Modelling and simulation of a solar tower power plant

M. Ewert[1], O. Navarro Fuentes[2]

[1]Master student of computer science at RWTH Aachen University,
Aachen, Germany, matthias.ewert@rwth-aachen.de.
[2]Master student of software systems engineering at RWTH Aachen University,
Aachen, Germany, omnieldis.navarro.fuentes@rwth-aachen.de.

Abstract

In this project a simulation and model for a solar tower power plant are made. The model describes the calculation of the solar position, cosine effects, atmospheric attenuation between heliostat and receiver, heliostat reflectivity and shading and blocking. The model has been done taking into account also a discretization of the surface of the heliostats. This discretization was used to find out how many reflected sample rays from each heliostat intersected the tower's receiver for calculating the total power arriving there. Inspired by the existing pairwise comparison of all heliostats to calculate shading and blocking, an improved approach for this operation is presented. The improvement in computation time using the new method for calculating shading and blocking is significant, specifically for hours very early in the morning and in the afternoon when the shading and blocking effects are more accentuated.

Keywords: Heliostats; Receiver; Shading and Blocking; Solar power; 2-d **tree, raytracing.**

1 Introduction

A model is presented for the calculation of the total solar power arriving at the tower's receiver. Factors that reduce power were taken into account including cosine effects, atmospheric attenuation between heliostat and receiver, heliostat reflectivity and shading and blocking. The computation of cosine effects is made according to Noone, C.J. and Torrilhon 2011 and is described in 3.1. The atmospheric attenuation between heliostat and receiver is shown in 3.4, and the heliostat reflectivity is assumed constant. The calculation of shading and blocking is described in section 5. A discretization of the heliostat's surface was also used to simulate the number of reflected sun's ray samples that reflected from each heliostat toward the tower's receiver.
A software programmed using C++ was created to compute the solar radiation received at the heliostats based on the solar position, the cosine effects, atmospheric attenuation between heliostat and receiver, heliostat reflectivity, shading and blocking effects, and the discretization of the heliostat's surface to finally calculate the total solar power arriving at the tower's receiver. This will be explained in next sections.
The structure of the sections is as follows. Section 2 shows the symbols of the different variables used. Section 3 describes the main aspects of the model used, namely the calculation of the solar position and insolation, cosine effects, atmospheric attenuation and heliostat reflectivity. Section 4 describes the coordinate system used, the process of alignment of the heliostats, the discretization of the heliostats' surface and the receiver intersection point of a sample ray from the heliostat. Section 5 presents a new approach for the computation of shading and blocking and describes the data structure used. Section 6

shows a generalization about the new method for calculating shading and blocking used on the project. Lastly, section 7 describes the results obtained on the simulation and model.

2 Nomenclature

ϕ	latitude, rad
θ	longitude, rad
α_{solar}	solar zenith, rad
γ_{solar}	solar azimuth, rad
α_{normal}	heliostat normal zenith, rad
γ_{normal}	heliostat normal azimuth, rad
η_{cos}	losses due to cosine effects
η_{aa}	losses due to atmospheric attenuation
η_{ref}	losses due to heliostat reflectivitiy
h_{receiver}	tower receiver's height, meter
s_{x}	number of horizontal samples
s_{y}	number of vertical samples
d_{i}	distance to the receiver, meter
$\boldsymbol{\tau}_{\text{solar}}$	solar vector
$\boldsymbol{x}_{\text{field}}$	global x-axis, vector
$\boldsymbol{y}_{\text{field}}$	global y-axis, vector
$\boldsymbol{z}_{\text{field}}$	global z-axis, vector
$\boldsymbol{x}_{\text{heliostat}}$	local heliostat x-axis, vector
$\boldsymbol{y}_{\text{heliostat}}$	local heliostat y-axis, vector
$\boldsymbol{r}_{\text{i}}$	reflective of heliostat i, vector
$\boldsymbol{n}_{\text{heliostat}}$	heliostat's normal
$\boldsymbol{n}_{\text{receiver}}$	receiver's normal
$\boldsymbol{p}_{\text{receiver}}$	position of the tower's receiver, vector
$\boldsymbol{p}_{\text{i}}$	position of the center of heliostat i, vector
$\boldsymbol{p}_{\text{sample}}$	position of a sample ray, vector

3 Model description

One of the main results of the simulation is to get a certain value for the solar power arriving at the tower's receiver. Therefore the computer-aided simulation of a solar power plant requires many computational procedures. Starting with the calculation of the position of the sun according to a geographical location on earth, date and time of day, the sun position as a solar vector $\boldsymbol{\tau}_{\text{solar}}$ can be determined. Another model used within the simulation is the Meteorological Radiation Model (MRM) for estimating the solar radiation. Furthermore, the declination of the heliostats is needed to get the correct reflection angles and then the ray tracer can calculate the sum of the rays being reflected to the solar tower's receiver. In addition there is a consideration of the cosine effects $\eta_{\cos}$, the atmospheric attenuation efficiency η_{aa} and the reflective efficiency η_{ref}. All relevant formulas are described in the following section.

3.1 Solar position

Each geographical position on earth has its coordinates, which are uniquely set by the latitude ϕ and the longitude θ. For example, the city of Aachen in Germany has the geographical coordinates (rounded in decimal degrees): $\phi = 50.78°$, $\theta = 6.08°$.

With reference to the position (latitude ϕ, longitude θ) and the date and time values of the current day, the solar position is given by two angles, the solar zenith α_{solar} and the solar azimuth γ_{solar}. In order to get these two angles, the algorithm for solar positions by the Plataforma Solar de Almeria (PSA) can be used [1]. Because the sun is far away from the earth, the sun rays are assumed to be parallel. The sun ray direction is given by the normalized solar vector

$$\boldsymbol{\tau}_{\text{solar}} = \begin{pmatrix} \sin(\alpha_{\text{solar}}) & \cos(\gamma_{\text{solar}}) \\ \sin(\alpha_{\text{solar}}) & \sin(\gamma_{\text{solar}}) \\ \cos(\alpha_{\text{solar}}) & \end{pmatrix}. \tag{1}$$

3.2 Insolation

The solar radiation can be computed by the Meteorological Radiation Model (MRM) [2]. As solar tower power plants are mostly installed in very sunny regions, the MRM in its first version was implemented, which provides values only for cloudless skies. The MRM is only one of many other existing models which could be used for the simulation.

3.3 Cosine effects

When the sun's rays hit the heliostats and then are reflected to the receiver, energy losses due to cosine effects $\eta_{\cos}$ occur. This effect depends on the solar position and the location of the individual heliostat relative to the receiver. The heliostat surface normal bisects the angle between the solar rays and a line from the heliostat to the tower. The effective reflection area of the heliostat is reduced by the cosine of one-half of this angle. It can easily be calculated using the law of reflection. The scalar product of the solar vector $\boldsymbol{\tau}_{\text{solar}}$ and the heliostat normal $\boldsymbol{n}_{\text{heliostat}}$ is related to the angle of incidence β_{inc} [4], so that

$$\eta_{\cos} = \cos(\beta_{\text{inc}}) = \langle \boldsymbol{\tau}_{\text{solar}}, \boldsymbol{n}_{\text{heliostat}} \rangle. \tag{2}$$

3.4 Atmospheric attenuation efficiency

Another loss of energy takes place due to atmospheric attenuation. This loss depends on the distance d_{i} between a heliostat and the receiver and assumes a visibility of 40 km. It is given by the formula [3]

$$\eta_{\mathrm{aa}} = \begin{cases} 0.99321 - 0.0001176\ d_{\mathrm{i}} + 1.97 \cdot 10^{-8}\ d_{\mathrm{i}}^2 & , d_{\mathrm{i}} \leq 1000m \\ \exp(-0.0001106\ d_{\mathrm{i}}) & , d_{\mathrm{i}} > 1000m \end{cases}. \tag{3}$$

3.5 Heliostat reflectivity

Due to its design each heliostat loses some energy as well. The mirror surface reflects most of the solar energy but also absorbs some of this energy. A standard value for the heliostat reflectivity η_{ref} is 0.88, which means that 88% of the energy is reflected. [4]

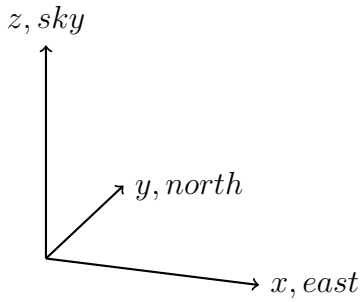


Figure 1: Coordinate system of the simulation.

4 Raytracing

4.1 Coordinate system

In the three-dimensional Cartesian coordinate system, the x-axis is facing from west to east, the y-axis from south to north, the z-axis from the ground to the sky, see figure 1. Thus the axes can be set as normalized three-dimensional vectors

$$\boldsymbol{x}_{\text{field}} = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}, \boldsymbol{y}_{\text{field}} = \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix}, \boldsymbol{z}_{\text{field}} = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}. \tag{4}$$

Each object in the simulation has its own position, which is expressed by a position vector $\boldsymbol{p}$. Hereby the solar tower's receiver is represented as a bounded plane. The center point of the receiver plane is placed on the x-y-center of the coordinate system, with the height h_{receiver}:

$$\boldsymbol{p}_{\text{receiver}} = \begin{pmatrix} 0 \\ 0 \\ h_{\text{receiver}} \end{pmatrix} \tag{5}$$

The same applies for the heliostats H_{i} on the field. Their position vectors $\boldsymbol{p}_{\text{i}}$ describe the center of each heliostat surface.

4.2 Alignment of the heliostats

It is assumed in the model that each heliostat aims towards the center of the receiver aperture. So the reflective vector $\boldsymbol{r}_{\text{i}}$ for each heliostat H_{i} from its surface center to the tower receiver's center is given by

$$\boldsymbol{r}_{\text{i}} = \boldsymbol{p}_{\text{receiver}} - \boldsymbol{p}_{\text{i}} = \begin{pmatrix} -\boldsymbol{p}_{\text{i, x}} \\ -\boldsymbol{p}_{\text{i, y}} \\ \boldsymbol{p}_{\text{receiver, z}} - \boldsymbol{p}_{\text{i, z}} \end{pmatrix}. \tag{6}$$

With the reflective vector $\boldsymbol{r}_{\text{i}}$ and the solar vector $\boldsymbol{\tau}_{\text{solar}}$, it is possible to get the normal of the heliostat $\boldsymbol{n}_{\text{i}}$. The general reflective formula is given by the reflective vector which equals two times the scalar product of the normal vector and the incoming vector (in this simulation the inverted solar vector) minus the incoming vector, see figure 2, so that

$$\boldsymbol{r}_{\text{i}} = 2 \cdot \langle \boldsymbol{n}_{\text{i}}, -\boldsymbol{\tau}_{\text{solar}} \rangle \cdot \boldsymbol{n}_{\text{i}} - (-\boldsymbol{\tau}_{\text{solar}}). \tag{7}$$

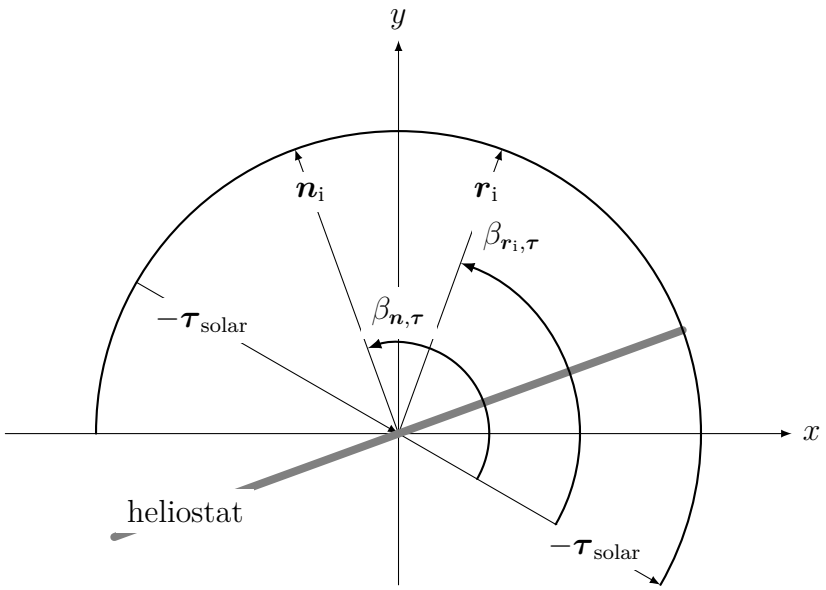


Figure 2: Calculating the normal vector.

The standard formula needs to be solved for the normal vector. Therefore the scalar product of the normal and the incoming vector must be considered and can be rewritten as

$$\langle \boldsymbol{n}_\mathrm{i}, -\boldsymbol{\tau}_\mathrm{solar} \rangle = \cos \beta_{\boldsymbol{n},\boldsymbol{\tau}} = \cos\left(\frac{\pi}{2} + \frac{\beta_{\boldsymbol{r}_\mathrm{i},\boldsymbol{\tau}}}{2}\right) = \cos\left(\frac{\pi}{2} - \frac{\arccos\langle \boldsymbol{r}_\mathrm{i}, -\boldsymbol{\tau}_\mathrm{solar} \rangle}{2}\right), \tag{8}$$

see figure 2. With this rewritten scalar product all values are known and the normal is given by

$$\boldsymbol{n}_\mathrm{i} = \frac{\boldsymbol{r}_\mathrm{i} - \boldsymbol{\tau}_\mathrm{solar}}{2 \cdot \cos\left(\frac{\pi}{2} - \frac{\arccos\langle \boldsymbol{r}_\mathrm{i}, -\boldsymbol{\tau}_\mathrm{solar} \rangle}{2}\right)}. \tag{9}$$

The normal vector $\boldsymbol{n}_\mathrm{i}$ can be expressed as a normal azimuth γ_normal and zenith α_normal by using the spheric coordinate system.

$$\gamma_\mathrm{normal} = \arctan(y, x) = \begin{cases} \arctan\left(\frac{y}{x}\right), & if\ x > 0 \\ sgn(y), & if\ x = 0 \\ \arctan\left(\frac{y}{x}\right) + \pi, & if\ x < 0\ and\ y >= 0 \\ \arctan\left(\frac{y}{x}\right) - \pi, & if\ x < 0\ and\ y < 0 \end{cases} \tag{10}$$

$$\alpha_\mathrm{normal} = \arccos(z) \tag{11}$$

As every heliostat in the simulation has its own position and declination angle, it is quite helpful to define a normalized x-vector $\boldsymbol{x}_\mathrm{heliostat}$ and a normalized y-vector $\boldsymbol{y}_\mathrm{heliostat}$ for each heliostat. The origin of these vectors is always the heliostat's surface center with the x-vector parallel to the horizontal edge of the heliostat and the y-vector parallel to the vertical edge. In order to get these two vectors, the global x-axis $\boldsymbol{x}_\mathrm{field}$ and y-axis

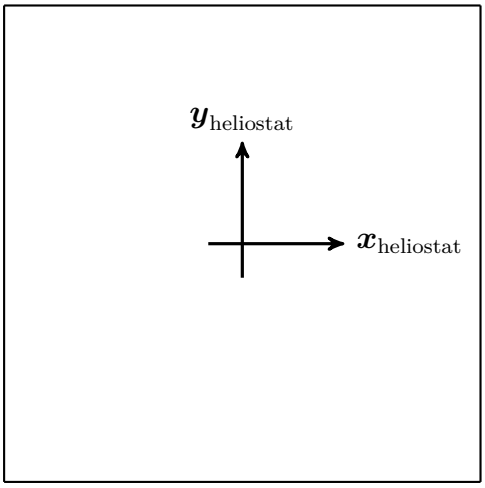


Figure 3: Normal vector axes of the heliostats.

$\boldsymbol{x}_{\text{field}}$ need to be rotated with the normal zenith α_{normal} over the global y-axis $\boldsymbol{y}_{\text{field}}$ and then with the normal azimuth γ_{normal} over the global z-axis $\boldsymbol{z}_{\text{field}}$:

$$\boldsymbol{x}_{\text{heliostat}} = \begin{pmatrix} \cos\gamma_{\text{normal}} & -\sin\gamma_{\text{normal}} & 0 \\ \sin\gamma_{\text{normal}} & \cos\gamma_{\text{normal}} & 0 \\ 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} \cos\alpha_{\text{normal}} & 0 & \sin\alpha_{\text{normal}} \\ 0 & 1 & 0 \\ -\sin\alpha_{\text{normal}} & 0 & \cos\alpha_{\text{normal}} \end{pmatrix} \cdot \boldsymbol{x}_{\text{field}}, \quad (12)$$

$$\boldsymbol{y}_{\text{heliostat}} = \begin{pmatrix} \cos\gamma_{\text{normal}} & -\sin\gamma_{\text{normal}} & 0 \\ \sin\gamma_{\text{normal}} & \cos\gamma_{\text{normal}} & 0 \\ 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} \cos\alpha_{\text{normal}} & 0 & \sin\alpha_{\text{normal}} \\ 0 & 1 & 0 \\ -\sin\alpha_{\text{normal}} & 0 & \cos\alpha_{\text{normal}} \end{pmatrix} \cdot \boldsymbol{y}_{\text{field}}. \quad (13)$$

4.3 Discretization of the heliostats

The heliostat's surface is partitioned into $s_{\text{x}} \cdot s_{\text{y}}$ parts with equal areas. The center point of the heliostat is given by $\boldsymbol{p}_{\text{i}}$ and k and l stand for the k+1-th and l+1-th rectangle in x-direction and y-direction, respectively, see figure 4.3. The position of an individual sample is then calculated by

$$\boldsymbol{p}_{\text{i,k,l}} = \boldsymbol{p}_{\text{i}} - \frac{width}{2} \cdot \boldsymbol{x}_{\text{heliostat}} - \frac{height}{2} \cdot \boldsymbol{y}_{\text{heliostat}} \quad (14)$$

$$+(\frac{1+2k}{2s_{\text{x}}}) \cdot width \cdot \boldsymbol{x}_{\text{heliostat}} \quad (15)$$

$$+(\frac{1+2l}{2s_{\text{y}}}) \cdot width \cdot \boldsymbol{y}_{\text{heliostat}}, \quad (16)$$

The number of rays per heliostat are given by the pre-defined values s_{x} for horizontal samples and s_{y} for vertical samples. If a sample does not hit the receiver due to shading, blocking or not hitting the receiver, the area belonging to this sample is not included in the total power sum.

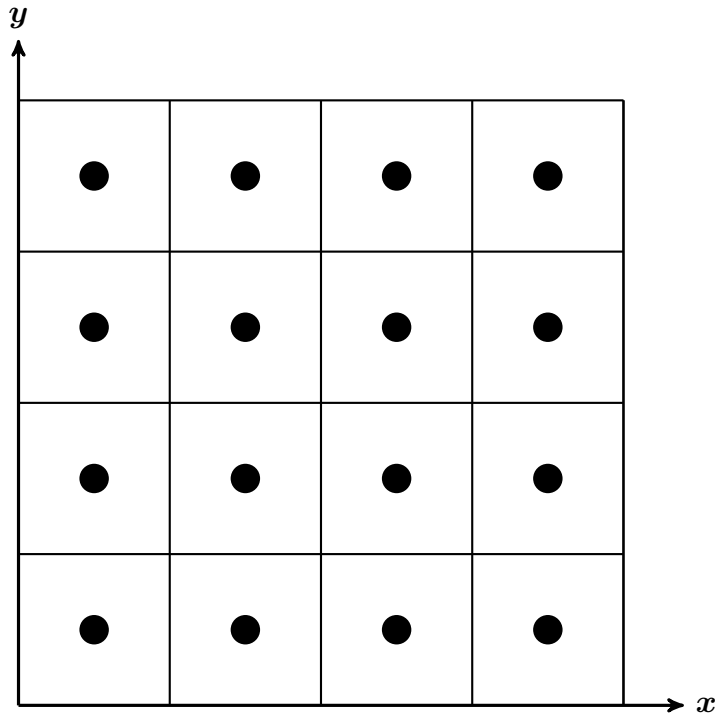


Figure 4: Discretization of the heliostats.

4.4 Tower shadow

The Tower is assumed to be a cuboid consisting out of four planes. Each sample solar ray is checked to determine if it hits one of these four planes and if not, it is included in the further computation.

4.5 Receiver intersection point

It is essential to know if a sample from the heliostat hits the receiver. In order to get this information, the mathematical calculation of a line-plane intersection can be used. Thus the receiver is first defined as a plane by

$$\boldsymbol{n}_{\text{receiver}} \cdot \begin{pmatrix} x \\ y \\ z \end{pmatrix} = \boldsymbol{n}_{\text{receiver}} \cdot \boldsymbol{p}_{\text{receiver}}. \tag{17}$$

For computing the intersection point with the receiver, the sample ray now needs to be tested to determine if there is a scalar q which satisfies

$$\boldsymbol{n}_{\text{receiver}} \cdot (\boldsymbol{p}_{\text{sample}} + q \cdot \boldsymbol{r}_i) = \boldsymbol{n}_{\text{receiver}} \cdot \boldsymbol{p}_{\text{receiver}}, \tag{18}$$

$$\leftrightarrow q = \frac{\boldsymbol{n}_{\text{receiver}} \cdot \boldsymbol{p}_{\text{tower}} - \boldsymbol{n}_{\text{receiver}} \cdot \boldsymbol{p}_{\text{sample}}}{\boldsymbol{n}_{\text{receiver}} \cdot \boldsymbol{r}_{\text{i}}} \tag{19}$$

For the scalar q, the intersection point is then

$$\boldsymbol{p}_{\text{hit, i}} = \boldsymbol{p}_{\text{sample}} + q \cdot \boldsymbol{r}_i. \tag{20}$$

Lastly, this intersection point needs to be tested to determine if it is in the range of real receiver's plane.

4.6 Error cone with Gaussian distribution

The error cone with Gaussian distribution is not considered within the simulation.

5 Shading and Blocking

In every solar power tower the radiation coming from the sun is reflected by heliostats towards a receiver. The efficiency of a solar tower is limited by the interception of the incident sunlight by neighbouring heliostats (shading), and the loss of illumination on the receiver due to the interception of reflected sunlight by other neighbouring heliostats (blocking). The computation of shading and blocking by a pairwise comparison of all heliostats has a time complexity $\mathcal{O}(n^2)$ [4], where n is the number of heliostats. Examples od solar towers with a large number of heliostats are the PS10 and PS20 solar towers, whose fields consist of 624 and 1255 heliostats, respectively. It is obvious that new methods for calculating the shading and blocking effects need to be found in order to reduce these computational costs. We propose to store the heliostats in a 2-d tree. This data structure helps us to reduce the computational costs for shading and blocking. While the subset of potentially shading heliostats must be updated for each time step, due to the motion of the sun, the subset of potentially blocking heliostats does not vary.

5.1 Construction of the 2-d tree

In the x-y plane a heliostat H_i can be represented by a bounding circle with diameter $\sqrt{width^2 + height^2}$. The 2-d tree is constructed as follows. First the median of the global x-components of all heliostats is found. A parallel line to the y-axis is drawn, which intersects the x-axis at this median. With the x-axis split, the heliostats are distributed into two subsets, $\mathcal{H}_{\text{left}}$ and $\mathcal{H}_{\text{right}}$.

- Every heliostat on the left side of the median is included in $\mathcal{H}_{\text{left}}$.
- Every heliostat on the right side of the median is included in $\mathcal{H}_{\text{right}}$
- Every heliostat intersected by the median is included in both $\mathcal{H}_{\text{left}}$ and $\mathcal{H}_{\text{right}}$.

As an example of this process lets consider the following set of heliostats' center coordinates: H_1(1,1), H_2(3,2), H_3(6,5), H_4(8,6), H_5(5,8), H_6(4,4), H_7(4,10). See figures 5 and 6.

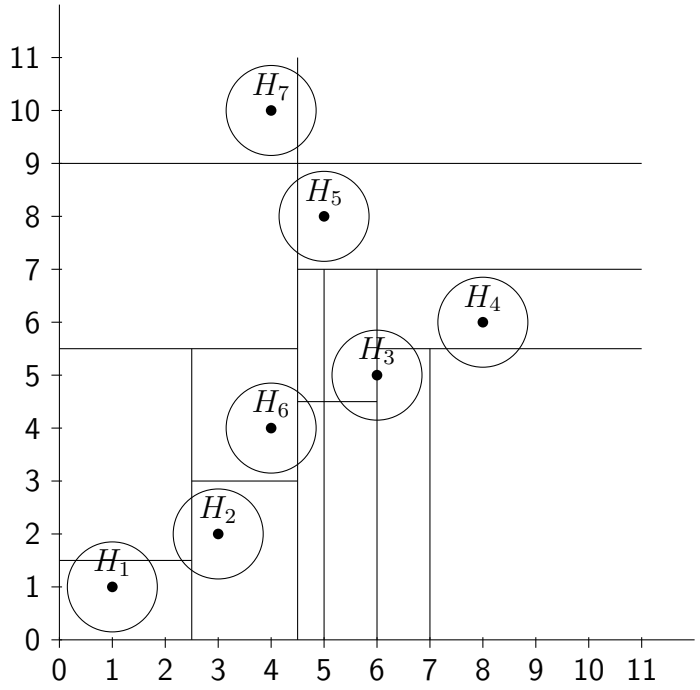


Figure 5: Plane after construction of the 2-d tree.

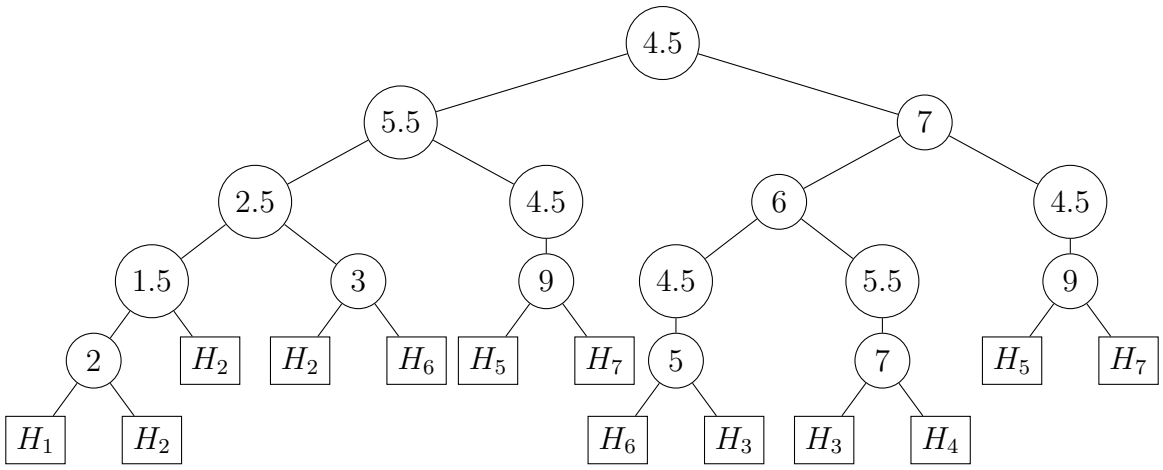


Figure 6: Corresponding tree.

The most expensive step in creating the tree is finding the median. First the set of heliostats $\mathcal{H}$ is sorted ascendently with respect to the x or y coordinate, depending on the level. Even levels are sorted by x-coordinate, and odd ones by y-coordinate. The median is calculated as the half of the sum of the smallest and greatest coordinates, at each level. The heapsort algorithm is used for sorting, which takes $\mathcal{O}(n \cdot \log(n))$. Assuming that the heliostats' x and y center coordinates are distributed uniformly, the tree will be optimal balanced with $\log(n)$ levels. To sort at the i^{th} level, with $\frac{n}{2^i}$ heliostats in each of 2^i sets, it takes time $\mathcal{O}((2^i \cdot \frac{n}{2^i} \cdot \log(\frac{n}{2^i})))$ in each level. The running time for all

the sorting in all levels of the tree is

$$\sum_{i=0}^{\log(n)} n{\cdot}\log\left(\frac{n}{2^i}\right) = n{\cdot}\left(\sum_{i=0}^{\log(n)} \left(\log(n) - \log 2^i\right)\right) \tag{21}$$

$$= n{\cdot}\left(\sum_{i=0}^{\log(n)} \left(\log(n) - i\right)\right) \tag{22}$$

$$= n{\cdot}\left(\sum_{i=0}^{\log(n)} (i)\right) \tag{23}$$

$$= n{\cdot}\left(\frac{\log(n){\cdot}(\log(n)+1)}{2}\right) \tag{24}$$

$$= n{\cdot}\left(\frac{\log(n){\cdot}(\log(n)+\log(2))}{2}\right) \tag{25}$$

$$= n{\cdot}\left(\frac{\log(n){\cdot}\log(2{\cdot}n)}{2}\right) \tag{26}$$

$$\tag{27}$$

Time $=\mathcal{O}\left(n{\cdot}\left(\frac{\log(n){\cdot}\log(2{\cdot}n)}{2}\right)\right)$..

5.1.1 Range search inside the 2-d tree

In a range search about a region delimited by coordinates x_1, x_2, y_1, y_2, the value stored at each intermediate node indicates which subtree should be searched.

- If it is greater or equal than x_2 or y_2, the search continues through the left child of the node.
- If it is smaller than x_1 or y_1, the search continues through the right child.
- If it is between x_1 and x_2, or y_1 and y_2, the search continues through both subtrees.

When a no child nodes is found, its coordinates are compared with those of the region, to check if the heliostat is inside of it. First the center's coordinates of the heliostat are checked. If they are within the region, then the heliostat is included. If they are not within the region then the sum and substraction of the heliostat's center coordinates with the heliostat's radius is made. If at least one of the results of the previous operations is within the region, the heliostat is included. The first check is used to determine if the heliostat center's coordinates are within the region. The second to determine which heliostats are overlapping over the region. A leaf node is found in time $\mathcal{O}(\log(n))$. If the region contains m nodes then for reporting them it takes time $\mathcal{O}(m{\cdot}\log(n))$, which is smaller than $\mathcal{O}(n^2)$ by pairwise comparison.

5.1.2 Improvements on the range search

The range search for shading depends on the solar azimuth angle. In the 2-d tree only reactangular searches are available, so for solar azimuth angles at around 0°, 90° and 180° the search is very efficient. For that reason we may have also cases in which a range search for some heliostats is inefficient, and is not better than a pairwise comparison of all

heliostats. To overcome these worst cases, several 2-d trees are used. The idea is to rotate the heliostats's center coordinates and create a new 2-d tree with the new coordinates. To rotate the coordinates of the heliostats, a rotation matrix R is used. R rotates points in the xy-cartesian plane counterclockwise through an angle θ about the origin of the cartesian coordinate system.

$$R = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{28}$$

A heliostat position vector p_i is rotated to the position:

$$\boldsymbol{p}_i' = R \cdot \boldsymbol{p}_i = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \boldsymbol{p}_i \tag{29}$$

For a total ordered set of angles $\Theta = \{\theta_0, \theta_1, ..., \theta_l\}$ between 0° and 90°, the corresponding 2-d trees $\mathcal{T} = \{T_0, T_1, ..., T_l\}$ were used, dependending on the solar azimuth angle, see table 1.

Solar azimuth mod 90° in range	Search tree	Search direction
$\left[0°, \frac{\theta_0+\theta_1}{2}\right]$	T_0	horizontal
$\left[\frac{\theta_0+\theta_1}{2}, \frac{\theta_1+\theta_2}{2}\right]$	T_1	horizontal
·	·	horizontal
·	·	horizontal
·	·	horizontal
$\left[\frac{\theta_{l\text{-}1}+\theta_l}{2}, 90\right]$	T_l	horizontal
$\left[90°, 90° + \frac{\theta_0+\theta_1}{2}\right]$	T_0	vertical
$\left[90° + \frac{\theta_0+\theta_1}{2}, 90° + \frac{\theta_1+\theta_2}{2}\right]$	T_1	vertical
·	·	vertical
·	·	vertical
·	·	vertical
$\left[90° + \frac{\theta_{l\text{-}1}+\theta_l}{2}, 180°\right]$	T_l	vertical

(30)

Table 1.

5.2 Reducing complexity on shading computation

A new method for the computation of the shading phenomenon is proposed. Every heliostat may be seen as a sphere due to rotationaland turning movement about its center. The number of comparisons of each heliostat is reduced, by finding a subset of heliostats, which could possibly shade the heliostat. The 2-d trees are used to make a rectangular range search of all the heliostats that are placed along the solar vector from. After calculating the subset of heliostats that could possibly shade the heliostat, computations are decreased again by taking into account the diameter of the heliostats and the distance d between both rays of the sun incident on the center of the heliostats.

Extending Belhomme[**?**] to $3D$, If d is smaller than $\sqrt{width^2 + height^2}$ then the heliostat H_j could shade the heliostat H_i, see figures 7 and 8. The distance d can be computed as follows,

$$d = \frac{|\tau_{\text{solar}} \times (\boldsymbol{p_j} - \boldsymbol{p_i})|}{|\tau_{\text{solar}}|} \tag{31}$$

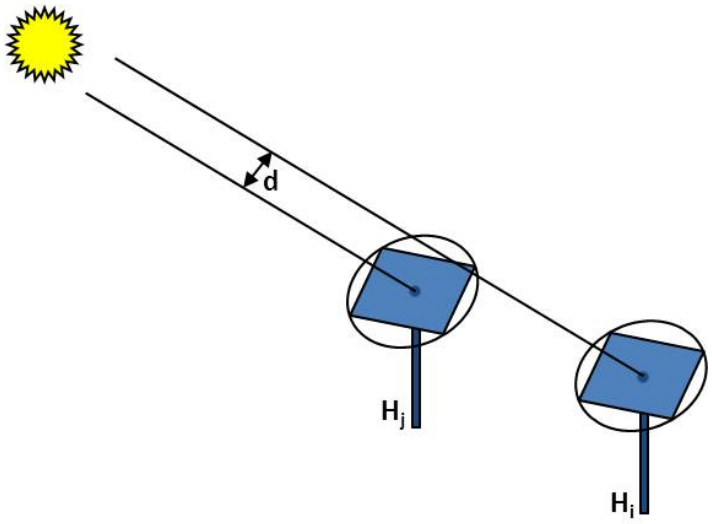


Figure 7: H_j shades H_i.

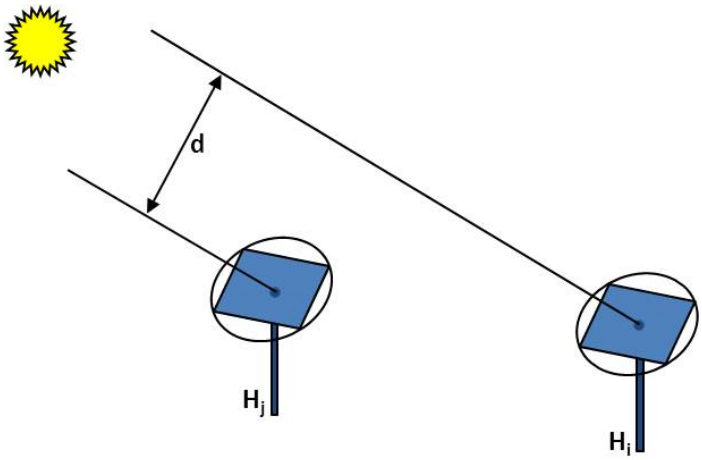


Figure 8: H_j does not shade H_i.

5.3 Blocking

As in 5.2, blocking is computed with the $3D$ extension of Belhomme[**?**]. Here the 2-d tree data structure explained earlier is used. First a subset of the heliostats that are

between the heliostat being analized and the tower is found, using a range search. For searching in the 2-d tree, the angle between the y-axis and the line from the heliostat to the tower is used, instead of the solar azimuth angle, to calculate the heliostats that potentially block the heliostat. The set is then reduced taking into account the distance d between the heliostat H_j and the reflected ray from the center of the heliostat H_i to the receiver. If d is smaller than $\sqrt{width^2 + height^2}$ then the heliostat H_j is included in the set of heliostats that potentially block the heliostat H_i, see figures 9 and 10. The distance d can be computed using the same formula as the shading section, but instead of τ_{solar}, $(\boldsymbol{p}_i - \boldsymbol{p}_{\text{receiver}})$ is used here.

$$d = \frac{|(\boldsymbol{p}_i - \boldsymbol{p}_{\text{receiver}}) \times (\boldsymbol{p}_j - \boldsymbol{p}_i)|}{|\boldsymbol{p}_i - \boldsymbol{p}_{\text{receiver}}|} \tag{32}$$

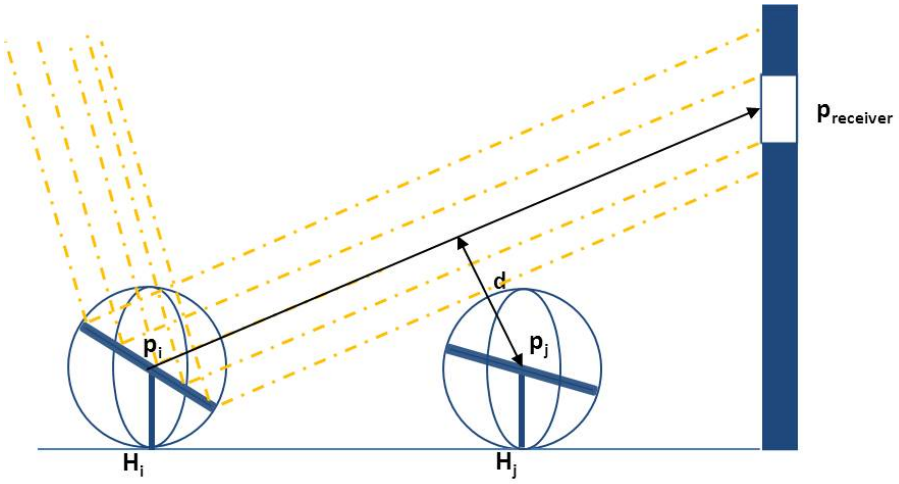


Figure 9: H_j does not block H_i.

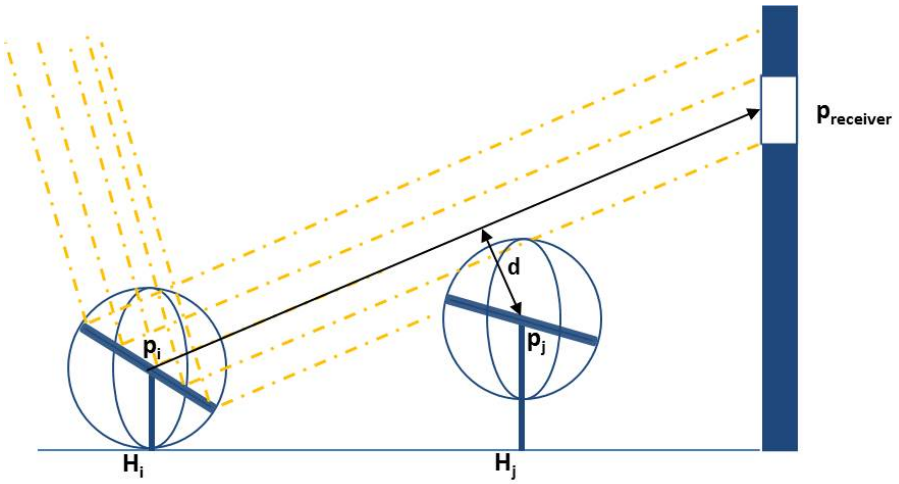


Figure 10: H_j blocks H_i.

6 Observations

In order to reduce the computational time related to blocking and shading of heliostats the 2-d tree is introduced. This data structure decreases the number of pairwise comparisons between heliostats by finding two subsets of potentially shading and blocking heliostats for each heliostat. A range search is made to find these subsets. The size of these subsets may be reduced to a minimum depending of how many 2-d trees are used. Specifically, as the number of 2-d trees increases, the number of elements contained in the range search decreases. This can be proven by observing that when only one 2-d tree is used, there are cases in which, very big rectangles regions appear on the plane. As a solution to this problem, additional 2-d trees with rotated coordinates are used, which depend on the position of the sun. In an ideal scenario with n heliostats, with 0 overlapping heliostats, with $\log(n)$ levels the 2-d related to the heliostats' center coordinates, given a set Θ of different solar azimuth angles, each one of them equidistant, the number of 2-d trees to be used can be described by the next function.

$$\ell = \begin{cases} \sqrt{n} & if\ n < |\Theta|^2 \\ |\Theta| & if\ n >= |\Theta|^2 \end{cases}$$

When the number of heliostats is small, it does not make sense to use many 2-d trees because even in the cases when big rectangles appear on the range searches they will not contain many heliostats. When the number of heliostats is greater, it is necesary to increase the number of 2-d trees because the big rectangles produced by the range searches will have many more heliostats than on the previous case and the objective is to make the rectangles of the range search as small as possible.

7 Results

Tests have been done to compute the total power arriving at the tower's receiver at three different periods of time during the day.

- First, in the morning.
- Second, hours around noon .
- Third, hours around the late afternoon.

These different periods of time have been selected to show the effects of shading and blocking on the total power arriving at the tower's receiver. Different numbers of sample rays have also been shown, see figures 11 to 19. It can be clearly seen that on the hours very early in the morning and very late in the afternoon the degree of shading and blocking is high. For that reason the total power arriving at the tower's receiver is smaller than on hours during close noon, when the shading and blocking is reduced to a minimum value, and greater values of power arrive at the tower's receiver. There is one common result for all the periods of time and it is that using a number of sample rays greater or equal than 256, the values of total power arriving at the tower's receiver are reduced only in an unsignificant value, it almost maintains constant. For that reason 256 may be seen as the upper bound for the number of sample rays to be used.

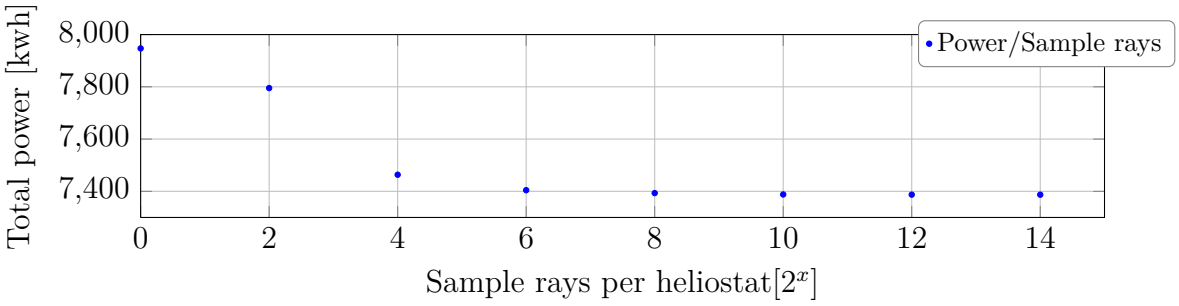


Figure 11: Comparison of total power arriving at tower's receiver from 5 AM to 6 AM.

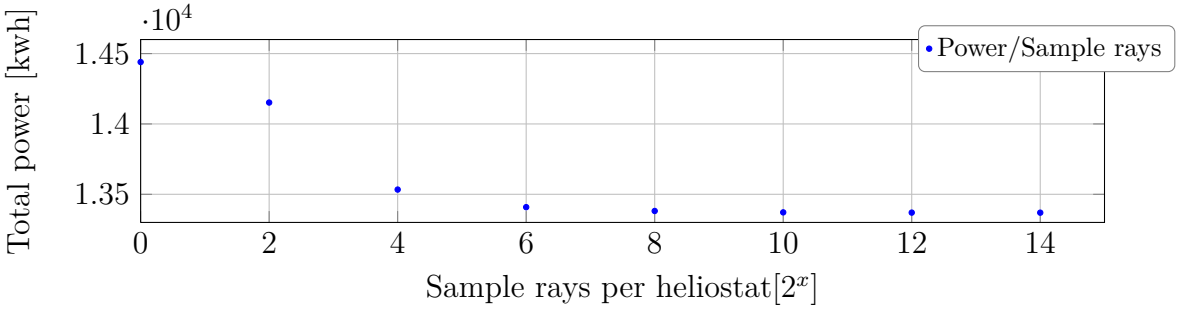


Figure 12: Comparison of total power arriving at tower's receiver from 6 AM to 7 AM.

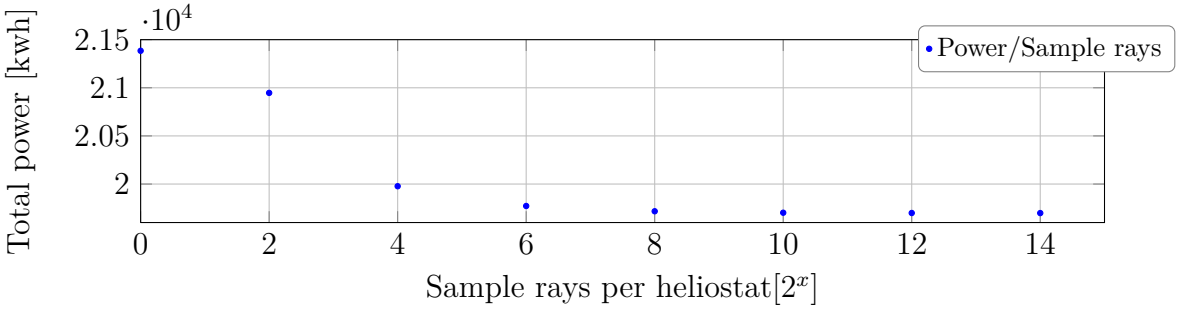


Figure 13: Comparison of total power arriving at tower's receiver from 7 AM to 8 AM.

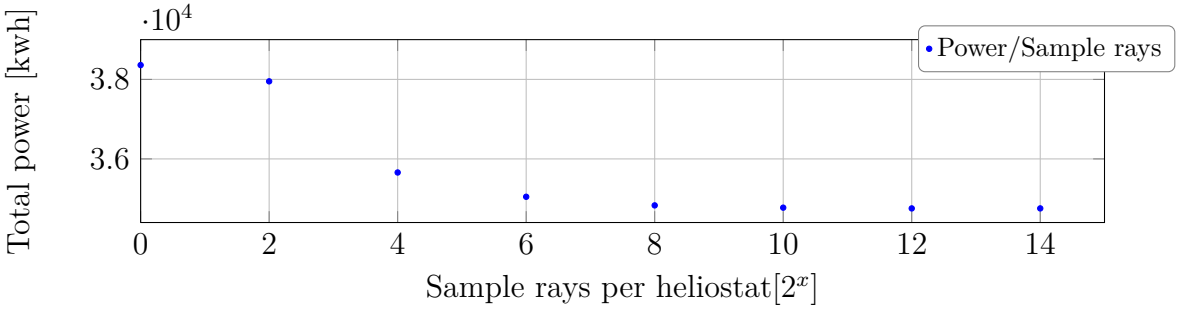


Figure 14: Comparison of total power arriving at tower's receiver from 11 AM to 12.

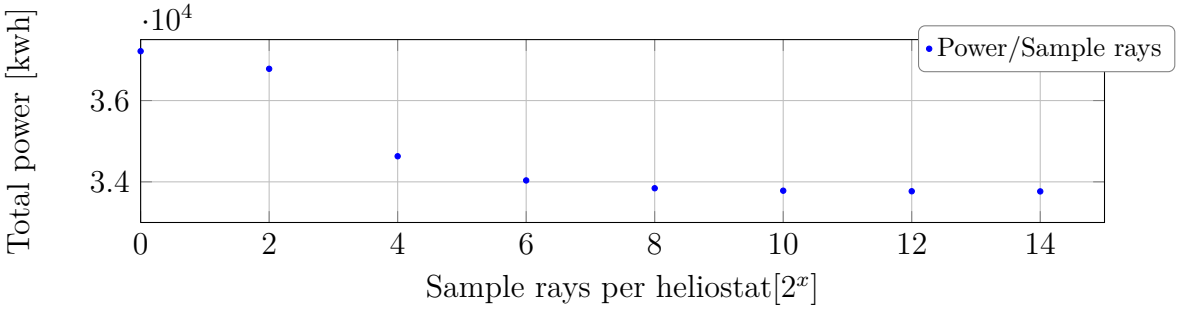


Figure 15: Comparison of total power arriving at tower's receiver from 12 to 1 PM.

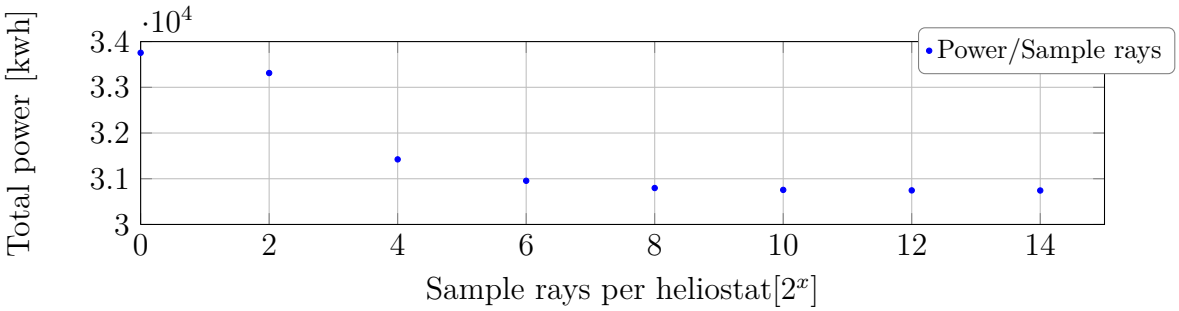


Figure 16: Comparison of total power arriving at tower's receiver from 1 PM to 2 PM.

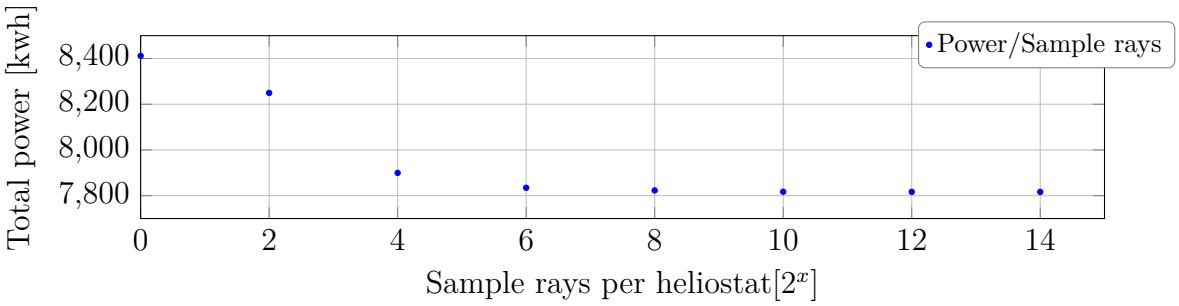


Figure 17: Comparison of total power arriving at tower's receiver from 5 PM to 6 PM.

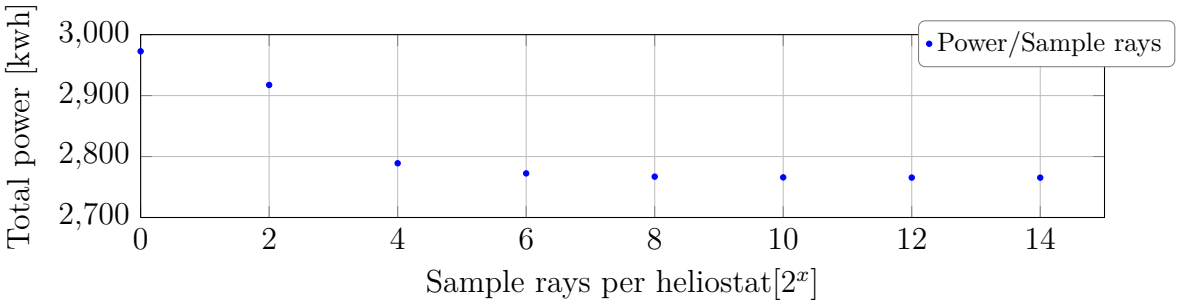


Figure 18: Comparison of total power arriving at tower's receiver from 6 PM to 7 PM.

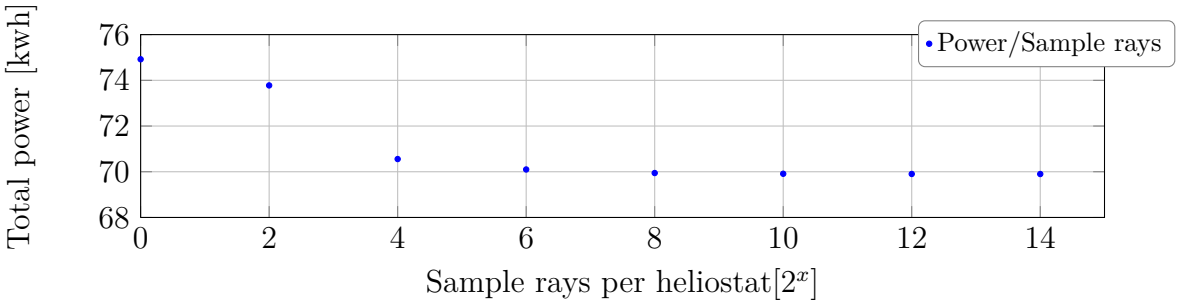
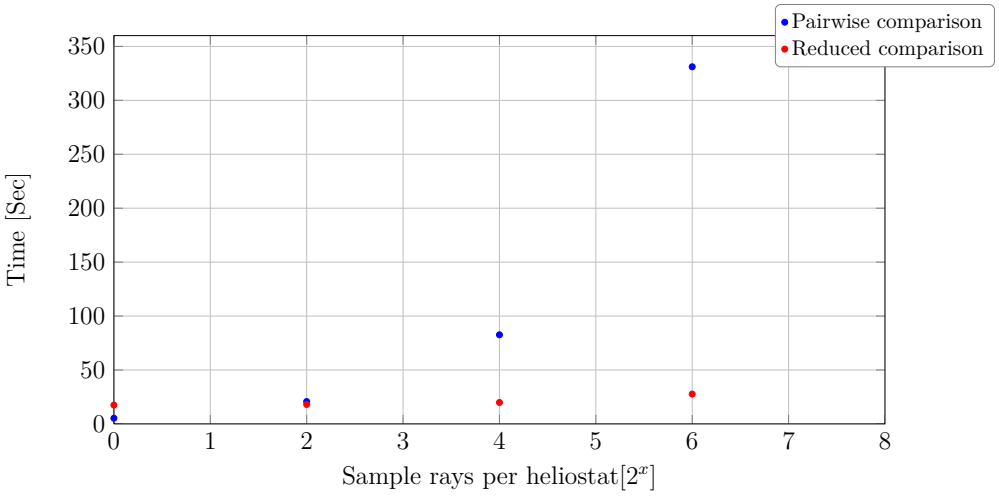
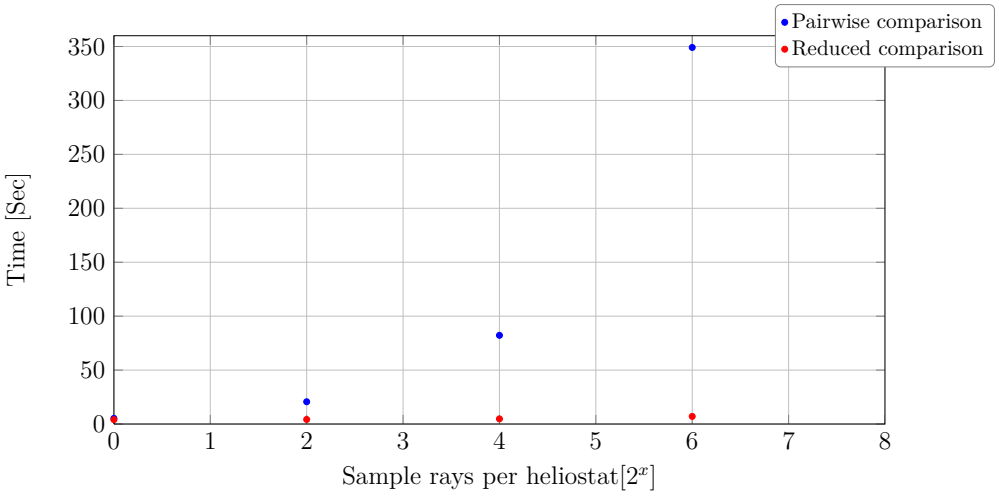


Figure 19: Comparison of total power arriving at tower's receiver from 7 PM to 8 PM.

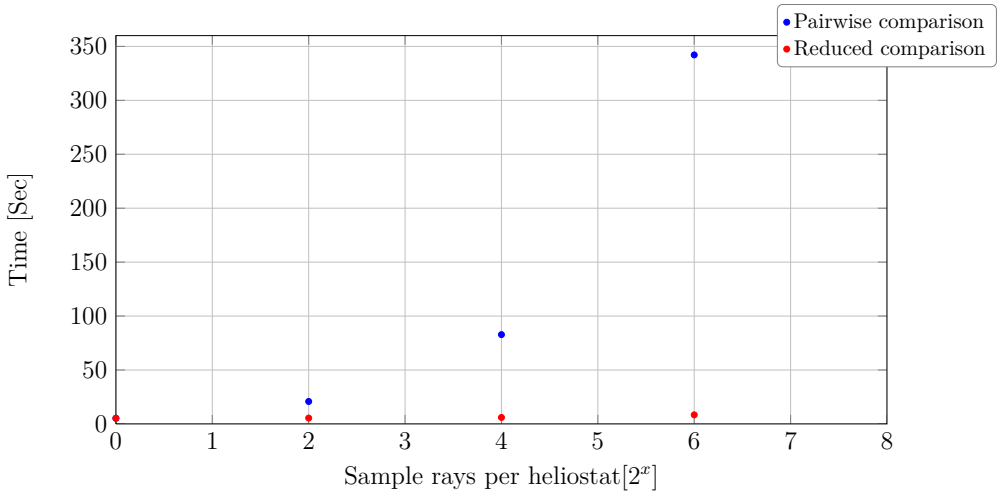
The simulations were executed first using a pairwise comparison of heliostats to find shading and blocking and second using the improved version with reduced number of comparisons. Significant results concerning the time needed for each simulation can be seen in figures 7, 7 and 7. As is clearly seen when the degree of shading and blocking is higher, using the pairwise comparison algorithm the time needed for the whole execution of the simulation is much greater than the time required for executing the whole simulation using the reduced comparison algorithm, see figures 7 and 7. When the degree of shading and blocking is not so high as in previous cases analyzed there is also an improvement on the time required for executing the simulation using the second approach, see figure 7.



Simulation at 05:00.



Simulation at 13:00.



Simulation at 19:00:00 to 19:00.

8 Verification

In the current state of our program, the models and the simulation itself have not been verified yet. For future considerations, the programm could be verified by using other simulation tools, such as SolTRACE.

References

[1] M. Blanco-Muriel, D.C. Alarcón-Padilla, T. López-Moratalla, and M. Lara-Coira. Computing the solar vector. *Solar Energy*, 70(5):431–441, 2001.

[2] V. Badescu. *Modeling solar radiation at the earth's surface: recent advances*. Springer Verlag, 2008.

[3] M. Schmitz, P. Schwarzbözl, R. Buck, and R. Pitz-Paal. Assessment of the potential improvement due to multiple apertures in central receiver systems with secondary concentrators. *Solar energy*, 80(1):111–120, 2006.

[4] C.J. Noone, M. Torrilhon, and A. Mitsos. Heliostat field optimization: A new computationally efficient model and biomimetic layout. *Solar Energy*, 2012.