos(incidenceAngles))).^2;
Rp = rp.*conj(rp);
end

%%%%%%%% nearestNeighbour.m
function nearestNeighbourIndex = NearestNeighbour(
    searchSpace, points)
%Computes the index of the nearest neighbour of the '
    point' by looking at
%point in the searchSpace. Uses Euclidean Distance
%searchSpace: d x k where there are k vectors in a d
    dimensional space
%point: d x p are p d dimensional vectors

p = size(points,2);
r = size(searchSpace,2);
nearestNeighbourIndex = zeros(1,p);

for i = 1:p
    %compute square distances of each point
    squareDistances = sum((searchSpace - repmat(points
        (:,i),1,r)).^2,1);

    %get the nearest neighbour index
    nearestNeighbourIndex(i) = find(squareDistances ==
        min(squareDistances),1);
end

%%%%%%%% rotateVectors.m
function vectorsOut = RotateVectors(vectorsIn,n,
    maxRotation)
```

```
%Given a set of vectors, rotates each one about itself
    to get a new set of
%expanded vectors. The angles given.


%lengths
m = 2*n+1;
vn = size(vectorsIn,2);

%base rotation matrix
base = ExpandAngles(0,n,maxRotation);

%expand out the input vectors
v = expand(vectorsIn,@(x)repmat(x,1,m));

%get rotation matrices
c = repmat(cos(base),1,vn);
s = repmat(sin(base),1,vn);

vectorsOut = sum([c;s].*v);
vectorsOut(2,:) = sum([-s;c].*v);


end

%%%%%%%%% rotateVectorsMat.m
function vectorsOut = RotateVectorsMat(vectorsIn,n,
    maxRotations)
%Given a set of vectors and angles, it rotates each one
     2*n+1 times.

%expand out the vectors
v = expandmat(vectorsIn,@(x)repmat(x,1,2*n+1));

%expand out the angles
angles = expand(maxRotations,@(x)x*(-n:n));

%find rotation matrix
c = cos(angles);
s = sin(angles);
vectorsOut = sum([c;s].*v);
```

```
vectorsOut(2,:) = sum([-s;c].*v);

end

%%%%%%%%% snellLaw.m
function transmissionAngles = snellLaw(incidentAngles,
    incidentIndex,transmissionIndex)
%SNELLLAW Computes the transmission angle given the
    incident angle

transmissionAngles = asin(incidentIndex*sin(
    incidentAngles)/transmissionIndex);


end

%%%%%%%%% vectsangle.m
function angle = vectsangle(v1,v2)
%Given two vectors, computes the angle between them
    using the dot product

angle = acos(norm(v1.*v2)/(norm(v1)*norm(v2)));

end

%%%%%%%%% DateValues.m
%% Materials
%aluminium
aluminium = struct();
aluminium.name = 'aluminium';
aluminium.refractiveIndex = 1.26232 + 7.1855i; %at 0.6
    um

%water
water = struct();
water.name = 'water';
water.density = 1000; %kg/m^3
water.heatCapacity = 4200; %J/kg.K
water.latentHeatEvaporation = 2257e3; %J/kg
water.boilingPoint = 373.13; %K
water.heatCapacityVapor = 2000; %J/kg.K
```

```
%oil
oil = struct();
oil.name = 'oil';
oil.density = 800; %kg/m^3
oil.heatCapacity = 2000; %J/kg.K
oil.heatConductivity = 0.15; %W/m.K
oil.thermalConductivityTable = struct();
oil.thermalConductivityTable.name = '
   thermalConductivity';
oil.thermalConductivityTable.temperature = [293 313 333
    353 373 393 413 433]; %K
oil.thermalConductivityTable.thermalConductivity =
   [0.145 0.144 0.140 0.138 0.137 0.135 0.133 0.132]; %
   W/ m K Thermal conductivity
oil.viscosityTable = struct();
oil.viscosityTable.name = 'viscosity';
oil.viscosityTable.temperature = [290 300 310 320 330
   340 350 400 450 500 550 600 700 800 900 1000]; %K
oil.viscosityTable.viscosity = [901 242 83.9 37.5 20.3
   12.4 8.0 5.6]*10^-6; %Kinematic viscosity m^2/s
oil.prandtlNumberTable = struct();
oil.prandtlNumberTable.name = 'prandtlNumber';
oil.prandtlNumberTable.temperature = [290 300 310 320
   330 340 350 400 450 500 550 600 700 800 900 1000]; %
   K
oil.prandtlNumberTable.prandtlNumber = [10400 2870 1050
    490 276 175 116 84];
oil.densityTable = struct();
oil.densityTable.name = 'density';
oil.densityTable.temperature = [290 300 310 320 330 340
    350 400 450 500 550 600 700 800 900 1000]; %K
oil.densityTable.density = [888 876 864 852 840 829 817
    806]; %kg.m^3

%Freon R-12
freon = struct();
freon.name = 'freon';
freon.density = 1400; %kg/m^3;
freon.heatCapacity = 1000; %J/kg.K
freon.latentHeatEvaporation = 165e3; %J/kg
```

```
freon.boilingPoint = 243; %K
%freon.heatCapacityVapor =

%Ethanol
ethanol = struct();
ethanol.name = 'ethanol';
ethanol.density = 789; %kg/m^3
ethanol.heatCapacity = 2434; %J/kg.K
ethanol.latentHeatEvaporation = 846e3; %J/kg
ethanol.boilingPoint = 351; % K
ethanol.heatCapacityVapor = 1891; %J/kg.K

%Ammonia
ammonia = struct();
ammonia.name = 'ammonia';
ammonia.density = 789; %kg/m^3
ammonia.heatCapacity = 2434; %J/kg.K
ammonia.latentHeatEvaporation = 846e3; %J/kg
ammonia.boilingPoint = 351; % K
ammonia.heatCapacityVapor = 1891; %J/kg.K

%copper
copper.emissivity = 0.88;
copper.thermalConductivity = 60; %W/mK

%air
air = struct();
air.name = 'air';
air.thermalConductivityTable = struct();
air.thermalConductivityTable.name = '
   thermalConductivity';
air.thermalConductivityTable.temperature = [290 300 310
    320 330 340 350 400 450 500 550 600 700 800 900
   1000]; %K
air.thermalConductivityTable.thermalConductivity =
   [0.0253 0.0261 0.0268 0.0275 0.0283 0.0290 0.0297
   0.0331 0.0363 0.0395 0.0426 0.0456 0.0513 0.0569
   0.0625 0.0672]; %W/ m K Thermal conductivity
air.viscosityTable = struct();
air.viscosityTable.name = 'viscosity';
```

```
air.viscosityTable.temperature = [290 300 310 320 330
   340 350 400 450 500 550 600 700 800 900 1000]; %K
air.viscosityTable.viscosity = [1.48 1.57 1.67 1.77
   1.86 1.96 2.06 2.6 3.18 3.8 4.45 5.15 6.64 8.25 9.99
    11.8]*10^-5; %Kinematic viscosity m^2/s
air.prandtlNumberTable = struct();
air.prandtlNumberTable.name = 'prandtlNumber';
air.prandtlNumberTable.temperature = [290 300 310 320
   330 340 350 400 450 500 550 600 700 800 900 1000]; %
   K
air.prandtlNumberTable.prandtlNumber = [0.714 0.712
   0.711 0.710 0.708 0.707 0.706 0.703 0.7 0.699 0.698
   0.698 0.702 0.704 0.705 0.709];
air.thermalDiffusivityTable.name = 'thermalDiffusivity'
   ;
air.thermalDiffusivityTable.temperature = [290 300 310
   320 330 340 350 400 450 500 550 600 700 800 900
   1000]; %K
air.thermalDiffusivityTable.thermalDiffusivity = [2.08
   2.21 2.35 2.49 2.64 2.78 2.92 3.70 4.54 5.44 6.39
   7.37 9.46 11.7 14.2 16.7]*1e-5; %m^2/s

%butane
butane = struct();
butane.name = 'butane';
butane.density = 601; %kg/m^3
butane.heatCapacity = 2.31e3; %J/kg.K
butane.latentHeatEvaporation = 385.2e3; %J/kg
butane.boilingPoint = 272.66; %K
butane.heatCapacityVapor = 1694; %J/kg.K

%% Environments and location
%lahore
lahore.latitude = [31 32 59]; %N
lahore.longitude = [74 20 37];%E
lahore.timezoneLongitude = 75; %E
lahore.monthlyIntensities = [180.96 221.22 238.40
   272.16 281.71 297.76 266.31 246.74 270.63 235.60
   201.71 169.01]*3.155; %kw/m^2
lahore.monthlyTemperatures = [18 22 27 34 39 38 35 34
   34 32 27 21]+273.16; %K
```

```
lahore.windSpeed = 0.05; %m/s
```

References

[1] Matlab 7.8.0.347 (r2009a).

[2] Methods of testing to determine the thermal performance of solar collectors.

[3] T. Sornakumar A. Valan Arasu. Design, manufacture and testing of fiberglass reinforced parabola trough for parabolic trough solar collectors. *Solar Energy*, 81:1237–1279, 2007.

[4] Salman Abasi Fakhar Abbas. Organic rankine cylcle engine suitable for low temperature thermal energy sources. In *Pakistan's Energy Options, Technology, Availability, Affordability and the Role of Regional Grids*, 2012.

[5] H. M. Guven F. Mistree R. B. Bannerot. A conceptual basis for the design of parabolic troughs for different design environments. *Transactions of the ASME*, 108:60–66, 1986.

[6] Halil M. Gaven Richard B. Bannerot. Optical and thermal analysis of parabolic trough solar collectors for technically less developed countries. Technical report, U.S. Agency for International Development, 1984.

[7] Halil M. Gaven Richard B. Bannerot. Determination of error tolerances for the optical design of parabolic troughs for developing countries. *Solar Energy*, 36:535–550, 1986.

[8] John A. Duffie William A. Beckman. *Solar Engineering of Thermal Processes*. John Wiley & Sons, 1980.

[9] Robert H. Cengel, Yunus A. Turner. *Fundamentals of Thermal-Fluid Sciences*. Thomas Casson, 2001.

[10] David K. Cheng. *Field and Wave Electromagnetics*. Tsinghua University Press, 2006.

[11] Wikipedia contributors. Wikipedia: List of solar thermal power stations. 25 may 2012.

[12] CIEMAT. CRES. DLR. FICHTNER Solar. FLABEG Solar. INABENSA (coordinator). SBP. Development of a low cost european parabolic trough collector, eurotrough. Technical report, The European Commission, 2001.

[13] Lifang Li Andres Kecskemethy A. F. M. Arif Steven Dubowsky. A novel approach for designing parabolic mirrors using optimized compliant bands. In *Proceedings of the ASME 2011 International Design Engineering Technical Conferences & Computers and Information in Engineering Conference*, 2011.

[14] P. Bendt et. al. Optical analysis and optimization of line focus solar collectors. Technical report, Solar Energy Research Institute, U.S. Department of Energy, 1979.

[15] Toni Ferreira. Some modifications to the design of a parabolic solar concentrator for construction in lesetho and their effects on power production. Master's thesis, Massachusetts Institute of Technology, 2005.

[16] R. Forristall. Heat transfer analysis and modeling of a parabolic trough solar receiver implemented in engineering equation solver. Technical report, National Renewable Energy Laboratory, 2003.

[17] William Stine. Michael Geyer. *Power From The Sun.* http://www.powerfromthesun.net/.

[18] Zhen Hongfie. Tao Tao. Dai Jing. Kang Huifang. Light tracing analysis of a new kind of solar concentrator. *Energy Conversion and Management*, 52:2373–2377, 2011.

[19] David P. Incropera, Frank P. DeWitt. *Fundamentals of Heat and Mass Transfer.* John Wiley & Sons, 2002.

[20] Nauman Javed. Solar powered sterling engine, a supplementary source of power generation for pakistan. In *Pakistan's Energy Options, Technology, Availability, Affordability and the Role of Regional Grids*, 2012.

[21] Henrik Wann. Jensen. A practical model for subsurface light transport. *SIGGRAPH conference proceedings.*

[22] Nasim Akhter Khan. *Solar Energy.* Ferozsons Rawalpindi Ltd., 1986.

[23] Andrew James Marston. Geometric optimization of solar concentrating collectors using quasi-monte carlo simulation. Master's thesis, University of Waterloo, 2010.

[24] Aden B. Meinel Marjorie P. Meinel. *Applied Solar Energy, An Introduction.* Addison - Wesly Publishing Company, 1976.

[25] C. Christopher Newton. A concentrated solar thermal energy system. Master's thesis, The Florida State University, 2007.

[26] Ricardo Vasquez Padilla. *Simplified Methodology for Designing Parabolic Trough Solar Power Plants*. PhD thesis, University of South Florida, 2011.

[27] Angela M. Patnode. Simulation and performance evaluation of parabolic trough solar power plants. Master's thesis, University of Wisconsion-Madison, 2006.

[28] J Prakash and H. P. Garg. *Solar Energy: Fundamentals and Applications*. Tata McGraw-Hill Education, 2000.

[29] Ahmed Sohail. Solar hybrid water heating systems for industrial applications. In *Pakistan's Energy Options, Technology, Availability, Affordability and the Role of Regional Grids*, 2012.

[30] G. H. Spencer and M. V. R. K. Murty. Genral ray-tracing procedure. *Journal of the Optical Society of America (1917-1983)*, 52:672, June 1962.

[31] Aldo Steinfeld Tom Melchior. Radiative transfer within a cylindrical cavity with diffusely/specularly reflecting inner walls containing an array of tubular absorbers. *Journal of Solar Energy Engineering*, 130:10–20, 2008.

[32] Ucilia Wang. The rise of concentrating solar thermal power.