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

Heuristic Layout Optimization of Central Receiver Systems

Heuristische Layout Optimierung von Solarturm-Kraftwerken

Masterarbeit
Informatik

Dezember 2022

Vorgelegt von Presented by	Florian Hövelmann Matrikelnummer: 369069 florian.hoevelmann@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 Modellierung und Verifikation RWTH Aachen University
Betreuer Supervisor	Dr. rer. nat. Pascal Richter Lehr- und Forschungsgebiet: Theorie der hybriden Systeme RWTH Aachen University

Contents

1 Introduction

Since the climate change is starting to show its drastic consequences, renewable energies are more important than ever. In Germany the 34.9% of the electrical energy comes from renewable sources which should increase to 40 to 45% by 2025 [26]. But there are of course a lot of challenges to overcome. One criticism is the unreliable energy production of most renewable energies. However, central receiver systems (CRS) offer the opportunity to store thermal energy and thus generate electricity even if no sun is shining. CRS consist of mirrors on heliostats reflecting the sunlight onto a receiver. An image of the Gemasolar [11] in Spain with 2650 heliostats is shown in Figure 1. The radiation heats up a liquid that then boils water to generate electricity at a turbine. The thermal energy of the liquid itself can be stored in tanks and used later.

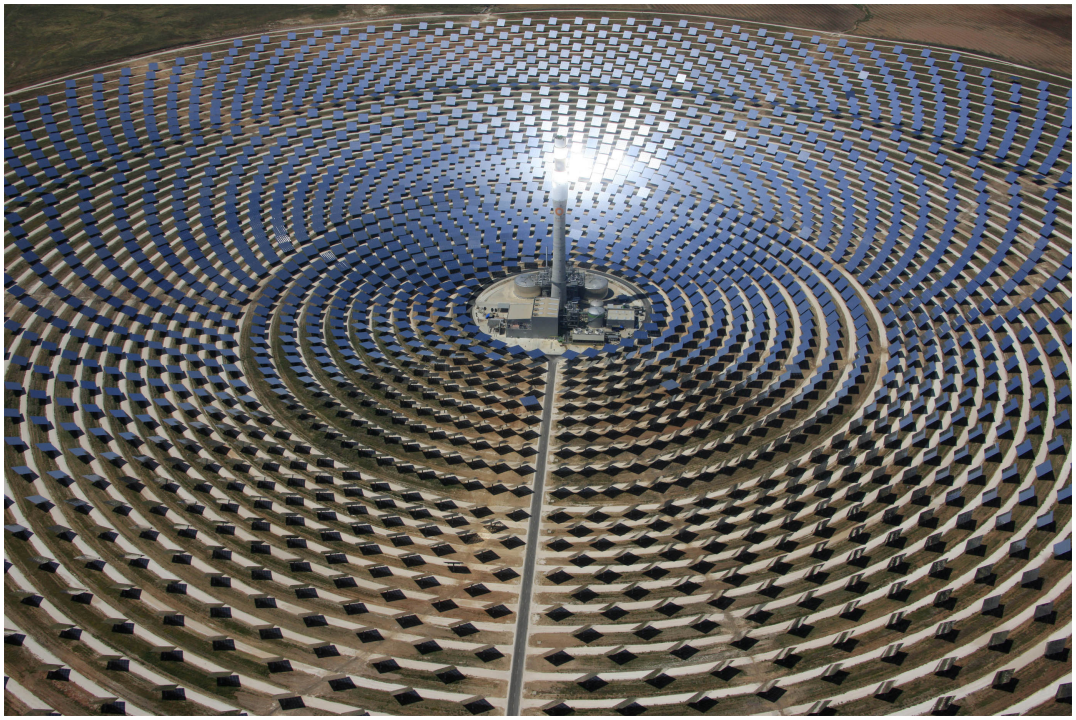


Figure 1: Image of the Gemasolar in Spain consisting of 2650 heliostats [11].

Desining and building a CRS is complex and expensive. Many aspects can significantly affect the efficiency of the plant. Therefore, it is crucial to estimate and optimize the energy production beforehand. An optimization of a new CRS is very cheap compared to the actual costs of the CRS. However, it requires an accurate model of the involved physics. Since many variants of the CRS are tested during optimization, the models also need to be fast. A lot of improvements have been made to the model as well as the optimization.

1.1 State of the art

Since the CRS modeling and the optimization is investigated in this thesis, the current state of the art of both is given in the following.

1.1.1 Modeling

The main differences between CRS simulation software concern the optical aspects of the plant. They include everything from the sunlight hitting the heliostat mirror, to the receiver collecting the radiation. The incoming radiation at the receiver is commonly computed by ray tracers. The ray tracing methods for central receiver systems can be divided into two categories, the non-deterministic Monte Carlo based ray tracers and the deterministic analytical ray tracers.

There exist several Monte Carlo ray tracers which mainly vary in the way how they generate solar rays. *Tonatiuh* [9], *MIRVAL* [36] and *SolTrace* [64] utilize a forward Monte Carlo ray tracing approach, where the rays are traced in forward direction from the sun to the heliostat onto the receiver. In general, the rays are generated on a plane above the heliostat. Improvements to this have been made by *STRAL* [1] and *TieSol* [30] which use the bidirection path tracing (BDPT) approach [45] where rays are generated on the heliostat surface. Due to the advantages of the bidirectional approach *SunFlower* [53] also implements two versions of such a ray tracer with different pseudo-random number generators.

Since Monte Carlo ray tracers rely on simulating a vast number of rays, they tend to have a long runtime. Analytical ray tracers, however, only use a few rays per heliostat and try to estimate the result based on a reflected flux. The fraction of flux hitting the receiver is computed by integration. *HELIOS* [8] can handle multiple flux functions. In contrast, *HFLCAL* [60] directly describes the flux on the receiver by a circular Gaussian distribution and evaluates it with a midpoint quadrature rule. *UNIZAR* [56] relies on the same integration method but describes the flux with the error function. *DELSOL* [34] uses a truncated expansion into Hermite polynomials as flux function. In one dimension the integral is evaluated analytically, while in the other dimension it is approximated with a 16-point Gaussian quadrature rule. The convolution ray tracer of *SunFlower* [53] uses a bivariate Gaussian distribution which is projected onto the image plane with a perspective projection. However, the integration method is what sets it apart from existing ray tracers. Here, a direct integration over the polygonal region representing the receiver is used which does not require a quadrature making it extremely fast [51]. In extend to this, the integrated convolution ray tracer improves the accuracy of the ray tracer without computational overhead.

A major improvement to the runtime of all ray tracers have been made by utilizing the GPU. Tools like *TieSol* [30], *QMCRT* [20] and *sbpRAY* [27] showed the benefits of parallelizing the ray tracers on the GPU. Likewise, for all ray tracers of *SunFlower* a GPU based version has been developed in [2].

1.1.2 Optimization

Since a CRS consists of many components, there are several optimization objectives. Most of them regard optical parts, as they are responsible for about 40% of the total losses [40]. Asselineau et al. [4] applied stochastic optimization and machine learning to find an optimal external receiver geometry. Similar methods have been applied to cavity receivers [3]. Another popular optimization technique in that area is particle swarm optimization [21] where particles represent sets of parameters. The particles share information and move according to specific rules in order to find an optimum of the objective function [40]. Genetic algorithms [57] also use parameter sets as candidates but optimize by mutation and selection. These optimization methods also have been applied to other optical aspects. For example, Wang et al. [62] used genetic algorithms to optimize the heliostat aiming strategy.

The heliostat field layout is another crucial aspect of the overall efficiency of a CRS. There are three main categories of heliostat field layout optimization methods. Pattern based methods, that place heliostats on a predefined pattern. They are described by a set of parameters which are then optimized. The patterns include simple cornfields [37], radial staggered arrangements [14], biomimetic spirals [43] and more. Optimization strategies like genetic algorithms [61], Nelder-Mead algorithm [47], or simple combinatorial searches [43] have been used to find the best set of parameters. However, the patterns typically generate more possible positions than needed. Thus, another optimization is required to find the best subset. Noone et al. [43] used the intuitive approach of simulating all heliostats and selecting the one with the highest efficiency. A more involved method has recently been proposed where a polygon defines the subset and each vertex is optimized with an evolutionary algorithm.

Free variable methods, on the other hand, directly optimize the heliostat positions and thus do not restrict them to a predefined pattern. As the search space is much larger than for pattern based methods, heuristics are used. The methods are based on genetic algorithms [35], particle swarm methods [22], evolutionary algorithm [52], and more. Other free variable algorithm use an initial solution and further optimize it by individually replacing the heliostats [10].

The last category includes field growth methods. As the name implies, they sequentially add heliostats to the field until the desired number is reached. The heliostats that maximize the overall efficiency are chosen at each step. Sánchez et al. [55] used a simplified model to calculate the efficiency at every position on a predefined grid and update the efficiencies to account for shading and blocking. To avoid placing heliostats in front of existing ones, the blocking effects are mirrored. Carrizosa et al. [12] avoid restricting the positions to a predefined grid by using a random set of positions to choose from at each step. However, it considers a much smaller set of possible positions at each step than in the grid based solution. Both of these methods approximate the efficiency at each position which might lead to a non optimal layout.

1.2 Contribution

In this thesis, the accuracy of all existing ray tracers is further enhanced and the analytical ray tracer accelerated with the help of extensive case studies. Additionally, all ray tracers are extended to incorporate more realistic sun shapes. It will be shown that the new sun shapes can accurately be approximated with Gaussians making it possible to use them in analytical ray tracer.

For the heliostat field layout optimization, the existing pattern optimization is tested on the GPU and further accelerated. Moreover, a new graph based field growth method has been developed. It is based on a new kind of ray tracing that has been implemented to all existing ray tracer of *SunFlower*. The new field growth method not only exactly evaluates the efficiency at each position, but also the efficiency loss introduced by placing a new heliostat. In combination, this leads to an optimal placement in the sense that replacing a single heliostat will never result in a more efficient layout. The exact algorithm is extended to more approximative methods to increase the number of possible positions. Furthermore, the proposed methods can also use patterns for the positions instead of a regular grid.

1.3 Outline

The thesis is structured as follows. Section 2 describes the components of a CRS and how they are modeled. A detailed description of the optical model is given in Section 2.1, as it is a crucial component for layout optimization. Section 3 describes the different optimization methods in *SunFlower* with the pattern-based approaches in Section 3.1 and the new graph based field growth method in Section 3.2. Comprehensive case studies investigating different aspects of the model and the optimization, are discussed in Section 4. A conclusion is drawn in Section 5.

2 Modeling solar central receiver systems

An accurate model of a central receiver system (CRS) is required to determine and later optimize its output. The CRS consists of different parts working together to generate electricity from collected solar radiation. These are considered by smaller individual models that will be discussed in the following. As shown in Figure 2, the optical model is the starting point and determines the amount of solar radiation hitting the receiver. A heat transfer fluid carries the thermal power to storage tanks which are described in the thermal and storage model. The thermal power is then converted into electricity, as depicted by the electrical model. The economical model captures all financial aspects of the CRS. Finally, a yearly operation of the CRS including different weather conditions is modeled by an annual integration.

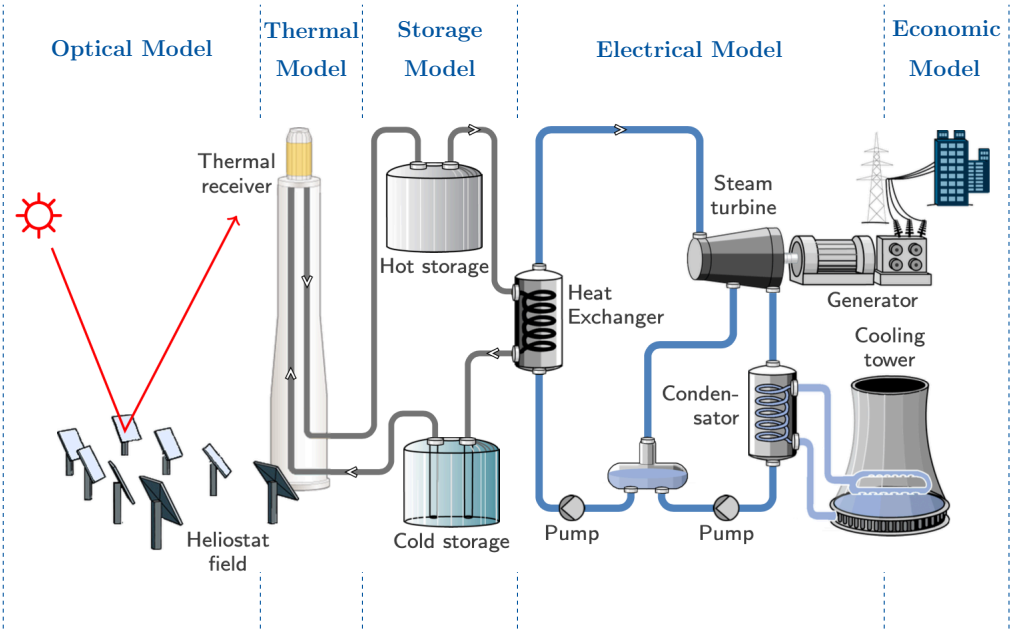


Figure 2: All components involved in modeling a CRS, taken from [25].

2.1 Optical model

The optical model is by far the computationally most expensive part when simulating central receiver systems. It includes every relevant aspect necessary to calculate the solar power reflected onto the receiver. However, the optical models used in the literature vary largely in terms of what type of systems can be represented as well as how the model is evaluated [36, 9, 64, 56]. Therefore, *SunFlower* aims to include various aspects present in current central receiver systems while offering a fast and accurate evaluation of the model. The presented model has been introduced by Richter et al. [50, 53] and extended in [25] and [29]. Further improvements and extensions have been made.

Environment The apparent movement of the sun is influenced by the geographical location of the CRS and thus needs to be specified. Additionally, the site boundaries as well as restricted areas can be given by a set of geographical coordinates that define a polygon. Topographical information about the site can automatically be generated from the Space Shuttle Radar Topography Mission [23]. All other positions and directions are defined in a Cartesian coordinate system where the x, y and z coordinates point toward east, north, and the sky, respectively.

From the suns azimuth γ_{solar} and altitude θ_{solar}, the solar vector $\vec{\tau}_{\text{solar}}$ can be computed as shown in Equation (1). Another important aspect of the sun is the direct normal irradiation I_{DNI} which describes the incoming radiation. It can either directly be defined or calculated using the Meteorological Radiation Model [32].

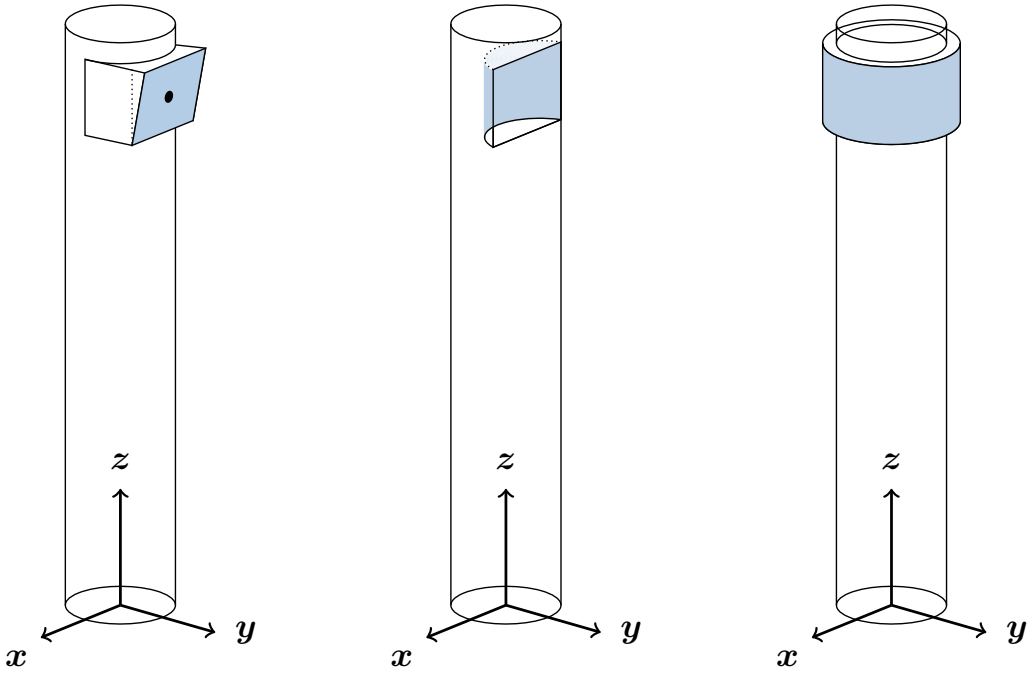
$$\vec{\tau}_{solar} = \begin{pmatrix} \sin\left(-\gamma_{\text{solar}}\right) \cdot -\cos\left(\theta_{\text{solar}}\right) \\ \cos\left(-\gamma_{\text{solar}}\right) \cdot \cos\left(\theta_{\text{solar}}\right) \\ \sin\left(\theta_{\text{solar}}\right) \end{pmatrix} \tag{1}$$

Heliostat Heliostats reflect the sunlight onto the receiver by tracking the sun and aligning their mirrors accordingly. In our model, two types of heliostat shapes can be represented. Rectangular heliostats and heliostats shaped like regular polygons. The later is becoming more important in recent power plants [7]. Both types consist of smaller facets that have either a rectangular or a triangular shape. For the alignment of the facet onto the scaffold, on-axis or off-axis canting is used [25]. Heliostats typically aim at a central point of the receiver. However, more involved aiming strategies trying to maximize the heat transfer to the receiver are also possible [33]. Since the vector of the incoming sun $\vec{\tau}_{\text{solar}}$ as well as the desired reflection vector $\vec{r}$ is known, the required normal $\vec{n}$ of the heliostat is given by

$$\vec{n} = \frac{\vec{r} + \vec{\tau}_{\text{solar}}}{|\vec{r} + \vec{\tau}_{\text{solar}}|} \tag{2}$$

Receiver After the solar radiation is reflected by the heliostats, it is collected at the receiver and turned into heat. To minimize blocking and shading effects, the receiver is mounted onto a tower at a certain height. Three types of receivers can be represented in our model, see Figure 3. A flat tilted receiver has a tilted rectangular area which is a simplification of the receiver Planta Solar 10 (PS10) [44]. A cylindrical cavity receiver consisting of multiple small rectangular receiver panels that are horizontally aligned according to parts of a regular polygon. The cavity receiver is a more accurate representation of the receiver at the PS10. Lastly, the cylindrical external receiver, as used in the power plant Gemasolar [11], has the shape of a regular polygon that is situated on the outside of the tower.

Optical losses Some solar radiation hitting the heliostats is lost by the time it reaches the receiver. The different types of losses will be discussed in the following.



(a) Flat tilted cavity receiver (b) Cylindric cavity receiver (c) Cylindric external receiver

Figure 3: All three receiver types included in the optical model. The Figure is derived from Richter [50, p. 11].

The **cosine effect** accounts for the tilted alignment of the heliostat to the incoming radiation and leads to the biggest loss. As stated at the beginning of this section, the radiation of the sun is described by the direct normal irradiation which gives the solar radiation received by an area perpendicular to the incoming sunlight. Since the heliostat is tilted, its area perpendicular to the sunlight is reduced by the cosine efficiency

$$\eta_{\cos} = \langle \vec{\tau}_{\text{solar}}, \vec{n} \rangle, \tag{3}$$

where $\langle \cdot, \cdot \rangle$ is the scalar product of two vectors.

Another effect having a major impact on the overall efficiency is called **blocking and shading**. There are different aspects responsible for the blocking and shading of solar rays. Heliostats casting a shadow on other heliostats or blocking their reflected rays have the biggest impact on the blocking and shading efficiency $\eta_{\text{s\&b}}$. However, the tower can also cast a shadow onto the heliostats and for a cavity receiver, some solar rays are blocked by the receiver opening window, see Figure 3b.

Heliostat reflectivity η_{ref} accounts for the absorption of the solar rays as well as the diffuse reflection that will not hit the receiver. In the literature, this effect is typically modeled by a constant value [48].

When the sunlight travels through the atmosphere it interacts with the molecules and thus losses parts of its power. The **atmospheric attenuation** efficiency η_{aa} models the remaining solar power using a formula derived by Schmitz et al. [58]

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

with d as the distance of the heliostat to the receiver.

The **optical errors** of the ideal reflected ray are the most difficult part to account for as they involve uncertainties. Since optical errors determine the direction of an actual solar ray, they influence the blocking efficiency as well as the intercept efficiency η_{int}. The latter describes which portion of the reflected solar power will hit the receiver. Three errors influence the direction of an actual solar ray, see Figure 4. The tracking error models the deviation of the heliostat alignment to the desired alignment. As there are typically two tracking axis, we consider a vertical and a horizontal tracking error which are both modeled by Gaussian distributions with standard deviations $\sigma^{\text{ver}}_{\text{tracking}}$, $\sigma^{\text{hor}}_{\text{tracking}}$, respectively. Another error influencing the normal of the heliostat is the roughness of its mirrors. The resulting slope error can deviate based on the position on the mirror. It is modeled using Gaussian distributions for the vertical and horizontal direction with standard deviations $\sigma^{\text{ver}}_{\text{slope}}(x, y)$ and $\sigma^{\text{hor}}_{\text{slope}}(x, y)$. Lastly, the shape of the sun has to be taken into account which is often done using a circular Gaussian distribution with standard deviation σ_{sun} [49]. But other sun shapes are also possible and will be discussed later in this section.

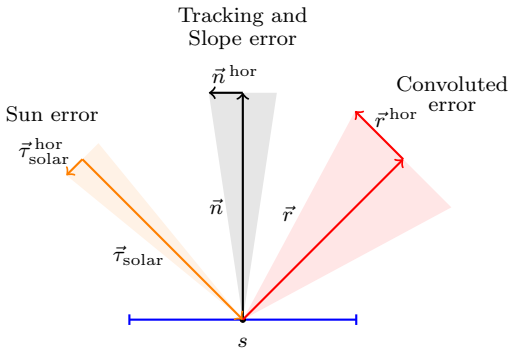


Figure 4: Horizontal disturbances of a reflected solar ray (yellow and red) on the surface of a heliostat (blue) [29].

All of these errors can be convoluted into a single error as stated in [58]. However,

slight differences were observed when using the convoluted error compared to modeling all errors individually. As a reason, we found that the influence of the normal perturbations onto the reflected ray depends on the incident angle of the incoming ray which is not included in [58]. Consider a solar ray hitting the mirror at a certain incident angle as shown by the orange vector in Figure 5a. As proven in the following, a perturbation of the normal in horizontal direction $\vec{n}^{\text{hor}}$ will keep the distance to the plane spanned by $\vec{n}^{\text{hor}}$ and $\vec{n}$ constant, see Figure 5b. Without loss of generality, assume $\vec{n} = (0, 0, 1)^T$, $\vec{n}^{\text{hor}} = (1, 0, 0)^T$ and the incoming ray $\vec{\tau}_{\text{solar}} = (\tau_x, \tau_y, \tau_z)^T$. From the law of reflection the unperturbed reflected ray $\vec{r}$ can be calculated to

$$\vec{r} = \vec{\tau}_{\text{solar}} - 2(\vec{\tau}_{\text{solar}}\vec{n})\vec{n} = \vec{\tau}_{\text{solar}} - 2(0, 0, \tau_z)^T. \tag{5}$$

While a perturbation of the normal vector in horizontal direction by a factor of α results in

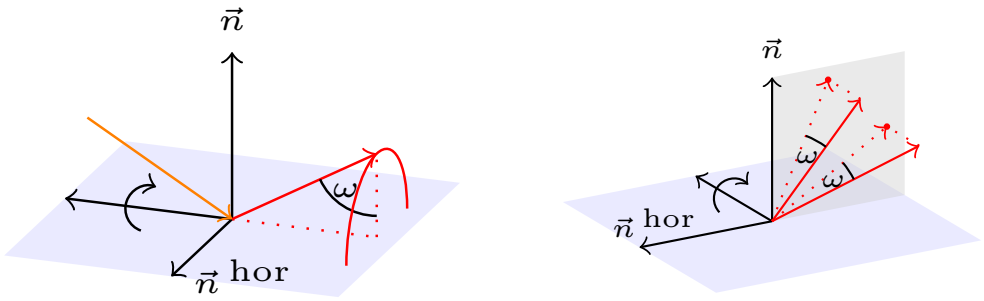
$$\begin{aligned}
\vec{n}_{\text{pert}} &= \frac{\vec{n} + \alpha\vec{n}^{\text{hor}}}{\|\vec{n} + \alpha\vec{n}^{\text{hor}}\|} = (\vec{n} + \alpha\vec{n}^{\text{hor}})\beta \\
\vec{r}_{\text{pert}} &= \vec{\tau}_{\text{solar}} - 2(\vec{\tau}_{\text{solar}}(\vec{n} + \alpha\vec{n}^{\text{hor}})\beta)(\vec{n} + \alpha\vec{n}^{\text{hor}})\beta \\
&= \vec{\tau}_{\text{solar}} - 2\beta^2(\vec{\tau}_{\text{solar}}\vec{n}\vec{n} + \vec{\tau}_{\text{solar}}\vec{n}\alpha\vec{n}^{\text{hor}} + \vec{\tau}_{\text{solar}}\alpha\vec{n}^{\text{hor}}\vec{n} + \vec{\tau}_{\text{solar}}\alpha\vec{n}^{\text{hor}}\alpha\vec{n}^{\text{hor}}) \\
&= \vec{\tau}_{\text{solar}} - 2\beta^2(\alpha\tau_z + \alpha^2\tau_x, 0, \tau_z + \alpha\tau_x)^T,
\end{aligned} \tag{6}$$

with $\beta = \|\vec{n} + \alpha\vec{n}^{\text{hor}}\|^{-1}$ as the normalization factor.

Both $\vec{r}$ and $\vec{r}_{\text{pert}}$ have the same y-coordinate and thus an equal distance to the xz-plane spanned by $\vec{n}^{\text{hor}}$ and $\vec{n}$. So the angle $\omega^{\text{hor}} = \omega$, shown in Figure 5b, between the reflected ray and its projection onto the xz-plane also remains unchanged. A normal perturbation in horizontal direction can hence be viewed as a rotation of the reflected vector around the perpendicular axis $\vec{n}^{\text{ver}}$, as illustrated. This causes the reflected ray to move around a circle, see Figure 5a. Smaller circle radii lead to less distance traversed by the reflected ray. From the geometry of the problem, it follows that the angle between the reflected ray and the plane of the circle is also given by ω^{hor}. Therefore, the radius of the circle is $\cos\omega^{\text{hor}}$. As the perturbed normal vector rotates around a circle of radius one, the distance traversed by the reflected vector is reduced by a factor of $\cos\omega^{\text{hor}}$. A similar argument applies for the vertical deviation. Additionally, a perturbation of the normal has twice as much impact on the reflected ray as an equivalent perturbation of the incoming sun ray, leading to a convoluted error of

$$
\begin{aligned}
\sigma^{\text{hor}} &= \sqrt{(\sigma_{\text{sun}})^2 + (2\cos\omega^{\text{hor}}\sigma_{\vec{n}}^{hor})^2} \\
&= \sqrt{(\sigma_{\text{sun}})^2 + (2\cos\omega^{\text{hor}})^2((\sigma_{\text{tracking}}^{\text{hor}})^2 + (\sigma_{\text{slope}}^{\text{hor}}(x,y))^2)}, \\
\sigma^{\text{ver}} &= \sqrt{(\sigma_{\text{sun}})^2 + (2\cos\omega^{\text{ver}}\sigma_{\vec{n}}^{ver})^2} \\
&= \sqrt{(\sigma_{\text{sun}})^2 + (2\cos\omega^{\text{ver}})^2((\sigma_{\text{tracking}}^{\text{ver}})^2 + (\sigma_{\text{slope}}^{\text{ver}}(x,y))^2)},
\end{aligned}
\tag{7}
$$

with $\sigma_{\vec{n}}^{hor} = \sqrt{(\sigma_{\text{tracking}}^{\text{hor}})^2 + (\sigma_{\text{slope}}^{\text{hor}}(x,y))^2}$ as the convoluted normal perturbation in horizontal direction and $\sigma_{\vec{n}}^{ver} = \sqrt{(\sigma_{\text{tracking}}^{\text{ver}})^2 + (\sigma_{\text{slope}}^{\text{ver}}(x,y))^2}$ for the vertical direction.



(a) The circle that the reflected ray is rotating around when being perturbed in horizontal direction.

(b) The constant angle $\omega = \omega^{\text{hor}}$ when perturbing the normal vector in horizontal direction.

Figure 5: Illustrations for the derivation of Equation (7). The mirror surface is shown in blue, the incoming sun ray in orange, the reflected ray in red, and the plane spanned by $\vec{n}$ and $\vec{n}^{\text{hor}}$ in gray.

Finally, the direction of the convoluted ray perturbation has to be calculated. So far the horizontal and vertical disturbances were given in terms of the normal vector. To obtain the corresponding directions for the reflected ray, the perturbation directions of the normal has to be rotated as illustrated in Figure 6. The rotation is defined by the angle β and the axis $\vec{a}$ being perpendicular to the normal and ray.

Sun shapes As stated earlier, other distributions that more accurately represent the sun shape than a circular Gaussian can also be modeled. Three new types of distributions were introduced for this purpose, all of which are circular symmetric. Namely, the Buie [63], Pillbox [63], and custom distributions. For the Buie sunshape the following probability density function was used.

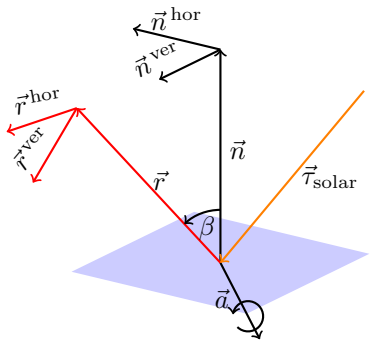


Figure 6: Illustration on how to compute the perturbation direction of the reflected ray from the perturbation of the normal [29].

$$\mathrm{buie}(\alpha, \mathrm{csr}) = \begin{cases} \frac{\cos 0.326 \cdot \alpha}{\cos 0.308 \cdot \alpha} & , \alpha \leq 4.65 \\ \alpha^{\gamma} \exp(\kappa) & , \mathrm{otherwise} \end{cases}, \tag{8}$$

$$\gamma = 2.2 \ln(0.52 \cdot \mathrm{csr}) \mathrm{csr}^{0.43} - 0.1, \kappa = 0.9 \ln(13.5 \cdot \mathrm{csr}) \mathrm{csr}^{-0.3} \tag{9}$$

with *alpha* as the angle in milliradians and csr as the circumsolar ratio [63]. The probability density function of the Pillbox distribution is much simpler and given by

$$\mathrm{pillbox}(\alpha, \mathrm{sw}) = \begin{cases} 1 & , \alpha \leq \mathrm{sw} \\ 0 & , \mathrm{otherwise} \end{cases}, \tag{10}$$

with sw as the sun width in milliradians. Lastly, the custom sun shape is defined as a stepwise linear probability density function which gets normalized afterwards.

All optical aspects described so far need to be evaluated using a ray tracer. *SunFlower* contains different types of ray tracers as described in the following.

2.1.1 Monte Carlo ray tracer

The task of a ray tracer is to calculate the total intercept power P_{int} at the receiver for a given direct normal irradiation I_{DNI} and sun position. To do so, the ray tracing techniques presented in the following generate representative rays on the surface of the mirror in a bidirectional ray tracing manner. Shading and blocking computations are sped up by precomputing a simplified version of the ray-heliostat interaction. As a result each heliostat has a set of potential blocking and shading heliostats [29]. The facet of a heliostat is evenly discretized into cells each with an intercept power of

$$P_{\text{int,cell}} = \underbrace{I_{\text{DNI}}\ A_{\text{cell}}\ \eta_{\text{cos}}\ \eta_{\text{ref}}\ \eta_{\text{aa}}}_{=:P_{\text{reflected}}}\ \eta_{\text{sb}}\ \eta_{\text{int}}. \tag{11}$$

with A_{cell} as the area of the heliostat cell and $P_{\text{reflected}}$ as the total power reflected by the cell.

For the computation of the total intercept power, there are three main challenges

- one ray is used to represent photon interactions of a certain cell,
- evaluation of the ray disturbance,
- computation of shading and blocking effects.

In the following the bidirectional Monte Carlo ray tracer of *SunFlower* is introduced [29] and extended to the new sun shapes. The Monte Carlo-based ray tracers are straightforward techniques to compute the concentrated solar power at the receiver. They are suitable for complex receiver geometries where most analytical methods are not applicable anymore [39]. But since they rely on the law of large numbers, a vast number of rays need to be simulated to achieve accuracy and reduce fluctuations. Therefore, Monte Carlo ray tracers tend to have a long processing time when an accurate result is needed.

A bidirectional-Monte Carlo ray tracer simulates a perturbed version of the ideal reflected ray, see Figure 7. For this, random numbers are generated from normal distributions with standard deviations as in Equation (7).

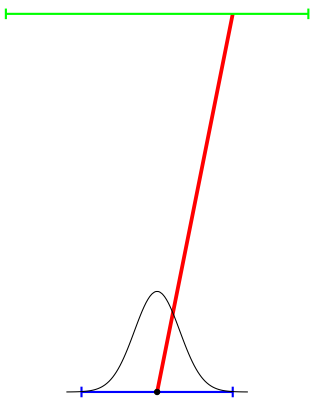


Figure 7: Illustration of the bidirectional-Monte Carlo ray tracing principle. The heliostat is shown in blue, the receiver in green, the Gaussian disturbance in black, and the generated ray in red [29].

The perturbed ray is now checked or shading and blocking. Therefore, it is traced in forward and backward direction. If the ray hits another heliostat, then η_{sb} for this

ray is set to 0, otherwise to 1. If the ray hits the receiver surface, then η_{int} is set to 1, otherwise to 0. Thus, only If the perturbed ray is not blocked nor shaded and it hits the receiver, its representative power $P_{\text{reflected}}$ from (11) is added to the total intercept power at the receiver.

Random generator for sun shapes As discussed in Section 2.1, our ray tracer should also be able to represent sun shapes other than a Gaussian. For an accurate representation first, the sun vector is perturbed, as specified by the new sun shapes, and then the reflected ray to account for tracking and slope errors. Therefore, a method to perturb a vector based on the new sun shapes, which are defined by their profile is needed. A two dimensional perturbation is calculated using polar coordinates. Since the sun shapes are circular, the angle φ is drawn from the uniform distribution $[0, 2\pi]$. As described in [63], the original sun shape profile then needs to be modified as follows

$$L(\theta) = \frac{\hat{L}(\theta)\sin\theta\cos\theta}{\int_0^{\pi/2}\hat{L}(\theta)\sin\theta\cos\theta d\theta}, \tag{12}$$

with $\hat{L}$ as the sun shape profile. Samples can then be drawn from the distribution by approximating it with a piecewise linear distribution. The three dimensional perturbed vector then is given by $(\tan\theta\cos\varphi, \tan\theta\sin\varphi, 1)^T$ which gets converted into the local coordinate system of the sun ray to obtain the desired disturbed sun ray.

2.1.2 Analytic ray tracer

In contrast to the Monte Carlo methods, an analytical ray tracer calculates the intercept efficiency η_{int} in a deterministic manner. The ray tracers presented in the following are expanding on the works in [29]. Instead of generating several perturbed rays from a given distribution, the aim is to integrate the distribution around the perfect reflected ray. With this, an exact evaluation of the ray disturbance can be obtained which eliminates a central error source of the ray tracer. Thus far fewer rays are needed to achieve accurate results. Moreover, due to the deterministic nature of an analytical ray tracer there are no fluctuations in the results.

The distribution of a ray reflection is commonly represented by a two dimensional probability density function on a plane orthogonal to the ideal ray direction called the image plane. As described in (7) we model the errors perturbing the ray using two independent Gaussian distributions. Therefore, our two dimensional probability density function is given by

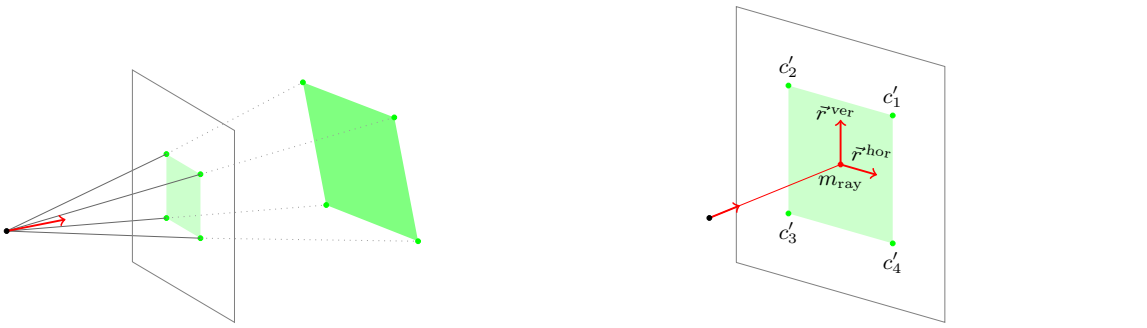
$$f(x,y) = \frac{1}{2\pi\sigma_{\text{span}}^{\text{hor}}\sigma_{\text{span}}^{\text{ver}}} \cdot \exp\left(-\frac{1}{2}\left(\left(\frac{x}{\sigma_{\text{span}}^{\text{hor}}}\right)^2 + \left(\frac{y}{\sigma_{\text{span}}^{\text{ver}}}\right)^2\right)\right), \tag{13}$$

with $\sigma_{\text{span}}^{\text{hor}} = \delta \cdot \tan\left(\sigma^{\text{hor}}\right)$, $\sigma_{\text{span}}^{\text{ver}} = \delta \cdot \tan\left(\sigma^{\text{ver}}\right)$ and δ as the distance of the ray origin to the image plane. Integrating over an area representing the receiver gives the probability

of hitting the receiver. In the following, two analytical ray tracers will be described that mostly differ in the function used to describe the ray disturbance. They have been introduced in [29] and are here generalized for the new sun shapes and refined to better account for shading and blocking effects.

Convolution method To compute the intercept efficiency η_{int}, the receiver is projected onto the image plane Ω using a perspective projection of each point of the receiver shape, see Figure 8a. Every projection requires only one matrix multiplication and gives the projected point in local coordinates of the image plane, see Figure 8b. Here, each corner c_i gets projected onto the plane resulting in c_i' whose coordinate system is defined by $\vec{r}^{\,\text{ver}}$, $\vec{r}^{\,\text{ver}}$ and the origin m_{ray}. The base vectors match those of the disturbances to the ideal ray which intersects the plane at m_{ray}. The resulting polygon D represents the region where each intersecting ray also hits the receiver surface. Note that the image plane can be set at an arbitrary distance since for all distances a valid representative region of the receiver exists. Thus the intercept efficiency η_{int} that the ray will hit the receiver is given by

$$\eta_{\text{int}} = \iint_D f(x, y)\ d\Omega. \tag{14}$$



(a) Perspective projection of a receiver shown in green that is required for the convolution method.

(b) Illustration of the local coordinate system on the image plane defined by $\vec{r}^{\,\text{ver}}$, $\vec{r}^{\,\text{ver}}$ and the origin m_{ray}.

Figure 8: Illustration on how to calculate the projected receiver area on the image plane where the two dimensional Gaussian is defined [29].

In Figure 9 the principle of the convolution ray tracer is demonstrated.

The basic idea to evaluate the integral in (14) is to integrate over the outer region of the polygon by dividing it into so called angular regions. With the extension in [18] arbitrary polygons can be handled. The integration over an angular region is partly not analytically possible. Therefore, a minmax polynomial fit to the non-integrable part of the original function is used [19]. The degree of the underlying polynomial directly effects the accuracy and the run-time. Further details can be found in [51].

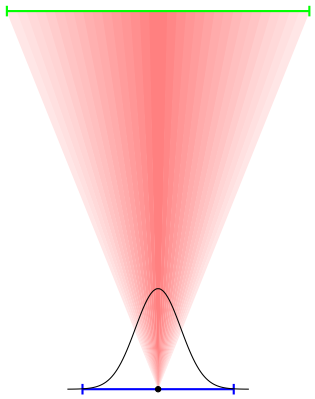


Figure 9: Illustration of our convolution ray tracer. The heliostat is shown in blue and the receiver in green. The shaded red cone illustrates an exact evaluation of the perturbation of the ideal reflected ray [29].

Commonly, analytical ray tracers like *HFLCAL* solve the integral in (14) with some kind of quadrature rule by discretizing the receiver into smaller pieces. The advantage of our approach is that it does not require any further discretization of the receiver since it directly integrates over the polygon. In such a way the overall run-time can be reduced significantly. Due to the gained speedup, our convolution ray tracer i
s capable of simulating multiple rays for each facet to improve accuracy.

For the shading and blocking efficiency η_{sb}, the convolution ray tracer uses the ideal non-perturbed ray and tests against neighboring heliostats. Furthermore, for cavity receivers partial blocking from the tower itself needs to be considered. To exclude these blocked regions, polygon D has to be cut into its truly visible parts. This can be done efficiently on the image plane itself using two cutting lines which also offer a fast way to test whether any cut is required.

As shown in [51], the convolution method is more accurate the more cells per facet are used. But increasing the number of rays effectively increases the computational costs. In the following, we present the integrated convolution ray tracer, which aims on integrating over several error cones, such that the accuracy increases without using more facet cells.

Integrated convolution method By deeper investigating the effect of evaluating multiple rays one can see that the simulation of an infinite amount of rays can be modeled by simply using a different probability density function. Consider a two dimensional example of the ray tracing problem as shown in Figure 10. When simulating two rays instead of one the original intercept efficiency η_{int} is split and the total power contribution to the receiver is given by

$$\eta_{\text{int}} = \frac{1}{2}\eta_{\text{int},1} + \frac{1}{2}\eta_{\text{int},2} = \frac{1}{2}\int_a^b \left(f(x-\mu_1) + f(x-\mu_2)\right)\ \mathrm{d}x, \tag{15}$$

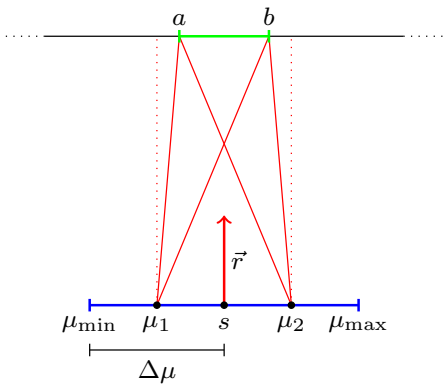


Figure 10: Simplified example of the power contribution of two heliostat cells shown in blue to the receiver spanning from a to b shown in green. μ_1 and μ_2 are the origins of the ideal reflected rays [29].

with f as the Gaussian distribution describing the disturbance, μ_1 and μ_2 as the origin of the two rays and $\eta_{\text{int},1}$ and $\eta_{\text{int},2}$ as the probability that they will hit the receiver. Here, a and b are coordinates of the projected receiver.

Extending the idea to an infinite amount of rays comes down to integrating over the cell length ℓ leading to

$$\begin{aligned}\int_a^b \lim_{n\to\infty} \frac{1}{n}\sum_{i=1}^{n} f(x-\mu_i)\,dx &= \int_a^b \frac{1}{\ell}\int_{-\ell/2}^{\ell/2} f(\mu-x)d\mu\,dx = \int_a^b F(x)\,dx \\ &= \int_a^b \frac{1}{2\ell}\left(\operatorname{erf}\left(\frac{\ell/2-x}{\sqrt{2}\sigma}\right) - \operatorname{erf}\left(\frac{-\ell/2-x}{\sqrt{2}\sigma}\right)\right)dx,\end{aligned} \tag{16}$$

with F as the new probability density function, see Figure 11. Here, μ_i are the new midpoints of the heliostat cells which are given by

$$\mu_i = \mu_{\min} + \frac{\Delta\mu}{2} + \Delta\mu(i-1),\ \Delta\mu = \frac{\mu_{\max}-\mu_{\min}}{n}. \tag{17}$$

Likewise a two dimensional version of the function F for our general ray tracing problem can be formulated. Given a cell of length ℓ and width w the new probability density function is given by

$$\begin{aligned}F(x,y) = \frac{1}{4\ell w}&\left(\operatorname{erf}\left(\frac{w/2-x}{\sqrt{2}\sigma_{\text{span}}^{\text{hor}}}\right) - \operatorname{erf}\left(\frac{-w/2-x}{\sqrt{2}\sigma_{\text{span}}^{\text{hor}}}\right)\right) \\ &\left(\operatorname{erf}\left(\frac{\ell/2-y}{\sqrt{2}\sigma_{\text{span}}^{\text{ver}}}\right) - \operatorname{erf}\left(\frac{-\ell/2-y}{\sqrt{2}\sigma_{\text{span}}^{\text{ver}}}\right)\right).\end{aligned} \tag{18}$$

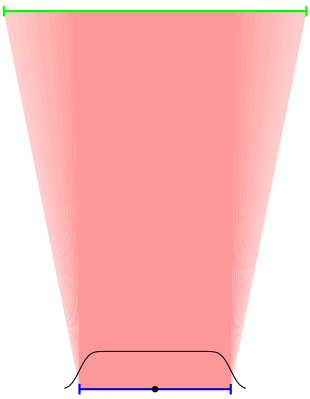


Figure 11: Illustration of our integrated convolution ray tracer [29].

Note that the function assumes the cell to be aligned with the axis of derivation. When this is not the case, a simplified version of the cell is used which is reduced to its effective reflected area and correctly aligned.

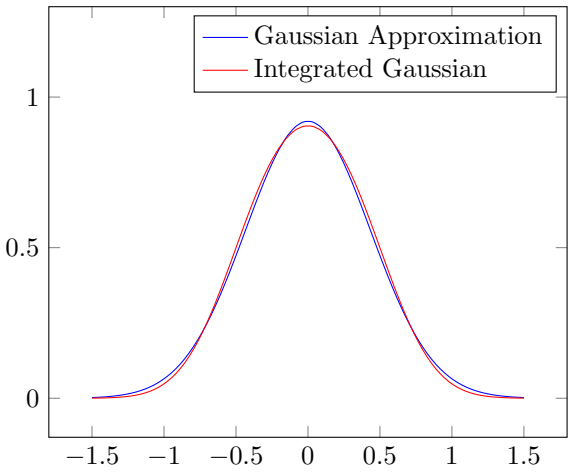


Figure 12: Approximation of the integrated Gaussian of a cell of length $l = 1$ m and standard deviation $\sigma_{\text{beam}} = 3$ mrad at a distance of 100 m [29].

But since an integral of the new probability density function over a polygon is hard to evaluate, an approximation is needed. To achieve an accurate approximation, we made use of the fact that with an increased distance to the image plane the function converges to a bivariate Gaussian function as in Equation (13). Here, the integrated version of the Gaussian distribution from each perturbation axis is approximated individually. An example from a cell of length $l = 1$ m and standard deviation $\sigma_{\text{beam}} = 3$ mrad with its approximation is shown in Figure 12. Therefore, given the length of the cell ℓ and standard derivation σ, a function that calculates the standard derivation σ_{appr} of the new Gaussian distribution approximating the integrated Gaussian distribution is

needed. Such a function is given in the following

$$\sigma_{\text{appr}} = \sigma \cdot g(\delta \frac{\sigma}{3\ell}), \quad with\ g(x) = \max\left(a \cdot \exp\left(\frac{b}{x+c}\right) + d, 1\right). \tag{19}$$

Coefficient	Value
a	$6.0654123395858638 \cdot 10^{-3}$
b	$1.0168091727137571 \cdot 10^{3}$
c	$1.3522384735784115 \cdot 10^{2}$
d	$9.8897960843463073 \cdot 10^{-1}$

Table 1: Coefficients of the sigma multiplier function g_{approx} from Equation (19).

Table 1 shows the values of the used constants in g. The derivation of g can be found here [29]. Note that g converges to one with the distance leaving the original standard derivation unchanged. A comparison between the integrated Gaussian distribution and an optimal Gaussian approximation as well as between an approximation using Equation (19) showed differences of less than 1.5% [29].

By approximating the integrated Gaussian distribution of each perturbation axis we can carry out the ray tracing the same way as in our Gaussian convolution method. However, since the rays are not originating from a single point anymore the use of a perspective projection to obtain the representative region of the receiver introduces some errors. But the error can be minimized by placing the image plane at the midpoint of the receiver.

A study done in [13] backs up the approximation method of our integrated convolution technique. They fitted the slope error in *HFLCAL* such that the solar flux matches the actual measured data. The resulting standard derivations increased for larger cell areas and decreased with the distance which corresponds to the behavior of our approximation.

To conclude, our integrated convolution ray tracer models the simulation of multiple rays using a simple adaptation of the bivariate Gaussian distribution.

Shading & Blocking So far, the integrated convolution ray tracer introduced in [29] addresses a fast and accurate computation of the intercept efficiency. However, shading and blocking efficiency calculations also have a large impact on the overall accuracy of the result. All other efficiencies are already accurately represented with only a single ray per heliostat facet, more details in Section 4.1.4. Therefore, additional

computational effort should only be spent on blocking and shading computations. Instead of evaluating only one ideal reflected ray for shading and blocking effects, our analytical ray tracer can handle multiple shading and blocking samples. A more exact evaluation would be to orthographically project shading and blocking heliostats onto the surface of the current heliostat cell and subtract overlapping areas. However, it is questionable whether this approach would be more performant.

Sun shape approximation Since our analytical ray tracers are formulated for a bivariate Gaussian distribution, other sun shapes can not be represented directly. An exact evaluation would require two new methods. First, a way to convolute the new sun shape with the bivariate gaussian for the tracking and slope error. Second, an efficient method to integrate the resulting two dimensional distribution over a polygonal region. Considering how much effort went into the development of a method to integrate a bivariate gaussian over a polygon, the second requirement is out of the scope of this thesis. But the same idea of the integrated convolution can be applied here. Instead of using the exact distribution directly, one can approximate the convoluted distribution with a bivariate Gaussian.

A straightforward approach to this problem is to simply approximate the sunshape before the convolution. Then the convolution can be carried out as before. However, as a simplified test case has shown, the resulting convoluted Gaussian has a different standard deviation than the flux profile obtained by separately perturbing the sun vector and heliostat normal. The test case consists of a rectangular 1 m^2 receiver discretized into 49x49 pieces and one ray with 100.000 samples originating 50 m away from the receiver. As a sun shape, the pillbox distribution with a sun width of 4.35 mrad was used combined with a tracking and slope error of 2 mrad. Since the standard deviation of the pillbox distribution is around 2.512 mrad, the simple approximation would result in a convoluted standard deviation of $\sqrt{2.512^2 + (2 \cdot 2)^2} \approx 4.72$ mrad. However, the standard deviation obtained from the experimental data is 4.59 mrad which is a difference of around 2.8%.

Thus, calculating the convolution and taking its standard deviation might lead to more accurate approximations. Since we already represent the profile of the new sun shapes as piecewise linear distributions, it seems reasonable to use a convolution method for one dimensional piecewise linear distributions. But even though the convolution of two bivariate Gaussians can be reduced to the convolution of the respective one dimensional Gaussian, the same does not hold for other circular distributions. For example, convoluting two one dimensional profiles of pillbox distributions results in a triangular shaped distribution. But, in the same test case as above with two pillbox distributions a different flux profile is obtained, see Figure 13. To verify the expected and actual result, the horizontal profile of a random generator for the 2D sun shape and one for the 1D pillbox profile are also shown. The same holds for the convolution of a pillbox with a bivariate Gaussian. Therefore, the convolution computation actu-

ally needs to consider the two dimensional distributions. Since no efficient analytical method has been found, a numerical calculation of the convolution is carried out.

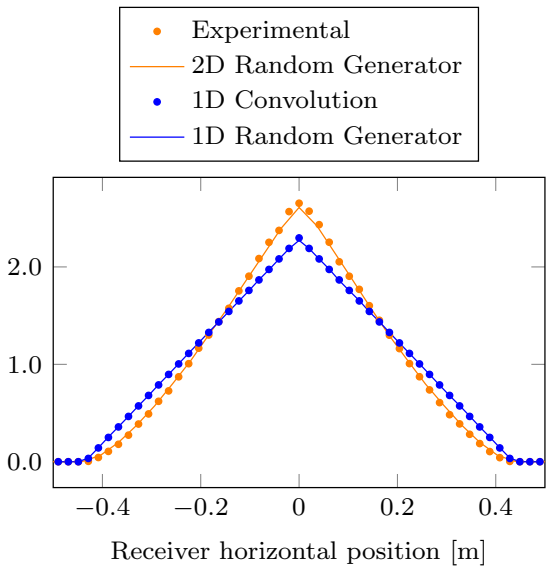


Figure 13: Convolution of two pillbox distribution with a rectangular 1 m^2 receiver discretized into 49x49 pieces and one ray with 100.000 samples originating 50m apart from the receiver. The convolution of the 1D pillbox profile is expected to have a triangular shape as confirmed by a random generator, shown in blue. But from the test case as well as a random generator for the actual 2D sun shape, a different profile is obtained shown in orange.

First, the two dimensional vector $(0, 0)$ gets individually perturbed by the specified distributions similar to the three dimensional case in the Monte Carlo methods, see Section 2.1.1. For each sample vector its length is calculated and the reverse operation of Equation (12) has to be applied to get the correct profile resulting in a standard deviation of

$$\sigma = \sqrt{\frac{\sum_{i=0}^{n} (s_i - \bar{s})^2}{n-1}}, s_i = \frac{1}{\sin l_i \cos l_i} \tag{20}$$

with l_i as the length in mrad of the i-th sample and $\bar{s} = \sum_{i=0}^{n} s_i/n$ as the mean.

To reduce memory usage, the samples are grouped in buckets instead of saving each of them for the calculation of the mean.

2.1.3 GPU Acceleration

Since each ray is evaluated independently, the ray tracers are highly concurrent. Therefore, they are very well suited to run on hardware specialized for parallelism like the

GPU. A summary on the how the CPU version of *SunFlower* was parallelized on the GPU is given in the following. The presented work done by Aldenhoff [2] has been refined and extended to include newly added features to the ray tracer.

A CPU consists of a small number of independent cores with large individual caches. In contrast, a GPU has a lot more cores sharing the control logic and caches. Memory consumption and transfer are typically the limiting factors for GPU programs. Therefore, a lightweight version of *SunFlower* was implemented in CUDA. Only data essential for the calculations is transferred from the CPU to the GPU. Because the GPU executes threads in so called warps sharing one program counter, any execution divergence results in synchronization and thus loss of parallelism. Branches were very common in the ray-heliostat and ray-receiver intersection calculation. Specifically, when traversing the bounding volume hierarchy. For better performance, an iterative synchronous traversal was implemented. Another frequently executed part of the code causing execution divergence is the bivariate polygon integration. The algorithm developed in [19] includes a variety of branches for the angular cases. Since the actual computation is mostly the same, a branchless version was implemented.

Now that all components of the most complex submodel of the CRS are fully described, the remaining models will be shortly discussed in the following sections.

2.2 Thermal model

The incident solar radiation $\dot{Q}_{\text{inc}}$ calculated by the optical model is transferred into thermal power at the receiver. It consists of panels containing tubes through which a heat transfer fluid (HTF) is flowing. As a HTF molten salt is often used and thus considered in our model. It is based on the works of Heiming [28] and Franke [25]. However, some of the incoming solar radiation is lost due to different types of effects. First, only a part of the radiation is absorbed by the receiver since some gets reflected $\dot{Q}_{\text{ref}}$. Additionally, the HTF losses thermal power to the surrounding air by convection $\dot{Q}_{\text{conv}}$. It is caused by the fluid flow as well as by gravity and thermal buoyancy. Since the convection loss depends on the receiver being used, each receiver type has its own model for the convection loss, see [15]. Lastly, the receiver panels radiate heat into the environment $\dot{Q}_{\text{rad}}$. Because the environment also radiates heat onto the receiver the actual loss depends on the temperature between the environment and the receiver. Therefore, our model differentiates between radiation loss to the ground and radiation loss to the sky [15]. In total the remaining thermal power of the HTF $\dot{Q}_{\text{htf}}$ is given by

$$\dot{Q}_{\text{htf}} = \dot{Q}_{\text{inc}} - (\dot{Q}_{\text{ref}} + \dot{Q}_{\text{rad}} + \dot{Q}_{\text{conv}}). \tag{21}$$

The second part of the thermal model is to predict the required mass flow rate of the HTF to obtain the desired temperature at the receiver outlet. The resulting equation for the mass flow rate has an implicit nature and is therefore solved iteratively. More details on the model can be found in [28].

2.3 Storage model

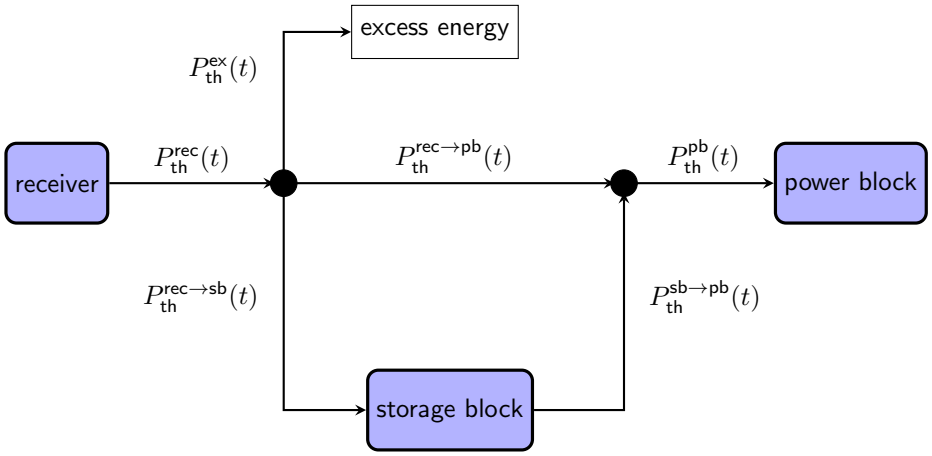


Figure 14: Possible power flows of the storage system, derived from [15]. The thermal power of the receiver can be stored, directly pumped into the power block, or discarded.

Instead of directly feeding the HTF to the power block, it can also be transported into a storage system. This approach has two benefits. First, electricity generation can be adjusted to the demand, and second excess power surpassing the capacity of the power block $P_{\text{th}}^{\text{pb\_max}}$ can be stored $P_{\text{th}}^{\text{sb}}$ instead of being wasted. The heated molten salt is stored in a hot storage tank and after its power is converted, it is transferred to a cold tank. As shown in Figure 14, the thermal power can flow from the receiver $P_{\text{th}}^{\text{rec}}$ to the storage block $P_{\text{th}}^{\text{rec}\to\text{sb}}$, to the power block $P_{\text{th}}^{\text{rec}\to\text{pb}}$ or be discarded as excess energy $P_{\text{th}}^{\text{ex}}$. Stored power in the storage block can then flow to the power block $P_{\text{th}}^{\text{sb}\to\text{pb}}$. A buffer strategy derived from Coumbassa [15] is considered which maximizes the power input. Depending on the amount of thermal power at the receiver, two operation modes are used. First, the generate mode, where the thermal power at the receiver is less than the maximum input power of the power block ($P_{\text{th}}^{\text{rec}} \leq P_{\text{th}}^{\text{pb\_max}}$). Then all thermal power can be transferred to the power block and thus nothing needs to be stored. Additionally, power in the storage tank can also be processed leading to

$$\begin{aligned} P_{\text{th}}^{\text{rec}\to\text{sb}} &= P_{\text{th}}^{\text{ex}} = 0, \\ P_{\text{th}}^{\text{rec}\to\text{pb}} &= P_{\text{th}}^{\text{rec}}, \\ P_{\text{th}}^{\text{sb}\to\text{pb}} &= P_{\text{th}}^{\text{pb\_max}} - P_{\text{th}}^{\text{rec}\to\text{pb}}. \end{aligned} \tag{22}$$

However, there is also a minimum input power required by the power block to produce electricity. If this requirement is not met, the thermal power is fed into the storage tank.

$$
\begin{aligned}
P_{\text{th}}^{\text{rec}\to\text{pb}} &= P_{\text{th}}^{\text{sb}\to\text{pb}} = 0, \\
P_{\text{th}}^{\text{rec}\to\text{sb}} &= \min\left(P_{\text{th}}^{\text{rec}}, P_{\text{th}}^{\text{sb\_max}}\right), \\
P_{\text{th}}^{\text{ex}} &= P_{\text{th}}^{\text{rec}} - P_{\text{th}}^{\text{rec}\to\text{sb}},
\end{aligned}
\tag{23}
$$

with $P_{\text{th}}^{\text{sb\_max}}$ as the capacity of the storage block.

On the other hand, the surplus mode considers the case that more thermal power is generated by the receiver than the power block is capable of handling ($P_{\text{th}}^{\text{rec}} > P_{\text{th}}^{\text{pb\_max}}$). The hot tank then stores as much power as possible and the residual power is discarded.

$$
\begin{aligned}
P_{\text{th}}^{\text{rec}\to\text{pb}} &= P_{\text{th}}^{\text{pb\_max}}, \\
P_{\text{th}}^{\text{rec}\to\text{sb}} &= \min\left(P_{\text{th}}^{\text{rec}} - P_{\text{th}}^{\text{rec}\to\text{pb}}, P_{\text{th}}^{\text{sb\_max}}\right), \\
P_{\text{th}}^{\text{sb}\to\text{pb}} &= 0, \\
P_{\text{th}}^{\text{ex}} &= P_{\text{th}}^{\text{rec}} - P_{\text{th}}^{\text{rec}\to\text{pb}} - P_{\text{th}}^{\text{rec}\to\text{sb}}
\end{aligned}
\tag{24}
$$

By applying the buffer strategy for each time step Δt, the thermal energy $Q_{\text{th}}^{\text{sb}}(t)$ stored in the hot tank at time t can be computed.

$$
Q_{\text{th}}^{\text{sb}}(t+\Delta t) = Q_{\text{th}}^{\text{sb}}(t) + \left(\eta_{\text{in}} P_{\text{th}}^{\text{rec}\to\text{sb}}(t) - \eta_{\text{out}}^{-1} P_{\text{th}}^{\text{sb}\to\text{pb}}(t) - \eta_{\text{loss}} Q_{\text{th}}^{\text{sb}}(t)\right) \cdot \Delta t, \tag{25}
$$

with η_{in}, η_{out} as the charging and discharging efficiency of the storage block, and η_{loss} as the heat loss factor.

Similarly, the thermal power at the power block is given by an iterative application of the buffer strategy while considering different losses, see [15] for more details. Therefore, the thermal power at the power block is given by

$$
P_{\text{th}}^{\text{pb}}(t) = P_{\text{th}}^{\text{rec}\to\text{pb}}(t) + P_{\text{th}}^{\text{sb}\to\text{pb}}(t). \tag{26}
$$

2.4 Electrical model

The thermal power calculated in the storage model is then converted into electrical power as described by the following electrical model [25]. It models the power block consisting of a heat exchanger, a steam turbine, a generator, a condensator, and a cooling tower. Since molten salt has a temperature far above 100°C, it evaporates water in the heat exchanger which then drives the turbine to generate electricity. A condensator in combination with a cooling tower cools the steam down to a liquid. To simplify the calculation of the power block efficiency η_{pb}, measurements from an actual 100MW power block provided by TSK Flagsol are used. These depend on the

turbine load as well as the ambient temperature. After testing the efficiency at different loads and temperatures, the results were bilinear interpolated to estimate efficiencies for intermediate values. Figure 15 shows the resulting efficiencies. Therefore, the efficiency η_{pb} is directly obtained from the bilinear interpolation, and the electrical power is given by

$$P_{\mathrm{el}}^{\mathrm{pb}}(t) = \eta_{\mathrm{pb}}(T_{\mathrm{amb}}, l) \cdot P_{\mathrm{th}}^{\mathrm{pb}}(t). \tag{27}$$

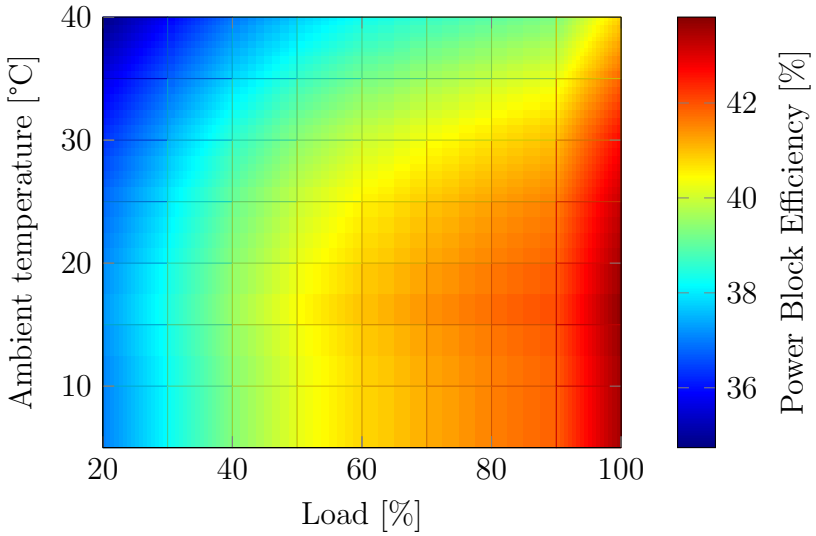


Figure 15: Interpolated efficiencies of a 100 MW power block for various ambient temperatures and loads taken from [25].

2.5 Economical model

In the last part of CRS model, its economical aspects are evaluated. Since the generated electricity of the plant is known from the electrical model, the income can be calculated given the hourly tariff of electricity $\pi_{\mathrm{ToE}}(t)$ to

$$C_{\mathrm{income}} = \sum_{d=1}^{365} \int_{0}^{24} P_{\mathrm{el}}^{\mathrm{pb}}(d,t)\pi_{\mathrm{ToE}}(t)dt \approx E_{\mathrm{el}} \cdot \pi_{\mathrm{ToE}}, \tag{28}$$

with E_{el} as the annual electrical energy production.

The modulation of the costs is the most complex part of the economical model. Most of the presented model is based on the works of [28] with a few simplifications [24]. Two types of costs are considered, the investment costs C_{invest} as well as the operation and maintenance costs C_{OM}.

Investment To estimate the investment costs, the cost of each component of the plant is summed up to

$$C_{\text{invest}} = C_{\text{land}} + C_{\text{hel}} + C_{\text{tower}} + C_{\text{rec}} + C_{\text{sb}} + C_{\text{pb}}. \tag{29}$$

Originally, two effects influencing each component costs were taken into account [28]. First, the scaling effect describing the scaling of all costs that are based on a reference project of different scale. Furthermore, the volume effect described the decreasing costs when larger quantities are bought. However, the required parameters are hard to estimate correctly and the volume effect is often the same since larger quantities are brought anyway. Therefore, these effects are not considered in the following simplified model

- The land costs C_{land} consist of terrain costs as well as costs for terrain improvement.
- According to [28] heliostat costs C_{heli} were put into two main categories. Direct costs for the foundation, manufacturing, wiring, communication, optical improvement of the mirrors as well as the installation of each heliostat. On the other hand, indirect costs for the engineering, facilities and tooling, and equipment lease. Considering our simplification, most of these costs can be combined to

  $$C_{\text{hel}} = c_{\text{hel}} \cdot N_{\text{hel}} + C_{\text{hel,ind}}, \tag{30}$$

  with c_{hel} as the combined direct cost factor for each of the N_{hel} and $C_{\text{hel,ind}}$ as the total indirect heliostat cost.
- C_{tower} are the total costs of the tower.
- Similarly, C_{rec} consists of all costs for the receiver.
- Since the costs of the storage system C_{sb} depends on the maximum storage capacity $Q_{\text{th}}^{\text{sb\_max}}$, they are given by

  $$C_{\text{sb}} = c_{\text{sb}} \cdot Q_{\text{th}}^{\text{sb\_max}}, \tag{31}$$

  with c_{st} as the costs per thermal power.
- Lastly the cost of the power block C_{pb} depends on its maximum output capacity $P_{\text{el}}^{\text{pb\_max\_out}}$ as follows

  $$C_{\text{pb}} = c_{\text{pb}} \cdot P_{\text{el}}^{\text{pb\_max\_out}}, \tag{32}$$

  with c_{pb} as the costs per electric power.

The operation and maintenance costs C_{OM} consists of staff, water, spare parts, and insurance costs which largely depend on the size of the plant. Since the investment costs also depend on the plants size, operation and maintenance costs are modeled as a fraction of C_{invest}

$$C_{\mathrm{OM}} = f_{\mathrm{OM}} \cdot C_{\mathrm{invest}}. \tag{33}$$

From the costs of the plant and its generated income, different economical performance measures can be calculated.

Levelized cost of electricity (LCOE) The levelized cost of electricity describes the running production costs of electricity. Thus, it does not directly consider the one-time investment costs. Due to its simplicity, it is commonly used to compare different power plants [28] and calculated as follows:

$$\mathrm{LCOE} = \frac{\text{Annual costs}}{\text{Annual energy production}} = \frac{C_{\mathrm{invest}} \cdot f_{\mathrm{annuity}} + C_{\mathrm{OM}}}{E_{\mathrm{a}}}, \tag{34}$$

with the annuity factor containing the nominal rate of interest r of the loan and the number of years N_y for the loan repayment which is assumed to be the whole project life time

$$f_{\mathrm{annuity}} = \frac{1}{\sum_{y=1}^{N_y} \frac{1}{(1+r)^y}} = \frac{(1+r)^{N_y} \cdot r}{(1+r)^{N_y} - 1}. \tag{35}$$

In simple terms, the annuity factor is used to determine the annual amount of money that has to be paid to compensate for the investment loan while taking time into account. It is calculated from the present value, i.e. how much the annual payments would be worth at the beginning of the loan. Therefore, it includes the fact that having some amount of money now is worth more than having the same amount of money in a few years. More specifically, the bank could use the money for other loans. Therefore, the present value PV_y of each of the annual loan repayments L_y decreases per year y due to the compounded interest ($\mathrm{PV}_y \cdot (1+r)^y = L_y$). To have a profitable deal for the bank, the summed present value should match the investment costs. Thus, the required annual payments can be calculated as a fraction of the investment cost as shown in Equations (34) and (35).

Net present value (NPV) To estimate the overall profit of the power plant, the net present value is used. As the name suggests, it measures how much the whole project including its annual revenue and investment costs, would be worth at the beginning of the project. The present value of the annual income and costs can be calculated as stated above by using the annuity factor. Since the present value is now of interest

and not the required annual loan repayment, the inverse of the annuity factor has to be used. When subtracting the investment costs we get

$$\begin{aligned} \text{NPV} &= \sum_{y=1}^{N_y} (\text{PV}_{\text{income,y}} - \text{PV}_{\text{costs,y}}) - \text{Investment costs} \\ &= \sum_{y=1}^{N_y} \frac{C_{\text{income}} - C_{\text{OM}}}{(1+r)^y} - C_{\text{invest}} = \frac{C_{\text{income}} - C_{\text{OM}}}{f_{\text{annuity}}} - C_{\text{invest}}, \end{aligned} \tag{36}$$

Internal rate of return (IRR) The internal rate of return calculates the required nominal interest rate r of Equation (36) to have an NPV of zero. Therefore, giving a measure at which interest rate the project would be profitable.

Paybakc Period (PP) Similarly, the number of years N_{PP} until the project generates profit can be calculated by setting Equation (36) to zero and solving for N_y resulting in

$$N_{\text{PP}} = \frac{\log\left(\frac{C_{\text{income}} - C_{\text{OM}}}{C_{\text{income}} - C_{\text{OM}} - r \cdot C_{\text{invest}}}\right)}{\log\left(1+r\right)}. \tag{37}$$

2.6 Annual integration

So far the optical model considered the static case of having a particular date and time for which the solar power should be calculated. But as discussed in the last sections the annual received energy is often of interest which is given by:

$$E_{\text{rec}} = \sum_{d=1}^{365} \int_0^{24} P(d,t)\, dt = \sum_{d=1}^{365} \int_0^{24} A_{\text{helios}} I_{\text{DNI}}(d,t) \eta(d,t) dt \tag{38}$$

with $P(d,t)$, A_{helios}, $I_{\text{DNI}}(d,t)$ and $\eta(d,t)$ as in Equation (11) and E_{rec} as the annual solar energy hitting the receiver. Even when approximating the integral over the day by a sum over the hours, the optical model has to be evaluated 8760 times for one annual simulation. Considering that an optimization of the heliostat field requires a lot of annual simulations, this approach is infeasible. Therefore, instead of computing the optical power for each hour of the year, specific dates and times are simulated and the remaining ones are obtained via interpolation. First, the temporal domain in Equation (38) is transformed into either an ecliptic or equatorial coordinate systems [54]. These are then transformed into a unit square which can be sampled by a regular or a non-regular grid. When using a regular grid, the computed sample efficiencies can be interpolated by a variety of different methods.

1. Nearest-neighbor interpolation which uses the closest neighboring sample.

2. Bilinear interpolation of the four surrounding points.

3. A spherical linear interpolation (SLERP) based method, that applies SLERP in both directions but uses efficiencies instead of quaternions.

4. Bicubic interpolation using cubic splines to achieve smoother interpolation surface [54].

For non-regular grids, Legendre polynomials are utilized. More details on the annual integration method can be found in [54].

3 Heliostat field layout optimization

Optical losses are responsible for about 40% of the total losses within a CRS [40]. The heliostat field layout has a crucial impact on optical losses. Therefore, optimizing the layout can significantly improve overall efficiency. As discussed at the beginning of this thesis, there are a variety of different approaches to optimization. In the following, three different optimization strategies will be discussed. Section 3.1 describes the pattern based optimization which places heliostat in a parameterized pattern and thus reduces the search space to a few parameters. The goal of pattern optimization is to find the best parameters and select the highest scoring subset of heliostat positions. One downside of pattern optimization is that an optimal field layout is most likely not within the pattern.

Therefore, pattern-free optimizations like field growth methods consider a far larger set of possible positions. They sequentially add the best performing heliostat to the field to obtain an optimal layout. Section 3.2 introduces our new graph based field growth method in detail.

Finally, in Section 3.3 local search optimization is discussed which further improves a given heliostat field by repositioning each heliostat individually.

3.1 Pattern based approaches

The pattern optimization discussed in the following is extending on the work of Richter et. al. [50, 24]. All original patterns and their improved versions are described by a few parameters. These are then optimized with our combinatorial or downhill simplex search methods, as described in Section 3.1.6.

3.1.1 North-South staggered

The north-south staggered pattern was developed as an improvement to the cornfield pattern by reducing empty space between the heliostats [37]. Instead of having equal rows as in typical cornfields, each row in the North-South staggered pattern is staggered to its neighbors offering a more dense packing. As indicated by its name, the pattern is aligned with the north-south direction due to its horizontal and vertical symmetry

as well as the sun's movement. Moreover, the distances d_i^{row} between rows and d_i^{col} columns of the pattern linearly increase with the distance to the center

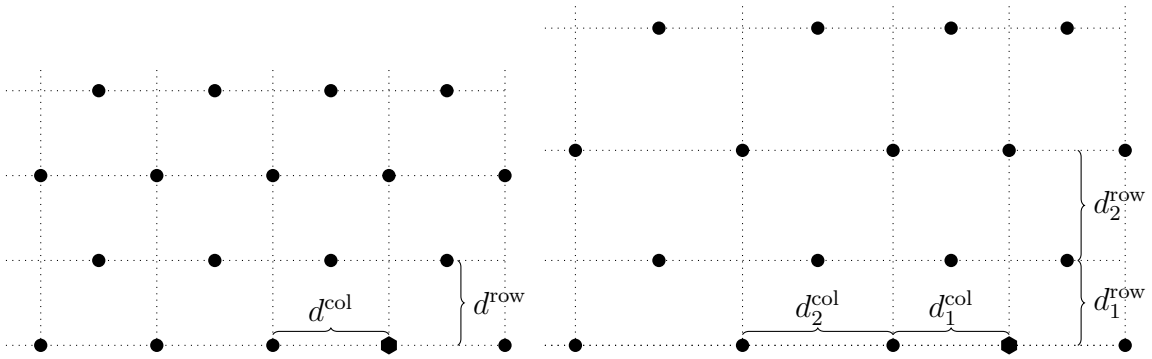
$$d_i^{\text{row}} = a^{\text{row}} \cdot i + b^{\text{row}} D, \quad d_i^{\text{col}} = a^{\text{col}} \cdot i + b^{\text{col}} D, \tag{39}$$

where D is the heliostat diameter and a^{row}, b^{row}, a^{col}, and b^{col} are the linear and constant factors for the row and column distance, respectively. The linear factors are extensions to the original North-South staggered pattern.

Figure 16 illustrates the north-south staggered pattern and the parameter for optimization can be found in Table 2.

Parameter	Description
a^{row}	constant factor of the row distance
b^{row}	linear factor of the row distance
$\mathbf{a^{col}}$	constant factor of the column distance
$\mathbf{b^{col}}$	linear factor of the column distance

Table 2: Parameters of the North-South staggered pattern, with parameters that are not existing in the original pattern shown in bold.



(a) Original (b) Improved

Figure 16: Illustration of the original and extended North-South staggered pattern, with d_i^{row} and d_i^{col} as the i-th row and column distance, respectively.

3.1.2 Radial staggered

The radial staggered pattern also places heliostats in a staggered fashion for closer packing [14]. However, they are now positioned on concentric rings, see Figure 17a. The pattern can be found in a lot of commercial CRS like the Gemasolar [11]. The requirements of the original pattern are simple. The radial distance between consecutive rows should be constant while heliostats are placed as dense as possible. Therefore, a staggered arrangement is needed, which mostly requires the same amount of heliostats

between each row. The only exception is when twice as many heliostats can be placed as on the row before. Rows having the same amount of heliostats are grouped into one zone. Finally, the first row is set to contain 35 heliostats.

From the requirements and some basic geometry, the following specifications were derived [14]. The minimal distance h between neighboring heliostats is given by $h = D+a$, with D as the heliostat diameter and a as a safety distance parameter. Therefore, the minimum radial increment Δr and the radius $r_{1,1}$ of the first row of zone one can be computed to

$$\Delta r = h\frac{\sqrt{3}}{2} = (D+a)\frac{\sqrt{3}}{2}, \quad r_{1,1} = \frac{h \cdot N_1}{2\pi}, \tag{40}$$

with $N_1 = 35$ as the number of heliostats in the first zone.

Since the heliostats are evenly spaced on each ring, the angular distance Δa_j between two consecutive heliostats on the j-th Zone is $\Delta a_j = 2\pi/N_j$. Each heliostat position is given in polar coordinates. The angle of the first heliostat of a row is displaced by $\Delta a_j/2$ to the previous rows first heliostat, see Figure 17a. Once the distance between two heliostats on a row is larger than $2h$, another zone starts with $N_j = 2N_{j-1}$ heliostats.

SunFlower also includes an improved version of the original radial staggered pattern, see Figure 17b. Here, the radial increment is multiplied by a growth factor g. Additionally, blocking effects are mostly avoided by requiring a minimum radial increment as done in [14]. This minimal increment is based on a simplified blocking model. From the intercept theorem, the radial increment $\Delta r_{i,j}$ of the i-th row of zone j can be calculated to

$$\Delta r_{i,j} = g \max\left(h\frac{\sqrt{3}}{2}, \frac{r_{i,j}h_{\text{hel}}}{h_{\text{tower}} - h_{\text{hel}}}\right), \tag{41}$$

with $r_{i,j}$ as the radius of the current row, $h$hel as the height of the heliostat, and h_{tower} as the height of the tower.

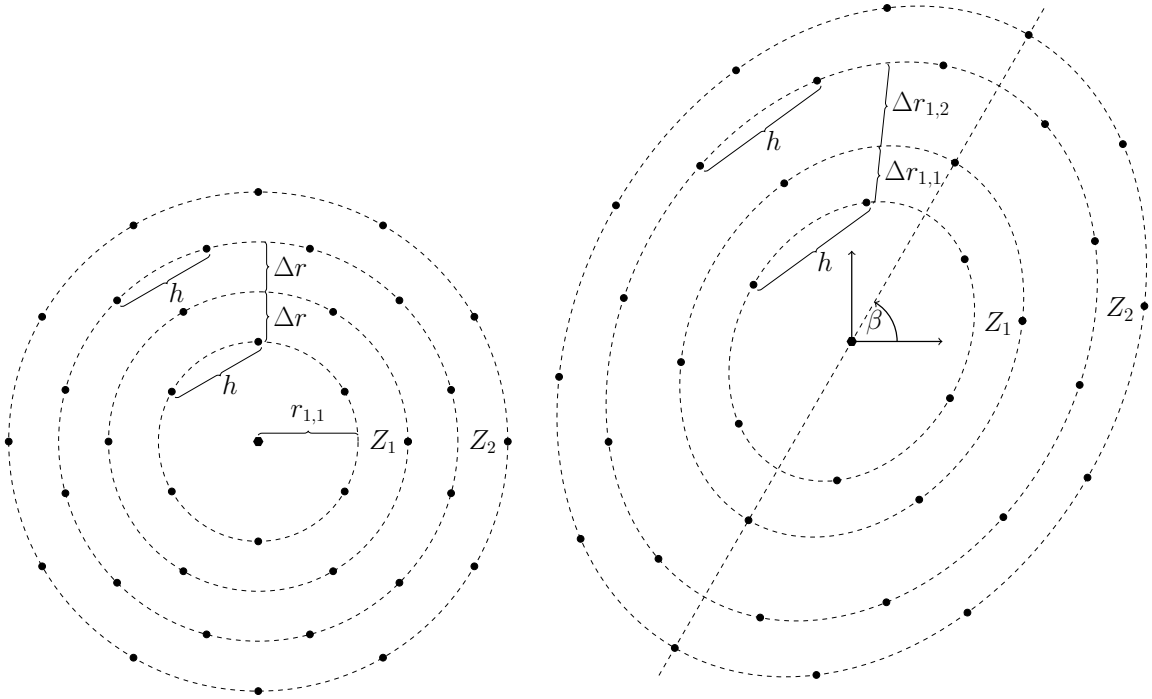
A more general formula for the number of heliostats in the j-th zone which is based on the radius $r_{i,j}$ of the zone is

$$N_i = \left\lfloor \frac{\pi}{\arcsin\left(\frac{h}{2r_{1,j}}\right)} \right\rfloor. \tag{42}$$

The equation above allows using different numbers of heliostats in the first and the following zones. Lastly, the whole pattern can be scaled in direction of an axis passing through the origin. The axis is defined by the angle β to the x-axis and s determines the scaling factor, see Figure 17b. For a given point p, the scaled version p' can be calculated by coordinate transformations. First, the scaling axis is aligned with the x-axis by rotating with an angle β in clockwise direction. Then the x-coordinates are scaled using s and the rotation is reversed leading to

$$\begin{pmatrix} p'_x \\ p'_y \end{pmatrix} = \begin{pmatrix} \cos(\beta) & -\sin(\beta) \\ \sin(\beta) & \cos(\beta) \end{pmatrix} \begin{pmatrix} s & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} \cos(-\beta) & -\sin(-\beta) \\ \sin(-\beta) & \cos(-\beta) \end{pmatrix} \begin{pmatrix} p_x \\ p_y \end{pmatrix}. \tag{43}$$

It should be noted that β only ranges from 0° to 180° since the scaling axis is symmetric at the origin. An illustration of the new scaled pattern can be found in Figure 17b and its parameters in Table 3.



(a) Original (b) Improved

Figure 17: Illustration of the original and the improved radial staggered pattern. Two zones Z_1 and Z_2 are shown where each heliostat has a distance of h to the next heliostat on the same row and the radial increment is given by Δr. The improved version uses varying radial increment and can be scaled along an axis defined by β.

3.1.3 Rose

One disadvantage of the radial staggered pattern for some heliostat fields is its radial symmetry. Depending on the geographical location of the field, the optical efficiencies can vary in north-south direction as well as east-west direction. To better account for these differences, the rose pattern segments the field into six sectors as illustrated in Figure 18. Sector six is the mirrored version of sector two, likewise for sectors five and three. Within each sector, heliostats are placed according to the radial staggered pattern. Safety distances between two sectors were added to avoid collisions. In the

Parameter	Description
a	constant safety distance
g	linear radius growth factor
$\boldsymbol{\beta}$	angle of the scaling axis to the x-axis
$\mathbf{s}$	scaling factor

Table 3: Parameters of the improved radial staggered pattern, with parameters that are not existing in the original pattern shown in bold.

original paper, each radial increment of each sector was optimized individually leading to a total of 172 parameters [17]. To simplify the optimization, our version of the rose pattern uses the parameters of the radial staggered pattern within each sector except the scaling parameters. Table 4 shows the parameters of the pattern.

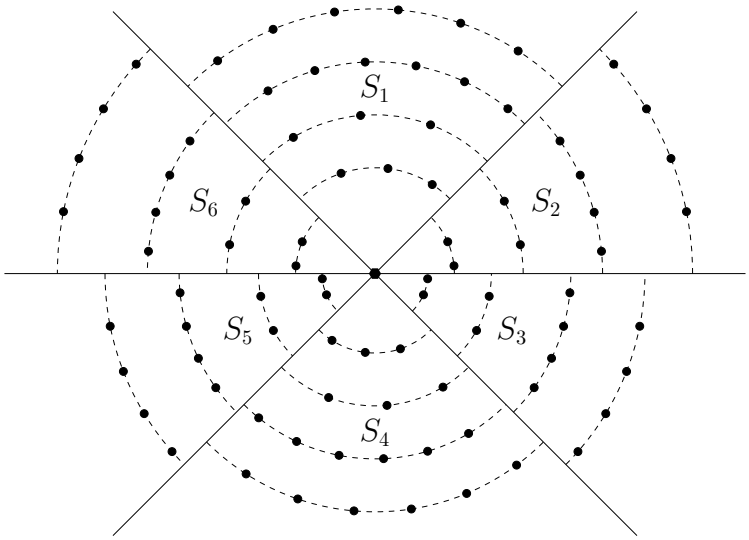


Figure 18: Illustration of the rose pattern, with S_i as the i-th sector. Within each sector the radial staggered pattern is used. Sector five and six are mirrored version of sectors three and two, respectively.

3.1.4 Hexagon

The original patent of the hexagon pattern [46] placed heliostats along concentric hexagons. The smallest distance d_i between the hexagons increases linearly, while each edge of a hexagon contains one more heliostat than the edge of the previous hexagon. This results in a staggered arrangement. The heliostats are evenly spaced on each edge with a distance of h_i between two heliostats and $h_i/2$ between a heliostat and a corner. On the edge of the first hexagon, only one heliostat is placed.

Parameter	Description
a_i	constant safety distance of sector $i \in [0, .., 4]$
$\mathbf{g_i}$	linear radius growth factor of sector $i \in [0, .., 4]$

Table 4: Parameters of the rose pattern, with parameters that are not existing in the original pattern shown in bold. Sectors two and six as well as three and five share the same parameter.

From the description above, a formula can be derived to obtain the heliostats position. An illustration of the hexagon pattern can be found in Figure 19a. The inner radius r_i of the hexagons determines the edge length k_i and the distance between heliostats h_i. From the increasing distance d_i between consecutive hexagons these values can be calculated to

$$d_i = a \cdot i + b, \quad r_i = \sum_{j=0}^{j=i} d_j, \quad k_i = \frac{2}{\sqrt{3}} r_i, \quad h_i = \frac{k_i}{i+1}. \tag{44}$$

To obtain the heliostats coordinates, first the corner positions $\vec{c}_{i,j}$ with $j \in [0, .., 5]$ are computed

$$\vec{c}_{i,j} = \begin{pmatrix} k_i \cos\left(j\frac{\pi}{3}\right) \\ k_i \sin\left(j\frac{\pi}{3}\right) \end{pmatrix}. \tag{45}$$

Starting at corner $\vec{c}_{i,j}$, the first heliostat is placed at a distance of $h_i/2$ in the direction of the next corner and the next i heliostats at a distance of h_i to one another.

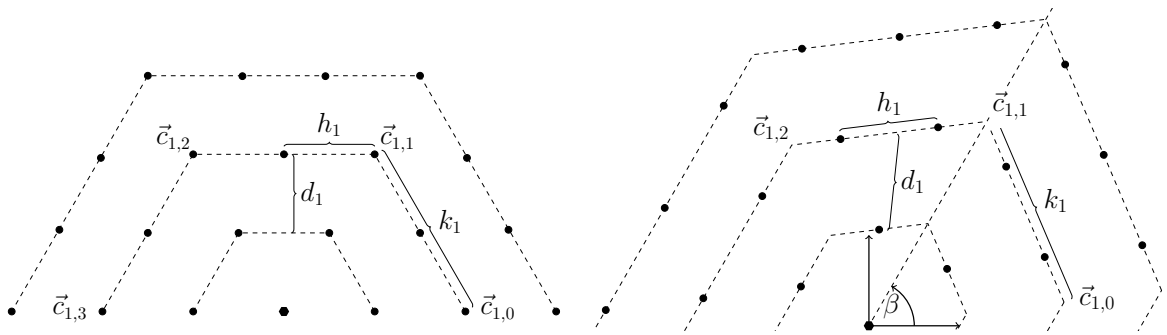
Further adjustments to the original layout have been made [24]. Heliostats are now also placed at the corner to ensure an equal distance between them throughout each hexagon. Similar to the modified radial staggered pattern, the distance d_i between two hexagons is increased to be at least the blocking distance leading to

$$d_i = \max\left(a \cdot i + b, \frac{r_i h_{\text{hel}}}{h_{\text{tower}} - h_{\text{hel}}}\right). \tag{46}$$

Finally, the pattern can be scaled at an arbitrary axis as defined in Equation 43. Figure 19b illustrates the modified hexagon pattern and Table 5 lists the parameters for optimization.

3.1.5 Spiral

As demonstrated in different areas of science, it can be beneficial to apply strategies that are inspired by nature. Sunflower seeds for example, are optimized for two objectives that are also relevant for CRS. One is to minimize shading of other seeds and the other is to maximize the number of seeds per area [5]. A sunflower achieves both by having a dense packing close to the center of the flower which decreases outwards. Furthermore, the seeds are arranged in a staggered way. Both properties arise when



(a) Original (b) Improved

Figure 19: Illustration of the Hexagon pattern and the improved version. The corners on the i-th hexagon are given by $\vec{c}_{i,j}$, the distance between hexagons by d_i, the hexagon side length by k_i and the scaling axis is defined by β.

Parameter	Description
a	constant factor of the hexagon distance
b	linear factor of the hexagon distance
$\boldsymbol{\beta}$	angle of the scaling axis to the x-axis
$\mathbf{s}$	scaling factor

Table 5: Parameters of the improved hexagon pattern, with parameters that are not existing in the original pattern shown in bold.

the seeds are placed in a spiral which is formed by gradually increasing the distance to the center and always placing a seed after a certain angle. The angle of interest is the golden ratio for angles, i.e. the ratio between the angle to the rest of a circle is the same as the rest to a full circle [5]. Noone et al. [43] first proposed to apply this pattern to heliostat fields.

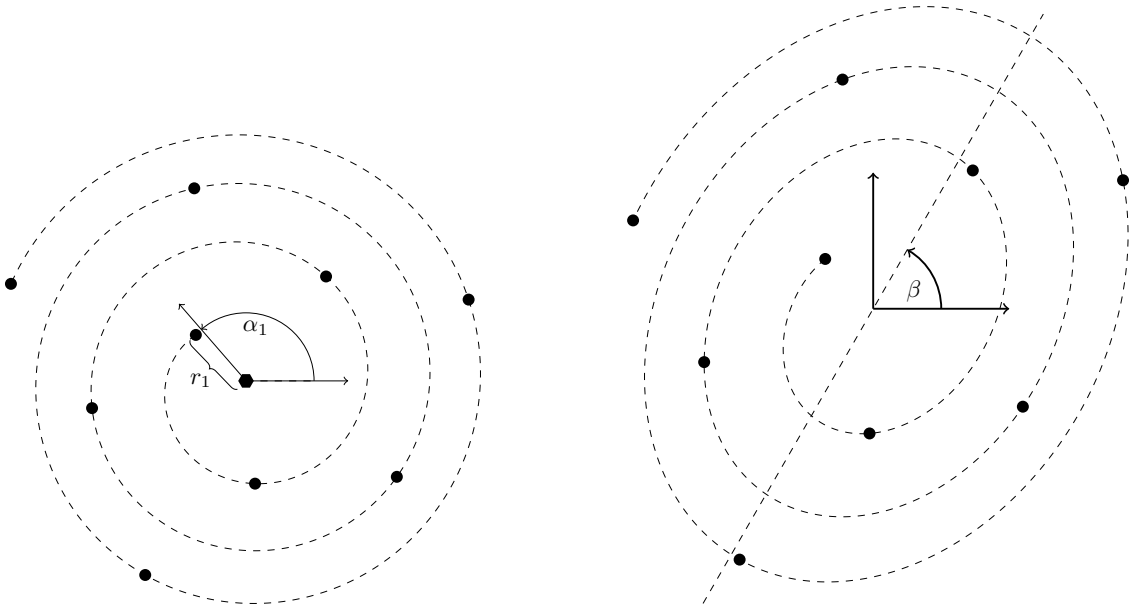
Based on the formulation of the pattern, it is easiest to describe the positions of the k-th heliostat in polar coordinates $(\alpha_k, r_k)^T$. To control the distribution of the pattern, the radius incorporates two parameters a and b leading to

$$r_k = a \cdot k^b, \quad \alpha_k = 2\pi k \left(\frac{1+\sqrt{5}}{2} \right)^{-2}. \tag{47}$$

Similar to the modified radial staggered and hexagon pattern, the spiral pattern can also be scaled as described in Equation (43). Figure 20 illustrates both the original pattern and the scaled one. The parameters of this pattern are given in Table 6.

3.1.6 Optimizer

Now that the different patterns have been specified, their parameters need to be optimized according to an objective function. A variety of different optimizers have been



(a) Original (b) Improved

Figure 20: Structure of the spiral pattern and the improved version. Here, α_1 is the golden ratio for angles, r_k the radius of the k-th heliostats polar coordinate, and β defines the scaling axis.

Parameter	Description
a	linear factor of the radius
b	exponential factor of the radius
$\boldsymbol{\beta}$	angle of the scaling axis to the x-axis
$\mathbf{s}$	scaling factor

Table 6: Parameters of the improved spiral pattern, with parameters that are not existing in the original pattern shown in bold.

used in the literature like simple combinatorial searches [43], genetic algorithms [6], neural networks [47]. The overall goal is not only to find the parameters maximizing the objective function but also to select the best subset of positions from the pattern. The latter is required as the patterns typically generate much more positions than needed. Therefore, there are two optimization stages. For a pattern with n parameters and a parameter combination $\vec{p} = (p_1, ..., p_n)^T$, find the subset that maximizes the objective function while containing the desired amount of heliostats. This stage is called parameter evaluation. In the second stage called pattern evaluation, the overall best parameter combination $\vec{p}_{\text{opt}}$ is determined. Thus, each evaluation in the second stage requires a new optimization of the first. Both optimization steps are described in the following.

Parameter evaluation There are different approaches to this task. A rather complex method has been proposed in [59]. Here, any heliostat inside a polygon is considered to be part of the solution. The polygon vertices are equiangular arranged to limit the search space. Then an evolutionary algorithm optimizes the shape of the polygon.

The downside of this approach is its run-time of several minutes as reported by the authors. During a full pattern optimization, the parameter evaluation is executed a lot of times. Therefore, a faster method is needed. Noone et al. [43] applied the intuitive approach of simulating all valid heliostats and selecting the most efficient ones. Afterward, another simulation is carried out with the selected ones to obtain the value of the objective function.

Combinatorial search A straight-forward solution to the pattern evaluation is to generate any combination of parameters and evaluate each of them. This combinatorial search has been used in [43]. Since the possible values for a single parameter are already infinite, only a certain range of values is considered that is then evenly discretized.

The set of possible combinations grows exponentially with the number of parameters. Therefore, this approach is quite inefficient. Another issue is the discretization of the values that might not contain the optimal values. Therefore, a smarter more flexible method is needed.

Downhill simplex search A more sophisticated and widely used optimization algorithm is the downhill simplex search algorithm by Nelder and Mead [41]. It is capable of optimizing multidimensional non-linear optimization problems without the need for gradients. Therefore, the simplex search is applicable for pattern optimization as also demonstrated in [47]. A summary of its basic principles is given in the following, more details can be found in [24].

The algorithm operates on a simplex, which is the simplest possible polytope in the search space. For example, the simplex of a pattern with 2 parameters would be a triangle and for 3 parameters a tetrahedron. Therefore, a simplex for a pattern of n parameters consists of $n+1$ points $\vec{p}_1, ..., \vec{p}_{n+1} \in \mathbb{R}^n$ each representing a possible combination of parameters. The idea of the downhill simplex search is to enclose an optimum within the simplex by expansion and reflection and then contract and shrink the simplex to obtain the optimum. The initial simplex start with one random point $\vec{p}_1$, while the others only differ in one dimension each. An exemplary simplex which is also used to illustrate each step can be found in Figure 21a. It then performs the following actions on the simplex:

1. Order the points according to the objective function f, $f(\vec{p}_1) \geq f(\vec{p}_2) \geq ... \geq f(\vec{p}_{n+1})$. Calculate the centroid p_m of all points except the worst, $\vec{p}_m = \sum_{i=1}^{n} \vec{p}_i$ and go to the next step.

2. Reflect the worst point on the centroid using the reflection coefficient $\rho > 0$, $\vec{p}_r = \vec{p}_m + \rho(\vec{p}_m - \vec{p}_{n+1})$, as illustrated in Figure 21b. If this is the new best point

$f(\vec{p}_r) > f(\vec{p}_1)$ go to the next step. If it performs better than the second worst point $f(\vec{p}_1) \geq f(\vec{p}_r) \geq f(\vec{p}_n)$ go to step six. Otherwise, go to step four.

3. Expand the reflection point using the expansion coefficient $\gamma > 1$, $\vec{p}_e = \vec{p}_m + \gamma(\vec{p}_r - \vec{p}_m)$, see Figure 21c. Replace the worst point by the reflection point or the expansion point depending on which performs better and go to step six.

4. Contract the reflected point if it perform better than the worst, leading to $\vec{p}_c = \vec{p}_m + \tau(\vec{p}_r - \vec{p}_m)$, with $0 < \tau < 1$ as the contraction coefficient, see Figure 21d. Otherwise, contract the worst point $\vec{p}_c = \vec{p}_m + \tau(\vec{p}_{n+1} - \vec{p}_m)$, see Figure 21e. Replace the worst point with the contracted point if it performs at least as good and go to step six. Otherwise, go to the next step.

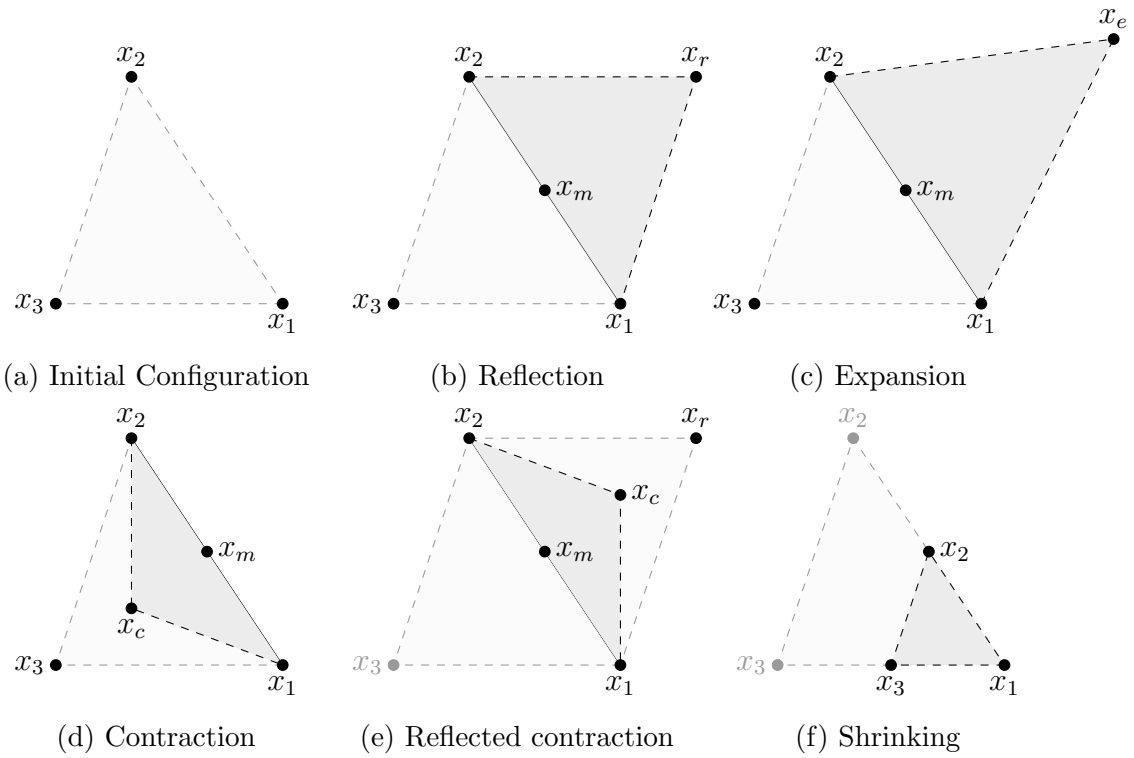
5. Shrink the entire simplex in the direction of the best point $\vec{p}_1$, resulting in $\vec{p}_i = \vec{p}_1 + \sigma(\vec{p}_i - \vec{p}_1)$, with $0 < \sigma < 1$ as the shrinking coefficient. Figure 21f illustrates a shrinked simplex.

6. Terminate if the maximum number of iterations is reached. If the standard deviation of the simplex points performance or the standard deviation in each dimension of the points is below a certain threshold, restart the algorithm. Otherwise, go to step 1.

The optimal values for each coefficient of the downhill simplex search method were found empirically in [24]. Additionally, the result of the downhill simplex method was verified by comparing it with an extensive combinatorial search.

Acceleration After adapting the optimizers to operate on the GPU ray tracer, their runtime decreased massively. The simulation part of the optimization then only took a third of the total runtime, revealing further potential for improvement. Two areas for further acceleration have been identified. The annual integration of the heliostat efficiencies and the collision detection of the generated heliostat positions. Within an annual integration, each heliostat efficiencies are individually integrated. Since each heliostat has a variety of different efficiencies and the annual integration estimates a value for each hour of the year, a lot of computation is required. However, after all efficiencies of a heliostat have been generated, their average, weighted by the I_{DNI}, is used to estimate the overall efficiency of the heliostat. Thus, the hourly efficiencies are not directly of interest. Instead of interpolating the efficiencies and taking their weighted average, a much less compute intensive approach is to calculate annual weights for each simulation point, which determine their contribution to the weighted average. For an annual integration of n simulation points, the annual weights $\vec{w} = (w_0, .., w_{n-1})^T \in \mathbb{R}^n$ are given by

$$\vec{w} = \frac{\sum_{\vec{p} \in f_{\text{an}}(\vec{q}_0, .., \vec{q}_{n-1})} \vec{p} \cdot I_{\text{DNI}}(\vec{p})}{\sum_{\vec{p} \in f_{\text{an}}(\vec{q}_0, .., \vec{q}_{n-1})} I_{\text{DNI}}(\vec{p})} \tag{48}$$



(a) Initial Configuration (b) Reflection (c) Expansion

(d) Contraction (e) Reflected contraction (f) Shrinking

Figure 21: Illustration of the different steps within the downhill simplex search method, taken from [24].

with $\vec{q}_0 = (1, 0, ..., 0)^T \in \mathbb{R}^n$,..., $\vec{q}_{n-1} = (0, ..., 0, 1)^T \in \mathbb{R}^n$, f_{an} as their annual integration based on the simulation points and $I_{\text{DNI}}(\vec{p})$ as the direct normal irradiation at $\vec{p}$.

To obtain the desired annual weighted average $\overline{\vec{\eta}}$ of the efficiencies $\vec{\eta}_0, .., \vec{\eta}_{n+1}$, they are multiplied with the annual weights leading to

$$\overline{\vec{\eta}} = (\vec{\eta}_0, .., \vec{\eta}_{n+1})^T \vec{w}. \tag{49}$$

Lastly, the collision detection that has been implemented so far, relied on testing the bounding sphere of each heliostat against the bounding sphere of all other heliostats. Even though the collision detection between two bounding spheres is simple to compute, the approach still required $\mathcal{O}(n^2)$ such operation. By utilizing the spatial data structure cell grid, far less collision detections are required [42]. The idea of a cell grid is to separate 2D space into evenly spaced cells and store references to objects contained in it. A cell grid is defined by its starting point $(c_{\text{xmin}}, c_{\text{ymin}})^T$, ending point $(c_{\text{xmax}}, c_{\text{ymax}})^T$, and the number of cells it contains d^2. For an object at $(x, y)^T$, the cell indices i, j are given by

$$i = \left\lfloor \frac{x - c_{\text{xmin}}}{c_{\text{xmax}} - c_{\text{xmin}}} d \right\rfloor, \quad j = \left\lfloor \frac{y - c_{\text{ymin}}}{c_{\text{ymax}} - c_{\text{ymin}}} d \right\rfloor. \tag{50}$$

In our case, the number of cells is chosen to be D^2, with D as the heliostat diameter. As a result, each cell can contain at most one heliostat, and the collision test only needs to be done with the direct neighbor cells, see Figure 22.

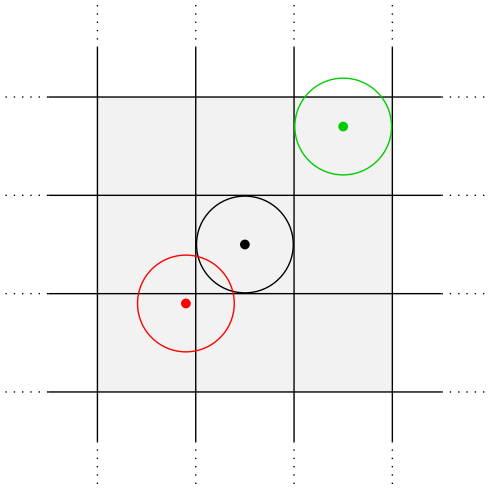


Figure 22: Illustration of the cell grid data structure to accelerate collision detection between placed heliostats. The relevant neighboring cells are shown in lightgray, the current heliostat in black, an overlapping heliostat in red, and a non-overlapping one in green.

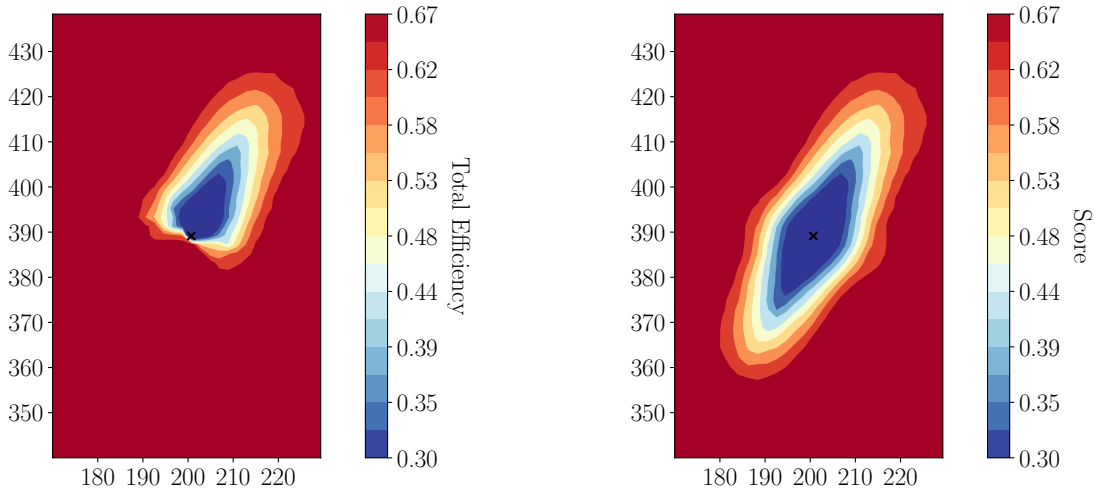
Another approach to optimize heliostat field layout is to sequentially add more heliostats to the field, which will be discussed in the following section.

3.2 Graph based field growth

The straightforward idea of field growth or heliostats growth methods is to sequentially add heliostats to the field by picking the next best spot. There are different implementations of the idea. A study done in [55] first generates an efficiency map of an empty field and then updates the map for each new heliostat to account for shading and blocking (S&B) effects. The map itself is a regular grid containing an estimate of the total heliostat efficiency at each spot. Another method [12] avoids having a finite grid by generating a set of random points at each step of the optimization. The best position is taken from the set and a new one is generated. Before describing our newly developed graph based field growth method, the optimization problem is viewed in more detail.

Consider a simplified version of the problem where only a single heliostat needs to be placed. Since the possible positions are continuous, even this simple problem is likely to be unsolvable. But if the positions are discretized, then each position can be evaluated and the best one chosen. Solving the same problem for two heliostats is, however, a more diffucult task. Intuitively, one could recompute the efficiencies at

each position after placing the first heliostat and again take the best, as done in [12]. Although this approach includes the influence of the first heliostat on other potential heliostats, it misses the fact that the second heliostat might shade or block the first one. Incorporating the latter appears to be far too compute intensive to be feasible. Given the current simulation model, the only option would be to simulate each possible combination of the first heliostat with any other heliostat. However, by further investigating and extending the simulation model, the influence that any potential second heliostat would have on the first one can be computed in just a single simulation. Therefore, each position not only has an efficiency gain obtained by the potential heliostat but also an efficiency loss on other existing heliostats. To incorporate both, a score is calculated by subtracting the loss from the gain. Figure 23a illustrates the efficiency map after placing the first heliostat and Figure 23b the corresponding score map.



(a) Efficiency map

(b) Score map

Figure 23: Comparison between the efficiency map and the score map of a single heliostat marked by the black cross. Both maps were obtained from a simplified test case, based on the PS10.

As shown, the score map looks almost like a mirrored image of the efficiency map, which has been utilized in [55]. Although a useful approximation, they still differ by up to ten percent, see Figure 24. Considering that most heliostat field optimizations of plants like the PS10 achieve less than 1% improvement [43, 10, 38], more precise values are necessary. Our graph based field growth algorithm aims to compute a fully accurate score map without introducing significant computational overhead. In the following sections, the necessary extension to the simulation model to efficiently calculate and update the score map are given.

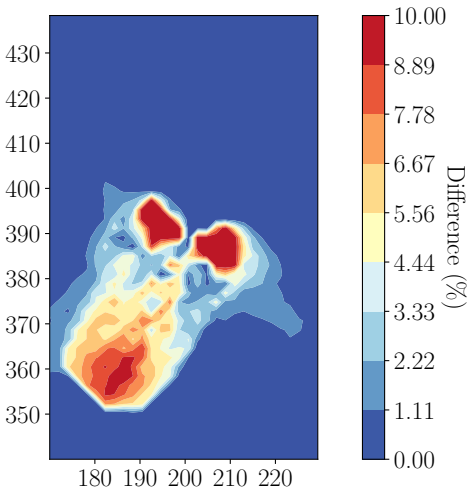


Figure 24: Difference between a mirrored efficiency map as used in [55] and the real score map from our graph based field growth algorithm.

3.2.1 Independent ray tracer

The only way that heliostats influence each other is through shading and blocking. Thus, a further look into the computation is needed. As explained in Section 2.1.1, each ray is evaluated individually for S&B with a precomputed set of potential S&B heliostats. Algorithm 1 shows a pseudo code implementation of the evaluation, with s as the ray source, $\vec{r}$ as the reflected ray direction, $\vec{\tau}_{\text{solar}}$ as the incoming sun direction, and H as the current heliostat.

Algorithm 1 Shading and Blocking calculation between heliostats

```
function CHECKSHADINGBLOCKING(s, r⃗, τ⃗_solar, H)
    for each potential shading heliostat H_S of H do
        if intersects(s, τ⃗_solar, H_S) then
            return True
    for each potential blocking heliostat H_B of H do
        if intersects(s, r⃗, H_B) then
            return True
    return False
```

For the computation of an efficiency map, two types of heliostats need to be differentiated. Active heliostats (●) which have already been placed and inactive ones (○) representing the possible next heliostats. The ray tracer should include S&B from active heliostats to any other heliostat ($\bullet \xrightarrow{sb} \bullet$, $\bullet \xrightarrow{sb} \circ$) but not any influence of inactive heliostats ($\circ \not\xrightarrow{sb} \bullet$, $\circ \not\xrightarrow{sb} \circ$). Both of these are given by simply limiting the potential S&B heliostat calculation to only include active heliostats.

To compute the desired score map, the efficiency reduction that any inactive heliostat would cause on active ones ($\circ \xrightarrow{sb} \bullet$) must be calculated without changing the

current efficiency of active heliostats. Therefore, each inactive heliostat needs a list of efficiencies that the influenced active heliostats would have if the heliostat would become active ($\bigcirc \xrightarrow{sb} \bullet \Rightarrow \bullet \xrightarrow{sb} \bullet$). The total efficiency loss is then given by the sum of the efficiencies of the influenced active heliostats minus the sum of the stored efficiencies. For example, if there is only one active heliostat, then each inactive heliostat stores their efficiency and the efficiency of the active heliostat, for the case that both are considered active. Then the score of the inactive heliostat is given by their efficiency subtracted by the efficiency loss onto the active heliostat.

The efficiency reduction list is not calculated during the simulation of the inactive heliostat but rather when simulating the active one. Hence, the simulation of an active heliostat results in a list of efficiencies that the active heliostat would have if one or none of the inactive ones would exist. Including inactive on active ($\bigcirc \xrightarrow{sb} \bullet$) interactions, requires adding inactive heliostats to the potential S&B computation of the active heliostats. However, instead of discarding a ray from an active heliostat intersecting an inactive one, it is further evaluated but marked as intersecting for the inactive heliostat. Thus, the S&B computation of the new independent ray tracer returns a list of boolean values stating whether the ray intersects certain inactive heliostats. Additionally, S&B between active heliostats is evaluated as usual. Algorithm 2 illustrates the independent S&B calculation, with n as the number of inactive heliostats potentially intersecting the current heliostat. Note that the last entry of the list SB is used to store whether the ray is blocked by any active heliostats.

Algorithm 2 Independent Shading and Blocking calculation between heliostats.

```
1:  function checkIndependentShadingBlocking(s, r⃗, τ⃗_solar, H)
2:      if checkShadingBlocking(s, r⃗, τ⃗_solar, H) then
3:          SB[0, .., n] ← {true, ..., true}
4:          return Eff_ind
5:      SB[0, .., n] ← {false, ..., false}
6:      for each potential shading heliostat H_S of H do
7:          if H_S is inactive and intersects(s, τ⃗_solar, H_S) then
8:              SB[getId(H_S)] ← true
9:      for each potential blocking heliostat H_B of H do
10:         if H_B is inactive and intersects(s, r⃗, H_B) then
11:             SB[getId(H_B)] ← true
12:     return SB
```

The remaining evaluation within the ray tracer also needs to be adjusted accordingly to calculate the efficiency that the heliostat H would have if one or none of the inactive heliostats would exist. Finally, the calculated heliostat efficiencies *Eff* as well as the efficiency reduction lists Eff_{ind} are returned.

For further acceleration, the independent ray tracer should be capable of only considering certain efficiency reductions. An unordered set of inactive heliostat ids can be passed to each active heliostat to implement this efficiently. Within the indepen-

dent S&B calculation, the Eff_{ind} list is reduced to the size of the unordered set. Each inactive potential S&B heliostat is checked whether it is included in the unordered set. Even though not specified in the definition of the independent ray tracer, it can also compute S&B interactions of type $\bigcirc \xrightarrow{sb} \bigcirc$. These are later used to determine all possible S&B interactions. Additionally, the ray tracer should be able to simulate a specified subset of the heliostats while the rest is only considered for their shading and blocking interactions onto the simulated ones.

Which heliostats need to be simulated when updating the score map, will be described in the next sections.

3.2.2 Shading and blocking graph

Before discussing the details on how to determine the heliostats that need to be simulated, a formal definition of the necessary variables is given. All of these variables are stored in a shading and blocking graph (S&B graph) which is a partially weighted directed graph. It should contain the efficiencies of active and inactive heliostats as well as any type of S&B interaction $\{\bullet, \bigcirc\} \xrightarrow{sb} \{\bullet, \bigcirc\}$. However, for performance reasons only interactions of type $\bigcirc \xrightarrow{sb} \bullet$ as well as the efficiencies are continuously updated. They contain all the necessary information to calculate the score of inactive heliostats. Let V be the set of all possible heliostats, $HF \subseteq V$ the set of active heliostats, then

- The S&B graph is given by $G = (V, E)$ and for all $H_i, H_j \in V$ with $H_i \xrightarrow{sb} H_j$ there exists an edge $(H_i, H_j) \in E$
- $Eff(H_i)$ is the total efficiency of a node $H_i \in V$ in the current heliostat field HF
- $Eff_{\text{red}}(H_i, H_j)$ for $(H_i, H_j) \in E$ with $H_i \in \overline{HF}, H_j \in HF$ $(\bigcirc \xrightarrow{sb} \bullet)$ is the efficiency of H_j if H_i would become active
- $Ch(H_i)$ for $H_i \in V$ is a set containing all children $H_j \in V$ of H_i in G
- $Pa(H_i)$ for $H_i \in V$ is a set containing all parents $H_k \in V$ of H_i in G
- $Loss(H_i) = \sum_{H_j \in Ch(H_i) \cap HF} (Eff(H_j) - Eff_{\text{red}}(H_i, H_j))$ for $H_i \in \overline{HF}$ is the efficiency loss introduced by H_i
- $Score(H_i) = Eff(H_i) - Loss(H_i)$ for $H_i \in \overline{HF}$ is the score of H_i

Note that even if not explicitly specified, all variables depend on the heliostat field HF.

For an efficient implementation of the S&B graph, a few requirements must be met. The data structure should only store references in the form of indices to the actual heliostat objects. Fast access to the node of a heliostat as well as its parents and children must be provided. In addition, a quick iteration over a nodes parents and children is needed, as it will be used extensively. Other variables like the efficiency of a node and the weight of an edge also require fast access. To fulfill all the requirements, the data

structure for an S&B graph is as follows. The graph consists of a dynamic list of nodes where the node of each heliostat H_i is stored at the i-th entry. Each node contains the heliostat's current efficiency, a dynamic list of edges to its parents, and a dynamic list of edges to its children. An edge consists of the index to the other heliostat and the efficiency reduction.

There are also some very interesting theoretical aspects of a more general form of the S&B graph. It is possible to adjust the independent ray tracer to calculate what efficiency H_i would have if any combination of heliostats H_k with a potential $H_k \xrightarrow{sb} H_i$ interaction would be active. Doing so for each heliostat contains all information needed to calculate the heliostat field efficiency of any possible combination of heliostats. Consider for example the PS10 in Spain [44]. The field could contain roughly 2300 heliostats of the type that was used in the plant. 624 heliostats were used. Suppose the overall efficiency of any possible heliostat field with 624 out of 2300 heliostats should be calculated. The naive approach of directly simulating each combination results in $\binom{2300}{624} \approx 1.47 \cdot 10^{582}$ full simulations which is an inconceivably large number, far more than there are atoms in the observable universe ($\sim 10^{80}$) [16]. However, with a general S&B graph, it would only take a single simulation generating about $2300 \cdot 2^{20} = 2.41^9$ efficiency values. Here, the factor 2^{20} comes from the fact that there are on average about 20 heliostats that shade or block a heliostat. After that simulation, it is only a matter of a few additions to compute the overall efficiency of any possible heliostat field, not just the ones consisting of 624 heliostats. The downside of this approach is of course its memory consumption of at least 20GB of data. Especially for dense fields with a lot more data this is impractical.

How the S&B graph is initialized and updated will be discussed in the following section.

3.2.3 Complete graph based field growth algorithm

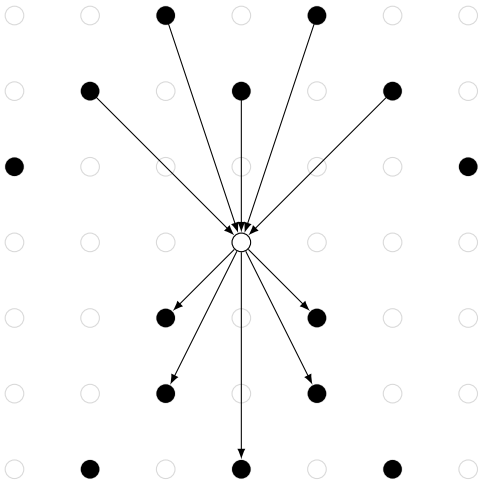


Figure 25: Illustration of the principle behind the field growth method. The score at each position does not only take the effects of the field onto the heliostat into account ($\bullet \xrightarrow{sb} \circ$) but also the heliostats influence onto the field ($\circ \xrightarrow{sb} \bullet$).

An illustration of the field growth method idea is shown in Figure 25. For the initialization of the S&B graph, all heliostats are considered inactive ($HF = \emptyset$). Their efficiencies and the S&B edges are computed by the independent ray tracer and the S&B graph is updated accordingly. Since all heliostats are inactive, their loss is zero and their score is given by their efficiencies ($Loss(H_i) = 0$, $Score(H_i) = Eff(H_i)$ for $H_i \in \overline{HF}$). The resulting S&B graph contains an edge for any possible shading and blocking interaction. So when adding new heliostats to the heliostat field HF, no new S&B edge is created. At most, existing edges might be modified or not be valid anymore.

After placing a new heliostat, its influence on the S&B graph must be considered, and the efficiencies as well as $\circ \xrightarrow{sb} \bullet$ interactions updated. All heliostats that overlap the newly placed one are removed from the graph. Neighboring heliostats can efficiently be calculated by a cell grid as introduced in Section 3.1.6. Now let H_i be the newly added heliostat, then there are three types of influences from H_i to the rest of the heliostats. First, the efficiencies of the children of H_i are reduced which is partially given in $Eff_{\text{red}}(H_i, H_j)$. Second, a new weight $Eff_{\text{red}}(H_k, H_i)$ for the edge from the inactive parents $H_k \in Pa(H_i) \cap \overline{HF}$ to H_i is added. Finally, the weight $Eff_{\text{red}}(H_l, H_j)$ of the active children's $H_j \in Ch(H_i) \cap HF$ inactive parents $H_l \in Pa(H_j) \cap \overline{HF}$ of H_i might be influenced since the interaction $H_i \xrightarrow{sb} H_j$ can reduce the potential S&B of H_l to H_j. Figure 26 illustrates the different types of influences from H_i. Only interactions of type $\circ \xrightarrow{sb} \bullet$ can possibly be removed, the rest are not changed. This ensures that any S&B interaction are contained in the graph.

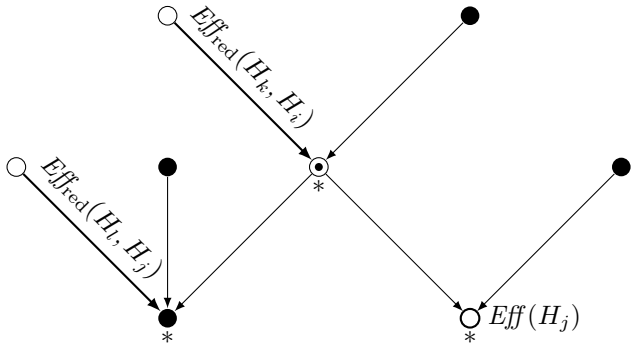


Figure 26: Influence of a new heliostat H_i ($\odot$) on the S&B graph, with H_j as the children of H_i, H_k as the parents, and H_l as the childrens parents. Values that need to be computed when updating the graph are written down. Only heliostats marked with a star need to be simulated, the rest is just considered for their S&B effects.

As shown, the influence of H_i closely resembles a Markov blanket within a bayesian network. Possible reasons and similarities are discussed in the following. As a reminder, the Markov blanket of a target node consists of all nodes that make the target conditionally independent of the rest [31]. Or in other words, the smallest set of nodes that must be observed such that the probability of the target node is fully determined. There is no direct correspondence from probability to any measure in the S&B graph. However, there are some similarities between probability and the score of a node which causes the resemblance to a Markov blanket. Calculating the score of a S&B node requires its efficiency and efficiency reduction on other active nodes. Therefore, the node's active parents, its active children, and active children's parents must be considered which corresponds to its Markov blanket, see Figure 27. When updating the S&B graph after adding H_i, its score is of course already known. However, the score of other inactive nodes is changing, each requiring their own Markov blanket to be fully determined, see Figure 28. But, in combination with the existing information in the S&B graph, only the nodes shown in Figure 26 are necessary. Due to the connection to a Markov blanket, we call the nodes of Figure 26 the S&B blanket of H_i.

To update the variables in the S&B graph, not all of those nodes need to be simulated. Only H_i and its children $Ch(H_i)$ have to be simulated. The rest will be used to determine their S&B influence on the simulated ones. Furthermore, all active simulated heliostats must independently evaluate their inactive parents. A pseudo code implementation is given in Algorithm 3.

In addition to the classical regular grid positions, the possible heliostats in V can also be taken from one of the patterns of Section 3.1. Since they generate non-overlapping heliostats, the heliostat diameter used to generate the pattern is scaled by an additional parameter.

Another extension to the general field growth algorithm is a relaxed score. Due to the greedy nature of the algorithm, it will result in fields that are spread out and not

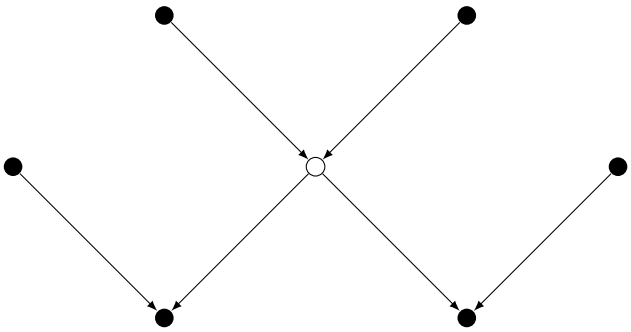


Figure 27: All heliostats necessary to calculate the score of an inactive heliostat (○). This corresponds to the Markov blanket of a node in a Bayesian network.

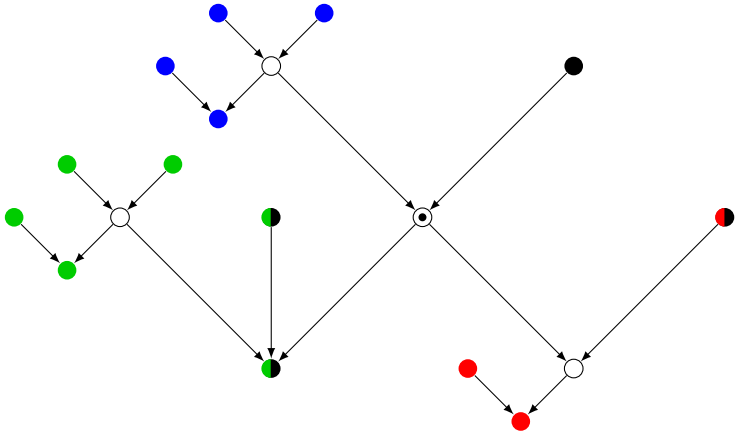


Figure 28: Full influence of a newly activated heliostat (⊙) onto the score of other inactive heliostats (○). The required nodes to calculate the score of the node itself, its inactive parents, inactive children and inactive childrens parents are shown in black, blue, red, and green, respectively. Given the existing knowledge in the S&B graph these reduce to the nodes shown in Figure 26.

take into account the loss of possibly efficient positions. Therefore, it can be beneficial to favor a dense packing. Thus, a density factor is added to the score computation given by

$$Density(H_i) = \begin{cases} \min\left(dis(H_i, H_c)/d_{\max}\right), 1) & |HF|/N < n_{heli} \\ 0 & \text{else} \end{cases} \tag{51}$$

with $dis(\circ, \circ)$ as the distance between two heliostats, H_c as the closest heliostat to H_i, $d_{\max}$ as the maximum distance to consider, N as the number of heliostats to place, and n_{heli} as the fraction of heliostats where the density loss should be applied.

Before continuing with some practical difficulties and how to overcome them, some theoretical aspects of our graph based field growth method are briefly viewed. Given a heliostat field HF and a set of possible positions V, the field growth method will

Algorithm 3 Field growth method on a given set V of possible positions.

```
1:  function FIELDGROWTH(V)
2:      toSim ← V
3:      active[0, .., n − 1] ← {false, ...false}
4:      Eff, Eff_ind ← independentRaytrace(V, toSim, active)
5:      G ← buildSBGraph(Eff, Eff_ind)
6:      while len(G.HF) < desiredHeliostats do
7:          H_i ← G.getHighestScoringHeliostat()
8:          G.removeIntersectingHeliostats(H_i)
9:          MB ← ∅
10:         toSim ← {H_i}
11:         MB.insert(G.getParents(H_i))
12:         for each children H_j in G.getChildren(H_i) do
13:             MB.insert(H_j)
14:             toSim.insert(H_j)
15:             for each parent H_l in G.getParents(H_j) do
16:                 if H_j is active or H_j is active then
17:                     MB ← MB ∪ {H_l}
18:         active ← G.determineIfActive(MB)
19:         Eff, Eff_ind ← independentRaytrace(MB, toSim, active)
20:         G.updateEfficiencies(Eff)
21:         G.updateEdgeWeight(Eff)
22:         G.recalculateScores()
```

find the heliostat H_i in V that, when added to HF, results in the highest overall field efficiency. However, this does not mean that the heliostat field $HF \cup \{H_i\}$ is the optimal heliostat field from V of that length. It rather is the best heliostat field possible when viewing HF as static and extending it by one heliostat. Since adding a new heliostat will lead to more and not less S&B, the next best score is always less or equal to the current best score. Therefore, the resulting heliostat field of the graph based field growth algorithm is locally optimal in the sense that replacing a single heliostat to any other position will never result in higher overall efficiency.

3.2.4 Practical difficulties

There are some practical difficulties arising with larger heliostat fields or in general with a huge number of possible heliostats. The independent ray tracer generates a lot of variables, especially at the initialization of the S&B graph. For an annual result, all of these variables need an hourly integration and their mean taken afterward. However, as discussed in Section 3.1.6, weights can be calculated for each simulated moment which drastically decreases the runtime and memory consumption. Furthermore, the calculated heliostat efficiencies need to be accurate and stable. Otherwise, the field growth method is not able to choose the next best heliostat. Monte Carlo based ray

tracers are not well suited for that purpose, as they require a lot of rays to reduce fluctuations within the total result. For a whole heliostat field, the effect is not as severe, since the total number of rays is large. However, for a single heliostat, only a few rays are simulated causing large fluctuations within the result. Analytical ray tracers do not rely on the law of large numbers and are therefore better suited.

The memory consumption of the independent ray tracer is much larger than the traditional ray tracer. Therefore, the GPU implementation of the independent ray tracer can differentiate between active and inactive heliostats as well as simulate a specified subset of the given heliostats. However, it lacks the possibility to calculate the independent efficiency list $\mathit{Eff}_{\text{ind}}$ as a GPU typically does not offer the necessary memory. As shown later, there are specific use cases for the GPU version of the independent ray tracer.

Even on the CPU, the memory needs of the independent ray tracer might become higher than what is offered by RAM. To reduce memory consumption without losing relevant information, two adjustments to the field growth method have been made. During the initialization, only relevant neighbors are considered for S&B effects shrinking the data used by the independent ray tracer. Furthermore, a minimum efficiency reduction can be specified. Any S&B below that is simply ignored. For that purpose the edge weight $\mathit{Eff}_{\text{red}}$ is extended to contain a value for all possible edges. For interactions other than $\circ \xrightarrow{sb} \bullet$, the efficiency reductions from the initialization are taken. They are not updated, but offer an upper limit as the actual efficiency reduction can not increase by placing more heliostats. Not only the memory consumption is reduced, but also the runtime as fewer heliostats need to be simulated when updating the S&B graph.

However, the memory consumption for very dense heliostat fields can still be too large. Further improvements have been made with two subvariants of the graph based field growth method. One utilizes an additional sub optimization and the other some representative heliostats to account for S&B of the new heliostat to the existing field.

3.2.5 Suboptimization

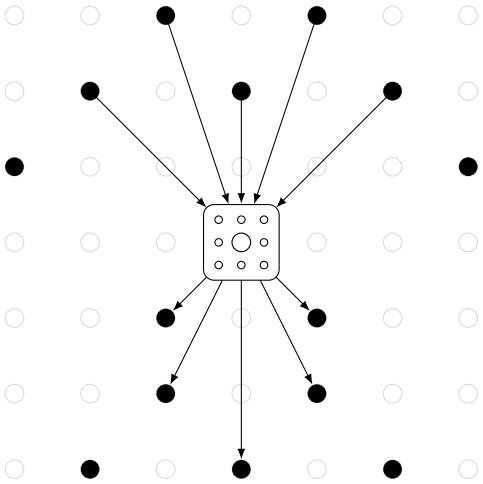


Figure 29: Illustration of the field growth suboptimization. After choosing the best scoring position, subpositions are generated and their scores computed. The best scoring subpostion is chosen.

The sub optimization version of the field growth algorithm allows to consider a lot of heliostat positions without keeping all the necessary information in memory. It is based on the assumption that the score map is smooth i.e., there are no drastic differences between the score of neighboring heliostats for a fine grid of possible positions. Thus, the best position of a fine grid is likely to be near the best position of a coarse grid. The sub optimization therefore first determines the highest scoring position of a coarse grid like the field growth method but with an additional sub optimization afterward. Here, the grid position of the best heliostat H_i is further discretized and the scores of the generated positions V_{sub} are determined, as illustrated in Figure 29. However, the newly generated positions are not contained in the S&B graph so their score need to be calculated individually. As discussed in the last section, when just the score of a node is of interest, only active heliostats of the S&B blanket are required, see Figure 27. But, no S&B interactions between existing heliostats (V) and newly generated ones (V_{sub}) are known. To accurately approximate them, the S&B interactions of all heliostats in V within a given distance to H_i are considered. So any child or parent of a helisotat within that radius is regarded as a child or parent of the heliostats in V_{sub}, respectively. Again, only the children as well as the heliostats in V_{sub} need to be simulated. The result of the independent ray tracer contains the efficiencies of all heliostats in V_{sub} in the heliostat field HF and their efficiency reduction onto the existing heliostats. Thus, the loss of the new heliostats and also their score can be computed. Afterward, the position of H_i is changed to the highest scoring sub position. Finally, the updating procedure of the field growth method is executed to maintain a valid S&B graph. So the field growth sub optimization requires two simulations for placing a heliostat, where one only contains a few heliostats to simulate.

Since a valid S&B graph is ensured after each heliostat placement, the local optimum property of the field growth algorithm is also given in the sub optimization version. The downside of this approach, however, is that the overall best sub position might not be at the best position in the S&B graph. To increase the chance of picking the best sub position, the following field growth version always maintains some information about the sub heliostats.

3.2.6 Representative field growth

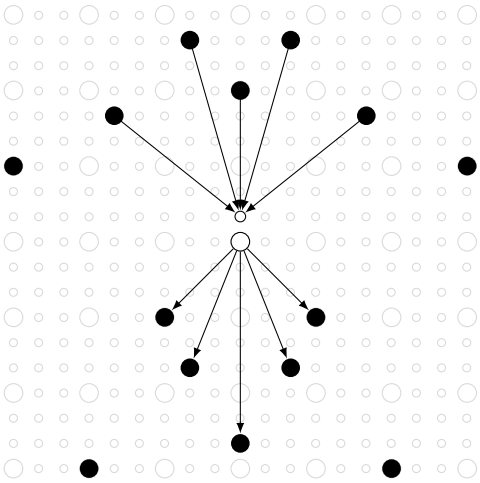


Figure 30: Principle behind the representative field growth method. The score of non-representative heliostats, marked by the small circle, contains an exact efficiency evaluation but approximates its influence onto the field by a neighboring representative heliostat, marked by the larger circle.

Similar to the sub optimization, the representative field growth method differentiates between a fine grid of positions and a coarse one. While the S&B interactions are also mostly considered between heliostats of the coarse grid V, the S&B graph $G = (V \cup V_{\text{sub}}, E)$ now contains all heliostats. The idea is to calculate efficiencies for all heliostats but the S&B of newly added heliostats onto existing ones is approximated by representative heliostats, see Figure 30. Any heliostat that is in the coarse grid V is regarded as a representative heliostat.

During the initialization, the potential efficiency reduction between the representative heliostats is needed. For non-representative heliostats (V_{sub}) only their efficiency must be determined. As stated in Section 2.1.3, these can quickly be calculated by the GPU version of the independent ray tracer. The loss of non-representative heliostat is approximated by the loss of the nearest representative heliostat. Again the best scoring heliostat is taken and its influence on the S&B graph is determined. S&B interaction of non-representative heliostats are only taken into account when they become active. They are first approximated using neighboring representative heliostats

and then determined by the independent ray tracer. When updating the S&B graph, all representative or active heliostats of the S&B blanket are simulated on the CPU to obtain the efficiency reductions. Furthermore, all non-representative heliostats within a certain distance of the evaluated representative heliostats are simulated on the GPU for acceleration.

Besides the benefits of incorporating a lot of possible heliostat positions, there is also a drawback to the representative field growth method. Since the loss for non-representative heliostats is approximated, their score might not be correct. Therefore, the local optimum property is not given anymore. However, it can be approached again by replacing heliostats. It is enough to always replace the worst heliostat, because if there is no better position for that heliostat then there is no better position for any other heliostat. To remove a heliostat H_i, it is regarded as inactive, all its neighboring positions only overlapping H_i are added again, and the S&B graph is updated to account for the changes. Then the usual procedure of adding the highest scoring heliostat follows.

3.2.7 Validation

The independent ray tracers are validated as follows. A set of 20 closely packed inactive heliostats were simulated by the independent ray tracer. Then their efficiencies on an empty field was computed using the traditional ray tracer and compared to the corresponding result from the independent ray tracer. Furthermore, any combination of two from the 20 heliostats was simulated by the traditional ray tracer. Again, the results were compared to the efficiency reduction of the independent ray tracer. By using the convolution based ray tracer for the validation, an exact comparison was made possible. The results of the independent ray tracer exactly match the ones from the traditional ray tracer.

A full validation of the graph based field growth algorithm was done by validating the relevant variables in the S&B graph after each step. For this, a new S&B graph was computed by simulating every heliostat as well as the efficiency reduction from inactive heliostats to active ones ($\circ \xrightarrow{sb} \bullet$). These are all the information to determine the score of each heliostat. The computed values are then compared to the values stored in the S&B graph of the field growth method. All settings of the CRS for the validation were based on the PS10 and the independent convolution ray tracer was used. No differences between the validated and the actual S&B graph were observed at any step of the field growth method.

Similarly, the sub optimization based field growth algorithm was validated. Since the S&B interactions of the sub heliostats were approximated, the validation result depend on how many neighboring heliostats were used. However, when including heliostats within a radius of two times the heliostat diameter almost no differences were observed. Likewise, the representative field growth method showed almost no differences to the validation when such a radius was used.

3.3 Local search

The local search optimization further improves the heliostat field generated by any optimization method. It is based on the works of Buck [10] with some adjustments done in [24]. The idea is to optimize each given heliostat individually, by replacing them within a region around their initial position. Instead of using a regular grid of possible new positions as done in [10], the local search algorithm generates a circular grid as shown in Figure 31. Exact details of the grid structure can be found in [24]. As before, the S&B effects from the field on the new positions, as well as from the new positions onto the field need to be considered. Therefore, a subgroup of heliostats within a certain distance is included. A parameter study was carried out to determine at which distance heliostats should be within the subgroup. However, one downside of this approach is the individual evaluation of each possible position. Since 16 possible new positions were typically considered, optimizing a single heliostat required 16 annual simulations of the subgroup. But with the new independent ray tracer, one heliostat can be optimized with a single annual simulation. Therefore, reducing the total number of simulations by a factor of 16. Similar to the field growth method, the score of each possible new position is calculated by computing its efficiency within the subgroup and subtracting its efficiency loss due to S&B of the subgroup. The highest scoring position is chosen and the next heliostat optimized. The local search algorithm terminates either if the improvement is below a certain threshold or a specified number of iterations is reached.

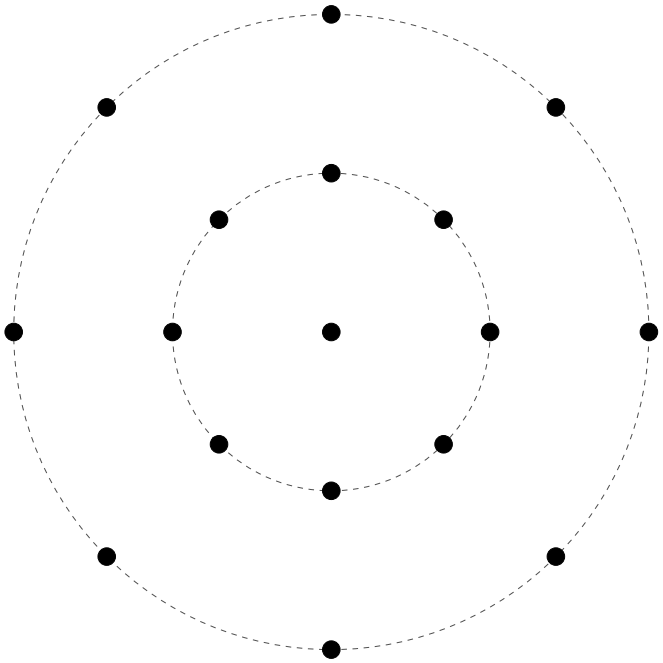


Figure 31: Circular grid of the possible positions when optimizing a heliostat with the local search algorithm, taken from [24].

4 Case Studies

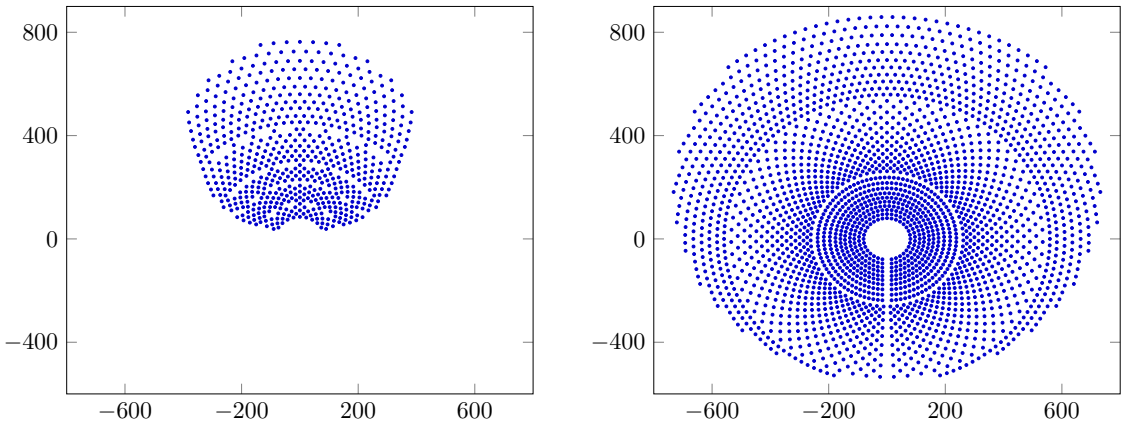
In the following, various aspects of the ray tracing methods are investigated. First, our optical model is validated against the Monte Carlo based ray tracing tool *SolTrace* in a large case study based on the PS10 and Gemasolar. Then, our new convolution of the sun slope and tracking error is compared against a separate perturbation of the sun and normal vector as well as the convolution equations from *HFLCAL* and *UNIZAR*. Afterward, the accuracies of all ray tracers as well as their CPU and GPU implementation are compared. An investigation of the solar plant efficiencies then gives deeper insights into our analytical ray tracer before getting to a direct run time comparison of all ray tracers. The final two studies examine the newly implemented sun shapes and how well they can be approximated in the convolution methods.

The used test cases are predicated on the solar plant Planta Solar 10 (PS10) in Spain [44] which utilizes 624 heliostats of the type Sanlúcar 120 as well as the Gemasolar in Spain having 2650 heliostats of type HE35 [11]. Figure 32 displays the positioning of the heliostats. Each heliostat of the PS10 consists of 28 facets with a total mirror area of about 120 m^2 [44]. The HE35 heliostat has 35 facets and a total mirror area of about $116m^2$ [11]. For the validation, two receiver types, namely the cylindrical cavity receiver and the flat receiver were used in the PS10 test cases. For each plant and receiver type, three different sun positions as shown in Table 8 were evaluated during the validation in order to account for various shading and blocking effects. All other case studies use the cavity receiver in the PS10 test case and both the PS10 and Gemasolar were annually simulated using actual weather data from the American Society of Heating, Refrigerating and Air-Conditioning Engineers (ASHRAE). The annual integration method as described in [54], was set such that the limit of the Monte Carlo ray tracer achieved an accuracy of 99.95% compared to simulating each hour of the year individually. Table 7 shows the general setup for the test cases and Table 9 the annual settings.

4.1 Modeling

4.1.1 Validation of SunFlower

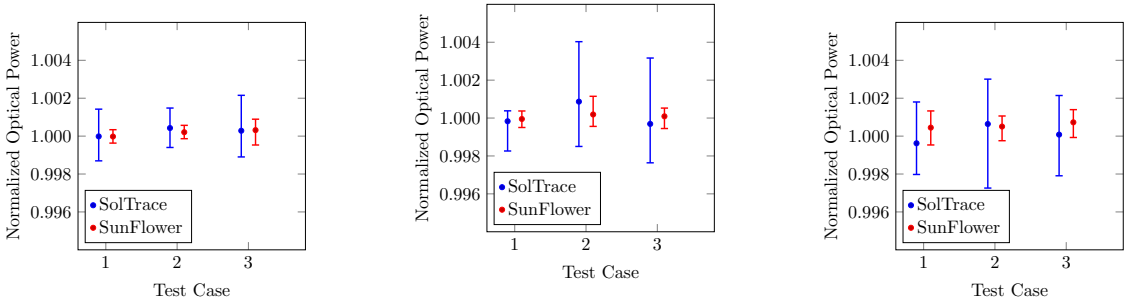
Since our definition of the cavity receiver is not included in *SolTrace*, we constructed it out of rectangular receivers. However, *SolTrace* does not model the effect of tower blocking and thus it had to be turned off in *SunFlower* during the tests. *SunFlower* and *SolTrace* simulated each test case ten times with roughly one million rays. Here, the bidirectional Monte Carlo ray tracer of *SunFlower* was used. The resulting total optical power was then normalized by the average power of *SunFlower* simulating ten million rays ten times. Figure 33 shows the minimal, maximal, and average results of both tools for all test cases. As presented, the average result as well as the minimum and maximum of *SunFlower* is always in between the minimal and maximal results of *SolTrace*. Moreover, the highest deviation of the average results from both tools



(a) PS10 (b) Gemasolar

Figure 32: Heliostat field layout of the tested solar plants.

is less than 0.09 % and the maximal fluctuation of the results of *SolTrace* is about 0.55 %, whereas for *SunFlower* it is only 0.18 %. *SunFlower* achieves higher precision since rays are generated on the heliostat surface. Therefore, no rays are wasted due to ground impacts as in the forward Monte Carlo ray tracer of *SolTrace*.



(a) PS10 with flat receiver (b) PS10 with cavity receiver (c) Gemasolar

Figure 33: Validation of *SunFlower* against *SolTrace* for test cases with various sun positions and solar plants. They are based on the test cases defined in Table 8 and Table 7. The average power of the bidirectional Monte Carlo ray tracer simulating ten million rays ten times was used as normalization.

4.1.2 Accuracy of convolution method

As described in Section 2.1, three types of errors influencing the actual reflection direction of an incoming solar ray are considered. The sun error describing a deviation of the incoming ray and the slope and tracking error for the deviation of the mirror normal. From the results, the deviated reflected ray can be computed. However, we convolute the perturbations and describe them in terms of the reflected ray. Therefore,

Parameter	PS10 flat	PS10 cavity	Gemasolar
Latitude	37.26°	37.26°	37.56°
Longitude	-6.14°	-6.14°	-5.33°
Sun Error	2.35 mrad	2.35 mrad	2.35 mrad
Global slope error vertical	1.4 mrad	1.4 mrad	1.4 mrad
Global slope error horizontal	1.4 mrad	1.4 mrad	1.4 mrad
Tracking error horizontal	1 mrad	1 mrad	1 mrad
Tracking error vertical	1 mrad	1 mrad	1 mrad
Heliostat type	Sanlúcar 120	Sanlúcar 120	HE35
Number of heliostats	624	624	2650
Heliostat reflectivity	88 %	88 %	88 %
Heliostat facet type	Flat	Flat	Flat
Canting	On axis	On axis	On axis
Tower type	Rectangular	Rectangular	Circular
Tower height	120 m	120 m	140 m
Tower length	18 m	18 m	-
Tower width	8 m	8 m	-
Tower diameter	-	-	8 m
Receiver type	Rectangular	Cavity	External
Number of receiver panels	1	4	18
Receiver panel width	13.78 m	3.445 m	1.476 m
Receiver panel height	12 m	12 m	10.5 m
Receiver raise height	-	2.5 m	-
Receiver tilt angle	11.5°	-	-

Table 7: Basic setup for the validation test cases. The settings are inspired by the PS10 plant and the Gemasolar plant.

a comparison to a separate perturbation is needed to evaluate the accuracy of the convolution. For this, Gemasolar and PS10 were simulated annually with different slope errors while keeping the sun error at 2.35 mrad and the tracking error at 0 mrad. The Monte Carlo ray tracer simulated each setting ten times with over ten million rays. First the sun and normal vector were perturbed separately and used as normalization. Besides the new convolution equation of *SunFlower*, the one from *HFLCAL* as well as the equation from *UNIZAR* were tested whose relevant parts are defined as follows [13]

$$\begin{aligned} \sigma_{\text{HFLCAL}} &= \sqrt{(\sigma_{\text{sun}})^2 + (\sigma_{\text{tracking}})^2 + (2\sigma_{\text{slope}})^2}, \\ \sigma_{\text{UNIZAR}} &= \sqrt{(\sigma_{\text{sun}})^2 + (\sigma_{\text{tracking}})^2 + 2(1+\cos\omega^2)(\sigma_{\text{slope}})^2}, \end{aligned} \tag{52}$$

where ω is the incident angle of the incoming solar ray. Note that *HFLCAL* and *UNIZAR* defines the tracking error in terms of the reflected ray and not in terms of the normal as done in *SunFlower*.

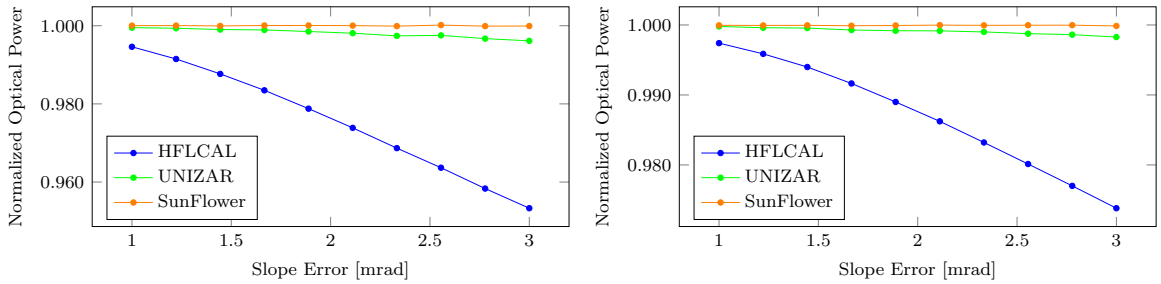
Test	Azimuth	Altitude	DNI
1	80°	30°	710 W/m^2
2	110°	60°	820 W/m^2
3	180°	70°	850 W/m^2

Table 8: Sun configuration for the validation test cases.

Method	Hours/day	Days/year
Bicubic	20	11

Table 9: Annual integration settings.

Figure 34 shows the normalized result of each equation for an annual simulation of the PS10 and Gemasolar. Clearly, the *HFLCAL* convolution leads to the largest differences to a separate perturbation of sun and normal vector with a maximum difference of 4.5%. *UNIZAR* performs much better but still has a maximum error of 0.4%. Moreover, for both equations, the error increases with the tracking and slope error. *SunFlower* however performs much better with a maximum difference of 0.01% making it up to 400 times more accurate than *HFLCAL* and 40 times more accurate than *UNIZAR*.



(a) Gemasolar (b) PS10

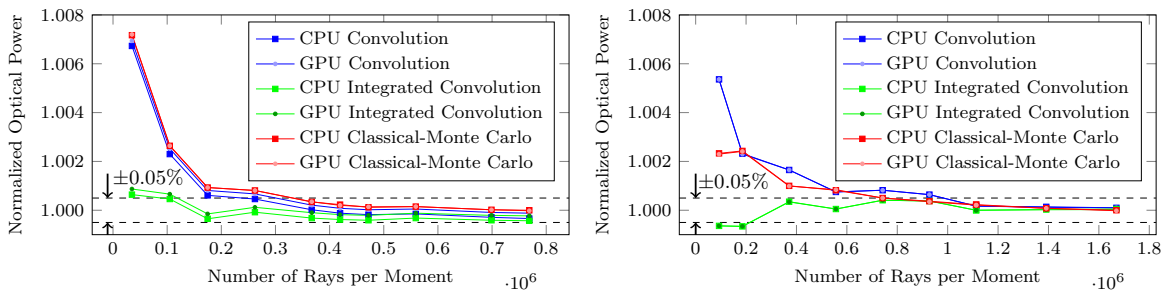
Figure 34: Comparison of different functions to convolute the sun, slope, and tracking error. All results are normalized by a separate perturbation of the sun and normal vector.

4.1.3 Comparison of convolution methods on GPU and CPU

Now that the Monte Carlo ray tracer of *SunFlower* has been validated against *SolTrace* as well as a separate perturbation of the sun and normal vector, a comparison against our convolution and integrated convolution ray tracer is given. All ray tracers have been implemented on the CPU as well as the GPU. It is of course crucial that both versions give similar results in the limit. To validate this, the annual PS10 and Gemasolar test case with different amounts of rays per sun position were simulated. The fluctuations of the Monte Carlo ray tracer were accounted for by simulating each setting ten times and taking the average result. Every result got normalized by the average of the CPU Monte Carlo ray tracer simulating over ten million rays ten times. Figure 35 shows the results of each ray tracer. First, all ray tracers approach the same limit within an

error of less than 0.05%. Not only do all ray tracers implementations have the same limit, but the GPU and CPU versions also give almost identical results for each setting with a maximum deviation of around 0.02%. Additionally, the convolution ray tracer and the Monte Carlo ray tracer give very similar results for each setting on the PS10 test case. The reason is that the convolution ray tracer is very similar to the limit, in terms of samples per ray, of the multi Monte Carlo ray tracer.

It should be noted that an annual simulation is an ideal setting for the Monte Carlo ray tracer with the reason being the law of large numbers. For example, the Gemasolar features 2650 heliostats each consisting of 5x7 facets which is simulated with 210 different sun positions. Thus, even when simulating only one ray per facet, over 18 million rays are used for the annual simulation. Therefore, fluctuations within the computations quickly cancel out eliminating the need to more accurately evaluate each ray, as done in the convolution ray tracer. Lastly, the integrated convolution ray tracer achieved much more accurate results than all other ray tracers. Even when taking only one ray per heliostat facet, the accuracy is very close to 99.95% on both test cases. Compared to the convolution ray tracer, the results are up to eleven and nine times more accurate for the PS10 and Gemasolar, respectively. In the following, we evaluate why exactly the integrated convolution ray tracer performs much better and increase its accuracy even further.



(a) PS10 (b) Gemasolar

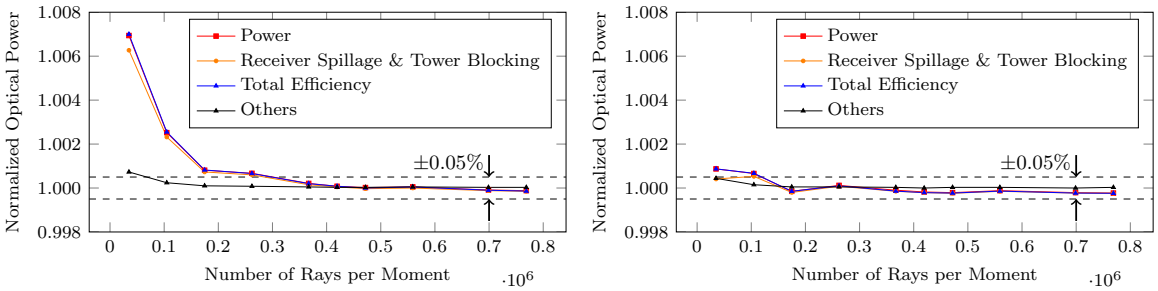
Figure 35: Comparison of different ray tracer versions for various number of rays on annual PS10 and Gemasolar simulations.

4.1.4 Efficiency Accuracies

To get a deeper understanding of the results of our analytical ray tracers, their efficiencies are investigated in the following. Each efficiency got normalized by corresponding average efficiency of the Monte Carlo ray tracer simulating over ten million rays ten times.

Figure 36 shows the relevant efficiencies for an annual simulation of the PS10. Since the total efficiency is calculated from the product of the decoupled individual efficiencies, it is potentially different than the actual computed power. However, as shown the difference is very small with at most 0.02%, indicating that the accuracies of the

individual efficiencies can be used to estimate the accuracy of the overall power. For simplicity, the receiver spillage and tower blocking efficiencies are combined into one. As shown on the left of Figure 36 it is exactly this efficiency that is not accurately represented by few rays. From the formulation of the integrated convolution ray tracer, one would expect these efficiencies to be more accurate when using few rays which is also what is observed on the right of Figure 36.



(a) Convolution

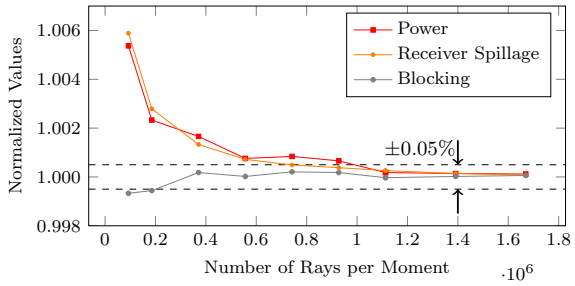
(b) Integrated Convolution

Figure 36: Accuracy of different efficiencies as well as the overall solar power of both analytical ray tracers for an annual simulation of the PS10. For better readability, only the relevant efficiencies are shown which are normalized by the corresponding efficiency of the Monte Carlo ray tracer with over a million rays.

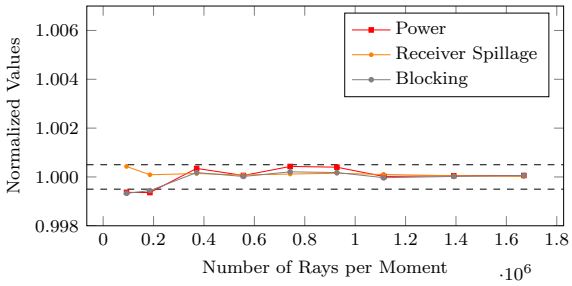
In the Gemasolar test case, different efficiencies are relevant for the accuracy of the total power, see Figure 37a. Here, the blocking efficiency is not accurate when simulating few rays. Since the integrated convolution ray tracer does not improve shading and blocking efficiencies compared to the convolution ray tracer, the blocking efficiency is expected to remain inaccurate for few rays, see Figure 37b. For some number of rays the integrated convolution ray tracer even gives less accurate power results than the convolution ray tracer, as the receiver spillage and blocking efficiency partly cancel out. Therefore, additional computational effort of the integrated convolution ray tracer should be spent only on the shading and blocking efficiencies rather than evaluating more rays completely. As discussed in Section 2.1.2, the convolution ray tracers offer the opportunity to take multiple shading and blocking samples per ray. Using four shading and blocking samples per ray almost completely eliminates the blocking inaccuracies, illustrating that one ray per facet can already be enough to achieve results with more than 99.95% accuracy.

4.1.5 Runtime Comparison

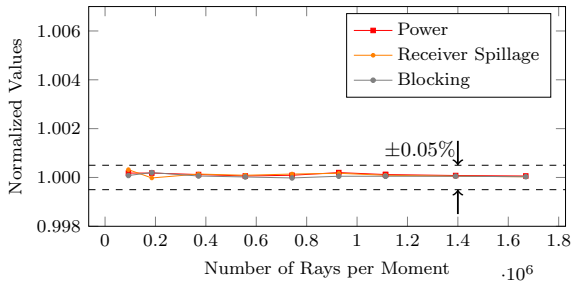
Now that various aspects of our ray tracers have been discussed and optimized, their runtime can be compared. Our goal is to achieve an accuracy of 99.95% in the shortest time possible for the PS10 and Gemasolar test case. Each setting was simulated ten times on each ray tracer and the average runtime and optical power were taken. Figure 38 shows the results of the case study. First, the convolution ray tracer performs



(a) Convolution



(b) Integrated Convolution



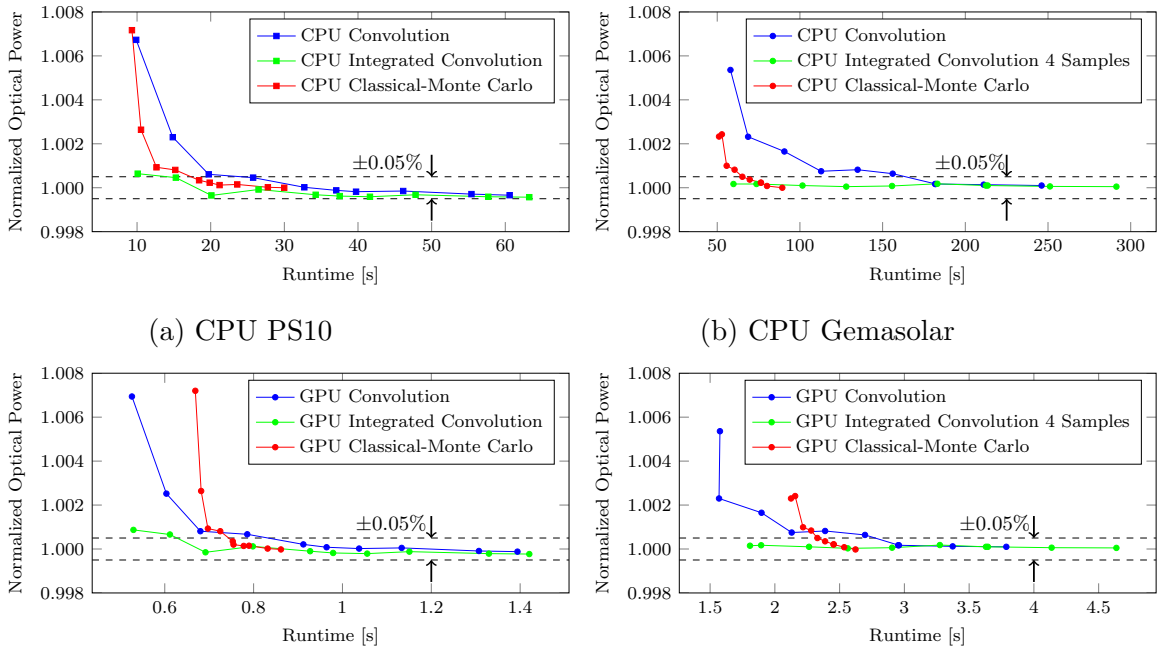
(c) Integrated Convolution using four Shading and Blocking samples

Figure 37: Accuracy of different efficiencies as well as the overall solar power of both analytical ray tracers for an annual simulation of the Gemasolar. For better readability, only the relevant efficiencies are shown which are normalized by the corresponding efficiency of the Monte Carlo ray tracer with over a million rays.

worse than the other two in all test cases. The results are similar to average of the classical Monte Carlo ray tracer while the computational overhead is larger.

However, with the extension to the integrated convolution ray tracer as well as the ability to use multiple shading and blocking samples per ray, far better results can be achieved. As shown, it achieved an accuracy of 99.95% faster than any other ray tracer on all test cases except for the Gemasolar test case on the CPU. Another interesting aspect is that the Monte Carlo ray tracer is slower on the GPU than the analytical ray tracers for the same amount of rays. A possible explanation is the initialization of the random number generator which is not needed in the convolution ray tracer. However, the Monte Carlo method scales much better for more rays on both CPU and GPU than the other ray tracers.

The GPU version of the integrated convolution ray tracer improved the runtime by a factor of more than 40 for the PS10 and 50 for the Gemasolar in the highest setting of rays tested. When achieving 99.95% accuracy the GPU version is more than 25 and 35 times faster for the PS10 and Gemasolar, respectively. Therefore, the PS10 and Gemasolar annual simulation took only around 0.7 and 1.8 seconds while achieving an accuracy of more than 99.95% compared to the limit of the Monte Carlo ray tracer.



(a) CPU PS10

(b) CPU Gemasolar

(c) GPU PS10

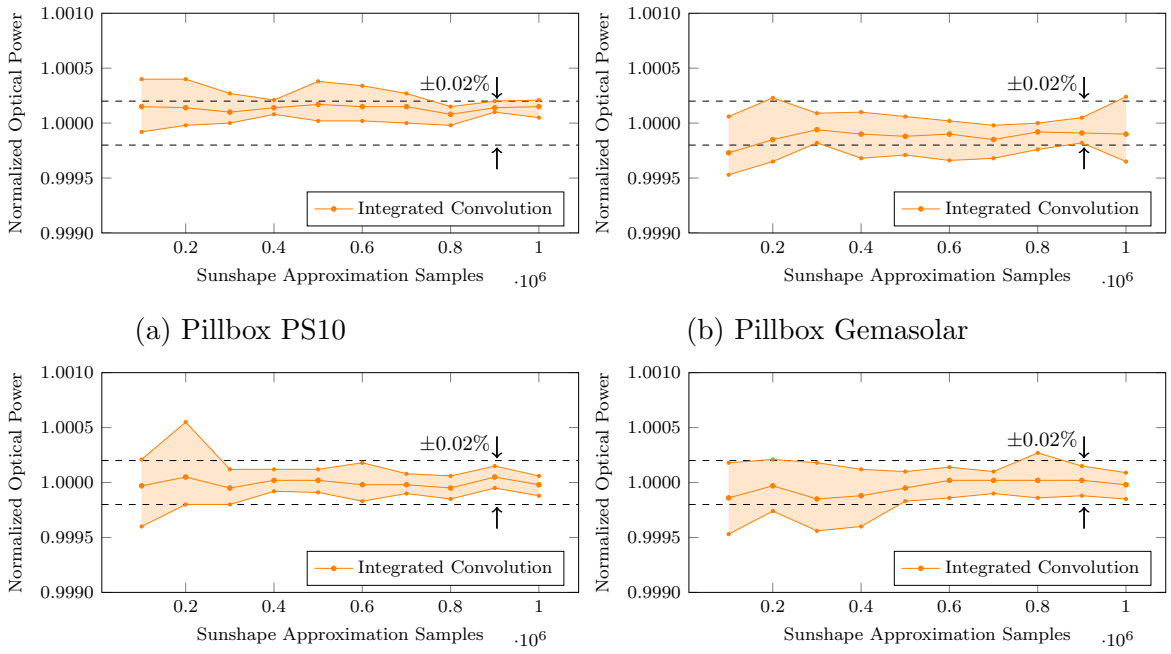
(d) GPU Gemasolar

Figure 38: Runtime comparison of different ray tracer implementations on an annual simulation of the PS10 and Gemasolar. The results got normalized by a Monte Carlo ray tracer simulating over ten million rays.

4.1.6 Approximating Sun Shapes

As discussed in Section 2.1.2, our analytical ray tracers rely on a bivariate Gaussian to represent the reflected ray disturbances. For sun shapes other than Gaussians, the convoluted two dimensional probability density function describing the ray perturbation is numerically calculated and its standard deviation used to accurately approximate it with a bivariate Gaussian. To validate the approximation procedure, the PS10 and Gemasolar annual test cases with a Pillbox and Buie sun shape were simulated. The sun width of the pillbox shape was set to 4.35 mrad and the csr of the Buie shape to 0.03. Since the accuracy of the approximation depends on the number of samples, different sample sizes were used. Figure 39 shows the result of the different test cases which got normalized by the Monte Carlo ray tracer simulating over ten million rays. The shaded orange area indicates the fluctuations of the integrated convolution ray tracer caused by the numerical approximation of the optimal bivariate Gaussian. Since we are interested in the limit result of the integrated convolution, over 1.5 million rays with 16 shading and blocking samples were used in each simulation. In all test cases an accuracy of more than 99.98% was achieved showing that other sun shapes can very well be approximated with a Gaussian without loosing noteworthy accuracy. For comparison, when simply approximating the pillbox and buie before the convolution, the difference between the integrated convolution and Monte Carlo ray tracer rises to

0.7% and 1.3%, respectively.



(a) Pillbox PS10

(b) Pillbox Gemasolar

(c) Buie PS10

(d) Buie Gemasolar

Figure 39: Results of the sun shape approximation from the integrated convolution ray tracer for an annual simulation of the PS10 and Gemasolar. Different amount of samples in the numerical sun shape approximations are used and the results normalized by the Monte Carlo results simulating over ten million rays.

4.2 Optimization

To compare the various heliostat field layout optimization methods, the PS10 and Gemasolar test cases from the last section were used.

4.2.1 Pattern-based

Before evaluating the different pattern optimizations, the runtime improvements to the optimizer are viewed. The PS10 test case was used with about 300.000 rays per moment. As shown in Figure 40, the GPU acceleration [2] of the ray tracers greatly reduced the overall runtime. However, by accelerating the annual integration method as well as heliostat collision detection, the runtime was further reduced by a factor of two.

In the PS10 test case, most pattern alone performed not as good as the original layout. Except the radial staggered pattern, which achieved an improvement of about 0.6%. This is not a surprise, as the original layout of the plant uses a radial staggered

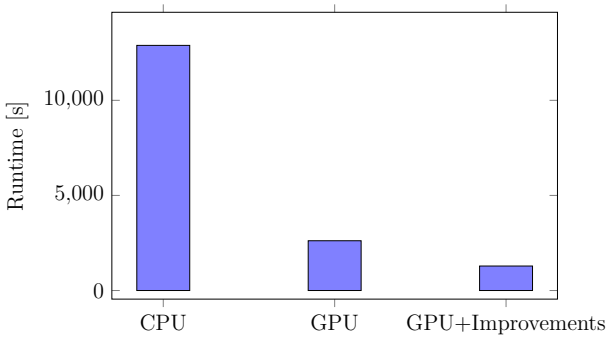


Figure 40: Runtime improvements to the pattern optimization. On the CPU the optimization took about 3 and a half hours, on the GPU 42 minutes, and with further acceleration 21 minutes.

arrangement [44]. Figure 41 shows the achieved annual optical power of all patterns as well as a subsequent local search. The annual optical power of the original plant is illustrated by the dashed line. For patterns with a scaling along an arbitrary axis as introduced in [24], their result with and without scaling is also given. The best result with an improvement of around 1.29%, was obtained by the scaled radial staggered pattern in combination with a local search, see Figure 43a.

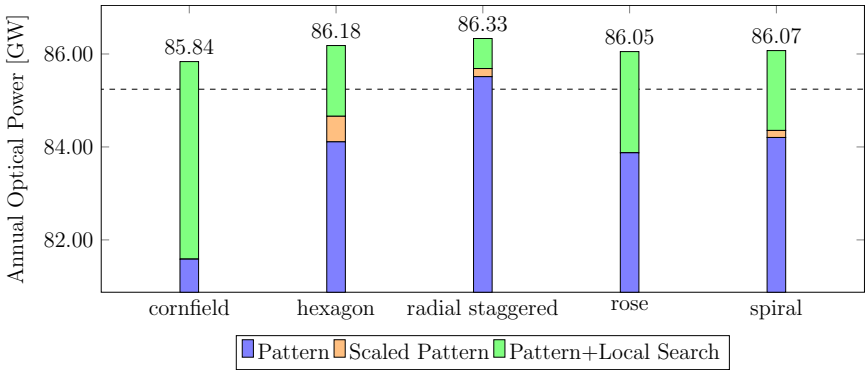


Figure 41: Optimization results of the different patterns for the PS10. The annual optical power of the original layout is shown by the dashed line. The best layout achieved an improvement of 1.29%.

The Gemasolar pattern optimization achieved higher improvements compared to the original layout, see Figure 42. The rose, spiral, and radial staggered pattern alone surpassed the original annual optical power shown by the dashed line. Again, the scaled patterns achieved slight improvements over the unscaled patterns. Like in the PS10 test case, the most efficient layout was obtained by a combination of the scaled radial staggered pattern and a local search. It results in about 5.5% more optical power than the original layout, see Figure 43b.

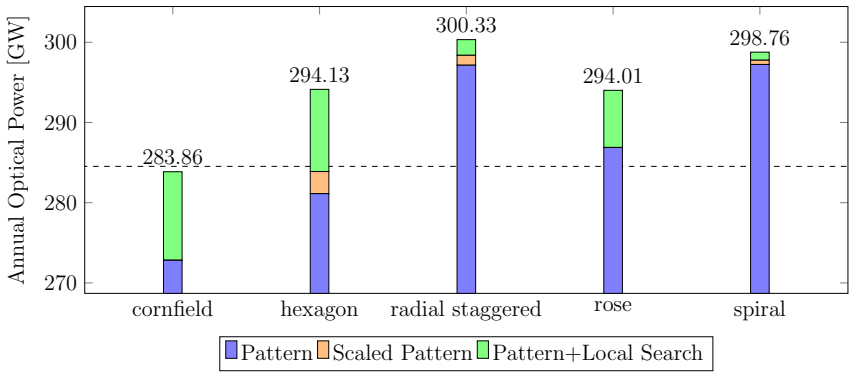
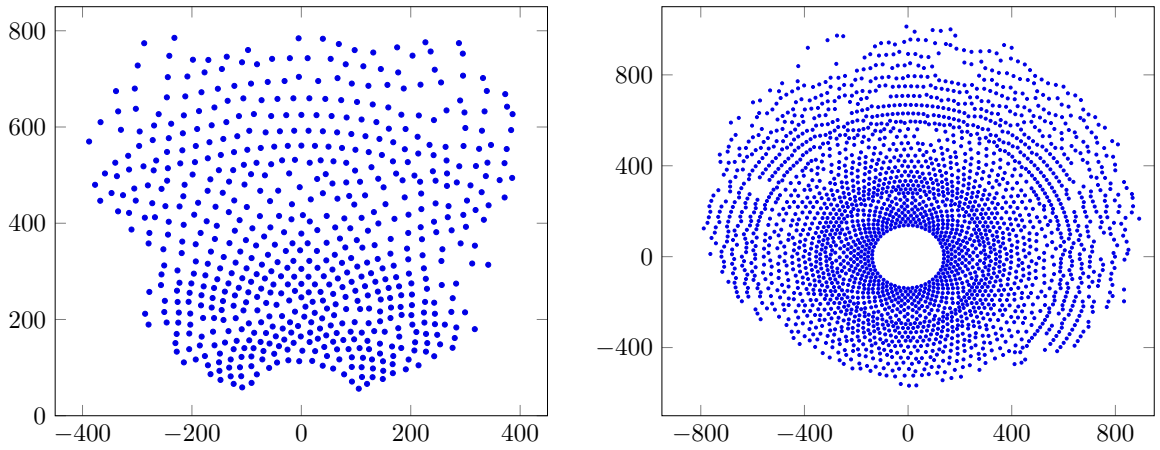


Figure 42: Optimization results of the different patterns for the PS10. The annual optical power of the original layout is shown by the dashed line. The best layout achieved an improvement of 5.5%.



(a) PS10 (b) Gemasolar

Figure 43: Best performing PS10 and Gemasolar layouts generated by the radial staggered pattern optimization with local search.

4.2.2 Graph based field growth

In the following, different aspects of the new graph based field growth algorithm and its variants are discussed. As stated in Section 3.2, the methods rely on very accurate and precise heliostat efficiencies. Following the findings from the simulation test cases, the best ray tracer to achieve both is the integrated convolution ray tracer. Therefore, the independent ray tracing extension of the integrated convolution ray tracer is used for all test cases.

A parameter study based on the PS10 has been carried out to find the optimal setting for the relaxed score calculation of our field growth method. The highest overall field efficiency was obtained by applying the relaxed score to the first half of the heliostats $n_{\text{heli}} = 0.5$, while using a maximal distance of $d_{\max} = 100$ m, see Figure 44. More details on the relaxed score can be found in Section 3.2.

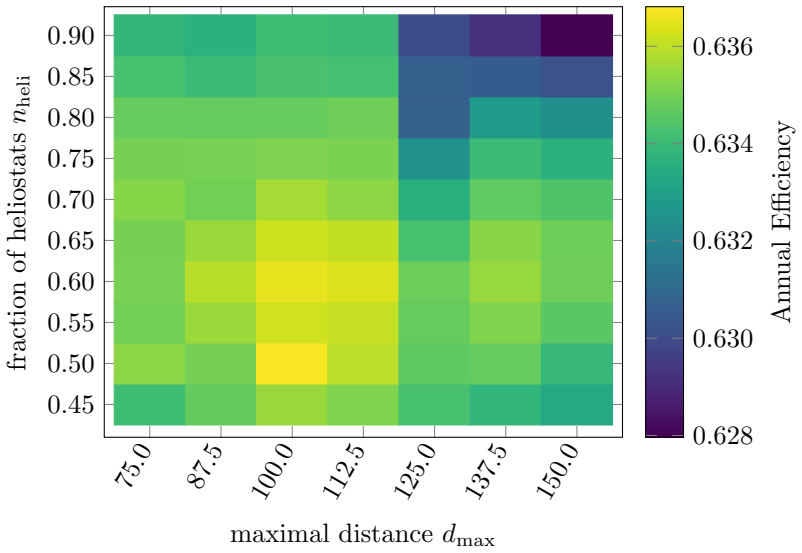


Figure 44: Parameter study for the relaxed score computation of the graph based field growth method.

The new field growth method also offers the possibility to only include heliostats with certain S&B weights, therefore reducing the simulated heliostats to the most important ones. Figure 45 compares the runtime of the algorithm on the PS10 for different minimum S&B weights. Additionally, the average efficiency difference of all heliostats to their actual value is given. As shown, the difference linearly increases with the minimum S&B weight. However, the runtime decreases only slightly after a S&B weight of 0.00015. Therefore, this weight is chosen in the following.

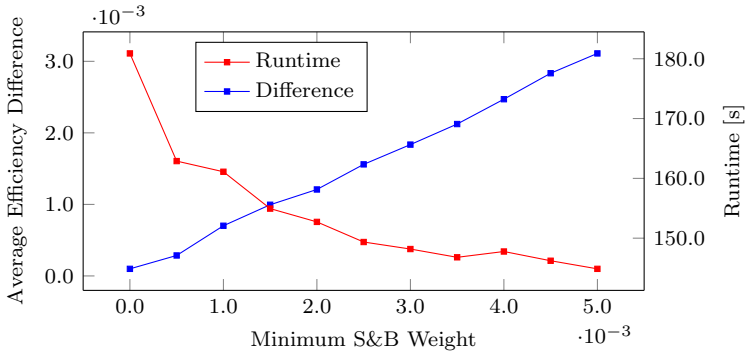


Figure 45: The effect of the minimum S&B weight on the runtime and average efficiency difference for the PS10. A field density of one was used.

Field density Another crucial parameter of our field growth method is the field density of the underlying grid. A field density of one means that the heliostat positions are packed as densely as possible without overlapping. For a field density of three

there are three times as many heliostats in both horizontal and vertical direction, thus nine times as many heliostats. For the PS10, this results in a total of 22344 heliostats. Both the annual efficiencies and runtime for different field densities are shown in Figure 46. Again, the dashed line is the result of the original layout. The runtime increases quadratically with the field density, due to the quadratically growing number of heliostats. A dense grid also leads to more heliostats being simulated at each step. However, even with a field density of 3.5, the algorithm only takes about 300 seconds more than the improved GPU pattern optimizer. It achieved an annual efficiency improvement of about 1.75% which is 0.4% higher than the best pattern with local search, see Figure 48a.

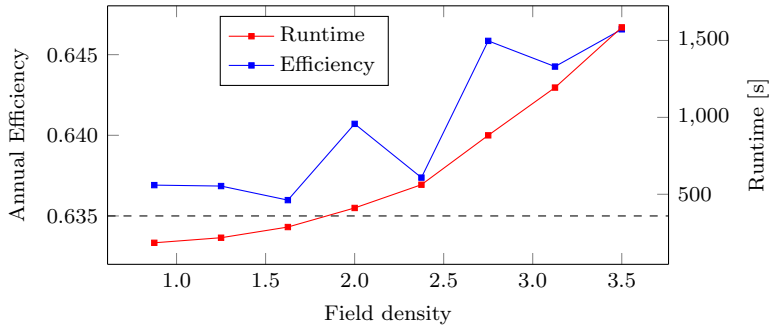


Figure 46: Runtime and annual efficiency of the field growth method for the PS10 at different field densities. The efficiency of the original layout is shown by the dashed line. The best layout achieved an improvement of 1.75%.

Similar results were obtained for the Gemasolar, see Figure 47. However, since the field is much larger than the one of the PS10, more heliostats are considered which influences the runtime. The layout optimization achieved an improvement of about 6.3%, 0.8% better than the best pattern optimization with local search. Figure 48b shows the generated layout.

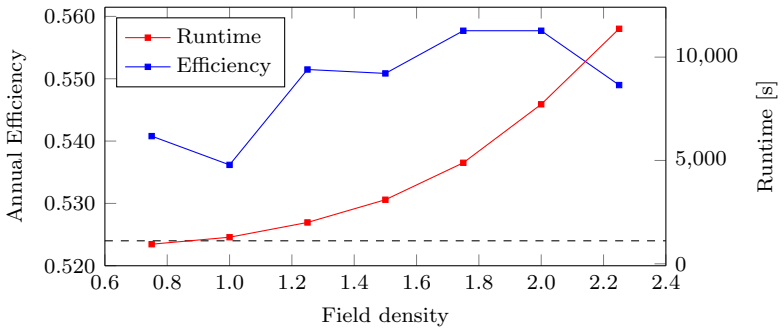
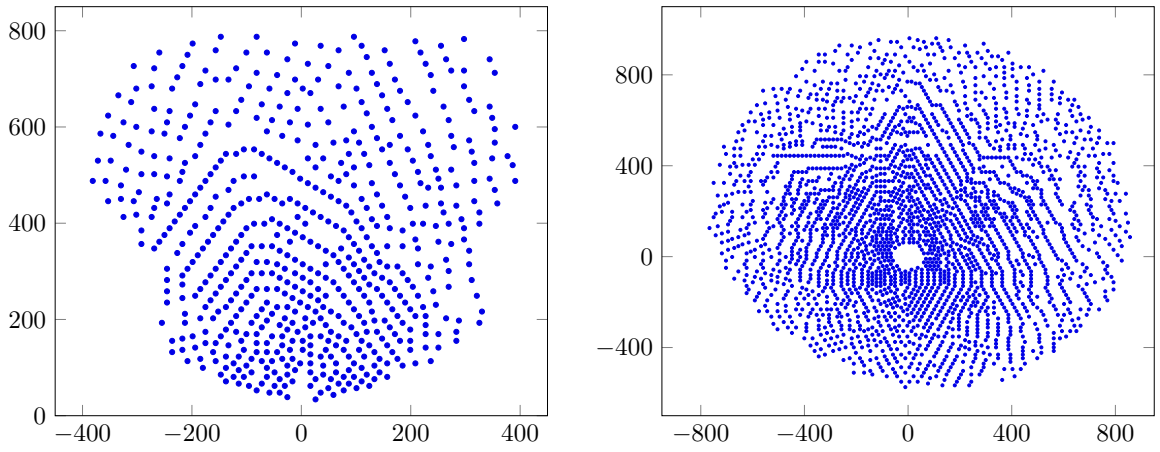


Figure 47: Runtime and annual efficiency of the field growth method for the Gemasolar at different field densities. The efficiency of the original layout is shown by the dashed line. The best layout achieved an improvement of 6.3%.



(a) PS10 (b) Gemasolar

Figure 48: Best performing PS10 and Gemasolar layouts generated by the graph based field growth method.

Field growth suboptimization Besides increasing the density of the underlying grid, the number of possible heliostat positions can also be increased by suboptimization. Figure 49 shows the achieved annual efficiency on the PS10 for different numbers of sub heliostats. The field density was chosen to be one and the dashed line indicates the achieved efficiency without suboptimization. As shown, the suboptimization first leads to a less efficient layout. However, after 20 sub heliostats, the field efficiency exceeds the one without suboptimization. Thus, the field growth suboptimization effectively increases the number of heliostat positions without additional memory consumption.

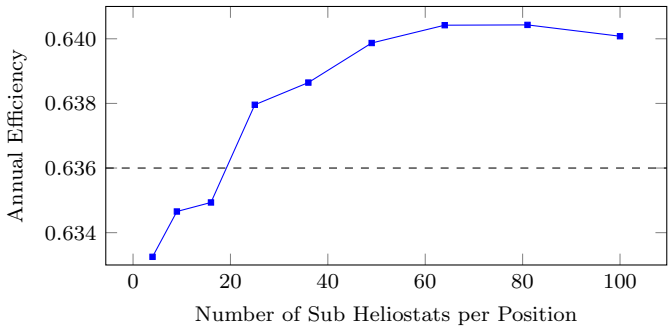


Figure 49: Annual efficiencies of the field growth suboptimization on the PS10. A field density of one was used and the dashed line indicates the efficiency from the field growth method.

Representative field growth Another approach to allow for more heliostat positions, is the representative field growth algorithm. To evaluate the method, the same PS10 test case was used with varying number of heliostats per representative heliostat. Each

heliostat of the densest non overlapping grid was considered a representative heliostat. The local optima property, as discussed in Section 3.2, was regained by replacing the worst heliostat up to 100 times. Figure 50 shows the achieved annual efficiency with the dashed line indicating the efficiency of the field growth method at a density of one. In the beginning, the resulting field efficiency increases significantly but then declines. A possible explanation is the score approximation in the representative field growth method. With increasing number of non-representative heliostats, more heliostats with inaccurate scores exist. Therefore, the algorithm is more likely to choose a non optimal heliostat resulting in less efficient layouts.

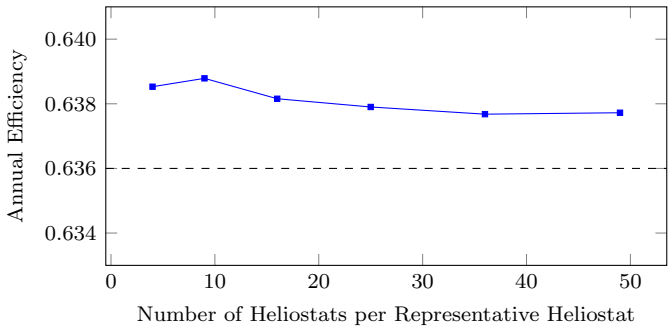


Figure 50: Annual efficiencies obtained by the representative field growth method on the PS10. The field density was set to one for the representative heliostats. The dashed line indicates the efficiency of the field growth method.

4.3 Discussion of results

The results of the simulation test cases show that the extension of the integrated convolution ray tracer improved the accuracies for a variety of test cases. Futhermore, the new equation to convolute sun, slope, and tracking errors now perfectly matches the results obtained by a separate perturbation of the sun and normal vector. In contrast to the equations used by *HFLCAL* and *UNIZAR* leading to a difference of up to 4.5% and 0.4%, *SunFlower* differed only 0.01% to a separate perturbation. Additionally, sun shapes like Buie and Pillbox can accurately be represented by a Gaussian making it possible to use them in analytical ray tracers. Therefore, the improved integrated convolution ray tracer is the recommended ray tracing method.

The GPU ray tracer as well as further acceleration decreased the runtime of the pattern based heliostat field layout optimization from 3 and a half hours to about 21 minutes. The most efficient layouts for the PS10 and Gemasolar improved the optical power by about 1.29% and 5.5%, respectively. Our new graph based field growth method achieved an improvement of 1.75% and 6.3%. The runtime for the PS10 optimization was similar to the pattern based with local search. However, the field growth method took 25 minutes longer on the Gemasolar test case. It should be noted

that the field growth algorithm is running on the CPU and not on the GPU like the pattern optimization and local search. In comparison to the CPU pattern optimization and local search, the field growth algorithm is at least 5 times faster. This is only made possible by the independent ray tracing extension. The field growth suboptimization and representative field growth methods allowed to include more positions, leading to better layouts. But for dense heliostat grids, both methods result in similar or even less efficient layouts than the one of our field growth method. This indicates that the approximative aspects of both methods limit their ability to find the best layout. Therefore, the exact evaluation of our graph based field growth algorithm is a crucial aspect and separates it from existing field growth methods. Due to the layout improvements, our new graph based field growth method should be preferred over pattern optimization.

5 Conclusion

In this work, the modeling and optimization of CRS were extended and improved. The accuracy of the integrated convolution ray tracer was enhanced by concentrating computational effort on the shading and blocking calculation. Additionally, the convolution of the bidirectional ray tracer was aligned to perfectly match the result of an individual accounting of optical errors. Besides more accurate results, the ray tracer also got extended to incorporate more sun shapes. For analytical ray tracers, a new approximation method of the sun shapes has been developed.

Within the optimization, the GPU ray tracers were adapted to work with the existing optimizers. The annual integration method and heliostat collision detection got accelerated. A new pattern was added to the pattern optimization. Most importantly, the new graph based field growth optimization method as well as two variants have been developed. For this, a novel way of ray tracing was implemented for the existing ray tracers. The graph based field growth method results in the most efficient layouts for both the PS10 and Gemasolar test case. An essential aspect is the exact computation of each heliostat's efficiency contribution, achieved by the algorithm itself and the underlying independent integrated convolution ray tracer.

5.1 Outlook

Acceleration One criticism of field growth methods in general is their runtime. Although our new graph based field growth method required similar runtime as the pattern optimization for the PS10, larger fields like the Gemasolar drastically increase the overall runtime. There are many opportunities to further accelerate the algorithm. The most promising is an efficient GPU implementation. More specifically, the independent GPU ray tracer should be extended to compute the efficiency reduction maps. Even though the GPU has much smaller memory capacities than the CPU, it is possible to get around this problem. One way is to not simulate all relevant heliostats

at once but only a certain fraction. Later the results are combined to get the desired computation. However, there are more issues than just the memory capacity for an efficient GPU implementation. It is also necessary to replace some data structures used to accelerate the CPU version. Another benefit of a fully GPU based field growth method is that it probably scales much better for larger fields.

The initialization of the S&B graph is another area for acceleration. Currently, all heliostats need to be simulated with the independent ray tracer. This offers an exact determination of which heliostats shade or block another one but is a very time consuming part of the optimization. A gradual generation of the full S&B graph might accelerate the initialization. However, then the relevant S&B heliostats need to be approximated which can increase the runtime or lead to less efficient layouts.

Heliostat selection A crucial aspect of pattern optimization is the selection of a subset of the generated heliostats. In *SunFlower* all heliostats are simulated and the ones with the highest efficiencies are chosen. However, this only partially includes S&B interactions into account while including ones that are not existing in the subset. A lightweight version of the field growth method might lead to more efficient subsets. One approach could be to independently simulate all heliostats and approximate combined S&B effects. So if two heliostats shade or block another heliostat then the combined effect is approximated by some function. When updating the graph, no simulation needs to be done.

Complete S&B Graph As discussed in Section 3.2, the independent ray tracer can be extended such that for each heliostat a set of efficiencies is computed considering any possible combination of heliostats. The resulting complete S&B graph contains all the necessary information to compute the efficiency of any possible layout from the grid. One problem is of course the immense memory consumption of such a graph. But with modern memory capacities and maybe some approximations, this can become manageable. The benefits would be tremendous, the efficiencies of any layout could be computed with a few additions thus in less than milliseconds. Far more complex optimization methods would be applicable. Our graph based field growth method, for example, could use backtracking to avoid local optima. However, for very dense and large fields a complete S&B graph will probably remain unfeasible. Approximations to the combined S&B effects could be used in that case.

References

[1] N. Ahlbrink, B. Belhomme, R. Flesch, D. Maldonado Quinto, A. Rong, and P. Schwarzbözl. Stral: Fast ray tracing software with tool coupling capabilities for high-precision simulations of solar thermal power plants. In *Proceedings of the SolarPACES 2012 conference*, 2012.

[2] L. Aldenhoff. Raytracer for central receiver systems using gpu. Master's thesis, RWTH Aachen University, 2021.

[3] C. Asselineau, J. Zapata, and J. Pye. Geometrical shape optimization of a cavity receiver using coupled radiative and hydrodynamic modeling. *Energy Procedia*, 69:279–288, 2015.

[4] C. Asselineau, J. Zapata, and J. Pye. Integration of monte-carlo ray tracing with a stochastic optimisation method: application to the design of solar receiver geometry. *Optics Express*, 23(11):A437–A443, 2015.

[5] P. Atela, C. Golé, and S. Hotton. A dynamical system for plant pattern formation: a rigorous analysis. *Journal of Nonlinear Science*, 12(6):641–676, 2002.

[6] M. Atif and F. Al-Sulaiman. Optimization of heliostat field layout in solar central receiver systems on annual basis using differential evolution algorithm. *Energy Conversion and Management*, 95:1–9, 2015.

[7] M. Balz, V. Göcke, T. Keck, F. von Reeken, G. Weinrebe, and M. Wöhrbach. Stellio–development, construction and testing of a smart heliostat. In *AIP conference proceedings*, volume 1734, page 020002. AIP Publishing LLC, 2016.

[8] F. Biggs and C. N. Vittitoe. Helios model for the optical behavior of reflecting solar concentrators. Technical report, Sandia Labs., Albuquerque, NM (USA), 1979.

[9] M. Blanco. Tonatiuh: An object oriented, distributed computing, monte-carlo ray tracer for the design and simulation of solar concentrating systems. Technical report, The University of Texas at Brownsville, 2006.

[10] R. Buck. Heliostat field layout improvement by nonrestricted refinement. *Journal of solar energy engineering*, 136(2), 2014.

[11] J. Burgaleta, S. Arias, and D. Ramirez. Gemasolar, the first tower thermosolar commercial plant with molten salt storage. *SolarPACES, Granada, Spain*, pages 20–23, 2011.

[12] E. Carrizosa, C. Domínguez-Bravo, E. Fernández-Cara, and M. Quero. A heuristic method for simultaneous tower and pattern-free field optimization on solar power systems. *Computers & Operations Research*, 57:109–122, 2015.

[13] F. Collado. One-point fitting of the flux density produced by a heliostat. *Journal of the American Statistical Association*, page 673–684, 2010. doi: 10.1080/01621459.1949.10483310.

[14] F. Collado and J. Guallar. Campo: Generation of regular heliostat fields. *Renewable energy*, 46:49–59, 2012.

[15] S. Coumbassa. Optimal storage strategy for hybrid concentrated solar thermal - photovoltaic plants. Master thesis, RWTH Aachen University, 2019.

[16] P. Davies. The tailor-made universe. *Sciences*, 18:6–10, 1978.

[17] L. Deng, Y. Wu, S. Guo, L. Zhang, and H. Sun. Rose pattern for heliostat field optimization with a dynamic speciation-based mutation differential evolution. *International Journal of Energy Research*, 44(3):1951–1970, 2020.

[18] A. Di Donato and R. Hageman. Computation of the integral of the bivariate normal distribution over arbitrary polygons. Technical report, Naval Surface Weapons Center, 1980.

[19] A. Di Donato, M. Jarnagin, and R. Hageman. Computation of the bivariate normal distribution over convex polygons. Technical report, Naval Surface Weapons Center, 1978.

[20] X. Duan, C. He, X. Lin, Y. Zhao, and J. Feng. Quasi-Monte Carlo ray tracing algorithm for radiative flux distribution simulation. *Solar Energy*, 211:167–182, 2020. doi: 10.1016/j.solener.2020.09.061.

[21] A. Farahmand, S. Payan, and S. Sarvari. Geometric optimization of radiative enclosures using pso algorithm. *International journal of thermal sciences*, 60: 61–69, 2012.

[22] O. Farges, J. Bézian, and M. El Hafi. Global optimization of solar power tower systems using a monte carlo algorithm: Application to a redesign of the ps10 solar thermal power plant. *Renewable Energy*, 119:345–353, 2018.

[23] T. Farr, P. Rosen, E. Caro, R Crippen, R. Duren, S. Hensley, M. Kobrick, M. Paller, E. Rodriguez, L. Roth, et al. The shuttle radar topography mission. *Reviews of geophysics*, 45(2), 2007.

[24] L. Fischer. Multi-step layout-optimization of heliostat fields in central receiver systems. Master's thesis, RWTH Aachen University, 2021.

[25] L. Franke. Modelling and optimization of large scale solar tower power plants. Master's thesis, RWTH Aachen University, 2018.

[26] Bindesministerium für Wirtschaft und Klimaschutz. Erneuerbare energien. `https://www.bmwk.de/Redaktion/DE/Dossier/erneuerbare-energien.html`. Accessed: 2022-12-01.

[27] D. Gebreiter, G. Weinrebe, M. Wöhrbach, F. Arbes, F. Gross, and W. Landman. sbpray–a fast and versatile tool for the simulation of large scale csp plants. In *AIP Conference Proceedings*, volume 2126, page 170004. AIP Publishing LLC, 2019. doi: 10.1063/1.5117674.

[28] G. Heiming. *Development of a techno-economic model for solar tower power plants*. Master thesis, MathCCES-RWTH Aachen, 2017.

[29] F. Hövelmann. Accelerated raytracer for solar tower power plants. Bachelor thesis, RWTH Aachen University, 2019.

[30] M. Izygon, P. Armstrong, C. Nilsson, and N. Vu. Tiesol–a gpu-based suite of software for central receiver solar power plants. *Proceedings of SolarPACES*, 2011.

[31] F. Jensen and T. Nielsen. *Bayesian networks and decision graphs*, volume 2. Springer, 2007.

[32] H. Kambezidis, B. Psiloglou, D. Karagiannis, U. Dumka, and D. Kaskaoutis. Meteorological radiation model (mrm v6. 1): Improvements in diffuse radiation estimates and a new approach for implementation of cloud products. *Renewable and Sustainable Energy Reviews*, 74:616–637, 2017.

[33] F. Kepp. Robust optimization of aiming strategies of heliostats in solar tower power plants, 2018.

[34] B. L. Kistler. A user's manual for delsol3: a computer code for calculating the optical performance and optimal system design for solar thermal central receiver plants. *Sandia National Laboratories, Sandia Report No. SAND86-8018*, 1986. doi: 10.2172/7228886.

[35] M. Lazardjani, V. Kronhardt, G. Dikta, and J. Göttsche. Simultaneous optimization of micro-heliostat geometry and field layout using a genetic algorithm. In *AIP Conference Proceedings*, volume 1734, page 020028. AIP Publishing LLC, 2016.

[36] P. Leary and J. Hankins. User's guide for MIRVAL: a computer code for comparing designs of heliostat-receiver optics for central receiver solar power plants. Technical report, Sandia Laboratories, 1979.

[37] F. Lipps and L. Vant-Hull. A cellwise method for the optimization of large central receiver systems. *Solar Energy*, 20(6):505–516, 1978.

[38] S. Lutchman, A. Groenwold, P. Gauché, and S. Bode. On using a gradient-based method for heliostat field layout optimization. *Energy Procedia*, 49:1429–1438, 2014.

[39] A. Mecit and F. Miller. Optical analysis of a window for solar receivers using the monte carlo ray trace method. *ASME 2013 7th International Conference on Energy Sustainability*, 2013. doi: 10.1115/ES2013-18186.

[40] K. Milidonis, M. Blanco, V. Grigoriev, C. Panagiotou, A. Bonanos, M. Constantinou, J. Pye, and C. Asselineau. Review of application of ai techniques to solar tower systems. *Solar Energy*, 224:500–515, 2021.

[41] J. Nelder and R. Mead. A simplex method for function minimization. *The computer journal*, 7(4):308–313, 1965.

[42] J. Nievergelt and P. Widmayer. Spatial data structures: Concepts and design choices. In *Handbook of Computational Geometry*, pages 725–764. Elsevier, 2000.

[43] C. Noone, M. Torrilhon, and A. Mitsos. Heliostat field optimization: A new computationally efficient model and biomimetic layout. *Solar Energy*, 86(2):792–803, 2012.

[44] R. Osuna, R. Olavarría, R. Morillo, M. Sánchez, F. Cantero, V. Fernández-Quero, P. Robles, T. López, S. Esteban, F. Céron, et al. Ps10, construction of a 11mw solar thermal tower plant in seville, spain. In *Proc. 13th IEA SolarPACES Symp.*, pages A4–S3, 2006.

[45] Eric P. and Yves D. Rendering participating media with bidirectional path tracing. In *Eurographics*, pages 91–100. Springer Vienna, 1996. doi: 10.1007/978-3-7091-7484-5_10.

[46] Q. Pham, C. Gregory, M. Slack, B. Gross, D. Reznik, and P. Arbogast. Heliostat array layouts for multi-tower central receiver solar power plants, 2014. US Patent 8,656,907.

[47] R. Pitz-Paal, N. Botero, and A. Steinfeld. Heliostat field layout optimization for high-temperature solar thermochemical processing. *Solar energy*, 85(2):334–343, 2011.

[48] R. Pitz-Paal, N. B. Botero, and A. Steinfeld. Heliostat field layout optimization for high-temperature solar thermochemical processing. *Solar Energy*, 85(2):334–343, 2011.

[49] A. Rabl. *Active solar collectors and their applications*. Oxford University Press, 1985.

[50] P. Richter. *Simulation and optimization of solar thermal power plants*. Dissertation, RWTH Aachen University, Aachen, 2017. URL `http://publications.rwth-aachen.de/record/690762`. Veröffentlicht auf dem Publikationsserver der RWTH Aachen University; Dissertation, RWTH Aachen University, 2017.

[51] P. Richter and F. Hövelmann. Computationally fast analytical ray-tracer for central receiver system. In *AIP Conference Proceedings*, volume 2445. AIP Publishing LLC, 2021. doi: 10.1063/5.0085714.

[52] P. Richter, D. Laukamp, L. Gerdes, M. Frank, and E. Abrahám. Heliostat field layout optimization with evolutionary algorithms. In *GCAI*, pages 240–252, 2016.

[53] P. Richter, G. Heiming, N. Lukas, and M. Frank. Sunflower: A new solar tower simulation method for use in field layout optimization. *AIP Conference Proceedings*, 2033(1):210015, 2018. doi: 10.1063/1.5067217.

[54] P. Richter, J. Tinnes, and L. Aldenhoff. Accurate interpolation methods for the annual simulation of solar central receiver systems using celestial coordinate system. *Solar Energy*, 213:328–338, 2021.

[55] M. Sánchez and M. Romero. Methodology for generation of heliostat field layout in central receiver systems based on yearly normalized energy surfaces. *Solar energy*, 80(7):861–874, 2006.

[56] A. Sánchez-González and D. Santana. Solar flux distribution on central receivers: A projection method from analytic function. *Renewable Energy*, 74:576–587, 2015.

[57] S. Sarvari. Optimal geometry design of radiative enclosures using the genetic algorithm. *Numerical Heat Transfer, Part A: Applications*, 52(2):127–143, 2007.

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

[59] P. Schöttl, S. Rohani, E. Leonardi, L. Pisani, I. Les, A. Mutuberria, and P. Nitz. Solar field heliostat selection based on polygon optimization and boundaries. In *AIP Conference Proceedings*, volume 2126, page 030053. AIP Publishing LLC, 2019.

[60] P. Schwarzbözl, R. Pitz-Paal, and M. Schmitz. Visual HFLCAL - a software tool for layout and optimisation of heliostat fields. In *SolarPACES Conference*, 2009.

[61] P. Talebizadeh, M. Mehrabian, and H. Rahimzadeh. Optimization of heliostat layout in central receiver solar power plants. *Journal of Energy Engineering*, 140 (4):04014005, 2014.

[62] J. Wang, L. Duan, and Y. Yang. An improvement crossover operation method in genetic algorithm and spatial optimization of heliostat field. *Energy*, 155:15–28, 2018.

[63] Y. Wang, D. Potter, C. Asselineau, C. Corsi, M. Wagner, C. Caliot, B. Piaud, M. Blanco, J. Kim, and J. Pye. Verification of optical modelling of sunshape and surface slope error for concentrating solar power systems. *Solar Energy*, 195: 461–474, 2020. doi: 10.1016/j.solener.2019.11.035.

[64] T. Wendelin. Soltrace: a new optical modeling tool for concentrating solar optics. In *ASME 2003 International Solar Energy Conference*, pages 253–260. American Society of Mechanical Engineers, 2003.