Theory of Hybrid Systems Informatik 2

RWTH AACHEN UNIVERSITY

Diese Arbeit wurde vorgelegt am
Lehr- und Forschungsgebiet Theorie der hybriden Systeme

Beschleunigte Zielpunktstrategie für Solarturmkraftwerke durch ganzzahlige lineare Optimierung unter Einsatz von Heuristiken

Accelerated aiming strategy for central receiver systems by integer linear optimization using heuristics

Bachelorarbeit
CES

August 2020

Vorgelegt von Presented by	Nils Speetzen Bachstr. 64 52066 Aachen Matrikelnummer: 369066 nils.speetzen@rwth-aachen.de
Erstprüfer First examiner	Prof. Dr. rer. nat. Erika Ábrahám Lehr- und Forschungsgebiet: Theorie der hybriden Systeme RWTH Aachen University
Zweitprüfer Second examiner	Prof. Dr. rer. nat. Thomas Noll Lehr- und Forschungsgebiet: Software Modeling and Verification Group RWTH Aachen University
Externer Betreuer External supervisor	Dr. rer. nat. Pascal Richter Steinbuch Centre for Computing Karlsruhe Institute of Technology

Eigenständigkeitserklärung

Hiermit versichere ich, dass ich diese Bachelorarbeit selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe. Die Stellen meiner Arbeit, die dem Wortlaut oder dem Sinn nach anderen Werken entnommen sind, habe ich in jedem Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht. Dasselbe gilt sinngemäß für Tabellen und Abbildungen. Diese Arbeit hat in dieser oder einer ähnlichen Form noch nicht im Rahmen einer anderen Prüfung vorgelegen.

Aachen, im August 2020

Nils Speetzen

Contents

1 Introduction **1**
- 1.1 State of the art . . . 1
- 1.2 Outline of the work . . . 2

2 Optical model **5**
- 2.1 Sun . . . 5
- 2.2 Heliostats . . . 6
- 2.3 Receiver . . . 9
- 2.4 Computing heliostat heat flux images . . . 14
  - 2.4.1 Original HFLCAL method . . . 14
  - 2.4.2 Extended HFLCAL method . . . 15
  - 2.4.3 Blocking of the receiver . . . 17
- 2.5 Real central receiver systems . . . 20
- 2.6 Heliostat images . . . 22
- 2.7 Allowed flux distribution . . . 24

3 Formulation of the optimization problem **25**
- 3.1 Aiming strategy problem definition . . . 25
- 3.2 Formulation as an ILP . . . 25
  - 3.2.1 Objective function . . . 26
  - 3.2.2 Decision variables . . . 26
  - 3.2.3 Constraints . . . 27

4 Solver software **29**
- 4.1 Branch-and-bound algorithm . . . 29
- 4.2 Solver overview . . . 30
- 4.3 C++ implementation . . . 31
- 4.4 Solver specific ILP model creation . . . 34
- 4.5 Case study for optimal solver configuration . . . 35

5 ILP acceleration using heuristics **39**
- 5.1 Grouping of heliostats . . . 39
- 5.2 Reduction of aim points . . . 46

6 Case study **49**
- 6.1 Heuristics on real CRS . . . 49
- 6.2 Warm start for the computation . . . 52

7 Conclusion and future work **55**

8 Appendix **57**

References **59**

1 Introduction

The importance of renewable energy sources and the efficiency of their utilization has increased significantly over the last few decades. A growing branch of the renewable energy industry is the usage of concentrated solar power (CSP), the worldwide generating capacity of CSP has more than doubled in the last eight years [1].

One approach to using CSP are central receiver systems (CRS). They consist of many mirrors which reflect the solar radiation onto a receiver. This receiver is mounted at the top of a tower and forwards the heat flux to a heat carrying medium, e.g. molten salt, which is transported through tubes behind the surface. Groups of small mirrors are attached to motorized frames, these assemblies are called heliostats and can be controlled by a central control unit to set the target point for the redirected sun beams on the receiver surface. A defining factor for the efficiency of a CRS is the quality of the aiming strategies used to align the heliostats. The goal is to achieve the maximum possible heat flux at the receiver while obliging certain safety constraints. Computing the solution to this problem using current methods takes a considerable amount of time and does not allow for real time corrections to the heliostat alignment to adjust to short-term environmental influences such as cloud shadows.

This work aims to improve the solution process of the heliostat alignment problem and to achieve close to real time performance. In a first, step the developed model is implemented in a solver-independent way, such that different solver implementations can be plugged in and their solving attempts can be evaluated against each other using the same underlying model. After chosing the best solver implementation for this problem, heuristic approximations are implemented and evaluated to reduce the overall runtime even more.

1.1 State of the art

Aiming strategies for CRSs have been subject to previous research, a rough outline of previous research is given below.

Early work approximates the exact solution using heuristics. Dellin et al. [7] propose fast heuristic aiming strategies, which aim to maximize the heat flux hitting the receiver while distributing the aim points of the heliostats in specific patterns. The heuristics are embedded within the context of an optimization software for the design of CRSs. Astolfi et al. [3] break the problem of optimizing the design of CRSs into smaller subproblems and propose heuristics to reduce peak heat flux densities at the receiver by up to 15% compared to other aiming strategies. Solutions are computed within 120 seconds.

Belhomme et al. [4] approach the aiming strategy problem using the ant colony optimization metaheuristic. Using this probabilistic technique, improvements of up to 10% over a reference case, where all heliostats aim at the center of the receiver, can be made within 38 minutes or more, depending on the number of aim points. Receiver constraints, such as a maximum allowed heat flux and spatial heat flux gradient are considered. Besarati et al. [5] use a genetic algorithm to flatten out the heat flux on the receiver surface while minimizing the standard deviation of the heat flux values, such that peak heat fluxes are prevented.

More recent work focuses on finding the exact solution of the aiming strategy problem. Ashley et al. [2] define the optimization problem as an integer linear program (ILP) and achieve, depending on the receiver resolution, close to real time results. Besides from a restricted maximum heat flux at any receiver point, an even heat flux distribution is forced by constraining the allowed range of heat flux on the receiver surface. Furthermore, cloud uncertainties are considered. Richter et al. [21] also work on a robust solution based on the ILP. A desired flux distribution is introduced, which allows to solve for a specific heat flux distribution at the receiver. High receiver resolutions of up to 192 aim points are used and several kinds of uncertainties are taken into account, as well as a limitation to spatial heat flux gradients. Kuhnke et al. [14] extend the work of Ashley et al. [2] and Richter et al. [21] by developing a mixed integer linear program formulation of the aiming strategy optimization including robustness to inaccuracies.

Only Ashley et al. [2] achieves a solution to the ILP in close to real time by using low resolutions and limited constraints, Kuhnke et al. [14] mentiones that the robust ILP model takes up to three hours to reach a gap of 5.86% and proposes heuristics for the robust model to achieve runtimes of about 60 seconds.

This work is based on the works of Kuhnke et al. [14] and aims to provide results in real time that are as close as possible to the optimal result. Instead of approximating a solution based on meta-/heuristics, the intention is to develop an heuristic approach to reduce the solution space of the underlying deterministic ILP in a sensible way such that much shorter runtimes are expected while keeping the gap to the optimal solution as small as possible.

1.2 Outline of the work

In Section 2 an optical model of the central receiver system is developed. This model can be used to compute the heat flux density at any point on the receiver surface for a given heliostat alignment. Based on this optical model, the optimization problem is formulated as an integer linear program (ILP) in Section 3. The formulation considers several safety constraints, e.g. the maximum allowed heat flux density at the receiver surface and the heat shield. The formulation is completely deterministic.

Section 4 covers how the solution to the ILP is computed by using up to five different solvers. The solvers are tuned towards optimal performance on the aiming strategy problem to achieve solutions in lower total runtimes. In Section 5 we perform an extensive investigation to accelerate the ILP solution process using heuristics. Approaches such as grouping heliostats and pre-computing a subset of available aiming points for each heliostat are tested on real szenarios in Section 6.

Lastly, in Section 7 we draw a conclusion about the proposed heuristics and used solver software. Finally, we propose possible future work to extend the research done in this work.

2 Optical model

The optical model presented in this section provides a computational method to determine the heat flux distribution at the receiver caused by every single heliostat. This model is based on the work of Kepp [13].

To obtain the total heat flux distribution at the receiver surface, the individual heat flux distributions from each heliostat, which we call heliostat images, are summed up. To compute these images we use a deterministic approach, which assumes the error cone for the heliostats to be a circular Gaussian distribution.

In the following, the sun, heliostats and receiver will be modeled and the HFLCAL (Heliostat Field Layout CALculation) method for computing heliostat images will be presented in the original and extended version.

2.1 Sun

The position of the sun is expressed using two angles as seen in Figure 1:

- θ_{solar} is the zenith angle. The zenith is the point in the sky directly above the observer, which means the elongation of the z-axis. θ_{solar} lies between 0 and $\pi/2$.
- γ_{solar} is the azimuth angle. This angle describes the clockwise rotation of the sun around the z axis, with $\gamma_{\text{solar}} = 0$ representing a perfect southern orientation. We define the bounds as $\gamma_{\text{solar}} \in [-\pi, \pi]$.

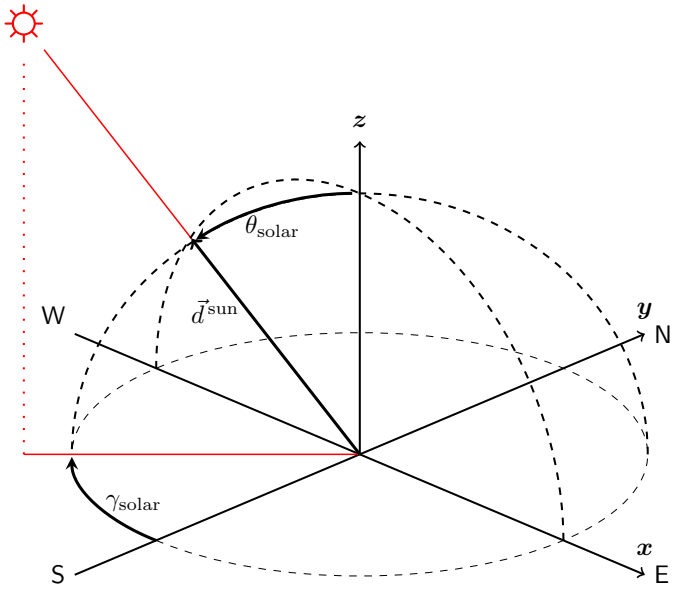


Figure 1: Position of the sun in dependency of the angles γ_{solar} and θ_{solar} [20].

The normalized vector pointing towards the sun now reads as

$$\vec{d}_{\text{sun}} = \begin{pmatrix} -\sin(\gamma_{\text{solar}}) \cdot \sin(\theta_{\text{solar}}) \\ -\cos(\gamma_{\text{solar}}) \cdot \sin(\theta_{\text{solar}}) \\ \cos(\theta_{\text{solar}}) \end{pmatrix}. \tag{1}$$

Central receiver systems make use of the direct radiation of the sun, which consists of the sunrays which are directly hitting the heliostats, while diffuse radiation cannot be utilized.

The intensity of the direct radiation hitting a surface is specified as the DNI $[\frac{\text{W}}{\text{m}^2}]$ (Direct Normal Irradiance). The DNI is measured on a surface perpendicular to the direction of the sunrays. Because these measurements in general happen at ground level, atmospheric effects on the irradiance are already considered.

The intensity I of the radiation hitting a surface depends on the size of its area which is perpendicular to the sunray direction. Since the intensity declines with the cosine of the incidence angle ϕ between the normal vector of the plane and the sunray direction, this effect is called the cosine loss and formalized as

$$I = \text{DNI} \cdot \cos(\phi). \tag{2}$$

Since the sun is not emitting its irradiation from a single point, but from its entire spherical surface, the sun shape error σ^{sunshape} is introduced. It represents small angular deviations of $\vec{d}_{\text{sun}}$, is given by 2.51 mrad [18] and stochastically independent.

2.2 Heliostats

Each heliostat consists of a motorized frame holding one or more small mirrors. The total mirror area of a heliostat can vary between 1m^2 and 140m^2.

The heliostats will be referenced by their indices. The set of heliostats therefore consists of

$$H = \{h_i \mid i \in \{1, ..., n_{\text{heliostats}}\}\} \tag{3}$$

with $n_{\text{heliostats}}$ defining the number of all heliostats.

We call the heat flux a heliostat reflects onto the receiver surface a heliostat (heat flux) image, since it can easily be displayed in image form. The following effects can lead to changes of the heliostat images:

- Shading: If the radiation of the sun does not or only partly hit heliostats due to obstacles being in the way, we speak of shading. The images of the corresponding heliostats are zero or respectively partly zero, as no radiation is reflected from their shaded areas. Shading can be induced by the solar tower itself or by clouds by casting their shadows onto the heliostat field.

Additionally, heliostats can shade each other, when they are aligned in such a way, that those closer to the sun cast a shadow on those further back.

- Blocking: If radiation is reflected by a heliostat, but hits another heliostat on its path to the receiver, the image is also zero at the corresponding area. This effect is known as blocking.

For the sake of simplicity we neglect both shading and blocking in this work.

The incidence angle for the cosine losses, introduced in Section 2.1, can now be computed for the heliostats. The incidence angle $\phi_{h,a}$ describes the angle between the sunray direction $\vec{d}_{\text{sun}}$ and the normal vector of heliostat h aiming at point a. The aiming vector is called $\vec{d}_{h,a}$ and defined as

$$\vec{d}_{h,a} = \vec{p}_a - \vec{p}_h \tag{4}$$

with $\vec{p}_a$ and $\vec{p}_h$ being the position vectors of heliostat h and its aimpoint a. Using this aiming vector and the sunray direction, we can compute the incidence angle as half the angle between $\vec{d}_{\text{sun}}$ and $\vec{d}_{h,a}$ as in Equation (5), since the normal vector $\vec{n}_{h,a}$ is exactly between those two, as shown in Figure 2.

$$\phi_{h,a} = \frac{1}{2} \cdot \arccos\left(\frac{\langle \vec{d}_{\text{sun}}, \vec{d}_{h,a} \rangle}{\|\vec{d}_{h,a}\|_2}\right) \tag{5}$$

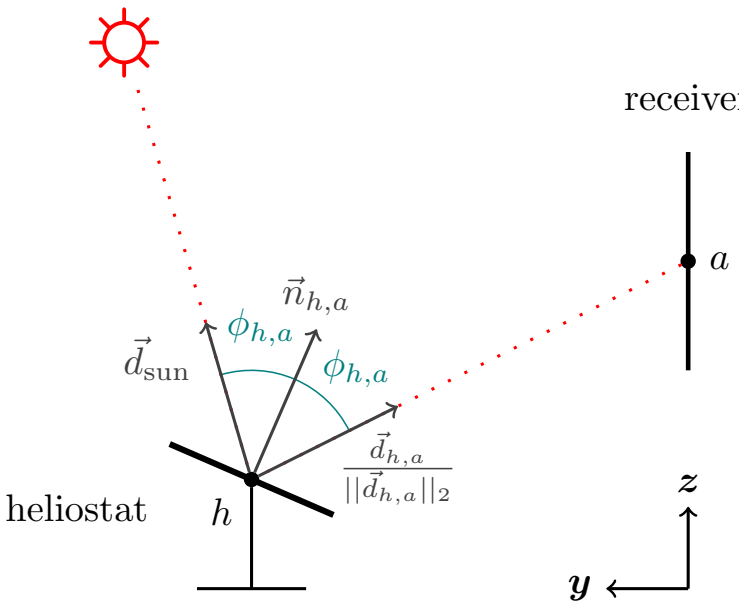


Figure 2: Computation of $\phi_{h,a}$ using $\vec{d}_{\text{sun}}$ and $\vec{d}_{h,a}$. $\vec{n}_{h,a}$ is the normal vector of the heliostats mirror surface.

The total amount of heat a heliostat reflects is called the beam power $P_{h,a}$ [W] and can be calculated as:

$$P_{h,a} = I_{h,a} \cdot \eta_{h,a}^{\text{att}} \cdot A_h \cdot \eta_h^{\text{reflectivity}} = \text{DNI} \cdot \cos(\phi_{h,a}) \cdot \eta_{h,a}^{\text{att}} \cdot A_h \cdot \eta_h^{\text{reflectivity}} \tag{6}$$

with irradiation $I_{h,a}$ from (2). $\eta_{h,a}^{\text{att}}, A_h, \eta_h^{\text{reflectivity}}$ are defined as follows:

The **atmospheric attenuation** $\eta_{h,a}^{\text{att}} \in [0,1]$ describes the effect, that the irradiation is reduced with the distance it has to travel through the atmosphere. The distance is computed as the norm of $\vec{d}_{h,a}$ from equation (4).

$$d_{h,a} = \|\vec{d}_{h,a}\|_2 \tag{7}$$

Depending on the distance, the atmospheric attenuation is computed using approaches of Belhomme et al. [4]:

$$\eta_{h,a}^{\text{att}} = \begin{cases} 0.99321 - 1.176 \cdot 10^{-4} \cdot d_{h,a} + 1.97 \cdot 10^{-8} \cdot d_{h,a}^2 & d_{h,a} \leq 1000[\text{m}] \\ \exp(-1.106 \cdot 10^{-4} \cdot d_{h,a}) & d_{h,a} > 1000[\text{m}] \end{cases} \tag{8}$$

A_h is the **total mirror area** of a heliostat and $\eta_h^{\text{reflectivity}}$ its **reflectivity**. The reflectivity of a surface represents the amount of incoming radiation which is reflected. The reflectivity $\eta_h^{\text{reflectivity}} \in [0,1]$ for the heliostat mirrors is desired to be as close to 1 as possible and given as a constant.

The heliostats do not perfectly reflect the irradiance of the sun onto the desired point but are prone to certain errors. The optical error σ^{optical} is caused by the roughness and small imperfections of the mirror surface.

Since the motor of a heliostat's frame is not perfectly accurate, the heliostats do not always aim exactly where they should. This effect and other related imperfections are summed up in the tracking errors $\sigma^{\text{tracking, hor}}$ and $\sigma^{\text{tracking, vert}}$. The total tracking error [22] is computed as

$$\sigma^{\text{tracking}} = \sqrt{\sigma^{\text{tracking, hor}} \cdot \sigma^{\text{tracking, vert}}}. \tag{9}$$

All errors are assumed to be known and stochastically independent for all $h \in H$. They are not subscripted with h, since they are the same for every heliostat. We combine them into a total error term σ^{total} as follows:

$$\sigma^{\text{total}} = \sqrt{(\sigma^{\text{optical}})^2 + (\sigma^{\text{sunshape}})^2 + (2 \cdot \sigma^{\text{tracking}})^2} \tag{10}$$

2.3 Receiver

The receiver of a central receiver system absorbs the heat flux which is reflected onto it and forwards the energy to a heat carrying medium.

The receiver surface itself naturally has to withstand very high temperatures, but it also has an upper limit to the heat flux it can absorb without taking permanent damage. This upper limit has to be considered for the aiming strategy optimization.

Additionally, the tower on which the receiver is mounted cannot deal with the average heat flux hitting a receiver, which usually varies between 200 to 1000 $\frac{\mathrm{kW}}{\mathrm{m}^2}$ [12]. Therefore, a heat shield protects the area around the receiver from missing heat flux. This heat shield is often made of ceramics and cannot withstand heat flux densities as high as the receiver, since it is not actively cooled. The safety constraints for receiver and heat shield are where the difficulty of the optimization problem stems from.

There exist different receiver designs for different power plant layouts. We focus on the shapes of the so called flat plate, cavity and external receiver's surfaces, which are depicted in Figure 3.

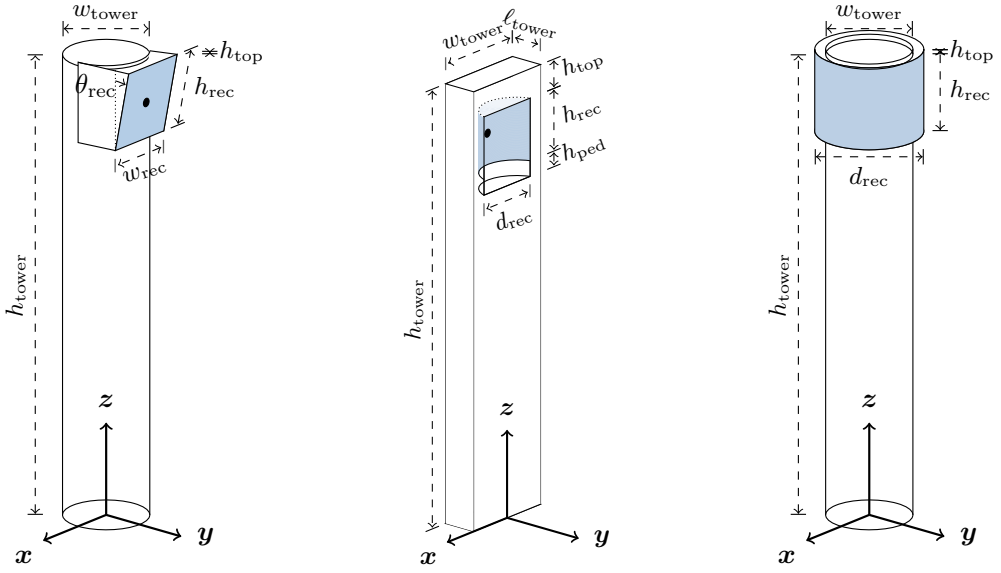


Figure 3: Model of different receiver types, from left to right: Flat plate, cavity and external receiver [20].

The **flat plate receiver** is shaped like a rectangle with width w_{rec} and height h_{rec}. The surface of the receiver is usually tilted towards the ground with angle θ_{rec} to minimize cosine losses through incident angles in vertical direction. The heat shield surrounds the rectangle in every direction and seemingly enlarges the rectangular receiver surface.

The **cavity receiver** is integrated into the tower as a cavity to counteract power losses due to radiation from the receiver itself and air cooling by convection [19]. The receiver surface of a cavity receiver is shaped like the inner surface of half a hollow cylinder with height h_{rec} and radius r_{rec}. In the case of the cavity receiver, the heat shield has the shape of a flat frame surrounding the cavity.

The **external receiver** has a surface, which is shaped like the outside of a hollow cylinder with height h_{rec} and radius r_{rec}. Due to its shape, the heat shield is only attached to its upper and lower edge while following the shape of the receiver.

Parametrization of the different receiver types

In the following the receiver surface is parametrized such that a point in $\mathbb{R}^3$ on the surface can be expressed by coordinates in the range of $(\hat{x}, \hat{y}) \in [0,1]^2 \subset \mathbb{R}^2$. All parameters and the origin of the coordinate system are depicted in Figure 3. The mapping f yields the point coordinates and n the normal vector at the point.

Flat plate receiver

The flat plate receiver has the simplest shape as it consists of one single rectangle. This rectangle can be tilted towards the ground by θ_{rec} to reduce cosine losses at the receiver surface.

$$\begin{aligned} & f : [0,1]^2 \subset \mathbb{R}^2 \to \mathbb{R}^3, \\ & (\hat{x}, \hat{y}) \mapsto \begin{pmatrix} w_{\text{rec}} \cdot (\hat{x} - 0.5) \\ \sin(\theta_{\text{rec}}) \cdot h_{\text{rec}} \cdot \hat{y} + (w_{\text{tower}}/2) \\ \cos(\theta_{\text{rec}}) \cdot h_{\text{rec}} \cdot \hat{y} + (h_{\text{tower}} - h_{\text{top}} - h_{\text{rec}}) \end{pmatrix} \\ & n : [0,1]^2 \subset \mathbb{R}^2 \to \mathbb{R}^3, \\ & (\hat{x}, \hat{y}) \mapsto \begin{pmatrix} 0 \\ \cos(\theta_{\text{rec}}) \\ -\sin(\theta_{\text{rec}}) \end{pmatrix} \end{aligned} \tag{11}$$

Cavity receiver

The cavity receiver consists of a rectangular opening within the tower which contains the cylindrical receiver surface. To reduce blocking losses, the cavity is extended downwards past the receiver surface by h_{ped}. To make evaluation of the heat shield easy, the normal vector of the left and right edges are set to be consistent with the heat shield in that they point directly towards the positive y direction.

Since many cavity receivers consist of discrete panels, which are aligned as a fraction of a regular polygon, the surface is approximated using the same fraction of a cylinder. If the radius of the cylinder is not given, it can be computed using the number of discrete panels n_{panels} which make up the receiver surface and their width w_{panels} as well as the theoretical number of panels in the full polygon $n_{\text{panels, poly}}$ as such:

$$r_{\text{rec}} = \frac{n_{\text{panels, poly}} \cdot w_{\text{panels}}}{2 \cdot \pi} \tag{12}$$

For an easier notation, the following helper variables are introduced:

$$p_{\text{rec}} = \frac{n_{\text{panels}}}{n_{\text{panels, poly}}} \tag{13}$$

$$p_{\text{offset}} = \frac{\frac{1}{2} - p_{\text{rec}}}{2} \tag{14}$$

The coordinates of the receiver surface points and the corresponding normal vectors are given as follows:

$$\begin{aligned}
&f : [0,1]^2 \subset \mathbb{R}^2 \to \mathbb{R}^3, \\
&(\hat{x}, \hat{y}) \mapsto \begin{pmatrix} -r_{\text{rec}} \cdot \cos(2 \cdot \pi \cdot (p_{\text{rec}} \cdot \hat{x} + p_{\text{offset}})) \\ -r_{\text{rec}} \cdot (\sin(2 \cdot \pi \cdot (p_{\text{rec}} \cdot \hat{x} + p_{\text{offset}})) - \sin(2 \cdot \pi \cdot p_{\text{offset}}) \\ h_{\text{rec}} \cdot \hat{y} + (h_{\text{tower}} - h_{\text{top}} - h_{\text{rec}}) \end{pmatrix} \\
&n : [0,1]^2 \subset \mathbb{R}^2 \to \mathbb{R}^3, \\
&(\hat{x}, \hat{y}) \mapsto \begin{cases} \begin{pmatrix} \cos(2 \cdot \pi \cdot (p_{\text{rec}} \cdot \hat{x} + p_{\text{offset}}) \\ \sin(2 \cdot \pi \cdot (p_{\text{rec}} \cdot \hat{x} + p_{\text{offset}}) \\ 0 \end{pmatrix} & \text{if } \hat{x} \in (0,1) \\ \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix} & \text{else} \end{cases}
\end{aligned} \tag{15}$$

External receiver

The external receiver surface can be represented by the mantle of a cylinder, the parametrization cuts the surface at its southern point.

$$\begin{aligned}
&f : [0,1]^2 \subset \mathbb{R}^2 \to \mathbb{R}^3, \\
&(\hat{x}, \hat{y}) \mapsto \begin{pmatrix} -r_{\text{rec}} \cdot \sin(2\pi \cdot \hat{x}) \\ -r_{\text{rec}} \cdot \cos(2\pi \cdot \hat{x}) \\ h_{\text{rec}} \cdot \hat{y} + (h_{\text{tower}} - h_{\text{top}} - h_{\text{rec}}) \end{pmatrix} \\
&n : [0,1]^2 \subset \mathbb{R}^2 \to \mathbb{R}^3, \\
&(\hat{x}, \hat{y}) \mapsto \begin{pmatrix} -\sin(2\pi \cdot \hat{x}) \\ -\cos(2\pi \cdot \hat{x}) \\ 0 \end{pmatrix}
\end{aligned} \tag{16}$$

Receiver heat flux map

The receiver surface is discretized as a grid of points at which the incoming heat flux is evaluated and the safety constraints can be secured. Hence, these points are called measurement points $m_{i,j}$. The set of measurement points on the receiver surface is given as

$$M^{\text{rec}} = \{m_{i,j} \mid i \in \{1, ..., n_m^{\text{horiz}}\}, j \in \{1, ..., n_m^{\text{vert}}\}\} \tag{17}$$

with n_m^{horiz} and n_m^{vert} being the number of measurement points horizontally and vertically respectively.

The coordinates and normal vectors for the discretization are computed by mapping the indices to the previously defined range of $(\hat{x}, \hat{y}) \in [0, 1]^2$. For this, an equidistant discretization is used:

$$\hat{x}_{m_{i,j}} = \frac{i - \frac{1}{2}}{n_m^{\text{horiz}}} \quad , \quad \hat{y}_{m_{i,j}} = \frac{j - \frac{1}{2}}{n_m^{\text{vert}}} \tag{18}$$

In addition to the set of measurement points which lie on the receiver surface M^{rec}, a set of measurement points at its borders M^{shield} is considered for the heat shield safety constraints. The combined set of all measurement points is called M.

$$M = M^{\text{rec}} \cup M^{\text{shield}} \tag{19}$$

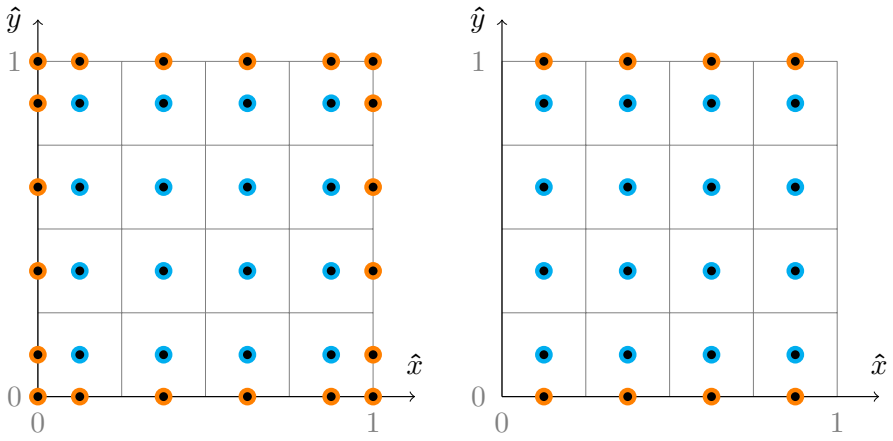
It is possible to only use measurement points for the heat shield which lie on the border of the receiver, since all of the heliostats are only allowed to aim directly at the receiver surface. The heat flux distribution caused by a heliostat gets smaller the further a point is away from its aim point, therefore the total heat flux anywhere on the heat shield is lower or equal to the total heat flux on the edges surrounding the receiver surface. Thus, if the heat flux at the border of the receiver surface is below the maximum allowed heat flux for the heat shield, the safety constraint is not violated.

Using this, we model the heat shield as a set of points directly on the border of the receiver using the same parametrization as above with

$$\hat{x} \in \{0, 1\} \vee \hat{y} \in \{0, 1\} \tag{20}$$

as displayed in Figure 4. Due to the cylindric nature of the external receiver, its heat shield only protects the tower above and below the receiver. Thus, we only add heat shield points in the $\hat{y}$ direction for this type of receiver.

It is worth noting that all safety constraint parameters have to be known at all measurement points. E.g. the maximum allowed heat flux values can be determined using thermodynamic simulations if they are not constant and are assumed to be given in this work.



(a) flat plate and cavity receiver (b) external receiver

Figure 4: Discretization of the receiver surface in $\hat{x}$, $\hat{y}$ coordinates with $n_m^{\text{horiz}} = n_m^{\text{vert}} = n_a^{\text{horiz}} = n_a^{\text{vert}} = 4$. The black dots indicate measurement points with the orange dots being the points added for the heat shield. For the external receiver, the left- and rightmost heat shield points are left out. Blue points are the aim points which can be targeted by heliostats.

Aim points for heliostats

Similar to the measurement points, aim points $a_{i,j}$ are used to discretize possible target points of the heliostats at the receiver surface. The heat flux distribution of a heliostat usually is the highest at its corresponding aim point. The set of aim points is given as

$$A = \{a_{i,j} \mid i \in \{1, ..., n_a^{\text{horiz}}\}, j \in \{1, ..., n_a^{\text{vert}}\}\} \tag{21}$$

with n_a^{horiz} and n_a^{vert} being the number of aim points horizontally and vertically respectively.

Contrary to the measurement points, no additional information about safety constraints has to be known at the aim points and therefore the number of aim points can be chosen arbitrarily, usually the number of aim points is chosen equally to the number of measurement points on the receiver surface:

$$\begin{aligned} n_a^{\text{horiz}} &= n_m^{\text{horiz}} \\ n_a^{\text{vert}} &= n_m^{\text{vert}} \end{aligned} \quad \Rightarrow |A| = |M^{\text{rec}}| \tag{22}$$

This way all the aim points $a \in A$ lie exactly on the measurement points $m \in M^{\text{rec}}$ (see Figure 4) and the heat flux maxima are measured. To fulfill this condition, it is recommended to adapt the resolution of the thermodynamic simulation to the desired number of aim points or, if the resolution is fixed, adapt the number of aim points accordingly.

2.4 Computing heliostat heat flux images

The heliostat heat flux image for heliostat h aiming at an aimpoint a shows the incoming heat flux $q_{h,a}^m$ $[\frac{\mathrm{W}}{\mathrm{m}^2}]$ for each measurement point m. In this section we present two methods to approximate $q_{h,a}^m$.

The heat flux images can also be used to obtain the power $P_{h,a}^m$ [W] absorbed by an area around a measurement point m. Using the two-dimensional midpoint quadrature rule, the heat flux can be integrated over the measurement point area A^m. This equals to

$$P_{h,a}^m = q_{h,a}^m \cdot A^m. \tag{23}$$

These power values can be summed up over all measurement points to receive the total power

$$P_{h,a}^{\mathrm{rec}} = \sum_{m \in M^{\mathrm{rec}}} P_{h,a}^m. \tag{24}$$

2.4.1 Original HFLCAL method

The HFLCAL (Heliostat Field Layout CALculation) tool [22] offers an analytic way to calculate the heat flux. It approximates the heat flux $q_{h,a}^m$ as

$$q_{h,a}^m = \frac{P_{h,a}}{2\pi\sigma_{h,a}^{\mathrm{eff}}} \cdot \exp\left(-\frac{(x_a^m)^2 + (z_a^m)^2}{2\sigma_{h,a}^{\mathrm{eff}}}\right) \cdot \left[\tfrac{1}{\mathrm{m}^2}\right] \tag{25}$$

- $(x_a^m)^2 + (z_a^m)^2$ equals the squared distance from the measurement point m to the aim point a in the x-z plane.
- $P_{h,a}$ is the beam power from (6) in Section 2.2.
- $\sigma_{h,a}^{\mathrm{eff}}$ is the effective error defining the distribution size.

The effective error $\sigma_{h,a}^{\mathrm{eff}}$ scales with the aim distance $d_{h,a}$ from (7):

$$\sigma_{h,a}^{\mathrm{eff}} = \frac{d_{h,a} \cdot \sigma^{\mathrm{total}}}{\sqrt{\cos(\phi_{h,a})}} \tag{26}$$

with $\phi_{h,a}$ being the incidence angle between the normal vector of the aim point $\vec{n}_a$ and the vector from the aim point to the heliostat $-\vec{d}_{h,a}$ in the x-y plane. σ^{total} describes the total error of the heliostats as defined in Equation (10).

2.4.2 Extended HFLCAL method

In the original HFLCAL method, the cosine losses of non-perpendicular beams are only included in the effective error term from Equation (26). The biggest problem of this approach is the limitation to only influence the overall magnitude of the heat flux image but not its shape, since it is a constant factor for the entire image. In order to get a more accurate representation, these losses can be computed separately by projecting the surface of the receiver onto a plane perpendicular to the beam.

This concept has been implemented by Kepp [13] and in this work we improve the approach to be more computationally efficient.

The idea of the extended HFLCAL method is to project any measurement point onto the plane which is perpendicular to $\vec{d}_{h,a}$ and anchored at a. On this plane the heat flux is given by the original HFLCAL method as in Equation (25) using the distances within this plane. To compute the heat flux around measurement point m, the area around m and m itself are projected onto the perpendicular plane, as displayed in Figure 5. The heat flux at the projected point m_o and the projected area A_o^m are then used to calculate the heat flux at m.

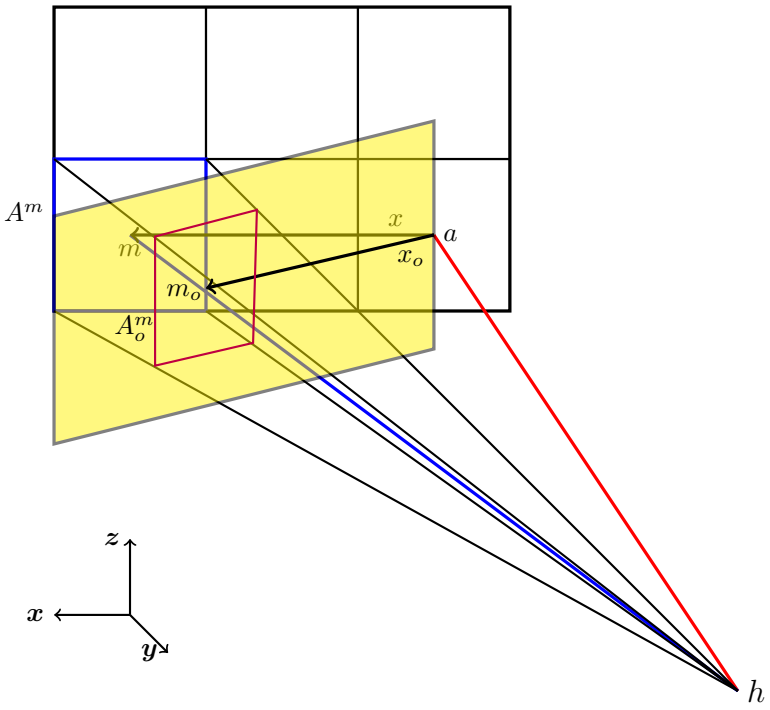


Figure 5: Projection of the measurement point area onto the plane perpendicular to the aim vector [13]. m and the associated area A^m are projected onto m_o and A_o^m. The projected distance x_o between a and m_o is used for the heat flux computation with the original HFLCAL method.

.

In comparison to Equation (26) the effective error is now only defined as

$$\sigma_{h,a}^{\text{eff}} = d_{h,a} \cdot \sigma^{\text{total}}. \tag{27}$$

To project any measurement point m onto the original plane, the vector from the heliostat to that point $\vec{d}_{h,m}$ is scaled such that the projection of the vector onto $\vec{d}_{h,a}$ has the length of $d_{h,a}$, which is visualized in Figure 6.

$$\vec{d}_{h,m_o} = \vec{d}_{h,m} \cdot \frac{d_{h,a}}{\langle \vec{d}_{h,m}, \vec{d}_{h,a} \rangle} \tag{28}$$

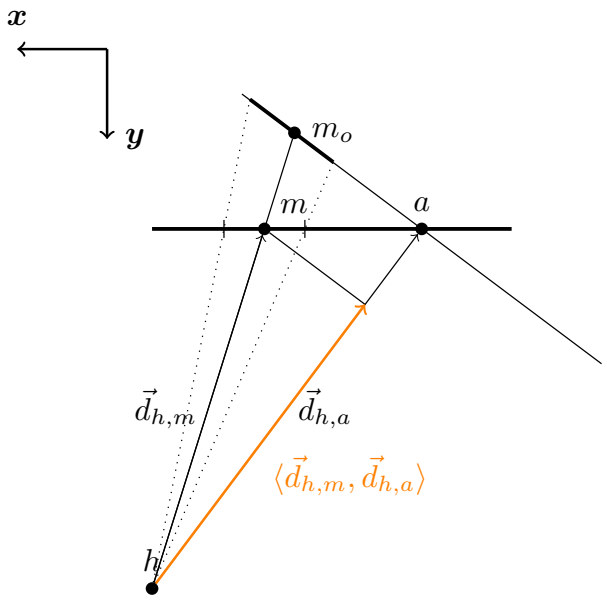


Figure 6: Measurement point m is projected onto the plane perpendicular to the aim vector. The Projection m_o is obtained by scaling vector $\vec{d}_{h,m}$ with the length-ratio of $\vec{d}_{h,a}$ to $\langle \vec{d}_{h,m}, \vec{d}_{h,a} \rangle$.

The projected point m_o is determined by adding $\vec{d}_{h,m_o}$ to the position of the heliostat, A_o^m is calculated by projecting the corners of A^m. For this, the virtual measurement points $m_{i+\frac{1}{2},j+\frac{1}{2}}$, $m_{i-\frac{1}{2},j+\frac{1}{2}}$, $m_{i+\frac{1}{2},j-\frac{1}{2}}$ and $m_{i-\frac{1}{2},j-\frac{1}{2}}$ are projected. The coordinates of these points can be computed with the parametrization from Section 2.3. The cross product $\times$ of the differences vectors of these points can be used to obtain A_o^m, since they are all within the same plane. For this, the area is split into two triangles as seen in Figure 7.

$$\begin{aligned} A_o^m =& \frac{1}{2} \cdot \| (m_{i+\frac{1}{2},j+\frac{1}{2}} - m_{i+\frac{1}{2},j-\frac{1}{2}}) \times (m_{i-\frac{1}{2},j-\frac{1}{2}} - m_{i+\frac{1}{2},j-\frac{1}{2}}) \|_2 \\ &+ \frac{1}{2} \cdot \| (m_{i+\frac{1}{2},j+\frac{1}{2}} - m_{i-\frac{1}{2},j+\frac{1}{2}}) \times (m_{i-\frac{1}{2},j-\frac{1}{2}} - m_{i-\frac{1}{2},j+\frac{1}{2}}) \|_2 \end{aligned} \tag{29}$$

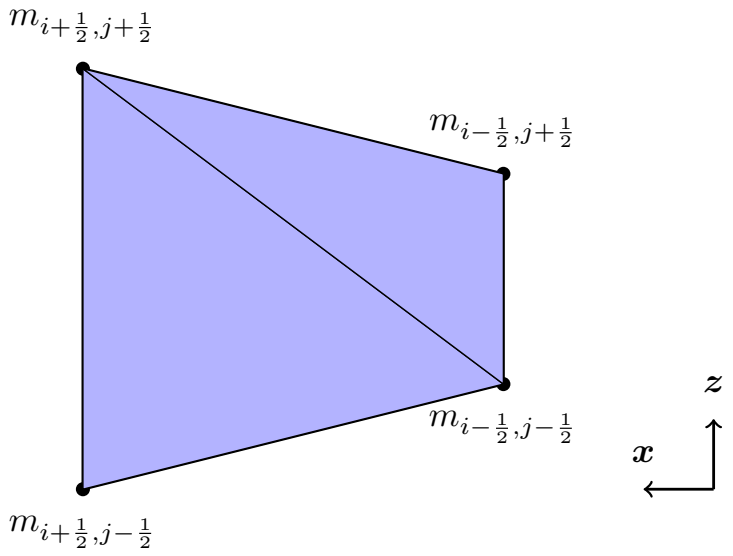


Figure 7: Visualization of the area computation using the projections of the corners of an area. The area is split into two triangles, their size can be easily computed using the cross product.

To finally scale the heat flux at a point correctly, the ratio of the projected area to the original area has to be included as a factor. This yields the following equation for the extended HFLCAL method:

$$q^m_{h,a} = \frac{P_{h,a}}{2\pi\sigma^{\text{eff}}_{h,a}} \cdot \exp(-\frac{(d^{m_o}_a)^2}{2\sigma^{\text{eff}}_{h,a}}) \cdot \frac{A^m_o}{A^m} \cdot \left[\frac{1}{\text{m}^2}\right] \tag{30}$$

The term $\frac{A^m_o}{Am}$ can be seen as a projection factor for the heat flux. Since the measurement points at the edges of the receiver have no associated areas, the projection factor is replaced with 1 and $\sigma^{\text{eff}}_{h,a}$ is computed as in Equation (26). This equals the intensity scaling done in the original HFLCAL method, with the difference being that m is still projected to m_o.

2.4.3 Blocking of the receiver

In reality some parts of the receiver can occlude others such that these do not receive any heat flux from a heliostat. We model this by checking whether a point on the receiver surface is facing the heliostat and whether the point is blocked by the receiver itself. In the following the methods to check for these cases are described. If a measurement point is within a blocked area, the heat flux measured at that measurement point will be assumed to be zero and blocked aim points cannot be targeted by heliostats.

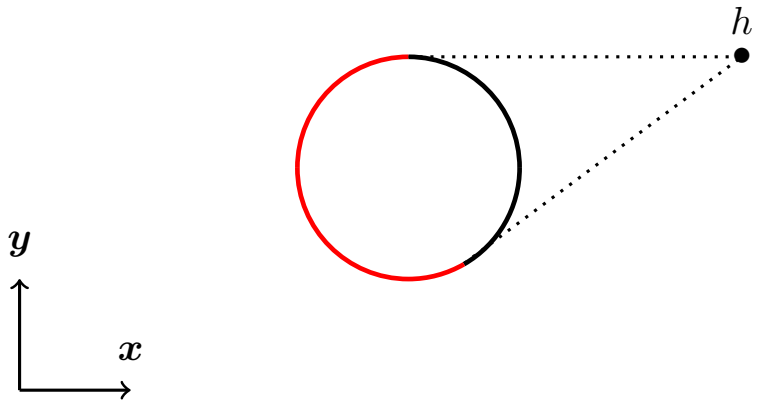


Figure 8: Points not facing a heliostat cannot receive any heat flux from its reflected radiation. The figure depicts a cut of the external receiver tower with a heliostat h positioned in such a way that the red area lies in shadow.

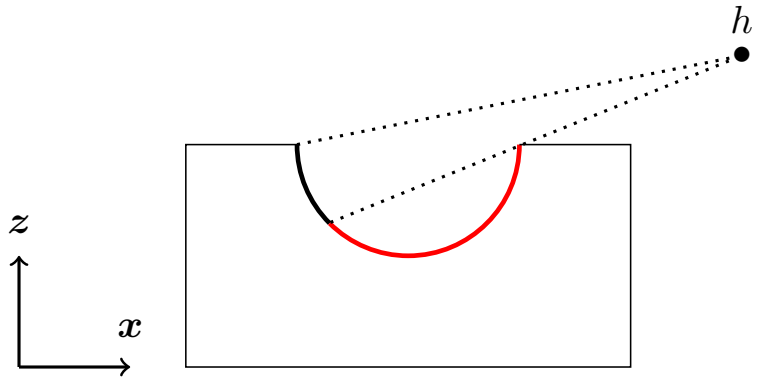


Figure 9: The cavity receiver's shape makes cast shadows possible. The figure depicts a cut of the cavity receiver tower with a heliostat h positioned in such a way that the red area lies in shadow.

First, the normal vector at the measurement point $\vec{n}_m$, defined in Section 2.3, will be used to check wether the receiver surface is facing the heliostat at this measurement point. The point is facing away and therefore will not receive any heat flux if the scalar product is positive,

$$\langle \vec{n}_m, \vec{d}_{h,m} \rangle > 0,$$

where $\vec{d}_{h,m}$ is the vector from the heliostat to the measurement point. This is illustrated in Figure 8 for the external receiver.

Secondly, the cavity receiver offers the possibility to cast a shadow onto measurement points from the sides of the cavity, as depicted in Figure 9.

We use the lower left and upper right corners ll and ur of the rectangular cavity to check for this by comparing the entries of the cross-product $\times$ of the aim vector and the corner points:

$$((\vec{d}_{h,m} \times \vec{d}_{h,ll})_x > 0) \neq ((\vec{d}_{h,m} \times \vec{d}_{h,ur})_x > 0) \tag{31}$$

$$((\vec{d}_{h,m} \times \vec{d}_{h,ll})_z > 0) \neq ((\vec{d}_{h,m} \times \vec{d}_{h,ur})_z > 0) \tag{32}$$

This inequality holds for any measurement point which is not in the shadow, as the irradiation passes between the lower left and upper right corners of the cavity, which can be seen in Figure 10.

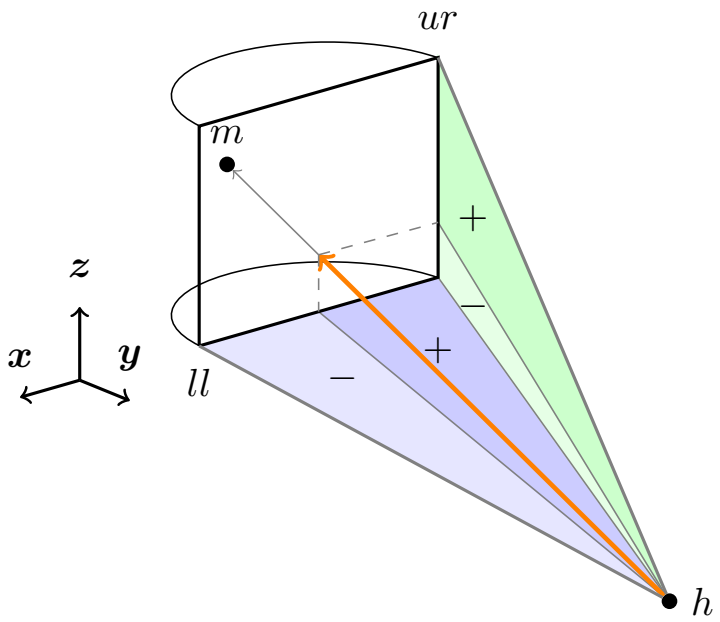


Figure 10: To check whether the ray from the heliostat h to the measurement point p is blocked by the receiver cavity, the signs of the cross-product of the vector $\vec{d}_{h,m}$ with the corners of the cavity is compared. If the signs for the x and z component switch, the ray is not blocked.

To minimize the losses by blocking, the cavity of the receiver can be extended further downwards, since the lower edge of the cavity would almost always cast a shadow onto the receiver surface. The height of this extension is called h_{ped} and can be seen in Section 2.3, Figure 3.

To ensure that the heatflux on the heat shield is never greater than on the edges of the receiver, we only allow the heliostat to aim for the aim points which are in line of sight of the heliostat. The blocked aim points are different for each heliostat and the corresponding subsets of A are called A_h^{visible} and A_h^{blocked}.

2.5 Real central receiver systems

To define testcases for this work, we use real central receiver systems (CRS) as a reference. All parameters are derived directly from the plant specifications to grant the most realistic behavior. A list of all parameters for the testcases can be found in Appendix, Table 4.

PS10

The Planta Solar 10 (PS10) [19] is a power plant in Spain, which was developed by Abengoa Solar and started operating in 2007. The CRS consists of 624 heliostats and a cavity receiver. The cavity receiver of PS10 consists of four rectangular panels which form the curved receiver surface. The panels are aligned as four sides of a nonagon and the total electrical power the plant produced adds up to 11MW.

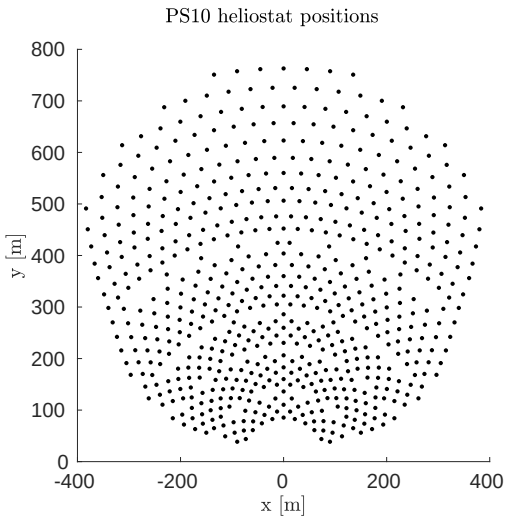
Gemasolar

The Gemasolar Thermosolar Plant [6], developed by Torresol Energy, is a significantly larger power plant than PS10 with 2650 heliostats spread around a cylindric external receiver. It produces 19.9MW of power and started operating in April 2011.

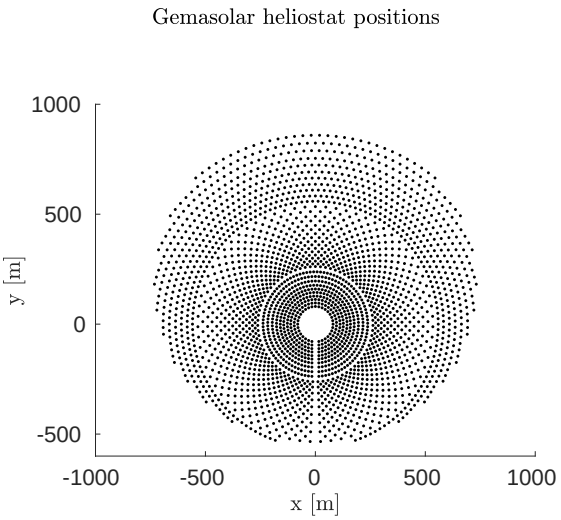
New Abengoa CRS

In comparison to PS10 and Gemasolar this CRS is still in planning. It has 8600 heliostats in total and uses a cylindric external receiver. We refer to this CRS as the Abengoa CRS, since it does not have an official name yet. For this CRS, a set of 1777 aim points is already given, which we use instead of the aim points from Section 2.3.

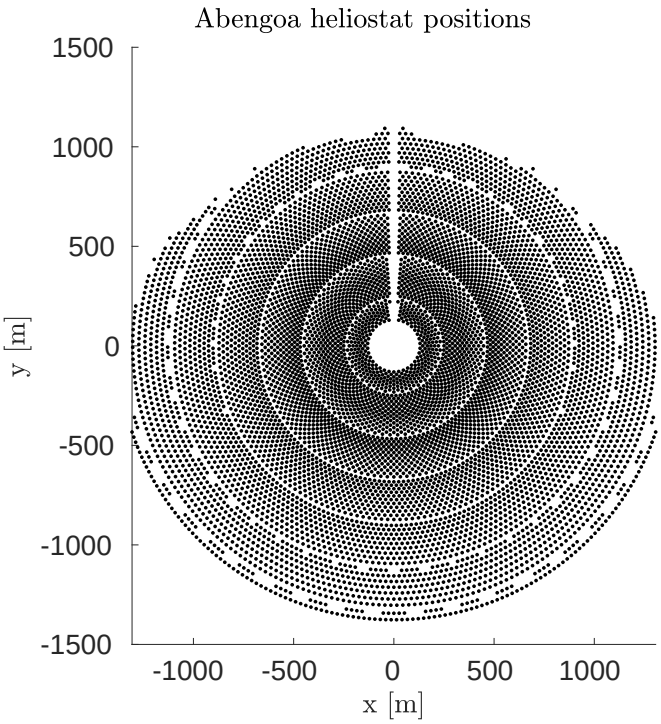
A top view of all three CRS heliostat layouts can be seen in Figure 11.



(a) Layout of the PS10 CRS.



(b) Layout of the Gemasolar CRS.



(c) Layout of the Abengoa CRS.

Figure 11: Layouts of the real central receiver systems. The receiver tower is always positioned at $(x, y) = (0, 0)$.

2.6 Heliostat images

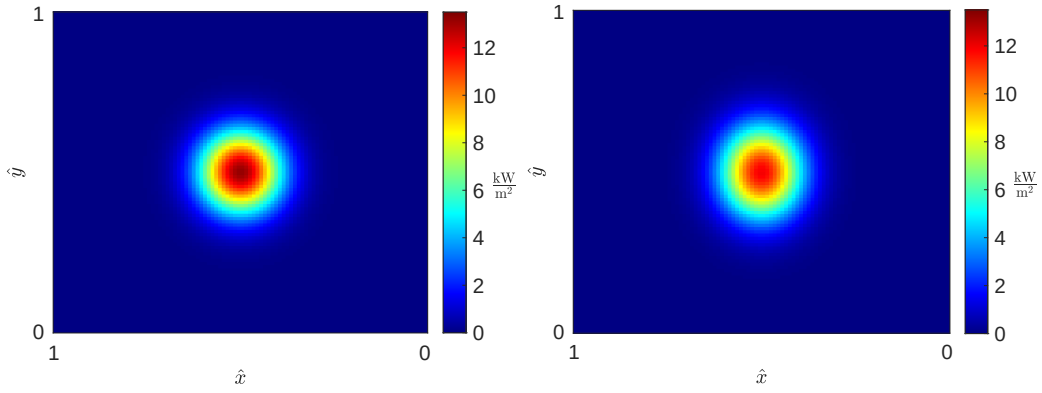
To visualize the functionality of the optical model, we take a look at heat flux images of a single heliostat for a flat plate and a cavity receiver. For the flat plate receiver we additionally look at two different heliostat positions. In all cases we use the original and the extended HFLCAL methods for the image computation. To stay close to reality, we choose our problem parameters according to the ones used in the PS10 CRS (see Appendix Table 4).

Figures 12a and 12b show the heat flux image for a heliostat positioned directly in front of the receiver. For these examples the heliostat aims at the center of the receiver, which can be verified by the central position of the peak of the heat flux. As expected, the heat flux decreases radially around the peak.

Comparing Figures 12a - 12d with each other shows two things. The heat flux arriving at the receiver becomes smaller, the further the heliostat is away from the receiver and the extended HFLCAL method produces much more believable results for diagonally placed heliostats.

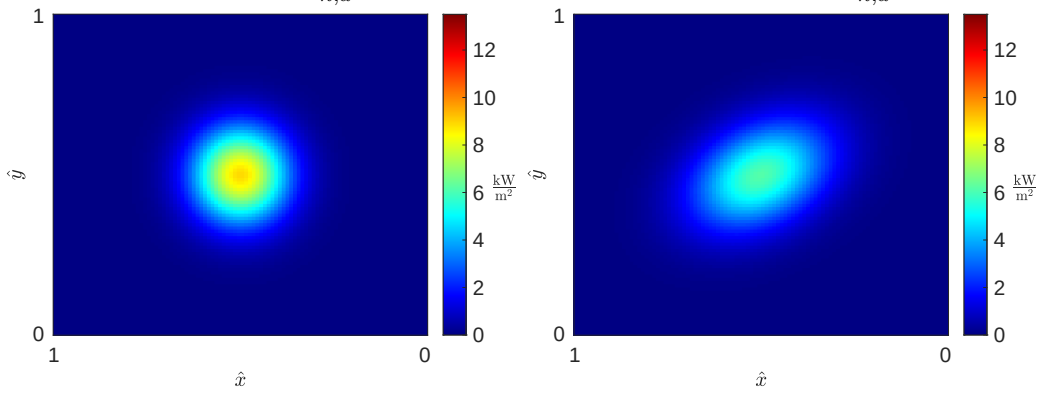
Figures 12e and 12f show that only the extended HFLCAL method implements blocking, which is explained in Section 2.4.3.

We conclude that both heat flux computation methods deliver satisfactory results. However, the extended HFLCAL method captures more detail than the HFLCAL method. Thus, we only use the extended HFLCAL method in this work.



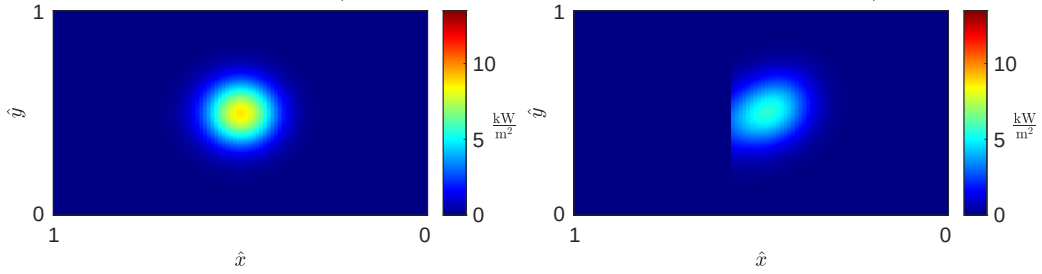
(a) Flat plate receiver, no tilt, heliostat at x=0, y=200 original HFLCAL, $P_{h,a}^{\text{rec}}$: 103.31 kW

(b) Flat plate receiver, no tilt heliostat at x=0, y=200 extended HFLCAL, $P_{h,a}^{\text{rec}}$: 95.09 kW



(c) Flat plate receiver, no tilt heliostat at x=200, y=200 original HFLCAL, $P_{h,a}^{\text{rec}}$: 97.01 kW

(d) Flat plate receiver, no tilt heliostat at x=200, y=200 extended HFLCAL, $P_{h,a}^{\text{rec}}$: 89.31 kW



(e) Cavity receiver, raised by 4m heliostat at x=200, y=200 original HFLCAL, $P_{h,a}^{\text{rec}}$: 69.85 kW

(f) Cavity receiver, raised by 4m heliostat at x=200, y=200 extended HFLCAL, $P_{h,a}^{\text{rec}}$: 45.81 kW

Figure 12: Examples for heliostat images and the PS10 CRS using a rectangular receiver 4. The left and right images show the same configurations using the original and extended HFLCAL method. Figures 12a - 12d use a flat plate receiver, 12e and 12f a cavity receiver. The heliostat is placed north of the receiver (x=0, y=200) in Figures 12a and 12b, for all other images it is placed at x=200, y=200. The heliostat always aims at the center of the receiver.

2.7 Allowed flux distribution

As mentioned before, each receiver has an upper limit for the heat flux hitting it, because its surface can only withstand a certain maximum temperature. Additionally, it might be desirable that a specific heat flux distribution is created at the receiver surface, which can depend on the layout of the tubes containing the heat carrying medium behind the receiver surface.

These ideas are combined by defining the allowed flux distribution (AFD), which is given as a $n_m^{\text{horiz}} \times n_m^{\text{vert}}$ matrix with an allowed heat flux value $q^{m,\text{AFD}}[\frac{\text{kW}}{\text{m}^2}]$ for each measurement point $m \in M^{\text{rec}}$.

Since most likely no heliostat alignment will exactly recreate the allowed heat flux distribution, a deviation below the given values is allowed. A deviation above the AFD could lead to damage done to the receiver and has to be prohibited. For the purpose of this work, all $q^{m,\text{AFD}}$ are assumed to be given by the receiver, as the computation of the AFD is an entirely different optimization problem: Since the AFD depends strongly on the current pump settings for the heat carrying medium, the operator has to find a balance between a high enough fluid throughput to allow for the most heat flux to be reflected onto the receiver and absorbed and a low enough throughput to not overshoot the total energy the CRS could possibly produce at that moment. The AFD values used in this work are given for the Abengoa CRS and obtained by trial and error for PS10 and Gemasolar.

3 Formulation of the optimization problem

3.1 Aiming strategy problem definition

The aiming strategy problem describes the task of finding the optimal aim point for each heliostat with the intent to use the solar power as efficiently as possible, while fulfilling certain constraints.

To be able to solve this problem, a few things have to be known, such as the CRS layout, heliostat and receiver properties. Furthermore, some parameters have to be chosen beforehand. This includes the measurement point resolution of the receiver, as well as the aim points for the heliostats.

The allowed flux distribution (AFD), which is the most important constraint for the optimization, is also assumed to be known. In reality, the AFD might change depending on the total power the receiver receives, which can lead to a two step process: In the first step the AFD is chosen based on an estimation of the total power the heliostats can reflect onto the receiver. In the second step this AFD is used to determine an aiming strategy. Since these two steps can be done separately, we only focus on the optimization problem for a given and fixed AFD.

The solution of the problem should tell the user at which aim point each heliostat aims.

3.2 Formulation as an ILP

In this section the aiming strategy optimization problem is formulated as an integer linear program (ILP). This ILP is built upon the one defined by Kepp [13] and the improvements made by Kuhnke et al. [14].

An ILP is a special form of linear program (LP), which has the following form:

$$\begin{aligned} &\text{maximize} && \langle \vec{c}, \vec{x} \rangle \\ &\text{subject to} && A \cdot \vec{x} \leq \vec{b} \quad \text{with} \quad \vec{x} \geq 0 \end{aligned}$$

$\vec{c}$ are the objective coefficients of the decision variables $\vec{x}$. A is called the constraint matrix, with $\vec{b}$ defining the upper bound for the constraints. The problem is called an ILP if all decision variables $\vec{x}$ are integer variables or a mixed integer linear problem (MILP) if only a subset of the decision variables is integer. The relaxation of an ILP describes the same problem without the condition that the variables have to be integer.

In the following the aim point optimization problem is expressed as an objective function, decision variables and a set of constraints for the decision variables, which together form an ILP.

3.2.1 Objective function

There are two ways of defining the objective function. First, the objective can be defined such that the solar power arriving at the receiver is maximized,

$$\text{maximize} \sum_{m \in M^{\text{rec}}} \left(q^m \cdot A^m\right). \tag{33}$$

Since an allowed flux distribution (AFD) is given, it is also plausible to rate the quality of the solution by how well it approximates this distribution.

$$\text{maximize} \sum_{m \in M^{\text{rec}}} \left(q^m - q^{m,\text{AFD}}\right) \tag{34}$$

Note that (33) and (34) are equivalent for equal A^m as in our discretization.

3.2.2 Decision variables

The heliostats H are allowed to aim at specific points at the receiver surface, called aim points A as defined in Section 2.3. Since the aim points which are targeted by the heliostats are the desired output from the optimization model's solution, we use these as decision variables. Thus, there exists one decision variable for each pair $(h, a), h \in H, a \in A$.

The decision variables $x_{h,a}$ determine if the heliostat $h \in H$ targets the aim point $a \in A$. This decision is binary, therefore all $x_{h,a}$ are binary, i.e.

$$x_{h,a} \in \{0, 1\} \quad \forall\, h \in H \quad \forall\, a \in A \tag{35}$$

If heliostat h targets aim point a then $x_{h,a} = 1$ otherwise it is 0.

Using all decision variables $x_{h,a}$, the heat flux q^m at any measurement point m can be computed as an accumulation of all $q^m_{h,a}$, for which the decision variable $x_{h,a} = 1$:

$$q^m = \sum_{h \in H} \sum_{a \in A} q^m_{h,a} \cdot x_{h,a} \quad \forall\, m \in M \tag{36}$$

3.2.3 Constraints

Maximum one aim point per heliostat

The heliostats can only aim at a single aim point, which translates to the following constraint:

$$\sum_{a \in A} x_{h,a} \leq 1 \quad \forall\, h \in H \tag{37}$$

Since it is also possible for a heliostat to target no aim point at all, the sum is not restricted to be equal to one. This enables the computation of solutions where some heliostats have to stay unused due to safety constraints.

Omit blocked aim points

A heliostat should only aim at an aim point which is in direct line of sight and not blocked in any way. Whether an aim point is blocked is determined by checking whether the receiver is facing the heliostat at this point and whether the receiver occludes the point itself, as described in Section 2.3. For a heliostat, we call the blocked aim points A_h^{blocked} and forbid the use of these points as a target:

$$x_{h,a} \equiv 0 \quad \forall\, a \in A_h^{\text{blocked}} \tag{38}$$

Allowed flux distribution

The allowed flux distribution (AFD) serves as an upper limit for the total heat flux density at the measurement points. The limit $q^{m,\text{AFD}}$ $[\frac{\text{kW}}{\text{m}^2}]$ at a measurement point m is explained as in Section 2.7.

The AFD constraint is formulated as follows:

$$\sum_{h \in H} \sum_{a \in A} q_{h,a}^{m} \cdot x_{h,a} \leq q^{m,\text{AFD}} \quad \forall\, m \in M^{\text{rec}} \tag{39}$$

Allowed flux distribution - heat shield

A heat shield is installed at the edges of the receiver, to prevent damage to the solar tower from heat flux missing the receiver. This heat shield usually has a lower allowed heat flux than the receiver surface, therefore the heat flux at the edges of the receiver is constraint to q^{shield} $[\frac{\text{kW}}{\text{m}^2}]$, see Section 2.3.

$$\sum_{h \in H} \sum_{a \in A} q_{h,a}^{m} \cdot x_{h,a} \leq q^{\text{shield}} \quad \forall\, m \in M^{\text{shield}} \tag{40}$$

4 Solver software

For solving the integer linear program (ILP) introduced in Section 3, different solver software will be tested. In this section we introduce the branch-and-bound algorithm, which is used by all solver implementations, and the properties of the investigated solvers. We also go into detail about the "wrapper" software, which has been implemented to describe the problem data in a way that allows for different solvers to be plugged in.

4.1 Branch-and-bound algorithm

The branch-and-bound algorithm (BAB) is a popular method of solving ILPs. It typically uses the simplex algorithm to compute exact solutions for the relaxed problem and creates branches afterwards to modify the solution to be integer. The following description of the BAB algorithm is based on the `glpk` reference manual [10].

The simplex algorithm, designed by G. Dantzig in 1947, is a solution algorithm for linear programs (LP). Using the simplex algorithm, the solution for the LP relaxation can be computed with a very low runtime. For the aiming optimization problem the relaxation will be that the decision variables can take any real value between 0 and 1, as long as the constraints are fulfilled. The solution may therefore not be immediately usable, due to its relaxed nature, but will be available in much shorter runtime.

Since the solution computed by the simplex algorithm is the global optimum, its objective value serves as an upper bound for the maximization objective of the non-relaxed problem.

Once a solution for the relaxation is computed, the BAB algorithm selects a single variable to create a branch for. Creating a branch means that additional constraints are added to split the problem into two subproblems.

$$var_{\text{relaxed}} = s \quad \Rightarrow \begin{cases} (c1) & var \leq \lfloor s \rfloor \\ (c2) & var \geq \lceil s \rceil \end{cases}$$

For both of these new subproblems, each one adding either constraint $c1$ or $c2$ to the original problem, a new relaxed solution is calculated. If one of these subproblems does not yield a feasible relaxed solution, it is eliminated, as well as when the objective value of the relaxed solution is worse than the best found integer solution. If the solution is better than the best found integer solution and all variables are integer, it is set to be the new best. If not, new subproblems will be created and the above steps are repeated.

Since traversing all generated subproblems would take a very long time, some branches of the problem tree can be cut off early. A subproblem can be eliminated, if the

relaxed solution of the subproblem is worse than the best found integer solution, since the solution of any further subproblems is always worse. This step is called pruning and can be repeated every time a new subproblem is created or a new best solution is found.

Once no subproblem remains, the best integer solution found is the global optimum of the original ILP. If the solution process is cancelled early, the best found solution can still be guaranteed to be close to the global optimum. How close is determined by the remaining gap, which represents the relative difference between the objective values of the best found solution and the best possible objective value.

Branch-and-cut

An extension to the BAB algorithm is the branch-and-cut (BAC) method. This method is often faster than BAB, which is due to the addition of cutting planes in the computation of the relaxed solution. These cutting planes add further constraints which do not limit the integer values the variables can take, but restrict the non-integer possibilities further. This leads to relaxed solutions, which are significantly closer to the best integer solution, and therefore earlier cutoffs for non-optimal branches.

Further deviations

To influence the BAB or BAC algorithm, their behavior can be changed in several ways. One option is to change the order, in which subproblems are investigated. While depth-first-search goes to one of the two created subproblems after branching, breadth-first-search chooses to first investigate the subproblems which were first created. Many different strategies for custom tree traversion orders are also possible and often depend on the use of heuristics to compute upper or lower bounds for the objective values within the branches. The BAC algorithm also offers the possibility to change the usage of different cut strategies.

4.2 Solver overview

The subset of all currently available solvers for ILP models that use the BAB and BAC algorithms contains `Gurobi`, `glpk`, `lpsolve`, `SCIP` and `coin-or`. Details on the used versions can be found in Table 1.

Gurobi [11] is a commercial software, but as it provides an academic license it can still be used for this thesis. Since `Gurobi` is known for its short runtimes, it is to be expected that it outperforms the other non-commercial solver tools.

glpk [10] (GNU Linear Programming Kit) is a package designed to solve linear programming and mixed integer programming problems. The GNU General Public License makes `glpk` free to use in any application as long as it is also distributed under this license.

lpsolve [17] is a mixed integer linear programming solver released under the GNU Lesser General Public License. This means that `lpsolve` may be used in commercial software, as long as it is not modified and used as a compiled library. `lpsolve` is entirely based around the simplex and BAB algorithm.

SCIP [8] [9] claims to be one of the fastest non-commercial solvers for mixed integer programming and mixed integer nonlinear programming. The `SCIP` solver is part of the *SCIP Optimization Suite*, which also contains `SoPlex`, a linear programming solver, which is used by `SCIP` and other tools that are not used in this thesis. The entire *SCIP Optimization Suite* is released under the ZIB (Zuse Institute Berlin) Academic License and free for academic use.

coin-or [15] (Common Optimization Interface for OR (Operations Research)) offers several tools in the field of operations research, one of them being the `Cbc` (Coin-or branch-and-cut) solver. The solver is an open-source mixed integer programming solver written in C++. The tools from the `coin-or` project build upon each other, such that the `Cbc` solver uses amongst other parts the `Clp` (Coin-or linear programming) solver. In this thesis `Cbc` will be used, since it offers BAB functionality. `Cbc` is published under the EPL (Eclipse Public License) which allows for commercial use of the unmodified program.

All solvers presented are used unmodified as callable libraries in a custom C++ implementation of the problem.

Solver	Licensing	Last Update	Used Version
`Gurobi`	commercial, academic	2020	9.0
`glpk`	GNU GPL (free)	2018	4.65
`lpsolve`	GNU LGPL (free)	2016	5.5.2.5
`SCIP`	ZIB (academic)	2018	6.0.2
`coin-or`	EPL (free)	2020	`Cbc` 2.10.4

Table 1: Used solver software, their licensing, the year of the last update and the used version.

4.3 C++ implementation

To implement the aiming strategy optimization, we use C++ and organize the code such that its class hierarchy represents the actual problem structure, while paying attention to modularity. This allows future work to extend or use the project easily.

The project's six most important components are the following:

- The central receiver system (CRS) is represented as a class, which manages its smaller components such as the receiver, the heliostats and the sun. This class works as a data structure with some additional functionality to provide e.g. receiver point coordinates.
- The raytracer class functions as an abstract class, from which new classes can be derived. Each derived class implements a different way of computing the heat flux. We include the original and extended HFLCAL method. Further methods, e.g. GPU raytracers, can be added easily.
- The ILP model constructs the integer linear program with the objective function and all its variables and constraints for an assigned CRS and raytracer. Additionally, it contains an instance of the solution class, which can be written by the solver. This allows for solver independent storage and evaluation as well as further usage of the solution, e.g. as a starting point for following computations.
- The solver class is again an abstract class, derived classes can attach different solver software to solve an ILP. A `read_model` function serves as an adapter to transform the ILP to the solver specific structure. We provide adapters for all solvers presented in Section 4.2.
- The solution class contains information like the computation time and used solver software in addition to the actual solution values.
- Data is a class which manages reading the input files and provides access to it via static functions.

An example program for loading a CRS and solving the aiming strategy problem can be implemented easily using this class structure and is displayed in Listing 1. The entire project can be found on `https://git.rwth-aachen.de/Computational-Renewable-Energy/SunFlower/Aiming-Strategy.git`.

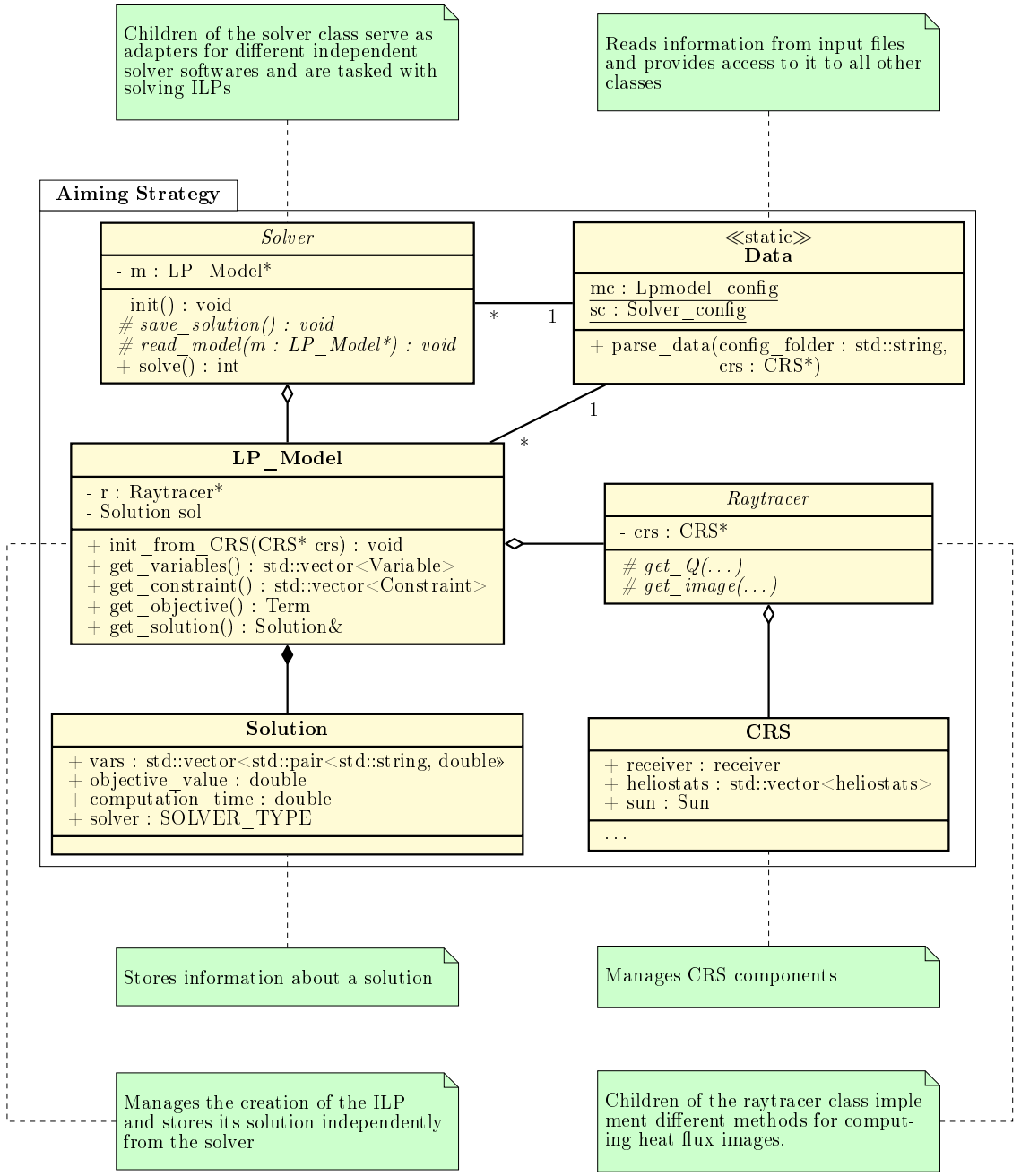


Figure 13: Overview of the class structure.
Code at `https://git.rwth-aachen.de/Computational-Renewable-Energy/SunFlower/Aiming-Strategy.git`

```cpp
#include <string>
#include "data.hpp"
#include "lpmodel.hpp"
#include "solver.hpp"
#include "raytracer.hpp"
#include "crs.hpp"

int main(int argc, char** argv) {
    assert(argc == 2);
    std::string input_folder = argv[1];

    CRS* crs = new CRS();                   // initialize the CRS
    Data::parse_data(input_folder, crs); // read input

    LP_Model* m = new LP_Model();          // initialize the ILP
        Model
    Solver* s = Solver::get();             // initialize the solver

    m->init_from_crs(crs);                 // create ILP from CRS
    s->read_model(m);                      // pass ILP to chosen
        solver
    s->solve();                            // solve the ILP
    m->write_solution();                   // output the solution

    delete s, crs, m;                      // cleanup
    return 0;
}
```

Listing 1: Simple example main function.

4.4 Solver specific ILP model creation

All of the solver softwares allow for an intuitive translation of the mathematical ILP model from Section 3 to the solver language. Furthermore, `Gurobi`, `lpsolve`, `SCIP` and `coin-or` offer the usage of a special constraint type called Special Ordered Set (SOS). An SOS is a set of variables of which only one or two can be non-zero. These are called SOS1 and SOS2 respectively. The ”maximum one aim point per heliostat” constraints from Equation 37 in Section 3 can be trivially formulated as an SOS1 constraint. In the documentation of `Gurobi` it is mentioned, that the usage of SOS constraints instead of linear constraints usually leads to worse solution times, which we can confirm for different testcases. Thus, we implement the constraints as less-or-equal constraints with an upper bound of 1, since all of the variables are binary.

4.5 Case study for optimal solver configuration

To obtain the best possible performance on a specific problem, several solver parameters can be set by the user. The influences of the different solver parameters were tested per individual solver on a reference problem, for this purpose the PS10 power plant is used with a flat plate receiver. Details about the used parameters can be found in Appendix, Table 4. The hardware used for all measurements in this work is a standard personal computer, the corresponding data can be found in Appendix, Table 5.

irace: iterated racing for automatic algorithm configuration

Since some solvers offer a great variety of parameters, it is not feasible to try to find good values for all of them manually. irace is a package designed by López-Ibáñez et al. [16] for automatic algorithm configuration. It allows the user to specify parameters, which are then tuned by iteratively running a given program using different configurations.

Tuning method

To get the best possible parameter configuration, we first manually adjust and evaluate some recommended options. The intuitive parameters e.g. include enabling the feasibility pump heuristic for glpk, which already reduces the runtime by a large margin. After doing these initial adaptations, all important parameters are fed into irace. For `SCIP` in particular, 110 parameters regarding the branch-and-cut heuristics and branching methods are examined using irace to find an optimal configuration. Finally, the results of irace are evaluated manually to find the parameters where changes lead to a major performance increase.

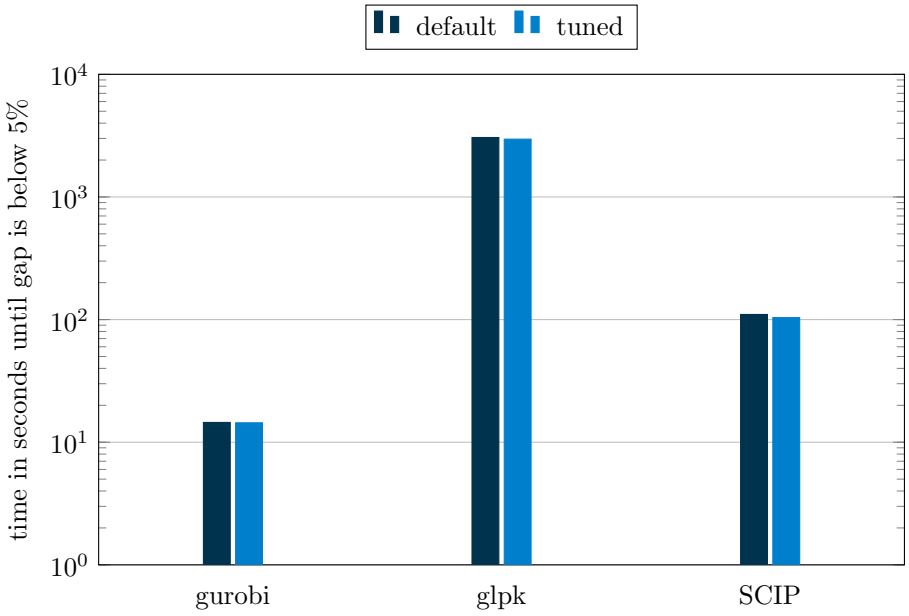
Tuned solver parameters

In Table 2 all changed parameters are listed. `Gurobi`, `glpk` and `SCIP` all achieve a solution within one hour with `Gurobi` being the fastest by far. `coin-or` did not obtain a solution with default settings in under 4 hours and `lpsolve` was not able to deliver a solution within a day, thus the evaluation with irace is not feasible and we ignore them in the parameter study from now on.

To show the improvements in solver runtime using the adapted parameters, we measure the time the solvers take until they reach a gap of less than 5% on the test problem.

Gurobi parameter	Default	Tuned	Brief description
GRB_IntParam_Goal	0	1	prioritize feasibility
glpk parameter	**Default**	**Tuned**	**Brief description**
fp_heur	0	1	enable feasibility pump heuristic
ps_heur	0	1	enable proximity search heuristic
gmi_cuts	0	1	enable gomory's cuts
br_tech	4	3	branch most fractional variable
bt_tech	3	1	depth first search
lpsolve parameter	**Default**	**Tuned**	**Brief description**
-	-	-	-
SCIP parameter	**Default**	**Tuned**	**Brief description**
heuristics/vbounds/priority	2500	3500	increase priority of vbounds heuristic
heuristics/vbounds/freq	0	1	set frequency of vbounds heuristic
heuristics/clique/freq	0	-1	disable clique cuts
coin-or parameter	**Default**	**Tuned**	**Function**
-	-	-	-

Table 2: Overview about tuned parameters for the different solvers.



	Average runtime in seconds		
	Gurobi	**glpk**	**SCIP**
Default	14.542	3053.982	110.117
irace	14.408	2961.020	103.806

Figure 14: Average runtime until a gap of 5% remains, using default and irace tuned parameters.

As stated above, `lpsolve` and `coin-or` are not appropriate for our kind of problem. Figure 14 shows that of the remaining three, `Gurobi` is by far the fastest solver, but both `glpk` and `SCIP` also manage to produce results in a larger but still acceptable time window. It is to be expected that the solution time scales with the problem size for all solvers similarly.

Due to this, we use gurobi for all our measurements in this work, since it delivers results the fastest. We also decrease the end gap down to 1% to 0.5%, depending on the size of the CRS. This allows for more reliable statements about the solutions.

5 ILP acceleration using heuristics

In the following, different heuristic approaches towards the optimization problem introduced in Section 3 are presented, which aim to speed up the solution process. The goal of these heuristic approaches is to achieve a solution, which is at most a small percentile worse than the solution without the use of heuristics.

The main idea is to reduce the size of the constraint matrix A, which can be done by reducing the total number of constraints and the decision variables.

For all approaches, we use the PS10 CRS to evaluate quality and performance.

5.1 Grouping of heliostats

To reduce the number of decision variables, heliostats can be combined into groups and all heliostats in one group are then only allowed to aim at the same aim point. The group's heat flux image at the receiver can be computed by summing up the heat flux images of all heliostats within the group. Therefore, the group acts as one single virtual heliostat for the optimization problem.

The set of groups is defined as

$$G = \{g_i \mid i \in \{1, ..., n^{\text{groups}}\}\} \tag{41}$$

with n^{groups} defining the total number of groups. Each group g_i represents a subset of heliostats

$$g_i \subseteq H \quad \text{with} \quad g_i \cap g_j = \varnothing \quad \forall\, i \neq j, \ \bigcup_i g_i = H \tag{42}$$

and sums up the heat flux images of its heliostats as follows:

$$q^m_{g_i,a} = \sum_{h \in g_i} q^m_{h,a} \tag{43}$$

The set of aim points the group can target is reduced to the intersection of the aim point sets visible from each heliostat.

$$A^{\text{visible}}_{g_i} = \bigcap_{h \in g_i} A^{\text{visible}}_h \tag{44}$$

When grouping heliostats, it is therefore important to make sure that the heliostats within a group are able to aim at overlapping sets of aim points. This can be achieved by letting the look-at-angle of the heliostats from the receiver tower influence the grouping.

Grouping the heliostats reduces the amount of decision variables by a factor of at least n^{groups}, while the restriction of aim points per group may lead to a further decrease, depending on the overlap of the heliostat aim point sets.

Angular grouping

We use agglomerative clustering to group the heliostats. This clustering method iteratively merges the two clusters with the lowest dissimilarity until the given number of clusters remains. The dissimilarity can be any function of two clusters, as long as it is symmetric. At the beginning, all clusters consist of single heliostats.

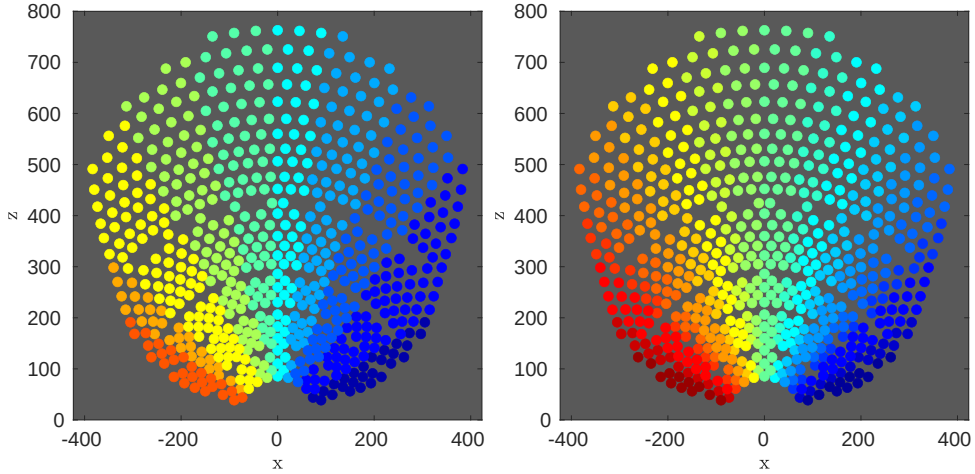
To formalize which heliostats should be within the same group, a custom dissimilarity function is defined. The dissimilarity of heliostats i and j is low, if they share a large set of possible aim points. A low dissimilarity therefore indicates heliostat pairs, which are well suited to be in the same group. The function

$$diss_{i,j} = \text{angdiff}(\alpha_i, \alpha_j) \tag{45}$$

lets the dissimilarity of two heliostats h_i, h_j depend on the angular difference between the look-at-angles α_i, α_j from the receiver tower.

The distance between the clusters, which is used in the clustering algorithm, is called linkage d and can be computed in many ways. In our case, computing the linkage, by taking the maximum dissimilarity between points within the clusters, works well:

$$d(g_r, g_s) = \max(diss_{i,j}) \quad i \in g_r, j \in g_s \tag{46}$$



(a) $n^{\text{groups}} = 10$ (b) $n^{\text{groups}} = 20$

Figure 15: Angular grouping of heliostats. Heliostats within the same group are drawn with the same color.

This is also called ”complete” linkage and minimizes the maximum dissimilarity within the groups. An other example is ”single” linkage, where the shortest distance between cluster points defines the distance of the clusters. In our case, when using the ”single” linkage approach, most heliostats are grouped into one big cluster, which is not desirable. In Figure 15, the heliostat clusters for ”complete” linkage can be seen.

Maximum distance grouping

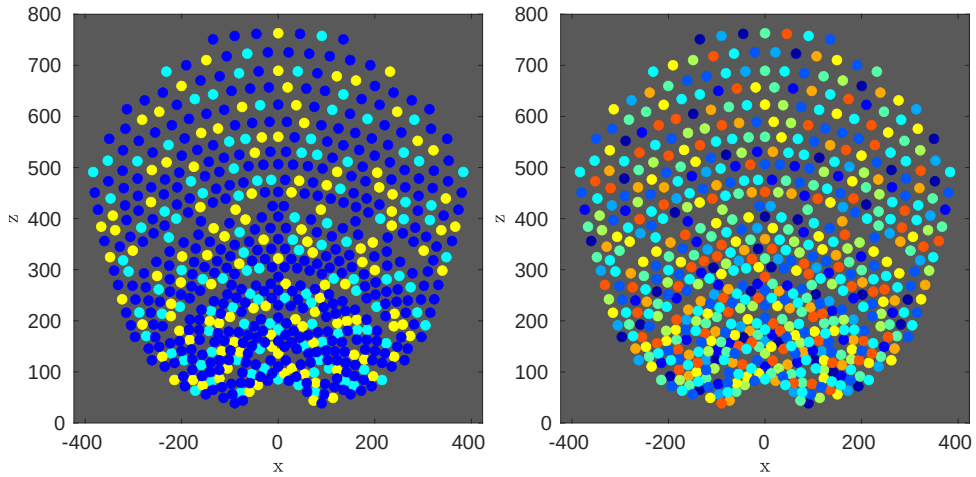
The groups resulting from the angular clustering consist of many heliostats in close proximity. This can be a problem, since it increases the influence of cloud shadows passing over the heliostat field. They now greatly reduce the heat flux hitting the receiver around the aim point of the covered groups. By spreading heliostats of one group, fewer of the heliostats lie within cloud shadows simultaneously and the heat flux distribution at the receiver is weakened homogenously.

To achieve this goal, a different dissimilarity function is introduced. This time, the focus lies on grouping heliostats with a large spatial distance, which can be done by changing the sign of the spatial distance:

$$diss_{i,j} = -||p_i - p_j||_2 \tag{47}$$

with p_i, p_j representing the position vectors of heliostats h_i and h_j.

Using this dissimilarity function and the same algorithm as above, a completely diffuse grouping is obtained, which can be seen in Figure 16. As the resulting clusters suffer from large angular distances within the groups, a combined dissimilarity function is derived below.



(a) $n^{\text{groups}} = 3$ (b) $n^{\text{groups}} = 10$

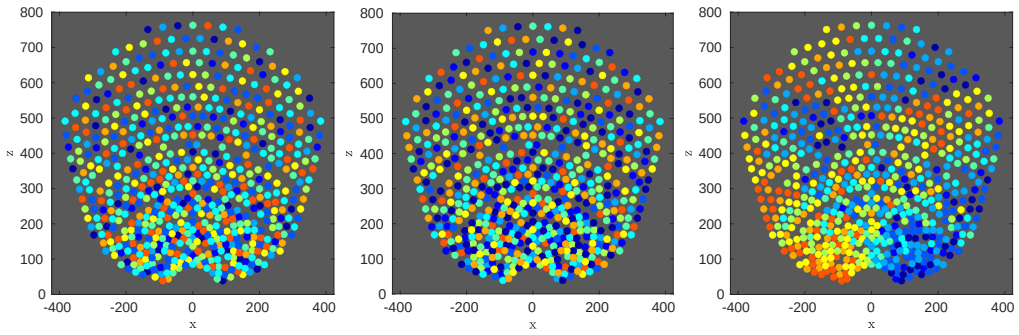
Figure 16: Grouping of heliostats based on maximizing the spatial distances within a group. Heliostats within the same group are drawn with the same color.

Combined grouping

To combine the upsides of both dissimilarity functions presented above, these functions are combined into a single one:

$$diss_{i,j} = \lambda_{\text{grouping}} \cdot \left(\frac{\text{angdiff}(\alpha_i, \alpha_j)}{\pi} \right)^2 - (1 - \lambda_{\text{grouping}}) \cdot \frac{||p_i - p_j||_2}{\max_{k,l} ||p_k - p_l||_2} \tag{48}$$

with $\lambda_{\text{grouping}} \in [0, 1]$. This dissimilarity function consists of two parts, which can be balanced using $\lambda_{\text{grouping}}$. First, the angular difference between the heliostats with respect to the solar tower should increase the dissimilarity between them, as only heliostats which lie in roughly the same direction can share a sufficient amount of aim points. To punish large angular differences, this term is additionally squared. Secondly, the euclidean distance between the heliostats should be as large as possible, therefore a large euclidean distance lowers the dissimilarity. This euclidean distance is normalized using the maximum distance between any two heliostats and the angular distance is normalized with π. Resulting heliostat groups for different $\lambda_{\text{grouping}}$ values can be seen in Figure 17.



(a) $\lambda_{\text{grouping}} = 0.0$ (b) $\lambda_{\text{grouping}} = 0.3$ (c) $\lambda_{\text{grouping}} = 0.6$

(d) $\lambda_{\text{grouping}} = 0.8$ (e) $\lambda_{\text{grouping}} = 0.9$ (f) $\lambda_{\text{grouping}} = 1.0$

Figure 17: Grouping of heliostats based on the combined dissimilarity function. Heliostats within the same group are drawn with the same color.

Optimal grouping configuration

The grouping of heliostats can be controlled by the two parameters $\lambda_{\text{grouping}}$ and c. The latter is defined as the compression factor of the heliostat set:

$$c_{\text{heliostats}} = \frac{n^{\text{heliostats}}}{n^{\text{groups}}} \tag{49}$$

It is to be expected that a high $\lambda_{\text{grouping}}$ value corresponds to higher possible objective values than a low $\lambda_{\text{grouping}}$ value, since the aimpoints are less restricted. The advantage of a low $\lambda_{\text{grouping}}$ are higher spatial distances within the heliostat groups, which reduces the impact of cloud shadows on the heat flux distribution at the receiver. An optimal value for $\lambda_{\text{grouping}}$ therefore is as low as possible, while not reducing the best possible objective value significantly.

Since $c_{\text{heliostats}}$ can be chosen arbitrarily, some sample values are chosen for further evaluation. Figure 18 shows the dependence of the solution quality of $c_{\text{heliostats}}$. In the following, a suitable $\lambda_{\text{grouping}}$ is determined using the sample set of $c_{\text{heliostats}} \in \{2, 4, 8, 16, 32\}$, as these cover the decline of solution quality for all $\lambda_{\text{grouping}}$. We use the PS10 central receiver system for evaluation, as the cavity receiver is expected to lead to higher differences in aim point sets for near heliostats than a cylindric receiver.

To evaluate the influence of $\lambda_{\text{grouping}}$ on the amount of remaining aim points, the average relative aim point set size $s_A \in [0, 1]$ is formalized:

$$s_A = \sum_{g_i} \left(\frac{1}{|g_i|} \cdot \sum_{h \in g_i} \frac{|A_{g_i}^{\text{visible}}|}{|A_h^{\text{visible}}|} \right) \tag{50}$$

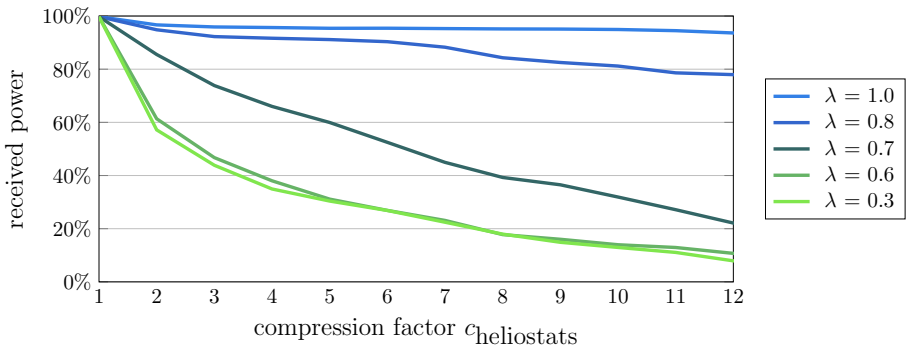


Figure 18: The influence of $c_{\text{heliostats}}$ and $\lambda_{\text{grouping}}$ on the reachable objective value. The solution quality decreases with an increase in $c_{\text{heliostats}}$ and a decrease in $\lambda_{\text{grouping}}$.

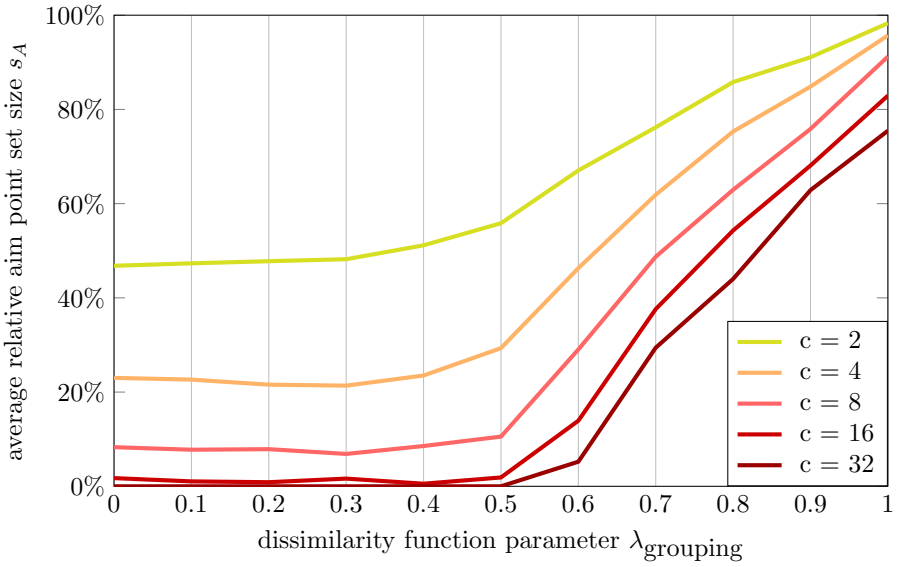


Figure 19: The influence of $\lambda_{\text{grouping}}$ and $c_{\text{heliostats}}$ on s_A. The set of available aim points per heliostat shrinks rapidly when lowering $\lambda_{\text{grouping}}$ until $\lambda_{\text{grouping}} \approx 0.5$. Increasing the compression factor $c_{\text{heliostats}}$ decreases s_A.

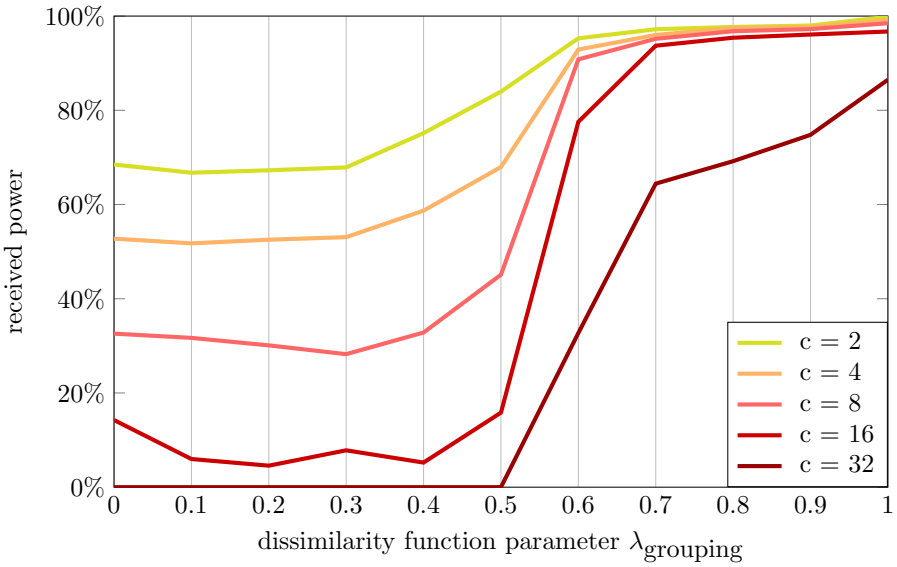


Figure 20: The influence of $\lambda_{\text{grouping}}$ and c on the reachable objective value. $\lambda_{\text{grouping}} > 0.7$ does not negatively impact the reachable objective value.

A value of $s_A = 0.6$ would indicate that heliostats can aim on average at 0.6 times as many aim points as if they were not grouped.

Figure 19 shows that s_A strongly depends on $\lambda_{\text{grouping}}$. Since s_A decreases rapidly for $\lambda_{\text{grouping}} < 1$, low $\lambda_{\text{grouping}}$ values are likely to decrease the solution quality. Figure 20 supports this by showing that the objective value rapidly decreases for $\lambda_{\text{grouping}} < 0.8$, regardless of the compression factor. A closer look reveals that the solution quality is above 95% for $s_A > 70\%$, although too high compression factors of $c_{\text{heliostats}} = 16$ or higher start to negatively impact the solution at any $\lambda_{\text{grouping}}$.

For $\lambda_{\text{grouping}} = 0.8$ and a compression factor of $c_{\text{heliostats}} <= 16$, the objective value stays above 95% of its normal value. $c_{\text{heliostats}} = 2$ with $\lambda_{\text{grouping}} = 0.8$ results in a solution quality of 98.2%.

Performance evaluation

To evaluate the effect of the compression factor $c_{\text{heliostats}}$ on the solver runtime, $\lambda_{\text{grouping}}$ is fixed at $\lambda_{\text{grouping}} = 0.8$. Figure 21 shows that the computation time decreases significantly when grouping the heliostats with any compression factor $c_{\text{heliostats}} \geq 2$ for the PS10 central receiver system. The speedup achieved stays almost constant at about 4, which is likely due to the relatively small size of the PS10 CRS. For larger total heliostat numbers, a greater achievable speedup is to be expected. For the PS10 CRS, suitable parameters for the grouping of heliostats, which keep the solution quality high enough but provide a noticeable speedup, are thus $\lambda_{\text{grouping}} = 0.8$ and $2 \leq c_{\text{heliostats}} \leq 5$.

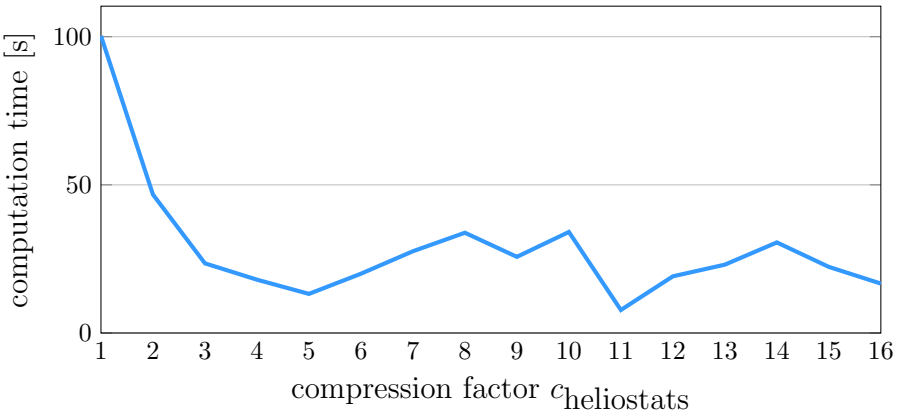


Figure 21: Runtime depending on $c_{\text{heliostats}}$ for $\lambda_{\text{grouping}} = 0.8$ for the PS10 central receiver system.

5.2 Reduction of aim points

Another approach to decrease the problem size is to reduce the number of aim points per heliostat instead of reducing the number of heliostats. While doing so, it is important to keep the initial aim point resolution and to reduce the number of aim points any individual heliostat can target. This keeps the solution quality higher while also allowing for finer control over the reduction in aim points.

In real scenarios, heliostats which are far away from the receiver tend to aim towards its center, while closer heliostats choose more diverse aim points. This leads to the idea of reducing the aim point set size of a heliostat in proportion to its distance from the receiver.

To do so, we introduce two compression variables $c_{\text{aimpoints,close}}$ and $c_{\text{aimpoints,far}}$, one for the relative aim point set compression for close heliostats and one for the far heliostats. For each heliostat or heliostat group the compression factor is then computed by using its relative distance to its aim points as in Equations (51) to (54).

$$g_{i,\text{mean}} = \frac{1}{|g_i|} \cdot \sum_{h \in g_i} h \tag{51}$$

$$a_{i,\text{mean}} = \frac{1}{|A_{g_i}^{\text{visible}}|} \cdot \sum_{a \in A_{g_i}^{\text{visible}}} a \tag{52}$$

$$\lambda_{i,\text{reduction}} = \frac{\|a_{i,\text{mean}} - g_{i,\text{mean}}\| - \min d_{a,h}}{\max d_{a,h} - \min d_{a,h}} \tag{53}$$

$$c_{i,\text{reduction}} = (1 - \lambda_{i,\text{reduction}}) \cdot c_{\text{aimpoints,close}} + \lambda_{i,\text{reduction}} \cdot c_{\text{aimpoints,far}} \tag{54}$$

$c_{i,\text{reduction}}$ now represents the compression factor for heliostat group i.

$$|A_{g_i}^{\text{reduction}}| = \max \left\{ 1, \left\lfloor \frac{1}{c_{i,\text{reduction}}} \cdot |A_{g_i}^{\text{visible}}| \right\rfloor \right\} \tag{55}$$

Which aim points are kept is determined by the usage of the k-means clustering algorithm. The aim points for $A_{g_i}^{\text{reduction}}$ are chosen to be the ones of $A_{g_i}^{\text{visible}}$ which are closest to the corresponding cluster centroid.

Instead of clustering the aim points from $A_{g_i}^{\text{visible}}$ in cartesian coordinates, they are clustered with regard to their horizontal and vertical angular difference from the group center $g_{i,\text{mean}}$. In current practice, the number of aim points is reduced by using a finer receiver discretization towards its edges and a more coarse one in its center. To emulate this, the angular differences $\phi_{a,\text{hor}}, \phi_{a,\text{vert}}$ are projected as follows in Equations (56) - (57):

$$\phi_{a,\text{hor}} = \text{sign}(\phi_{a,\text{hor}}) \cdot (\text{abs}(\phi_{a,\text{hor}}))^x \tag{56}$$
$$\phi_{a,\text{vert}} = \text{sign}(\phi_{a,\text{vert}}) \cdot (\text{abs}(\phi_{a,\text{vert}}))^x \tag{57}$$

with x being an arbitrary exponent. For $x > 1$ the outer aim points are stretched further apart than the inner ones, making them more likely to be clustered into smaller groups, as displayed in Figure 22.

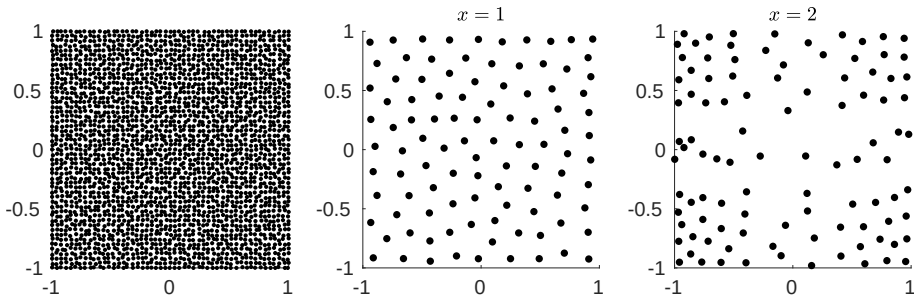


Figure 22: K-means centroids for a noisy point grid (left), computed with previously mapping the coordinates using exponent $x = 1$ (middle) and $x = 2$ (right). The right image shows finer clusters on the edges and more coarse clustering in the center.

To find a suitable x we evaluate its influence on the solution quality. Figure 23 shows that for $c_{\text{aimpoints}} = 8$, $x = 2$ leads to the best solution.

The effect of the aim point reduction can be seen in Figure 24 for the PS10 CRS. The plots reveal that the influence of the compression factors of close and far heliostats is almost entirely symmetric. Therefore, we combine them as $c_{\text{aimpoints}} = c_{\text{close}} = c_{\text{far}}$. For PS10, the value $c_{\text{aimpoints}} = 6$ is a suitable example, since the computation time is already reduced drastically from around 100 seconds to about 13 and the solution quality only decreases by about 2.7%.

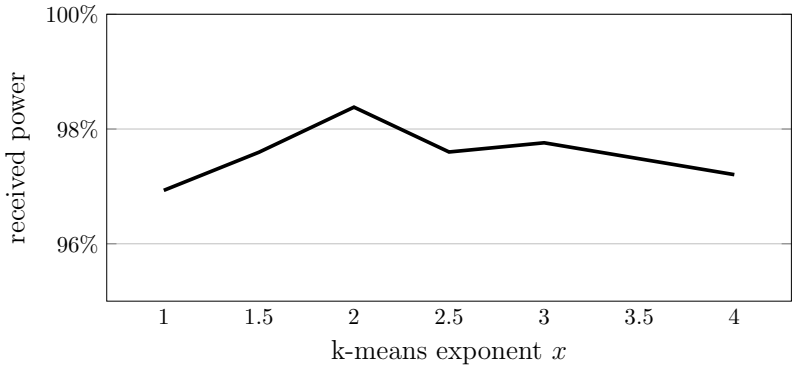
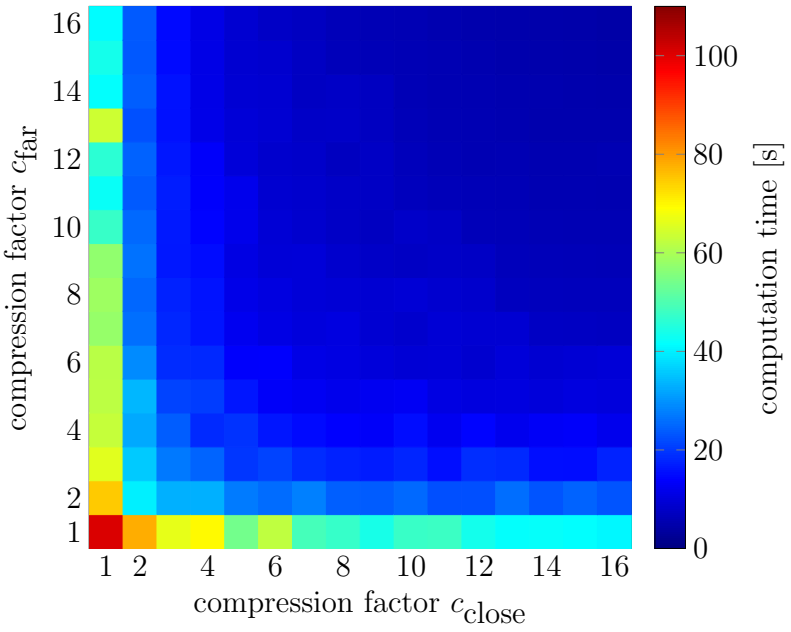
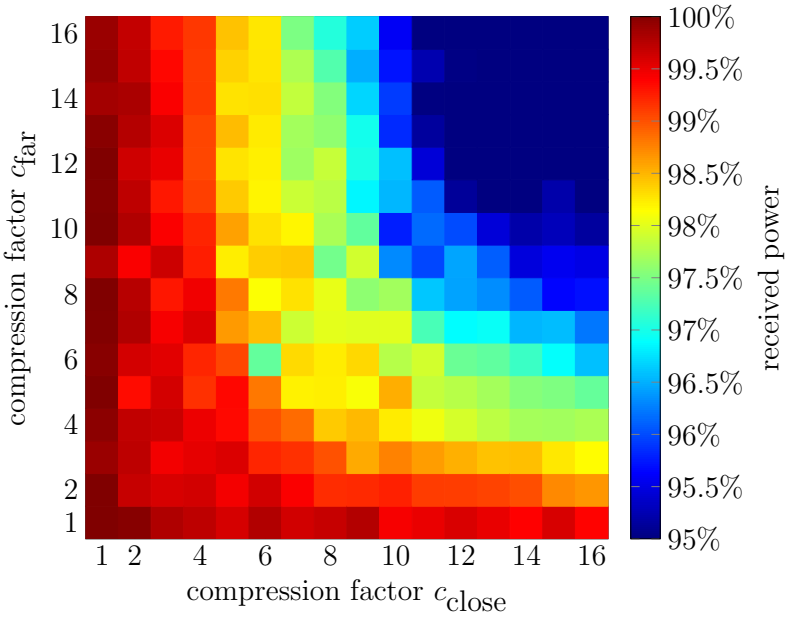


Figure 23: Effect of the exponent for the k-means aim point clustering on the objective value for PS10 and $c_{\text{aimpoints}} = 8$.



(a) Influence of the compression factors $c_{\text{aimpoints,close}}$ and $c_{\text{aimpoints,far}}$ on the computation time.



(b) Influence of the compression factors $c_{\text{aimpoints,close}}$ and $c_{\text{aimpoints,far}}$ on the solution quality.

Figure 24: Effect of the aim point reduction on the computation time and solution quality for the PS10 CRS.

6 Case study

To show the effects of the methods developed in this thesis, we apply them to three different central receiver systems, which are introduced in Section 2.5.

As prerequisite to the simulation runs, we have to determine the measurement and aim point sets, as well as the allowed flux distribution (AFD). These settings are kept constant for each considered central receiver system (CRS), while applying the different heuristic approaches, to compare their performance to the default problem, which represents the optimal solution without any heuristic optimization. For the heliostat grouping, we use $\lambda_{\text{grouping}} = 0.8$ for all testcases, as discussed in Section 5.

6.1 Heuristics on real CRS

The parameters that have been varied over the course of the following computations are the heuristic compression factors $c_{\text{heliostats}}$ and $c_{\text{aimpoints}}$. The diagrams show the solution quality in relation to the non-optimized solution as well as the computation time in seconds for all combinations of compression factors with $c_{\text{heliostats}}, c_{\text{aimpoints}} \in [1, 16]^2 \subset \mathbb{N}^2$. Detailed information about the parameters can be found in Appendix Table 4.

PS10: 624 heliostats, 135 aim points, 187 measurement points

The performance diagrams for PS10 can be seen in Figure 25. It can be observed, that the solution quality decreases consistently with rising compression factors, while the computation time decreases rapidly. For high compression factors, spikes in the computation time can be seen, which appear once the groups are large enough, such that each one produces a significant heat flux spike at the receiver surface, while also having a limited amount of aim points remaining.

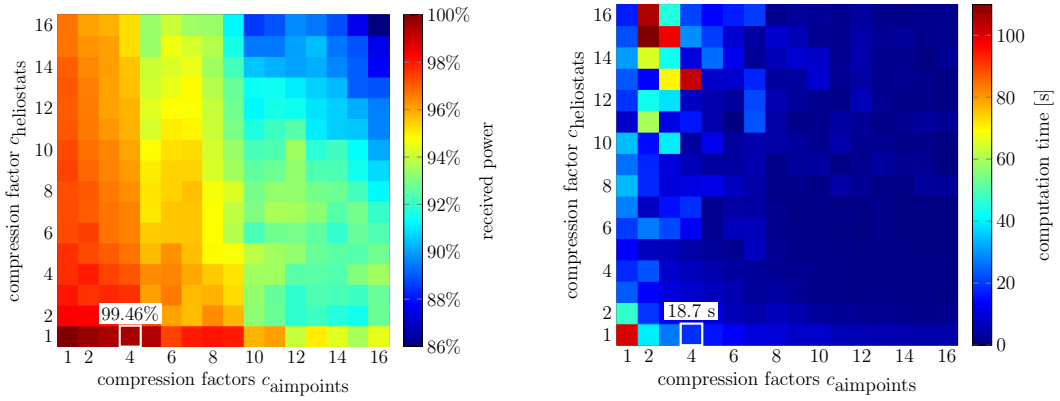


Figure 25: Solution quality (left) and computation time (right) for PS10 in dependence of the heuristic compression factors $c_{\text{heliostats}}$ and $c_{\text{aimpoints}}$.

For PS10, a viable set of compression factors are $c_{\text{heliostats}} = 1$ and $c_{\text{aimpoints}} = 4$, as these provide a solution quality of 99.46% and reduce the computation time from 100 seconds down to 18.7 seconds. Further increasing the compression factors would still decrease the computation time but also lead to larger decreases of the solution quality, which is not necessary, since a computation time of 18.7 seconds is already really low.

Gemasolar: 2650 heliostats, 135 aim points, 165 measurement points

The Gemasolar CRS shows a stronger decrease in solution quality with $c_{\text{aimpoints}}$ than $c_{\text{heliostats}}$, which can be explained by the larger number of heliostats available, while the aim point set size is kept the same as for PS10. Since the receiver is a cylindric external receiver, the heliostats can only target about half the aim points by default, which makes a further reduction in aim points impact the solution quality strongly. As a result, $c_{\text{aimpoints}}$ should not be chosen too high for this testcase, a feasible set of compression factors are $c_{\text{heliostats}} = 8$ and $c_{\text{aimpoints}} = 4$. These reduce the solution quality to 96.61% and the computation time to 6.3 seconds from the original 174.1 seconds.

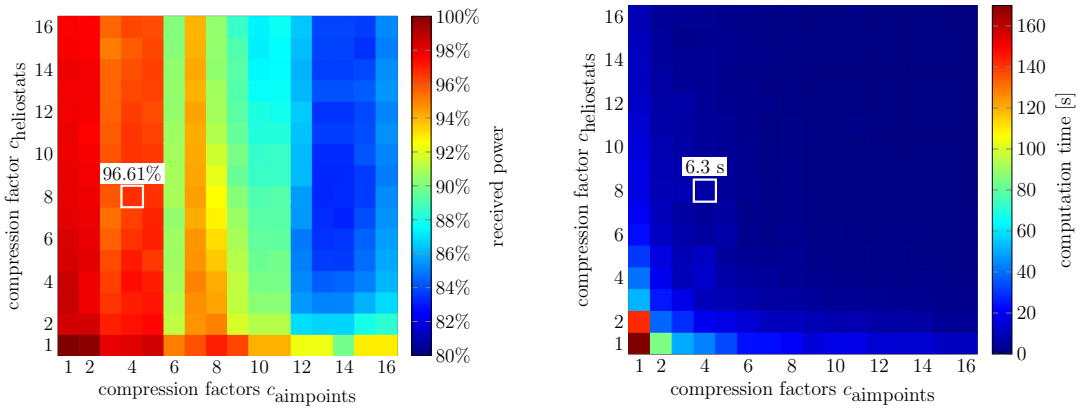


Figure 26: Solution quality (left) and computation time (right) for Gemasolar in dependence of the heuristic compression factors $c_{\text{heliostats}}$ and $c_{\text{aimpoints}}$.

Abengoa CRS: 8600 heliostats, 1777 aim points, 288 measurement points

The Abengoa CRS is the large scale testcase of the three. Because of the very high number of heliostats as well as aim points, solving the aiming strategy problem without the use of heuristics is not possible due to hardware limitations. For the same reasons, we only use a rougher grid of heuristic parameters. Instead of using the solution for no heuristics as a reference to evaluate the solution quality, we therefore use the highest solution achieved, which is for $c_{\text{heliostats}} = 4$ and $c_{\text{aimpoints}} = 1$. The computation time for this solution is already as high as 878 seconds, which can be reduced drastically without a real loss in solution quality by applying higher

compression factors. Due to the size of this CRS in both heliostat and aim point count, reducing their numbers only marginally affects the solution quality, since most restrictions can be balanced out easily. The compression factors $c_{\text{heliostats}} = 8$ and $c_{\text{aimpoints}} = 8$ already lead to a computation time of 83.9 seconds and keep the solution quality at 99.3%, further increasing them to up to $c_{\text{heliostats}} = 16$ and $c_{\text{aimpoints}} = 16$ only reduces the solution quality by an additional 0.2% and decrease the computation time down to 18.2 seconds.

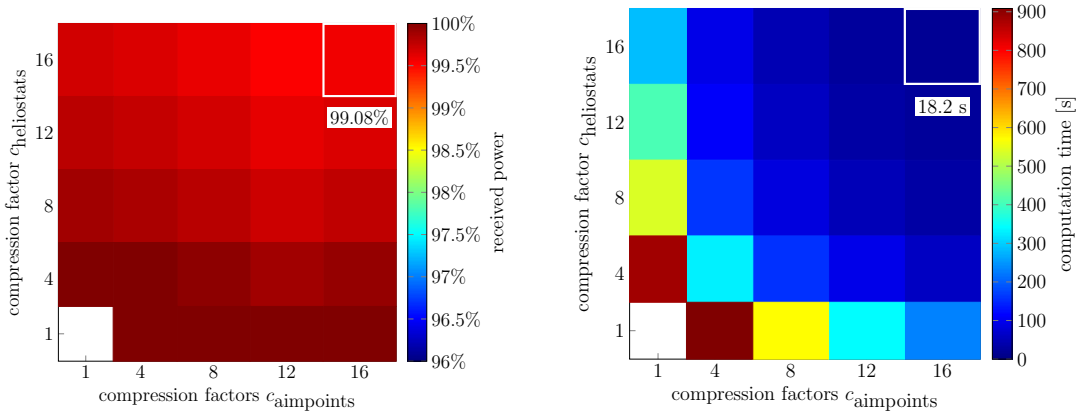


Figure 27: Solution quality (left) and computation time (right) for the Abengoa CRS in dependence of the heuristic compression factors $c_{\text{heliostats}}$ and $c_{\text{aimpoints}}$. The solution for $c_{\text{heliostats}} = c_{\text{aimpoints}} = 1$ was not computed due to hardware limitations.

Recommended compression factors

The recommended compression factors determined for the testcases of the case study are shown in Table 3.

	$c_{\text{heliostats}}$	$c_{\text{aimpoints}}$	solution quality	speedup
PS10	1	4	97.10%	5.3
Gemasolar	8	4	96.61%	27.6
Abengoa	16	16	99.08%	48.2

Table 3: Recommended compression factors for the three testcases.

6.2 Warm start for the computation

In practice, the aiming strategy problem for a CRS is solved repeatedly to always adjust to the current environmental conditions. Should a previous solution be known, it can be used as an initial solution in the next time step. This requires a check for constraint validations, since the solution can be invalidated by e.g. the sun rising, which increases the heat flux all the heliostats reflect onto the receiver.

To imitate the real world scenario, we reinitialize the solver and provide only the solution values for the decision variables as initial values, while not giving any information about whether this is a solution or not. Since the solution has to be recomputed from scratch if the past solution is invalid, we only provide solutions which are guaranteed to be viable.

Figure 28 shows the decrease in solution time for PS10, when providing the solution from the exact same problem. To note is the decrease of spikes in computation time for high compression factors. For Gemasolar this is also observable, although far less impactful, as one can see in Figure 29.

In the tests above we solved the exact same problem twice and input its own solution as an initial solution. Since in reality the problem does not stay exactly the same, the solution is generally unknown and the initial solution can only come from a different, already solved, problem. It is important to ensure the validity of the initial solution, since a change in problem parameters could lead to higher heat flux values or stricter safety constraints. To get an idea for the effect, providing an initial solution has, we use a simple testcase on the PS10 CRS, which guarantees the validity of the initial solution. For this, we use the initial solution for the normal problem, which is computed exactly the same as before, and use it as an initial solution for the same problem with a set of heliostats disabled, as seen in Figure 30. Similar changes could happen in reality when clouds pass over the heliostat field. For the disabled heliostats, all reflected heat flux is set to zero, which only reduces the total heat flux image at the receiver surface. This keeps the solution valid, as we keep the allowed flux distribution the same.

Figure 31 shows the comparison between the computation times for the new problem with and without the initial solution from the unmodified problem. The total computation times drop a very small amount when using the initial solution, but in general the time save is negligible. Thus, we conclude that providing an initial solution to the aiming strategy problem does not necessarily result in noticeably faster computation times.

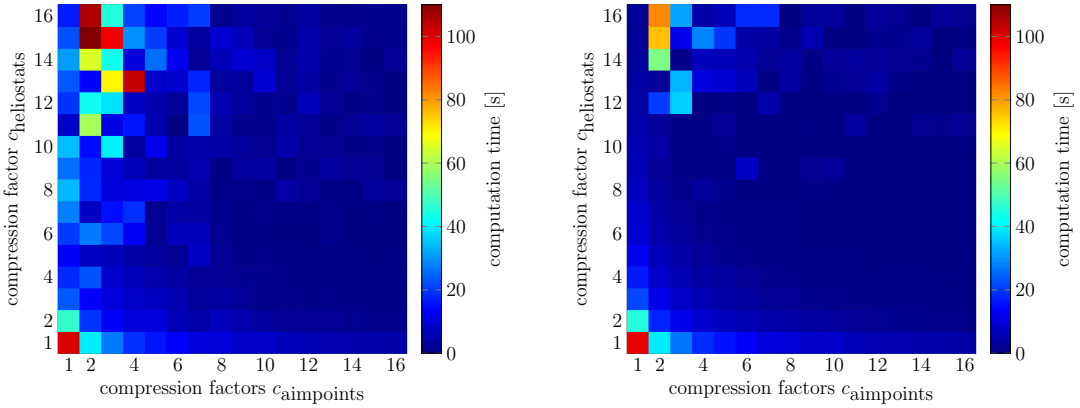


Figure 28: Normal computation time (left) and computation time with an initial solution (right) for PS10 in dependence of the heuristic compression factors $c_{\text{heliostats}}$ and $c_{\text{aimpoints}}$.

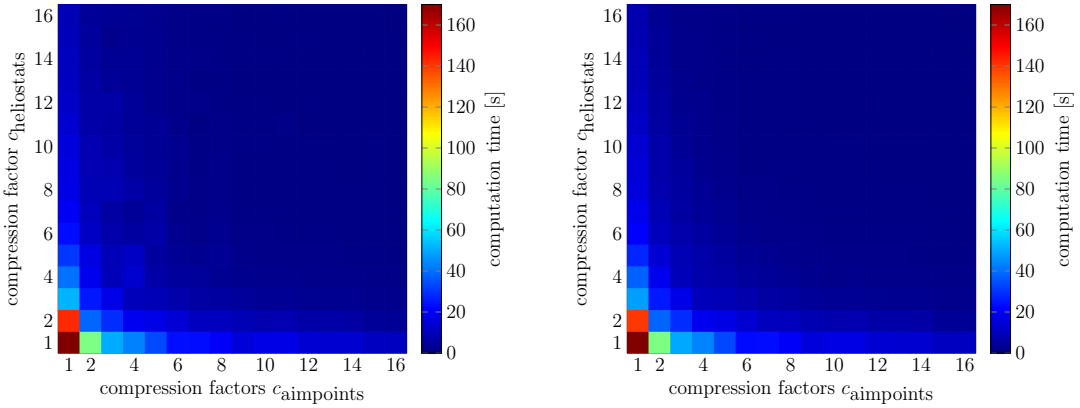


Figure 29: Normal computation time (left) and computation time with an initial solution (right) for Gemasolar in dependence of the heuristic compression factors $c_{\text{heliostats}}$ and $c_{\text{aimpoints}}$.

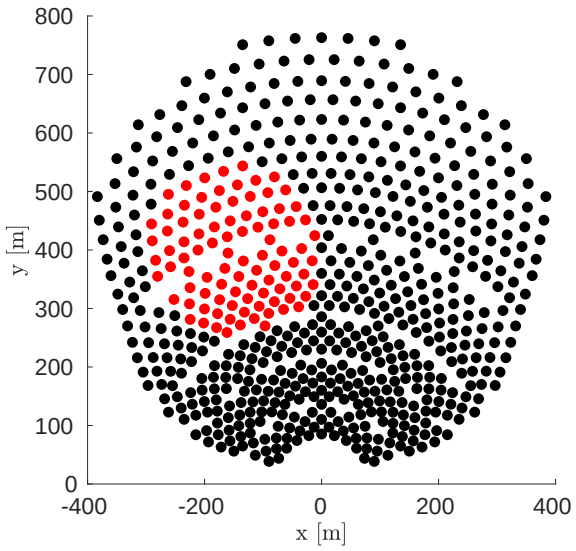


Figure 30: Disabled heliostats for the warm start testcase on PS10 are drawn red.

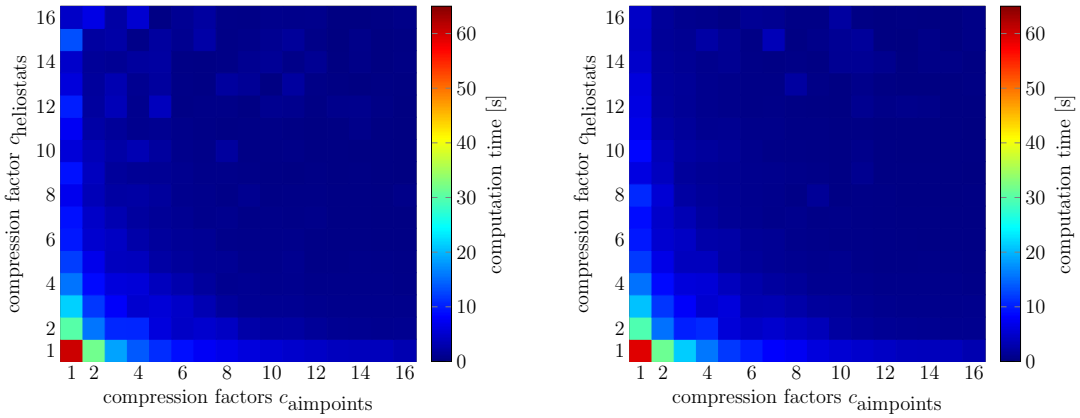


Figure 31: Normal computation time (left) and computation time with an initial solution (right) for PS10 with some disabled heliostats in dependence of the heuristic compression factors $c_{\text{heliostats}}$ and $c_{\text{aimpoints}}$. The initial solution is the solution from the original problem without disabled heliostats.

7 Conclusion and future work

Within this work, we refined an optical model to compute the heat flux at the receiver surface for any heliostat alignment. To compute heat flux images as efficiently as possible, the extended HFLCAL method was improved.

Afterwards, the aiming strategy problem was formulated as an integer linear problem. We developed a C++ framework for the solution of this integer linear problem, using different existing solver softwares. The comparison and tuning of these solvers showed that `Gurobi` as well as `glpk` and `SCIP` can be used to solve the aiming strategy problem for medium sized central receiver systems, while `lpsolve` and `coin-or` cannot deliver a solution in feasible time.

Different heuristic approaches were developed to reduce the size of the problem by either letting groups of heliostats only aim at the same aim point or reducing the number of available aim points per heliostat. These heuristic approaches showed promising results on real central receiver systems, as the computation time was vastly reduced, while keeping the solution quality above a very high percentage of the best obtainable solution. The optimal heuristic parameters vary between the different testcases, but the tendency of strongly decreasing computation times always stayed similar.

The heuristic approaches lead to even better results the larger the central receiver system is, in terms of number of heliostats and aim points as well as measurement points, which makes them viable for many new CRS. In the largest real testcase, for which a solution without the use of heuristics could not be obtained due to hardware limitations, the use of heuristics reduces the runtime to less than 20 seconds while keeping the solution quality above 99% of the best one achieved.

For possible future work we propose the addition of a GPU raytracer for the heliostat heat flux image generation, as well as support for arbitrary receiver surface shapes. Additionally, one could investigate warm starting the optimization with old solutions possibly violating safety constraints but which solve a problem very similar to the new one. An approach for this could be to disable selected critical heliostats in the old solution to make it feasible.

8 Appendix

Sun parameters	Symbol	Unit	PS10 - flat	PS10	Gemasolar	Abengoa
Sunshape error	σ^{sunshape}	mrad	2.51	2.51	2.51	2.35
Direct normal irradiation	DNI	$\frac{\text{W}}{\text{m}^2}$	1050	1050	1050	1016
Azimuth angle	γ_{solar}	deg	0	0	0	160.9
Elevation angle	θ_{solar}	deg	14	14	14	14.4
Heliostat parameters						
# heliostats	$\|H\|$		624	624	2650	8600
Mirror surface area	A_h	m^2	121	121	115.72	138.672
Pedestal height	-	m	5.17	5.17	5.675	7.095
Optical error	σ^{optical}	mrad	2.9	2.9	2.9	1.22
Reflectivity	$\eta_h^{\text{reflectivity}}$		0.88	0.88	0.93	0.91956
Tracking error horizontal	$\sigma^{\text{tracking, hor}}$	mrad	1.3	1.3	1.3	0.71
Tracking error vertical	$\sigma^{\text{tracking, vert}}$	mrad	2.6	2.6	2.6	0.38
Receiver parameters						
Type			flat	cavity	cylindric	cylindric
Tilt angle	θ_{rec}	deg	11.5	-	-	-
Circle fraction	p_{rec}		-	4/9	-	-
Receiver pedestal height	h_{ped}	m	-	2	-	-
Receiver diameter	d_{rec}	m	14	14	8.1	16.2
Receiver height	h_{rec}	m	12	12	10.6	18.5
Tower height	h_{tower}	m	115	115	140	229.25
Tower height above receiver	h_{top}	m	2.74	2.74	2.74	18.5
Receiver discretization						
# horizontal aim points	n_a^{horiz}		15	15	15	1777
# vertical aim points	n_a^{vert}		5	5	5	
# horizontal meas. points	n_m^{horiz}		15 + 2	15 + 2	15	32
# vertical meas. points	n_m^{vert}		5 + 2	9 + 2	9	7
Optimization parameters						
Allowed heat flux	q^{AFD}	$\frac{\text{kW}}{\text{m}^2}$	270	293	1100	1290
Allowed heat flux (shield)	q^{shield}	$\frac{\text{kW}}{\text{m}^2}$	250	250	500	500
Relative allowed flux	q_{rel}^m		1 (const)	1 (const)	1 (const)	0.5 - 1
Solver solution gap			1%	1%	0.7%	0.5%

Table 4: Central receiver system data for PS10 (flat / cavity), Gemasolar and the Abengoa CRS. Empty entries are not needed.

Component	Details
Processor	intel i5 6500, 4x3.2 GHz (1 core used)
RAM	16 Gb

Table 5: Hardware data

References

[1] Renewable capacity statistics 2020. https://www.irena.org/-/media/Files/IRENA/Agency/Publication/2020/Mar/IRENA_RE_Capacity_Statistics_2020.pdf, 2019. last visited on 2020-08-30.

[2] Thomas Ashley, Emilio Carrizosa, and Enrique Fernández-Cara. Optimisation of aiming strategies in solar power tower plants. *Energy*, 137:285–291, 2017.

[3] Marco Astolfi, Marco Binotti, Simone Mazzola, Luca Zanellato, and Giampaolo Manzolini. Heliostat aiming point optimization for external tower receiver. *Solar Energy*, 157:1114–1129, 2017.

[4] Boris Belhomme, Robert Pitz-Paal, and Peter Schwarzbözl. Optimization of heliostat aim point selection for central receiver systems based on the ant colony optimization metaheuristic. *Journal of solar energy engineering*, 136(1):011005, 2014.

[5] Saeb M Besarati, D Yogi Goswami, and Elias K Stefanakos. Optimal heliostat aiming strategy for uniform distribution of heat flux on the receiver of a solar power tower plant. *Energy Conversion and Management*, 84:234–243, 2014.

[6] Juan Burgaleta, Santiago Arias, and Diego Ramirez. Gemasolar, the first tower thermosolar commercial plant with molten salt storage. *Solarpaces*, 69, 01 2011.

[7] TA Dellin, MJ Fish, and CL Yang. User's manual for delsol2: a computer code for calculating the optical performance and optimal system design for solar-thermal central-receiver plants. Technical report, Sandia National Labs., Albuquerque, NM (USA); Sandia National Labs., Livermore, CA (USA), 1981.

[8] Ambros Gleixner, Michael Bastubbe, Leon Eifler, Tristan Gally, Gerald Gamrath, Robert Lion Gottwald, Gregor Hendel, Christopher Hojny, Thorsten Koch, Marco E. Lübbecke, Stephen J. Maher, Matthias Miltenberger, Benjamin Müller, Marc E. Pfetsch, Christian Puchert, Daniel Rehfeldt, Franziska Schlösser, Christoph Schubert, Felipe Serrano, Yuji Shinano, Jan Merlin Viernickel, Matthias Walter, Fabian Wegscheider, Jonas T. Witt, and Jakob Witzig. The SCIP Optimization Suite 6.0. Technical report, Optimization Online, July 2018. URL http://www.optimization-online.org/DB_HTML/2018/07/6692.html.

[9] Ambros Gleixner, Michael Bastubbe, Leon Eifler, Tristan Gally, Gerald Gamrath, Robert Lion Gottwald, Gregor Hendel, Christopher Hojny, Thorsten Koch, Marco E. Lübbecke, Stephen J. Maher, Matthias Miltenberger, Benjamin Müller, Marc E. Pfetsch, Christian Puchert, Daniel Rehfeldt, Franziska Schlösser, Christoph Schubert, Felipe Serrano, Yuji Shinano, Jan Merlin Viernickel, Matthias Walter, Fabian Wegscheider, Jonas T. Witt, and Jakob Witzig. The SCIP Optimization Suite 6.0. ZIB-Report 18-26, Zuse Institute Berlin, July 2018. URL http://nbn-resolving.de/urn:nbn:de:0297-zib-69361.

[10] GNU. Gnu linear programming kit, version 4.65, 2012. URL https://www.gnu.org/software/glpk/glpk.html. last visited on 2020-02-23.

[11] LLC Gurobi Optimization. Gurobi optimizer reference manual, 2020. URL http://www.gurobi.com. last visited on 2020-02-23.

[12] Soteris A. Kalogirou. Chapter ten - solar thermal power systems. In Soteris A. Kalogirou, editor, *Solar Energy Engineering*, pages 521 – 552. Academic Press, Boston, 2009. ISBN 978-0-12-374501-9. doi: https://doi.org/10.1016/B978-0-12-374501-9.00010-8. URL http://www.sciencedirect.com/science/article/pii/B9780123745019000108.

[13] Fynn Kepp. *Robust Optimization of Aiming Strategies of Heliostats in Solar Tower Power Plants*. 2018.

[14] Sascha Kuhnke, Pascal Richter, Fynn Kepp, Jeff Cumpston, Arie MCA Koster, and Christina Büsing. Robust optimal aiming strategies in central receiver systems. *Renewable Energy*, 2019.

[15] R. Lougee-Heimer. The common optimization interface for operations research: Promoting open-source software in the operations research community. *IBM Journal of Research and Development*, 47(1):57–66, Jan 2003. ISSN 0018-8646. doi: 10.1147/rd.471.0057.

[16] Manuel López-Ibáñez, Jérémie Dubois-Lacoste, Leslie Pérez Cáceres, Thomas Stützle, and Mauro Birattari. The irace package: Iterated racing for automatic algorithm configuration. *Operations Research Perspectives*, 3:43–58, 2016. doi: 10.1016/j.orp.2016.09.002.

[17] Peter Notebaert Michel Berkelaar, Kjell Eikland, 2016. URL http://lpsolve.sourceforge.net/5.5/. last visited on 2020-02-23.

[18] Corey J Noone, Manuel Torrilhon, and Alexander Mitsos. Heliostat field optimization: A new computationally efficient model and biomimetic layout. *Solar Energy*, 86(2):792–803, 2012.

[19] Rafael Osuna, Rafael Olavarrıa, Rafael Morillo, Marcelino Sánchez, Felipe Cantero, Valerio Fernández-Quero, Pedro Robles, T López, Antonio Esteban, Francisco Céron, et al. Ps10, construction of a 11mw solar thermal tower plant in seville, spain. In *Solar-PACES Conference, Seville, Spain, June*, pages 20–23, 2006.

[20] Pascal Richter. *Simulation and optimization of solar thermal power plants*. PhD thesis, RWTH Aachen University, 2017.

[21] Pascal Richter, Fynn Kepp, Christina Büsing, and Sascha Kuhnke. Optimization of robust aiming strategies in solar tower power plants. *AIP Conference Proceedings*, 2126(1):030045, 2019. doi: 10.1063/1.5117557. URL https://aip.scitation.org/doi/abs/10.1063/1.5117557.

[22] Peter Schwarzbözl, Robert Pitz-Paal, and Mark Schmitz. Visual hflcal - a software tool for layout and optimisation of heliostat fields. 09 2009.