Theory of Hybrid Systems Informatik 2 | RWTH AACHEN UNIVERSITY

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

Accelerated Raytracer for Solar Tower Power Plants

Beschleunigte Raytracer für Solarturmkraftwerke

Bachelorarbeit
Informatik

September 2019

Vorgelegt von Presented by	Florian Hövelmann Stephanstraße 43 52064 Aachen 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 Modelierung und Verifikation (MOVES) RWTH Aachen University
Externer Betreuer External supervisor	Dr. rer. nat. Pascal Richter Steinbuch Centre for Computing Karlsruhe Institute of Technology

Eigenständigkeitserklärung

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

Aachen, im September 2019

Florian Hövelmann

Contents

List of Figures IV

List of Tables V

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

2 Optical model 4
- 2.1 Environment . . . 4
- 2.2 Heliostats . . . 4
- 2.3 Receiver . . . 5
- 2.4 Optical losses . . . 7
- 2.5 Generation of representative solar rays . . . 11
- 2.6 Blocking and shading computations . . . 14
- 2.7 Discretization of the receiver . . . 15
- 2.8 Ray tracing pipeline . . . 15

3 Monte Carlo ray tracing methods 17
- 3.1 Classical-Monte Carlo method . . . 17
- 3.2 Multi-Monte Carlo method . . . 19
- 3.3 Quasi-Monte Carlo method . . . 20
- 3.4 Tower blocking . . . 21

4 Convolution methods 23
- 4.1 Analytic image method *HFLCAL* . . . 23
- 4.2 Gaussian convolution method . . . 29
  - 4.2.1 Computation of a region representing a receiver piece . . . 29
  - 4.2.2 Integration of the bivariate Gaussian distribution over a polygon 36
  - 4.2.3 Tower blocking . . . 47
- 4.3 Integrated convolution method . . . 50
  - 4.3.1 Approximating the integrated Gaussian distribution . . . 55
  - 4.3.2 Use of the approximation . . . 59

5 Case study 61
- 5.1 Validation . . . 61
  - 5.1.1 Fluxmap comparison . . . 64
  - 5.1.2 Validation of the new integrated convolution method . . . 64
- 5.2 Optimal setting for quasi- and multi-Monte Carlo . . . 66
- 5.3 Accuracy vs runtime . . . 69
- 5.4 Characteristics of the convolution methods . . . 72
- 5.5 Acceleration of the ray tracers . . . 74
  - 5.5.1 Preselection . . . 74

5.5.2 CPU parallelization 74

6 Conclusion and Outlook **76**
6.1 Conclusion 76
6.2 Outlook 76
6.2.1 Distant dependent heliostat discretization 76
6.2.2 Combined receiver cells 77
6.2.3 Reversed convolution methods 77
6.2.4 GPU Parallelization 78
6.2.5 Other distributions 79

References **80**

List of Figures

1 Solar tower power plant Planta Solar 10 (PS10) in Spain [29]. The figure is derived from Richter et al. [34, p. 2] . . . 1
2 Different tower types with different receiver types. The Figure is derived from Richter [33, p. 11] . . . 6
3 Regular polygon with nine edges, an edge length of e and a circumscribed circle of diameter d which is drawn in gray. . . . 7
4 Horizontal shape of a cavity receiver with four panels. It is based on a regular polygon with nine edges as illustrated in gray. . . . 7
5 Illustration of our cylindrical receiver types . . . 8
6 Error cones representing the optical errors [33, p. 17]. . . . 10
7 Two dimensional perturbation for horizontal deviations. The lengths of the deviation vectors illustrate the amount of perturbation. . . . 12
8 Illustration of the required rotation to obtain the convoluted deviation directions. The shortened deviation vectors are actually of unit length. 13
9 Discretization of a heliostat facet into 5 by 5 cells. From each cell a representative ray originates, which intensity is weighted by its area [18, p. 16]. . . . 14
10 Simplified pipeline for the evaluation of the solar radiation at the receiver. 16
11 Problem of evaluating a representative ray traveling in the direction of $\vec{r}$ with a deviation direction of $\vec{r}^{\,\mathrm{hor}}$. The Gaussian function defining the deviation is shown in black, the heliostat cell in blue, the receiver cell in green and the thin red lines illustrate the error cone. . . . 17
12 Basic principle of the classical-Monte Carlo method. The generated perturbed ray is shown in red. . . . 18
13 Transformation from angle of horizontal deviation to length. . . . 18
14 Basic principle of the multi-Monte Carlo method. Here, multiple samples, shown in red, are taken from the error cone. . . . 19
15 Basic principle of the quasi-Monte Carlo method. Here, the generated samples are more uniform and thus better represent the error cone with few samples. . . . 20
16 Blockage of the cavity receiver. The horizontal and vertical blocking edges are drawn in thick red and the rest of the receiver window in thick black. Here, l is the lifting height of the receiver to its lower window edge and b_{vis} a definitely visible point. . . . 21
17 Transformation of the one dimensional Gaussian distribution of Section 2.4 from angle to spanned region. . . . 25
18 Bivariate Gaussian distribution of Equation (4.2) on the ray image plane with the distance δ and standard deviations $\sigma_{\mathrm{beam}}^{\mathrm{ver}}$, $\sigma_{\mathrm{beam}}^{\mathrm{hor}}$ being exemplary set such that $\sigma_{\mathrm{span}}^{\mathrm{ver}} = \sigma_{\mathrm{span}}^{\mathrm{hor}} \approx 1$ m [33]. D marks a possible region to integrate over. . . . 25
19 Simplified example where the receiver pieces marked in green lies in the ray image plane. . . . 26

20 Illustration of the required calculations to compute the local coordinate m'_y of the midpoint m of a receiver piece marked in green. Analogously the local coordinate m'_x can be computed. 27
21 Basic principle of the Gaussian convolution method. The shaded red region illustrates an exact evaluation of the error cone for the current receiver piece. 29
22 Two dimensional version of the angular region D_{ang} defined by the angles α_1 and α_2. 30
23 Required projection to obtain the angles α_1 and β_1 of a 31
24 Example to illustrate the need of signed angles as the angles α_1 and α_2 are equal even though they refer to different corners. 32
25 Exemplary representative region D_{span} illustrated in light green on the ray image plane of a receiver piece shown in green. 32
26 Example of the perspective projection problem with the center of projection lying in the origin. The vector $\vec{n}$ is the normal of the image plane onto which the point q_1 and q_2 are to be projected. 33
27 Problem of calculating local coordinates of the receiver corners on the ray image plane. The coordinate system is defined by m_{ray}, $\vec{r}^{\text{hor}}$ and $\vec{r}^{\text{ver}}$. 35
28 Angular regions of a polygon with four vertices. The figure is derived from [1]. 37
29 Example of a general angular region before the rotation derived from [1]. 39
30 Rotated angular region which is defined by the length R and the angles θ_1 and θ_2 derived from [1]. 39
31 Illustration for the calculation of the angles θ_1 and θ_2. 40
32 Calculation of angular regions including the cases where the constraints of Equation (4.46) are not given. The figure is derived from [1]. 45
33 Simplified example of the cutting process for the cavity receiver to obtain its visible pieces. The currently checked receiver piece is shown in dark blue and the checked corners are marked with a dot. 48
34 Illustration of all possible cases when cutting a receiver piece horizontally. Not visible corners are marked with a cycle and visible ones with a dot. 49
35 Basic principle of our integrated convolution method. Here, the whole heliostat cell of the representative ray is taken into account which requires a different probability density function, shown in black, for evaluation. 50
36 Simplified two dimensional examples to illustrate the process of evaluating more than one ray per heliostat cell. 51
37 Probability density function of the integrated Gaussian distribution from a heliostat cell of length $l_{\text{cell}} = 1$ m at different distances with standard derivations σ_{span} as in Equation (4.3) from $\sigma_{\text{beam}} = 3$ mrad. 53

38 Integrated two dimensional Gaussian distribution for a cell of width $w_{\text{cell}} = 2$ m and length $l_{\text{cell}} = 2$ m at a distance $\delta = 50$ m and standard deviations $\sigma^{\text{ver}}_{\text{beam}} = \sigma^{\text{hor}}_{\text{beam}} = 3$ mrad resulting in $\sigma^{\text{ver}}_{\text{span}} = \sigma^{\text{hor}}_{\text{span}} \approx 0.15$ m [18]. 54
39 The integrated Gaussian distribution of Figure 37(b) approximated by a Gaussian distribution. 55
40 Values of g_{new} and the approximation g_{approx}. 57
41 RMSE between the integrated Gaussian distribution and the optimal Gaussian approximation g as well as the approximation using the sigma function g_{approx}. 58
42 RMSE heatmaps for the approximation steps at a distance of 100 m. . 59
43 Adapted heliostat cell drawn in blue of the original cell drawn in black. 60
44 Heliostat field layout of the PS10 plant 61
45 Normalized results of SunFlower and SolTrace for all test cases. 63
46 Interpolated fluxmaps of both tools for test case two with a total power difference of less than 0.05 %. 63
47 Relative difference between both fluxmaps. 65
48 Old results of the convolution methods [18]. 66
49 Results of the renewed convolution methods. 66
50 Fluctuations of the classical-Monte Carlo ray tracer and the multi-Monte Carlo ray tracer taking five samples per ray illustrated by the shaded region. 67
51 Average results of different Monte Carlo based ray tracers with respect to the total number of samples. 68
52 Fluctuations, illustrated by the shaded regions, of the classical- and quasi-Monte Carlo method using one sample per ray. 68
53 Average results of the different ray tracing methods for the PS10 test cases of Section 5.1 with the correct number of facets. 69
54 Average results of different ray tracing techniques for test case 2^* in dependence on the total number of rays. 70
55 Average results of different ray tracing techniques for test case 2^* in dependence on the total runtime. 71
56 Average results of different ray tracing techniques for test case 6^* in dependence on the total number of rays. 71
57 Average results of different ray tracing techniques for test case 6^* in dependence on the runtime. 72
58 Positions of the close and far heliostats. 73
59 Normalized results of both convolution methods for the far and close heliostat. 73
60 Average results of the multi-Monte Carlo method using thousand samples per ray and the Gaussian convolution method. 74
61 Acceleration of a Monte Carlo and a convolution based ray tracer using the two preselection techniques on test case 2^*. 75

62 Acceleration of a Monte Carlo and a convolution based ray tracer using a CPU parallelization with a minimal critical section on test case 2*. . 75
63 Illustration of the required cell length such that the a-sigma regions of two neighboring cells intersect. 77
64 Illustration of a reversed ray and its correlation to the original ray. The reversed ray originates from the receiver piece and travels in $-\vec{r}$ direction. 78

List of Tables

1 Coefficients of the sigma multiplier function g_{approx} from Equation (4.62). 57
2 Parameters for the flat tilted receiver 62
3 Parameters for the cylindrical cavity receiver 62
4 Basic setup for the six validation test cases. The settings are inspired by the PS10 plant and derived from a similar test case [18]. 62
5 Unique settings for each test case. 63
6 Exact results of *SunFlower* for each test case. 64
7 Exact results of *SolTrace* for each test case. 64
8 Overview over the test cases in [18]. 65

1 Introduction

In times of global warming, the need for renewable energy is greater than ever. It is indisputable that most of the CO_2 emissions are caused by fossil fuel burning for power generation and the transport sector [5]. For this reason, a lot of countries are mandating the use of renewable energy which is predicted to deliver 30 % of the electricity by 2035 [5]. In particular, the development of concentrating solar power (CSP) technology is growing faster than any other renewable technology [5].
This work concentrates on central receiver systems (CRS) which are one of the four types of CSP systems. They consist of movable mirrors, called heliostats, reflecting and concentrating the sun light onto a solar receiver mounted at the top of a tower as shown in Figure 1.

Figure 1: Solar tower power plant Planta Solar 10 (PS10) in Spain [29]. The figure is derived from Richter et al. [34, p. 2]

The receiver absorbs the solar radiation and transfers the resulting heat via a heat transfer fluid (HTF) to a power conversion system. Here, the heat is used to turn water into steam which powers a steam turbine to obtain electricity.

In order to increase the performance of a CRS, the annual energy production (AEP) needs to be maximized. A big factor of the AEP is the heliostat field layout, i.e., the positioning of the heliostats. However, as with any optimization process, a measure of its current performance is needed. This is where the optical model comes in. It tries to predict the collected solar power at the receiver for a given setting. A major part of the optical model is the ray tracer which traces the light from the sun to the heliostat onto the receiver.

1.1 State of the art

There is a variety of tools available for the simulation and optimization of solar tower power plants. A short overview which mainly focus on the used ray tracing technique is given below. It is inspired by the work of Garcia et al. [19], Richter [33] and Franke [18].

The most commonly used ray tracing method is the Monte Carlo method. However, there are different implementations of it that vary on how the solar rays are generated. In tools such as *Tonatiuh*, *MIRVAL* and *SolTrace* the rays originate on a plane above the heliostats [9, 22, 44]. To increase computational efficiency, *STRAL* and *TieSol* generate rays on the heliostat surface itself [3, 11]. By this, no rays are wasted onto the ground and the first intersection calculation is omitted. Moreover, *TieSol* parallelizes the ray tracing process on the graphic processing unit (GPU), making it extremely fast [19].

The other categories of ray tracers are using analytical approaches. They generally represent the reflected solar flux, i.e., the sun light energy per area, of a heliostat by some mathematical function. For example, *HFLCAL* expresses the flux with a circular Gaussian distribution [41]. The solar power at the receiver is then calculated by numerically integrating over the receiver area. *UNIZAR* describes the solar flux with the error function and relies on a similar numerical integration method as *HFLCAL* [37]. *DELSOL* is using a truncated expansion in Hermite polynomials as a solar flux distribution [21]. Here, the two dimensional integral is solved analytically in one dimension and numerically in the other [21]. Another analytical ray tracing tool is *HELIOS*. It is capable of describing each disturbance of the ray by a different two dimensional distribution. The resulting total disturbance is then approximated with a numerical convolution of the two dimensional distributions using the fast Fourier transform [8]. A weighting scheme is used to evaluate the integral needed for the solar power calculation.

1.2 Outline

In this thesis, different ray tracing methods will be discussed, accelerated and validated. Our optical model is presented in Section 2. Moreover, a detailed definition of the problems for which ray tracers are used is given. In Section 3, the widely used Monte Carlo ray tracing method as well as two extensions are introduced. Afterwards, a well known analytical ray tracer called *HFLCAL* [41] will be discussed. Extending on the ideas of *HFLCAL*, our Gaussian convolution method which evaluates the perturbation of each ray accurately, is presented in Section 4.2. In addition to this, Section 4.3 provides a new ray tracing technique called the integrated convolution method. It reduces the error of taking one ray to represent photon interaction of a complete cell without the need of simulating more rays. Section 5 gives a cross validation of our optical model and investigates different aspects of our newly developed ray tracers. Furthermore, acceleration strategies of all ray tracers are reviewed. Finally, future improvements are listed in Section 6.

2 Optical model

With the optical model the radiation on a solar tower receiver for a given day and time can be calculated. The presented model is expanding on the works of Franke [18] and Richter et al. [33, 35] with a more detailed focus on ray tracing techniques.

In the following, a short overview of this model is given. Section 2.1 describes environmental aspects such as the site area or how the sun is modeled. In Section 2.2 and Section 2.3 the heliostat model as well as different receiver types are being introduced. Optical losses due to effects such as blocking and shading or atmospheric attenuation are presented in Section 2.4, the required calculations in Section 2.5 and 2.6. After introducing the two main options of evaluating the solar flux in Section 2.7 all relevant information for the ray tracing techniques is as a final point summarized in Section 2.8.

2.1 Environment

In our model a Cartesian coordinate system is used where the x axis points towards East, the y axis towards North and the z axis into the sky. The site area is defined by a polygon where its boundary points can be given in Cartesian or geographic coordinates. Every object placed into the model including its expansion is checked to be inside the site polygon but outside the expansion of all other already included objects.

Information about the relative position of the sun is defined by the time-dependent azimuth γ_{solar} and altitude θ_{solar}. Moreover, the direct normal irradiation I_{DNI} expresses the amount of incoming radiation per area. From this the direction vector $\vec{\tau}_{\text{solar}}$ of the sun is calculated using Equation (2.1) from [25] as displayed below.

$$\vec{\tau}_{solar} = \begin{pmatrix} \sin\left(-\gamma_{\text{solar}}\right) \cdot -\cos\left(\theta_{\text{solar}}\right) \\ \cos\left(-\gamma_{\text{solar}}\right) \cdot \cos\left(\theta_{\text{solar}}\right) \\ \sin\left(\theta_{\text{solar}}\right) \end{pmatrix} \tag{2.1}$$

2.2 Heliostats

A solar field consist of N heliostats H_i aiming to concentrate the sunlight at a tower mounted receiver by tracking the position of the sun. Each heliostat mirror consists of small mirrors, called facets. Therefore, the whole mirror area A_i of heliostat H_i can be depicted by the sum of all facet areas. Their positions and alignments are described with a local heliostat coordinate system having the mirror center-position p_i of the heliostat as origin. The x-axis $\vec{x}_i$ is parallel to the horizontal edge of the heliostat H_i and the y-axis $\vec{y}_i$ is parallel to the vertical edge. The normal vector $\vec{n}_i$ of the heliostat scaffold defines the z-axis. This representation has the benefit that only the orientation of the local coordinate system changes for different sun positions and not the alignment of the facets, as they are fixed on the heliostat scaffold.

Since all heliostats are aiming towards the receiver aiming point $p_{\text{aim},i}$ the normalized reflective vector $\vec{r}_i$ is defined by

$$\vec{r}_i = \frac{p_{\text{aim},i} - p_i}{|p_{\text{aim},i} - p_i|}. \tag{2.2}$$

Furthermore, the normal vector $\vec{n}_i$ can be computed with the law of reflection as the incoming vector $\vec{\tau}_{\text{solar}}$ and outgoing vector $\vec{r}_i$ are known

$$\vec{n}_i = \frac{\vec{r}_i + \vec{\tau}_{\text{solar}}}{|\vec{r}_i + \vec{\tau}_{\text{solar}}|}. \tag{2.3}$$

With all that the local coordinate system is given by

$$\vec{x}_i = \frac{\vec{n}_i \times (0,0,1)^T}{|\vec{n}_i \times (0,0,1)^T|}, \quad \vec{y}_i = \vec{n}_i \times \vec{x}_i, \quad \vec{z}_i = \vec{n}_i. \tag{2.4}$$

However, when $\vec{n}_i$ is close to $(0,0,1)^T$ the cross product $\vec{n}_i \times (0,0,1)^T$ results in the zero vector. In this case the local coordinate system is calculated as follows,

$$\vec{y}_i = \begin{pmatrix} -p_{i,x} \\ -p_{i,y} \\ 0 \end{pmatrix} \quad \vec{x}_i = \vec{n}_i \times \vec{y}_i, \quad \vec{z}_i = \vec{n}_i, \tag{2.5}$$

with $p_{i,x}$, $p_{i,y}$ as the global x and y coordinates of heliostat H_i, respectively.

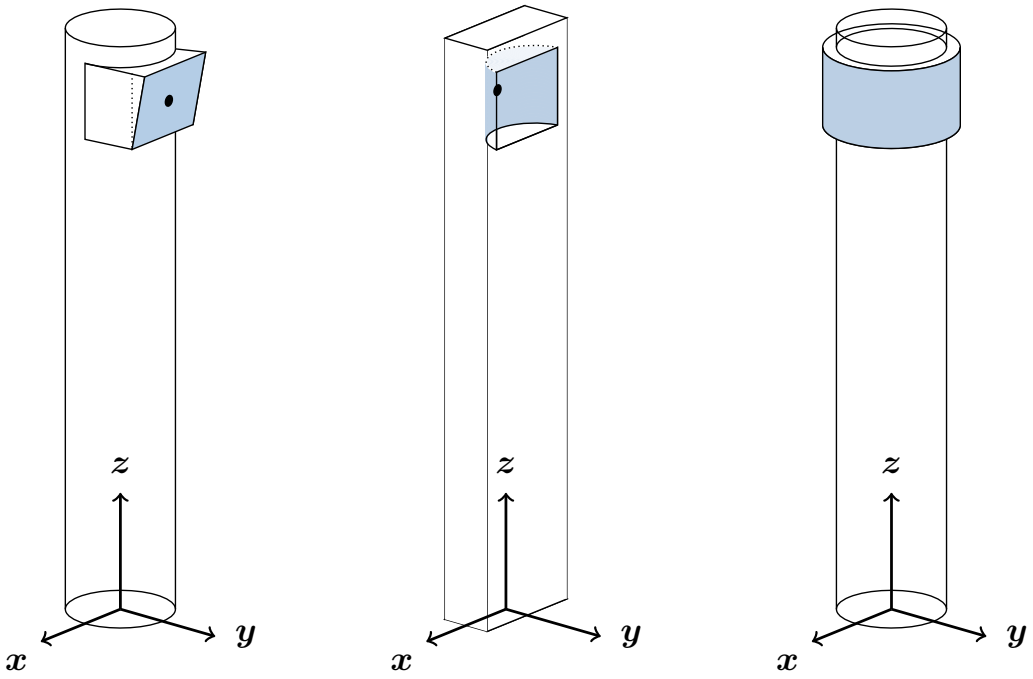
We differentiate two ways to position and align facets on the heliostat scaffold which is called canting. With **on-axis canting**, the ideal focusing alignment of all facets for the case that the sunlight emerges from the receiver is used. This results in the facets being positioned around a paraboloid with a focus at the receiver.

In **off-axis canting** for a fixed sun vector $\vec{\tau}_{\text{solar}}$ the facets are aligned to focus the light at the receiver. However, since the sun, heliostat and receiver are not on one common axis, the facets are positioned at the side of a paraboloid.

2.3 Receiver

We consider the tower to be either a cylinder or a cuboid at position p_{tower}. The receiver collecting the reflected sun light is transforming their radiation into heat and is mounted just below the towers top. In our model, three types of receivers can be represented, see Figure 2.

- **Flat tilted receiver**
  It has a tilted rectangular receiver area to whose center all heliostats are aiming. The receiver is defined by its width, height, tilt angle and orientation angle.



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

Figure 2: Different tower types with different receiver types. The Figure is derived from Richter [33, p. 11]

The following two cylindrical receivers are inspired by the solar tower power plants Planta Solar 10 (PS10) [29] and Gemasolar [12] both operating in Spain.

Both of them are predicated on the definition of regular polygons, i.e. polygons having equal angles at every vertex and all edges have the same length [2]. Given the edge length e and the number of edges n the diameter d of the circumscribed circle around the polygon is calculated as follows [2]

$$d = \frac{e}{\sin\left(\frac{\pi}{n}\right)}. \tag{2.6}$$

A regular polygon with nine edges and its circumscribed circle is shown in Figure 3.

- **Cylindric external receiver**
  The horizontal shape of our external receiver is described using a regular polygon which is lifted to a certain height h in order to obtain a receiver as in Figure 5(b). A comparable receiver is found in the Gemasolar plant [12]. Here, the edges are referred to as panels. Thus, the receiver is defined by the number of panels as well as the panel width and height. Heliostats are now assumed to aim towards the closest point at the horizontal center line of the aperture.

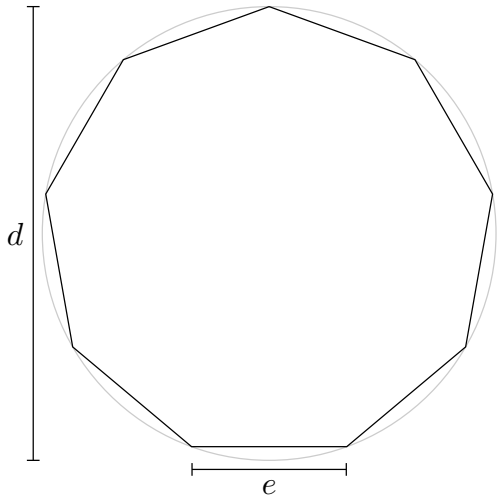


Figure 3: Regular polygon with nine edges, an edge length of e and a circumscribed circle of diameter d which is drawn in gray.

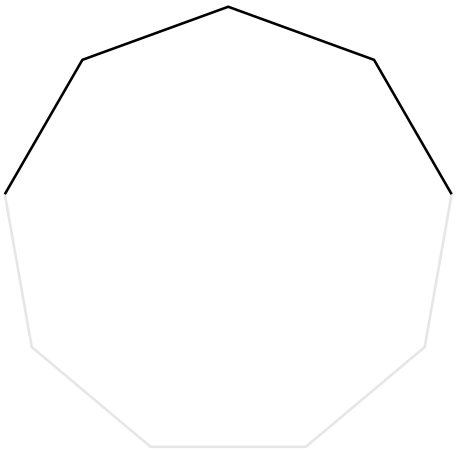


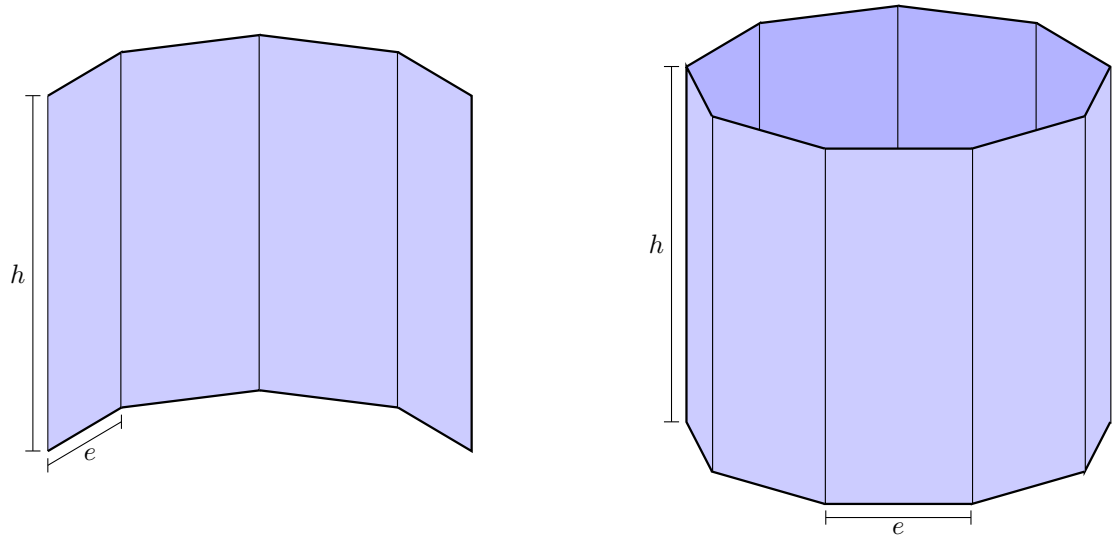
Figure 4: Horizontal shape of a cavity receiver with four panels. It is based on a regular polygon with nine edges as illustrated in gray.

- **Cylindric cavity receiver**
  Our cavity receiver is also defined by the number of panels n, each's width e and height h. However, now the circumscribed circle of a polygon with $2n + 1$ edges and an edge length of e is used. Therefore, n edges of such a polygon describe the horizontal shape of our cavity receiver. Additionally, the cavity receiver is also lifted by the height l to the lower edge of its entrance. This is important for blocking calculations which will be discussed in Section 3.4. By this definition the receiver is similar to the one used in the PS10 plant. Its horizontal shape is shown in Figure 4 and the resulting receiver in Figure 5(a). Again, all heliostats are assumed to focus the light on its center of aperture.

2.4 Optical losses

Sun rays have to be traced from the sun to the heliostat onto the receiver in order to calculate the amount of radiation at the receiver. There are different losses influencing the rays which will be discussed in the following.

Blocking and shading effects appear due to the fact that heliostats as well as the tower are casting a shadow and some heliostats might block the reflected rays of other heliostats. Instead of checking every ray against every heliostat, a set of potentially shading or blocking heliostats is precomputed for each heliostat. Only for those heliostats, a possible blocking or shading will be evaluated. The sets are computed as follows.



(a) Cylindric cavity receiver with four panels

(b) Cylindric external receiver with nine panels

Figure 5: Illustration of our cylindrical receiver types

- **Tower shading**
  A simplified shadow of the tower is used to compute the set of potentially tower shaded heliostats. To do so, the minimal distance between the line of the sun ray hitting the heliostat's center p_i and a line from the tower position p_{tower} straight into the sky has to be calculated. Now, a heliostat is potentially tower shaded if the minimal distance between two lines is less or equal to half of the heliostat expansion plus half of the tower expansion.

- **Heliostat shading**
  For heliostat shading a similar approach can be applied. However, now the minimal distance between the line of the sun ray hitting p_i and the center p_j of a neighbouring heliostat H_j is used. A distance smaller than the expansion of a heliostat implies that H_i is potentially shaded by H_j.

- **Heliostat blocking**
  The computation of potentially blocked heliostats is almost the same as for heliostat shading. The only difference is that the line of the sun ray is replaced by a line going through the aiming point p_{aim} of the heliostat and its center point p_i.

In Section 2.6 we will discuss shading and blocking computation more in detail.

With cylindrical cavity receivers another blocking type is introduced, called **tower blocking**. It occurs due to the fact that some solar rays potentially hitting the receiver would first intersect with the tower. As shown in Figure 2(b), some rays might not

go through the rectangular entrance of the receiver which is referred to as the receiver window. This is the only blocking and shading type that has to be handled differently depending on the ray tracer used.

Cosine effects describe the reduced projected area of the facets due to their tilted alignment. The reflected area is reduced by the cosine of the angle of incidence [28], i.e., the angle between the heliostat surface normal and the incoming sun rays. Therefore, the loss is modeled by

$$\eta_{\cos,i} = \langle \vec{\tau}_{\text{solar}}, \vec{n}_i \rangle, \tag{2.7}$$

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

Heliostat reflectivity tries to model the losses due to the mirrors not perfectly reflecting the sun rays, i.e., they scatter the light and also absorb parts of it. In our model a constant value η_{ref} encounters this effect as it is often used [31].

Atmospheric attenuation efficiency considers the power losses as the light travels through the atmosphere. It depends on the distance d_i from the position p_i of the heliostat and the aiming point $p_{\text{aim},i}$ and can be computed with the formula derived by Schmitz et al. [39]

$$\eta_{\text{aa},i} = \begin{cases} 0.99321 - 1.176 \cdot 10^{-4} d_i + 1.97 \cdot 10^{-8} d_i^2 & , d_i \leq 1000m \\ \exp(-1.106 \cdot 10^{-4} d_i) & , d_i > 1000m \end{cases}. \tag{2.8}$$

Optical errors encounter deviations of the computed reflection ray to an actual solar ray. The direction of a ray hitting the surface of a heliostat can deviate since the sun is a sphere. Based on the idea of Rabl [32] we model this error as a Gaussian distribution with a standard deviation σ_{sun}. Moreover, most of the time the heliostat will slightly differ from the intended alignment which leads to a tracking error. Since the heliostat mirror has a certain roughness it is reflecting a small outgoing cone instead of an exact ray which is referred to as error cone, see Figure 6. This slope error as well as the tracking error can again be described using Gaussian distributions. As heliostats have a vertical and a horizontal tracking axis, two Gaussian distributions with standard deviations $\sigma^{\text{ver}}_{\text{tracking}}$ and $\sigma^{\text{hor}}_{\text{tracking}}$ are required for accurate modeling. Moreover, the slope error is also not always equal in horizontal and vertical direction and deviates for different positions (x, y) on the mirror surface. Thus, two matrices $\sigma^{\text{hor}}_{\text{slope}}(x, y)$ and $\sigma^{\text{ver}}_{\text{slope}}(x, y)$ are used to express the slope error. With the Central Limit Theorem [40], stating that the convolution of two or more distribution functions converge to a Gaussian distribution, the optical errors are modeled with one Gaussian distribution for each direction by combining the standard deviations to

$$\begin{aligned} \sigma^{\text{hor}}_{\text{beam}} &= \sqrt{(\sigma_{\text{sun}})^2 + (\sigma^{\text{hor}}_{\text{tracking}})^2 + (\sigma^{\text{hor}}_{\text{slope}}(x, y))^2}, \\ \sigma^{\text{ver}}_{\text{beam}} &= \sqrt{(\sigma_{\text{sun}})^2 + (\sigma^{\text{ver}}_{\text{tracking}})^2 + (\sigma^{\text{ver}}_{\text{slope}}(x, y))^2}. \end{aligned} \tag{2.9}$$

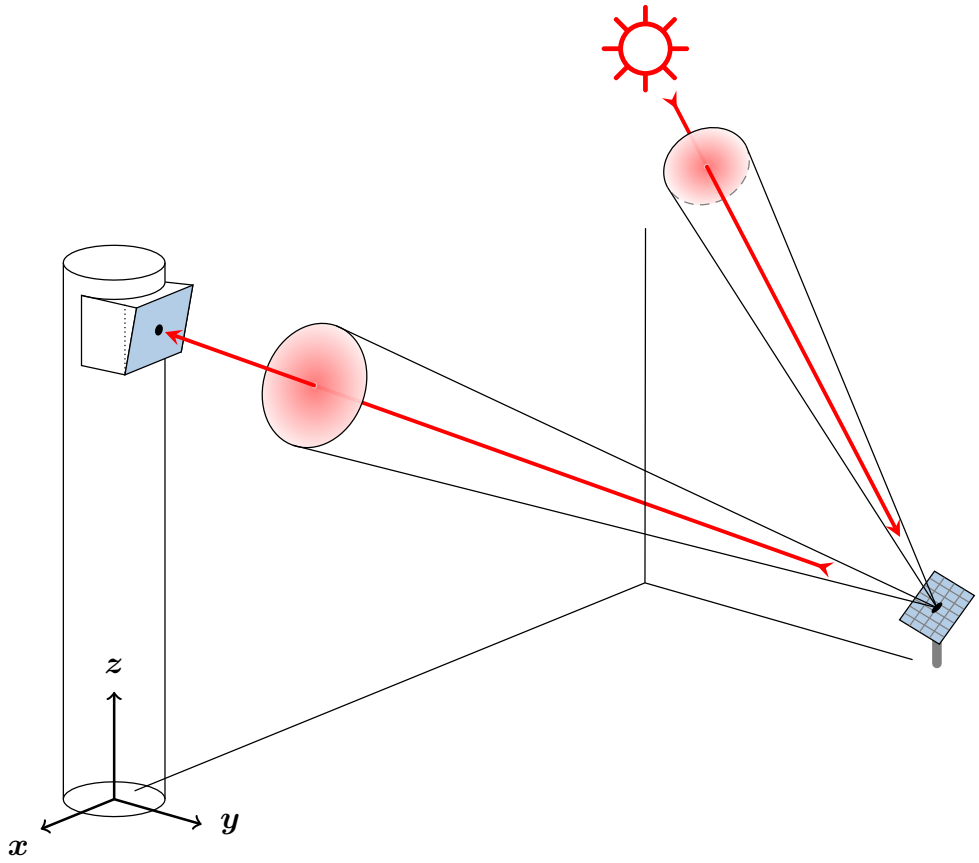


Figure 6: Error cones representing the optical errors [33, p. 17].

The values of these Gaussian distributions can be interpreted as the relative likelihood of certain angles between the perfect reflected ray and an actual ray. Thus, the standard deviations $\sigma_{\text{beam}}^{\text{hor}}$ and $\sigma_{\text{beam}}^{\text{ver}}$ are given in milliradian.

As two separate Gaussian distributions are used to describe horizontal and vertical deviation of the perfect reflected ray, the error cone is actually of elliptical shape instead of being circular.

The handling of optical errors is the main difference between the ray tracing techniques that will be discussed shortly. Prior to that, the generation of representative solar rays as well as blocking and shading calculations will be discussed as most of them are handled similarly for each ray tracer.

2.5 Generation of representative solar rays

To generate solar rays each heliostat facet is uniformly discretized into smaller cells as shown in Figure 9. Cells of a curved facet are assumed to be flat. The perfect reflected rays originate at the center s of their corresponding heliostat cell and their directions $\vec{r}$ can be computed with the facet piece normal $\vec{n}$ and sun direction $\vec{\tau}_{\text{solar}}$ by the law of reflection.

$$\vec{r} = \vec{\tau}_{\text{solar}} - 2 \, \langle \vec{\tau}_{\text{solar}}, \vec{n} \rangle \, \vec{n}. \tag{2.10}$$

Indices to determine the currently viewed heliostat cell are omitted to simplify notations in the following.

For each heliostat cell a representative ray is created. Therefore, in contrast to other tools such as *Tonatiuh*, *MIRVAL* and *SolTrace*, the rays are generated directly on the heliostats mirror surface instead of on a plane above them as described in [7]. This saves computing the first intersection and no rays are wasted due to ground impacts. In our model the incoming radiation I_{DNI} is assumed to spread evenly among each facet so the power of the ray can be computed with their cell area A_{cell}. In respect to the optical losses discussed in Section 2.4 the representative ray power P_{ray} of the current heliostat cell is given by

$$P_{\text{ray}} \;=\; I_{\text{DNI}} \;\cdot\; A_{\text{cell}} \;\cdot\; \eta_{\text{cos}} \;\cdot\; \eta_{\text{ref}} \;\cdot\; \eta_{\text{aa}}. \tag{2.11}$$

Now that the perfect reflected ray is specified, a more detailed description on how to perturb it is needed. In particular the horizontal and vertical direction of deviation must be specified. The deviation directions on the heliostat normal and the sun direction are assumed to be orthogonal to $\vec{n}$ and $\vec{\tau}_{\text{solar}}$, respectively. Due to the law of reflection the resulting convoluted deviation directions are orthogonal to the ray direction. For a two dimensional perturbation this already fully specifies the deviation direction, see Figure 7. Note that as the Gaussian distribution describing the deviation is axis-symmetric, any of the two orthogonal directions may be used. However, in three dimensions there are two deviation directions thus requiring more information to be able to determine them correctly.

First, the horizontal and vertical deviation directions of the mirror normal corresponding to the tracking and slope error need to be defined. This is done using the edges of the heliostat mirror where the horizontal edges corresponds to the horizontal deviation direction. The definition is not arbitrary but rather chosen due to the direction of the horizontal tracking axis. Similarly the vertical deviation direction is given by the vertical tracking axis. However, the directions of the slope errors usually do not match the directions of the tracking errors. But they can always be expressed as a combination of the horizontal and vertical tracking directions as they all lie on

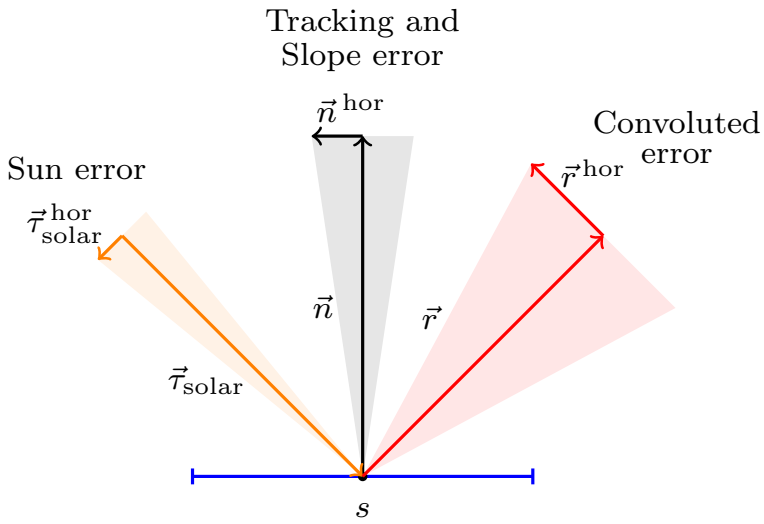


Figure 7: Two dimensional perturbation for horizontal deviations. The lengths of the deviation vectors illustrate the amount of perturbation.

the same plane. Thus, the slope errors in $\sigma^{\mathrm{hor}}_{\mathrm{slope}}(x, y)$ and $\sigma^{\mathrm{ver}}_{\mathrm{slope}}(x, y)$ are assumed to be transformed such that their direction matches the ones of the corresponding tracking errors.

With the horizontal and vertical deviation directions of the mirror normal, the resulting deviation directions of the ray can be calculated. One way to do this is by projecting the ray onto the plane spanned by the mirror normal and one deviation direction. Afterwards, the argument for a two dimensional deviation can be applied meaning the resulting deviation direction is given by an orthogonal direction to the ray on the plane. The cross product of this deviation direction and the ray direction then determines the remaining deviation.

However, one can also apply the same rotation needed to obtain the ray direction from the mirror normal to the deviation directions of the mirror normal, in order to get the resulting deviation directions of the ray. Figure 8 illustrates the required rotation. The rotation matrix $R_{\vec{a},\beta}$ for rotating a vector around an arbitrary axis $\vec{a}$ by an angle β is defined as [38]

$$\begin{aligned} R_{\vec{a},\beta} &= \cos(\beta) \cdot I + (1 - \cos(\beta)) \cdot (\vec{a} \otimes \vec{a}) + \sin(\beta) \cdot R_{\vec{a}}, \\ R_{\vec{a}} &= \begin{pmatrix} 0 & -a_z & a_y \\ a_z & 0 & -a_x \\ -a_y & a_x & 0 \end{pmatrix}, \end{aligned} \tag{2.12}$$

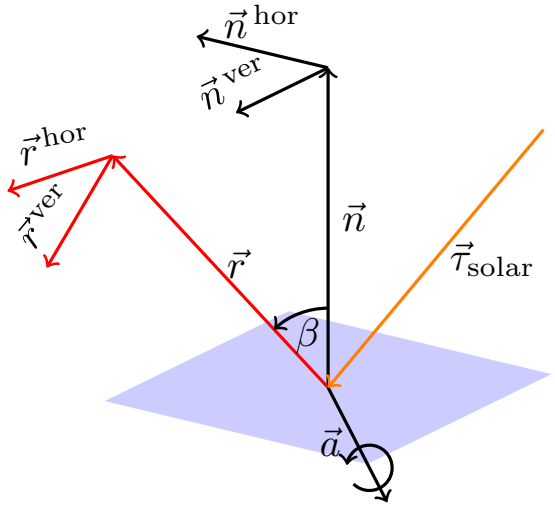


Figure 8: Illustration of the required rotation to obtain the convoluted deviation directions. The shortened deviation vectors are actually of unit length.

with $\vec{a} = \begin{pmatrix} a_x \\ a_y \\ a_z \end{pmatrix}$, I as the identity matrix and $(\cdot \otimes \cdot)$ as the outer dot product.
In our case the rotation axis $\vec{a}$ as well as the angle β are calculated by

$$\begin{aligned} \vec{a} &= \vec{n} \times \vec{r}, \\ \beta &= \arccos\left(\langle \vec{n}, \vec{r} \rangle\right), \end{aligned} \tag{2.13}$$

with $\vec{n}$ as the normal of the mirror and $\vec{r}$ the normalized ray vector.

The order of $\vec{n}$ and $\vec{r}$ is important when computing $\vec{a}$ as $R_{\vec{a},\beta}$ performs a counterclockwise rotation in the direction of a $\vec{a}$ [38]. Therefore, switching the order and applying $R_{\vec{a},\beta}$ on $\vec{n}$ would no longer result in $\vec{r}$ as intended. Finally, the normalized vectors $\vec{r}^{\,\mathrm{hor}}$ and $\vec{r}^{\,\mathrm{ver}}$ defining the deviation directions of the ray are given by

$$\begin{aligned} \vec{r}^{\,\mathrm{hor}} &= \frac{R_{\vec{a},\beta}\vec{n}^{\,\mathrm{hor}}}{|R_{a,\beta}\vec{n}^{\,\mathrm{hor}}|}, \\ \vec{r}^{\,\mathrm{ver}} &= \frac{R_{\vec{a},\beta}\vec{n}^{\,\mathrm{ver}}}{|R_{a,\beta}\vec{n}^{\,\mathrm{ver}}|}, \end{aligned} \tag{2.14}$$

with $\vec{n}^{\,\mathrm{hor}}$ and $\vec{n}^{\,\mathrm{ver}}$ as the horizontal and vertical deviation of the mirror normal, respectively.

The discretization of the heliostat facet into cells as well as the deviation of the corresponding representative ray lead to the two main sources of errors for each ray tracer. Firstly, the fact that one ray is used to represent photon interactions of a heliostat cell and secondly the evaluation of the error cone of each representative ray.

How both of these errors are handled differentiates between the ray tracing techniques. However, they all have in common that the error of using one representative ray per cell area can effectively be reduced by increasing the number of discretization points leading to smaller cell areas.

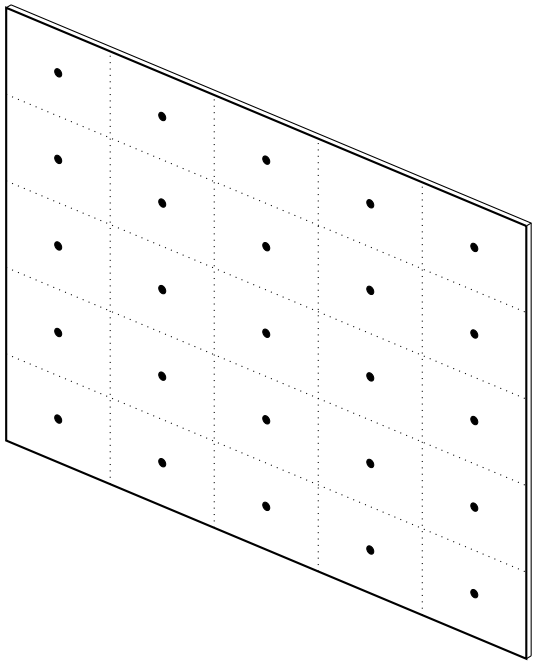


Figure 9: Discretization of a heliostat facet into 5 by 5 cells. From each cell a representative ray originates, which intensity is weighted by its area [18, p. 16].

2.6 Blocking and shading computations

As already mentioned, blocking and shading calculations are almost equally for each ray tracer with the only difference being the tested ray. For the Monte Carlo methods, a perturbed version of the representative ray is evaluated whereas in convolution based methods the representative ray itself is used. The ray is then traced, i.e., it is checked for intersections with potential blocking heliostats. Moreover, to encounter shading effects, intersections of the incoming sun ray with potential shading heliostats as well as the tower are evaluated. The sets of potentially blocking or shading heliostats were precomputed as described in Section 2.4. Algorithm 1 illustrates the required steps.

Algorithm 1 Blocking and Shading

```
1: function notBlockedOrShaded(H, s, r⃗)
2:     for each heliostat H_j in H.potentialBlockingHeliostats do
3:         if H_j.blocks(s, r⃗) then
4:             return False
5:     for each heliostat H_j in H.potentialShadingHeliostats do
6:         if H_j.shades(s, τ⃗_solar) then
7:             return False
8:     if H.potentialTowerShaded() then
9:         if tower.shades(s, τ⃗_solar) then
10:            return False
11:    return True
```

2.7 Discretization of the receiver

Now, there are basically two options to evaluate the flux at the receiver. When only the total sum of solar power is of interest, the receiver is used as it is. On the other hand, to get an understanding of how well the solar flux is focused, a so called flux map is needed [24]. Here, the receiver surface is discretized into smaller pieces. An evaluation of a discretized receiver, however, generally requires more processing time. The discretization itself is described by the number of desired receiver pieces in horizontal and vertical direction. For our flat receiver the rectangular receiver surface is then discretized accordingly and in the case of a cylindrical receiver each panel is discretized.

2.8 Ray tracing pipeline

Since our optical model is introduced and the required background information discussed we can set up our ray tracing pipeline, see Figure 10. Besides the steps that are part of the ray tracers this pipeline also sums up the necessary steps before using a ray tracer to evaluate the solar radiation at the receiver. Therefore, it can also be understood as a short summary of our optical model.

- **Setup**
  In this step all relevant data is loaded, the heliostat field generated and each heliostat is checked whether it intersects with another nearby heliostat. Furthermore, the tower is placed and the receiver build. This also includes the discretization of the receiver and each heliostat into smaller receiver pieces and heliostat cells. Afterwards, information about the sun is processed, the heliostats are aligned accordingly and the sets of potential blocking and shading heliostats are computed.

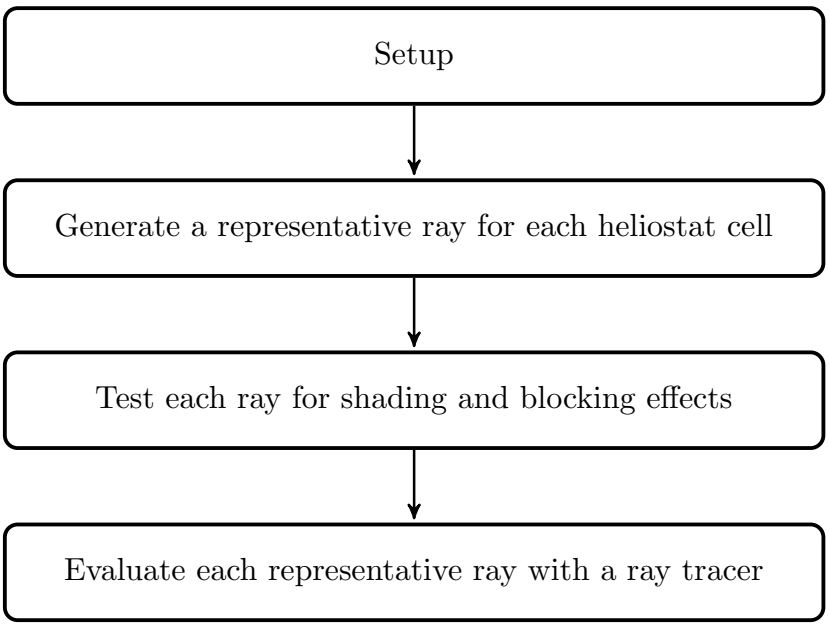


Figure 10: Simplified pipeline for the evaluation of the solar radiation at the receiver.

- **Generation of representative rays**
  For every heliostat cell a representative ray is generated. The ray originates in its cell center s and has the direction $\vec{r}$ of the perfect reflected ray.

- **Testing rays for shading and blocking effects**
  Depending on which ray tracing method is used either the perfect reflected ray or a perturbed ray is tested for shading an blocking effects. In the case of a Monte Carlo ray tracer the perturbed ray is tested and for a convolution ray tracer the perfect reflected ray is tested. Only rays that are not blocked or shaded are evaluated in the next step.

- **Evaluating the representative ray**
  Finally, the main part of a ray tracer being the evaluation of each representative ray is utilized. The evaluation process of each ray tracer will be discussed in the following sections.

Before getting into details on how a representative ray is evaluated a quick recap of the relevant information is given. The current representative ray originates at the point s, travels in the direction of the normalized vector $\vec{r}$ and has the power P_{ray}. Its perturbation is given by the normalized vectors $\vec{r}^{\text{hor}}$, $\vec{r}^{\text{ver}}$ for horizontal and vertical deviation, respectively. In addition, the deviation itself is described by two Gaussian functions for either direction with standard deviations $\sigma_{\text{beam}}^{\text{hor}}$ and $\sigma_{\text{beam}}^{\text{ver}}$. Figure 11 shows a simplified version of the problem of evaluating a representative ray in the horizontal deviation plane which is later used to illustrate the principle of each ray tracing method.

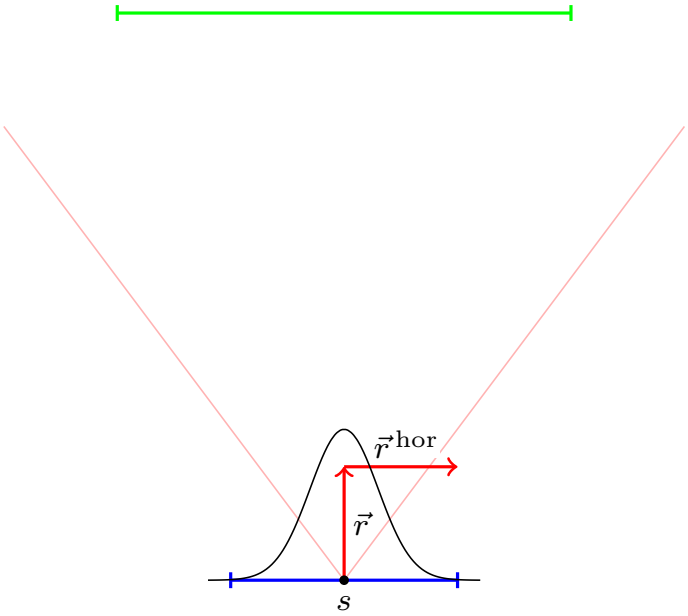


Figure 11: Problem of evaluating a representative ray traveling in the direction of $\vec{r}$ with a deviation direction of $\vec{r}^{\,\mathrm{hor}}$. The Gaussian function defining the deviation is shown in black, the heliostat cell in blue, the receiver cell in green and the thin red lines illustrate the error cone.

3 Monte Carlo ray tracing methods

The Monte Carlo ray tracing methods are straightforward techniques to compute the concentrated solar flux distribution on a receiver. They rely on the law of large numbers, meaning a large number of randomized rays are simulated to approximate optical errors. In general, Monte Carlo methods have been used to solve mathematical and physical problems by repeated random sampling [26]. Since these ray tracing techniques replicate real interactions of photons, they tend to be more accurate then analytical methods when enough rays are generated [44]. Furthermore, as ray intersections can easily be calculated for different geometry, Monte Carlo ray tracing methods are suitable for complex cases where analytical techniques are not applicable anymore [24]. However, considering the computational effort involved in simulating a vast number of rays, these methods have a comparatively long processing time.

The Monte Carlo ray tracing methods discussed in the following sections mainly differentiate in how the perturbed rays are generated.

3.1 Classical-Monte Carlo method

In classical-Monte Carlo ray tracing the perfect reflected ray gets perturbed with Gaussian noise based on the distributions in Section 2.4, see Figure 12. Currently they refer

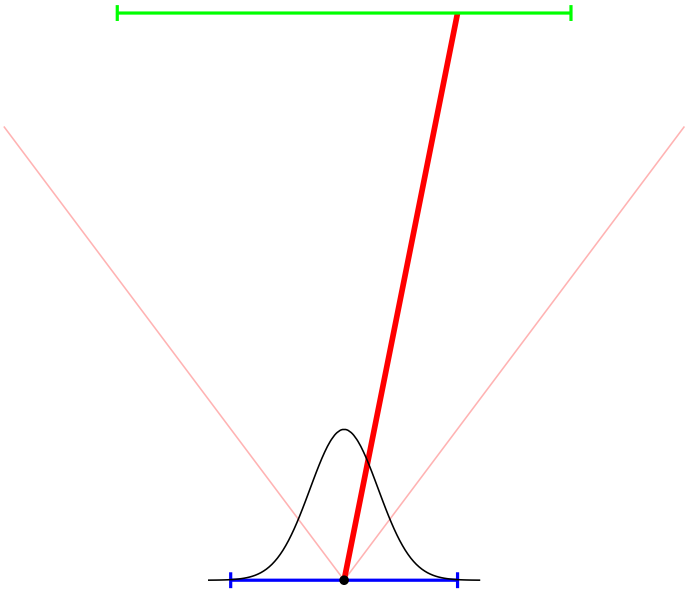


Figure 12: Basic principle of the classical-Monte Carlo method. The generated perturbed ray is shown in red.

to certain angles of deviation in either of the deviation directions $\vec{r}^{\,\mathrm{hor}}$ and $\vec{r}^{\,\mathrm{ver}}$. To simplify calculations, the Gaussian distributions are transformed to relate to the resulting lengths of deviation, see Figure 13. In particular, their standard deviations are converted to

$$\begin{aligned} \sigma_{\mathrm{beam}}^{\mathrm{hor'}} &= 1 \cdot \tan\left(\sigma_{\mathrm{beam}}^{\mathrm{hor}}\right), \\ \sigma_{\mathrm{beam}}^{\mathrm{ver'}} &= 1 \cdot \tan\left(\sigma_{\mathrm{beam}}^{\mathrm{ver}}\right). \end{aligned} \tag{3.1}$$

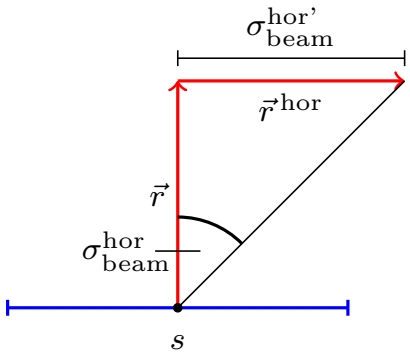


Figure 13: Transformation from angle of horizontal deviation to length.

Then two random deviation lengths d^{hor} and d^{ver} from each of the two Gaussian distributions are generated and added to the ray vector $\vec{r}$ as follows

$$\vec{r}_{\mathrm{dev}} = \vec{r} + d^{\mathrm{hor}}\vec{r}^{\,\mathrm{hor}} + d^{\mathrm{ver}}\vec{r}^{\,\mathrm{ver}}. \tag{3.2}$$

The resulting ray is checked for blocking and shading effects as described in Section 2.6. Only if no shading or blocking is affecting the ray, it is traced against the receiver. Algorithm 2 shows the basic steps of this method for calculating the total solar power.

Algorithm 2 Classical-Monte Carlo ray tracer

function CLASSICALMONTECARLO($s, \vec{r}, \vec{r}^{\text{hor}}, \vec{r}^{\text{ver}}, \sigma^{\text{hor}}_{\text{beam}}, \sigma^{\text{ver}}_{\text{beam}}$)
  $solarPower \leftarrow 0$
  $\vec{r}_{\text{dev}} \leftarrow \text{perfectReflection}(s,\ \vec{\tau}_{solar})$
  $\sigma^{\text{hor'}}_{\text{beam}} \leftarrow \tan(\sigma^{\text{hor}}_{\text{beam}})$
  $\sigma^{\text{ver'}}_{\text{beam}} \leftarrow \tan(\sigma^{\text{ver}}_{\text{beam}})$
  $d^{\text{hor}} \leftarrow \text{generateRandomValueFromGaussian}(\sigma^{\text{hor'}}_{\text{beam}})$
  $d^{\text{ver}} \leftarrow \text{generateRandomValueFromGaussian}(\sigma^{\text{ver'}}_{\text{beam}})$
  $\vec{r}_{\text{dev}} \leftarrow \vec{r} + d^{\text{hor}}\vec{r}^{\text{hor}} + d^{\text{ver}}\vec{r}^{\text{ver}}$
  if NOTBLOCKEDORSHADED($H, s, \vec{r}_{\text{dev}}$) **then**
    if intersect(Ray($s, \vec{r}_{\text{dev}}$), Receiver) **then**
      $P_{\text{ray}} \leftarrow I_{DNI} \cdot A_{cell} \cdot \eta_{cos} \cdot \eta_{ref} \cdot \eta_{aa}$
      $solarPower \leftarrow solarPower + P_{\text{ray}}$
  return $solarPower$

3.2 Multi-Monte Carlo method

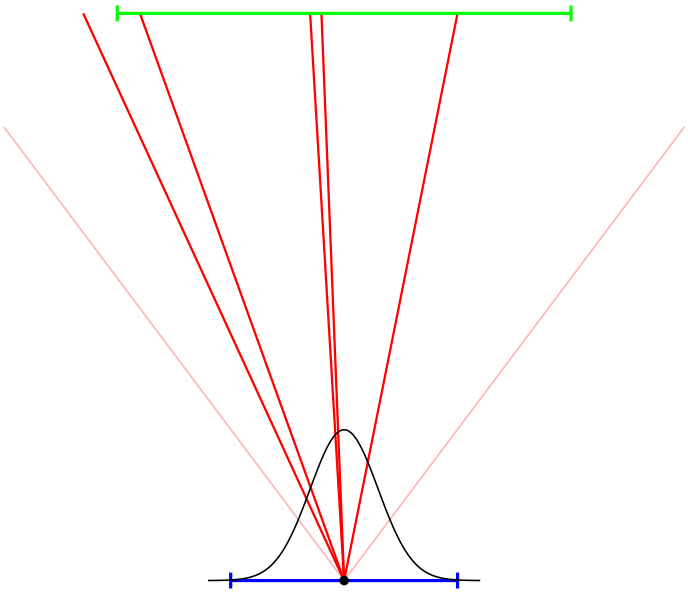


Figure 14: Basic principle of the multi-Monte Carlo method. Here, multiple samples, shown in red, are taken from the error cone.

In the classical-Monte Carlo method one ray is used to evaluate the error cone of a

representative ray and therefore photon interactions of a whole cell area. To further improve the accuracy, a multi-Monte Carlo ray tracer takes multiple samples of the error cone for each representative ray, as shown in Figure 14. Therefore, the power P_{ray} of Equation (2.11) has to be split equally among all samples.

3.3 Quasi-Monte Carlo method

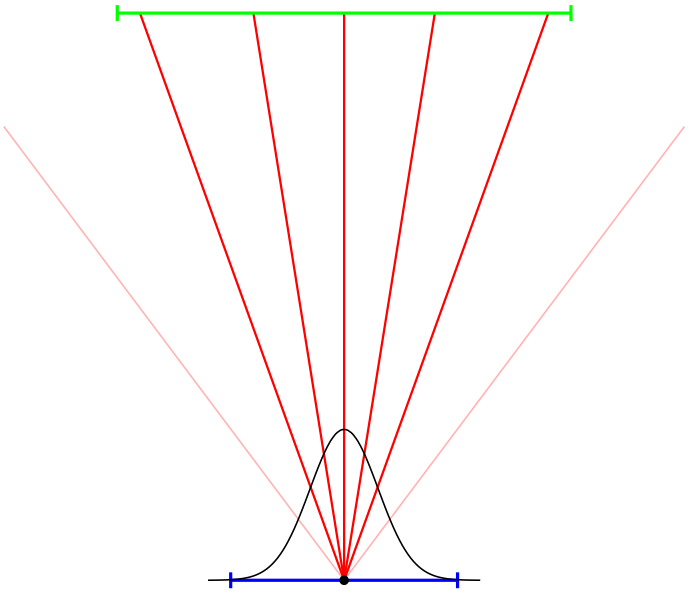


Figure 15: Basic principle of the quasi-Monte Carlo method. Here, the generated samples are more uniform and thus better represent the error cone with few samples.

As a refinement of the multi-Monte Carlo method, a quasi-Monte Carlo ray tracer enhances convergence to the actual solar power by using a different sequence to generate deviation lengths from the Gaussian distributions. The idea is that fewer samples are needed in order to represent the error cone, as illustrated in Figure 15. Here, quasi-random sequences are used instead of pseudo-random sequences as in multi- and classical-Monte Carlo. However, the name is misleading since these sequences do not attempt to imitate the behavior of random sequences but rather aim to produce more uniform elements [13]. Uniformity of a sequence is measured in discrepancy where a low discrepancy indicates better uniformity. For this reason and as this causes the improvement in convergence, quasi-random sequences are also referred to as low-discrepancy sequences. In numerical integration, the quasi-Monte Carlo method converges at a rate $O(N^{-1} \log N^k)$ for some constant k, compared to $O(N^{-\frac{1}{2}})$ for classical-Monte Carlo integration [13].
Our quasi-Monte Carlo ray tracer uses the well known sobol sequence as quasi-random sequence because of its efficient implementation [15].

3.4 Tower blocking

As stated in Section 2.4 besides the blocking and shading effects tested by Algorithm 1 there also exists tower blocking in the case of a cavity receiver. Here, all rays first have to go through the receiver window. Due to the receiver window being a rectangle, at most two of its edges can cause the rays of one heliostat cell to be blocked against meaning that the rays would not go through the window. First, there is a vertical blockage because of the closest vertical edge of the receiver window to the ray origin. Second, the lower horizontal edge causes a horizontal blockage. Both of these blockages are determined by using two planes B^{hor} and B^{ver} each corresponding to one blockage. Figure 16 illustrates the blockage with l as the lifting height of the receiver, as discussed in Section 2.3.

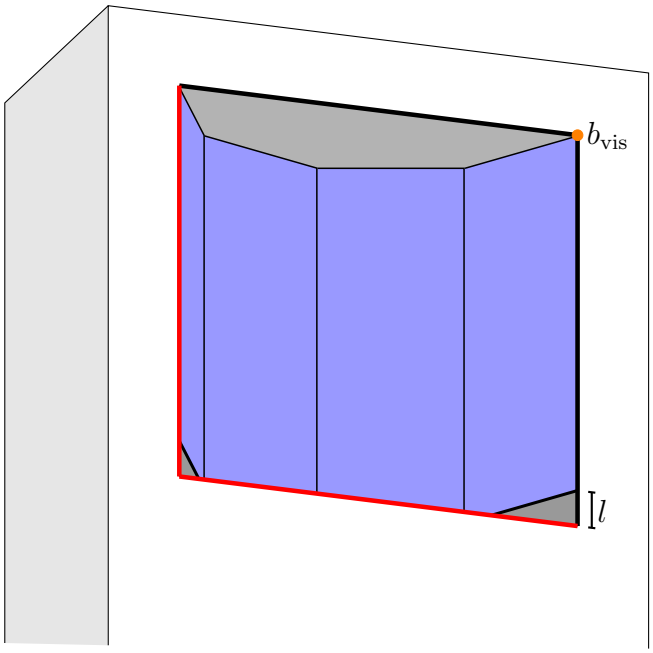


Figure 16: Blockage of the cavity receiver. The horizontal and vertical blocking edges are drawn in thick red and the rest of the receiver window in thick black. Here, l is the lifting height of the receiver to its lower window edge and b_{vis} a definitely visible point.

Each plane is defined by the two endpoints b_1 and b_2 of the corresponding blocking edge as well as the ray source s. From this the normals $\vec{n}_{\text{B}}^{\text{hor}}$, $\vec{n}_{\text{B}}^{\text{ver}}$ of each plane can be calculated leading to

$$\begin{aligned} \vec{n}_{\text{B}}^{\text{hor}} &= \frac{b_1^{\text{hor}} - b_2^{\text{hor}} \times s - b_2^{\text{hor}}}{|b_1^{\text{hor}} - b_2^{\text{hor}} \times s - b_2^{\text{hor}}|} \Rightarrow B^{\text{hor}} = \{p \in \mathbb{R}^3 | \langle p, \vec{n}_{\text{B}}^{\text{hor}} \rangle - \langle s, \vec{n}_{\text{B}}^{\text{hor}} \rangle = 0\}, \\ \vec{n}_{\text{B}}^{\text{ver}} &= \frac{b_1^{\text{ver}} - b_2^{\text{ver}} \times s - b_2^{\text{ver}}}{|b_1^{\text{ver}} - b_2^{\text{ver}} \times s - b_2^{\text{ver}}|} \Rightarrow B^{\text{ver}} = \{p \in \mathbb{R}^3 | \langle p, \vec{n}_{\text{B}}^{\text{ver}} \rangle - \langle s, \vec{n}_{\text{B}}^{\text{ver}} \rangle = 0\}, \end{aligned} \tag{3.3}$$

with $b_1^{\text{hor}}, b_2^{\text{hor}}$ as the endpoints of the lower horizontal edge and $b_1^{\text{ver}}, b_2^{\text{ver}}$ as the endpoints of the closest vertical edge to the ray origin s.

Depending on which side of the planes a ray lies it is either blocked or not which is referred to as the visibility of a ray. The side of the plane on which a point p lies is determined by the sign of $\langle p, \vec{n}\rangle - \langle s, \vec{n}\rangle$ being the plane equation with $\vec{n}$ as the normal of the plane. If the expression is zero the point lies on the plane, see Equation (3.3). However, if the sign is positive or negative the point lies on the positive or negative side of the plane, respectively. Therefore, a point which is definitely visible is needed and any point having the same sign as the visible point for both plane equations of B^{hor} and B^{ver} is also visible.
The corner b_{vis} of the receiver window rectangle not included in any of the two blocking edges is such a point, see Figure 16. With this the visibility of a point p for each plane can be determined as follows

$$\begin{aligned} \text{sign}(\langle p, \vec{n}_{\text{B}}^{\text{hor}}\rangle - \langle s, \vec{n}_{\text{B}}^{\text{hor}}\rangle) == \text{sign}(\langle b_{\text{vis}}, \vec{n}_{\text{B}}^{\text{hor}}\rangle - \langle s, \vec{n}_{\text{B}}^{\text{hor}}\rangle) \Rightarrow p \text{ is horizontally visible,} \\ \text{sign}(\langle p, \vec{n}_{\text{B}}^{\text{ver}}\rangle - \langle s, \vec{n}_{\text{B}}^{\text{ver}}\rangle) == \text{sign}(\langle b_{\text{vis}}, \vec{n}_{\text{B}}^{\text{ver}}\rangle - \langle s, \vec{n}_{\text{B}}^{\text{ver}}\rangle) \Rightarrow p \text{ is vertically visible.} \end{aligned} \tag{3.4}$$

If p is horizontally and vertically visible it is considered visible.
Now the visibility of a perturbed ray is determined by the visibility of any point on the perturbed ray except for its source as it lies on both blocking planes. Therefore, the point $s + \vec{r}_{\text{dev}}$ is tested in our Monte Carlo ray tracers and if it is not visible the ray is considered to be blocked. This needs to be done for every generated ray in order to correctly account for tower blocking.

4 Convolution methods

In contrast to the Monte Carlo ray tracing methods, convolution based methods are using an analytical approach to compute the flux distribution on the receiver. This makes them deterministic meaning simulations with the same configuration will always produce the exact same result which is a crucial part when they are used for optimization purposes. Moreover, they require less rays to be evaluated for achieving the same accuracy as Monte Carlo based methods [16].

The idea behind convolution based ray tracing methods is to describe the flux density produced by a cell of a heliostat using a mathematical function. The resulting solar power at a receiver piece is then calculated by integrating the flux density function over a specific area representing the receiver piece [16]. Which flux density function is used and how the integral is evaluated is the main difference between the convolution based methods.

In the following, *HFLCAL* [41] a well known analytical flux density model and our adaptation will be discussed. However, even with the descriptions in [16, 41] the exact calculation steps of the model are not known. Therefore, the specified steps partly originate from our interpretation of the model. After that our Gaussian convolution ray tracer will be introduced which evaluates the integral far more accurately. Finally, our integrated convolution ray tracer is presented in Section 4.3.

4.1 Analytic image method HFLCAL

The general principle of *HFLCAL* as well as our Gaussian convolution method is to determine the probability P_{hit} that the current perfect reflected ray will be perturbed in such a way that it hits the current receiver piece. Indices to specify the current receiver piece are omitted to simplify notations, however, the evaluation process is done for every receiver piece. Once P_{hit} is known the incoming power P_{rec} at the receiver piece from the current heliostat can be calculated by

$$P_{\text{rec}} = P_{\text{hit}} \cdot P_{\text{ray}}, \tag{4.1}$$

where P_{ray} is the power of the perfect reflected ray, see Section 2.5.

Calculating P_{rec} for every receiver piece effectively reduces one main error source of every ray tracer being the evaluation of the error cone of a representative ray.

To determine the probability P_{hit} a more detailed look on how the perfect reflected ray gets perturbed is needed which is given in the following. Afterwards the representation of a receiver piece as well as the corresponding integration of the flux density

function will be discussed.

As described in Section 2.5, the perfect reflected ray gets disturbed in horizontal and vertical direction in means of two independent Gaussian distributions. Therefore, the deviation in both direction can be described by multiplying the two distributions leading to a bivariate Gaussian distribution with no correlation between the two directions. This results in the following probability density function

$$\begin{aligned} f(x,y) &= f^{\text{hor}}(x) \cdot f^{\text{ver}}(y) \\ &= \frac{1}{2\pi\sigma^{\text{hor}}_{\text{beam}}\sigma^{\text{ver}}_{\text{beam}}} \cdot \exp\left(-\frac{1}{2}\left(\left(\frac{x}{\sigma^{\text{hor}}_{\text{beam}}}\right)^2 + \left(\frac{y}{\sigma^{\text{ver}}_{\text{beam}}}\right)^2\right)\right), \end{aligned} \tag{4.2}$$

where x and y are the angles in horizontal and vertical direction between the perfect reflected ray and an perturbed ray.

Multiplying the probability density function of Equation (4.2) with the ray power P_{ray} gives the desired flux density function. The flux density function can be understood as a mathematical description of the error cone. *HFLCAL* originally uses a circular Gaussian distribution to describe the flux density so $\sigma^{\text{hor}}_{\text{beam}} = \sigma^{\text{ver}}_{\text{beam}}$ but as our model uses different deviations for both directions we adapted the definition [41].
Instead of describing the disturbance in angular space, i.e., with the angles x and y one can also refer to the resulting spanned region on a plane orthogonal to the ray direction at a certain distance δ to the ray origin. This plane is referred to as the ray image plane in the following. Figure 17 illustrates the transformation for a one dimensional Gaussian distribution and Equation (4.3) gives the required calculation.

$$\begin{aligned} \sigma^{\text{ver}}_{\text{span}} &= \delta \cdot \tan\left(\sigma^{\text{ver}}_{\text{beam}}\right) \\ \sigma^{\text{hor}}_{\text{span}} &= \delta \cdot \tan\left(\sigma^{\text{hor}}_{\text{beam}}\right) \end{aligned} \tag{4.3}$$

Technically, the ray image plane as well as the probability density function on that plane is infinite leading to an infinite error space rather than an error cone. However, Gaussian distributions have the property that 99.7% of their values lie within a 3σ region around their mean [2]. In case of the probability density function in Equation (4.2), this means that the chance for the representative ray to be perturbed in such a way that it intersects a $9 \cdot \sigma^{\text{hor}}_{\text{span}}\sigma^{\text{ver}}_{\text{span}}$ region around its unperturbed intersection with the ray image plane, is 99.4%. When referring to that region the error space is cut to an elliptical shaped error cone.

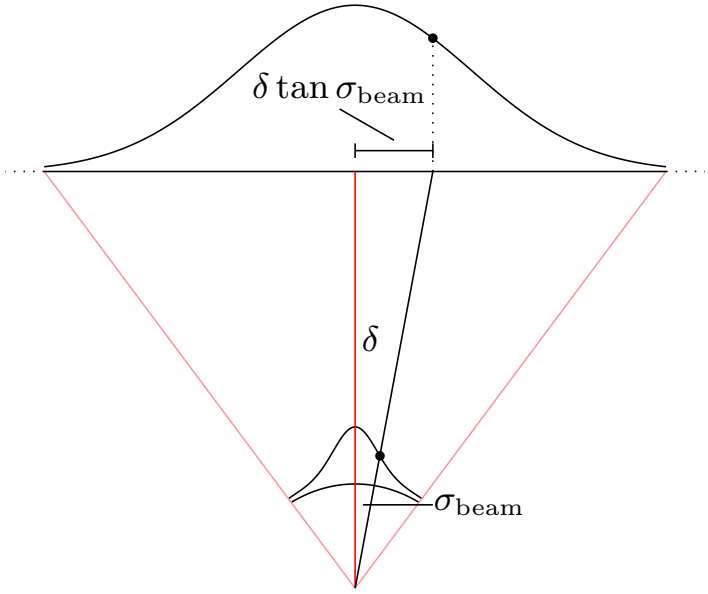


Figure 17: Transformation of the one dimensional Gaussian distribution of Section 2.4 from angle to spanned region.

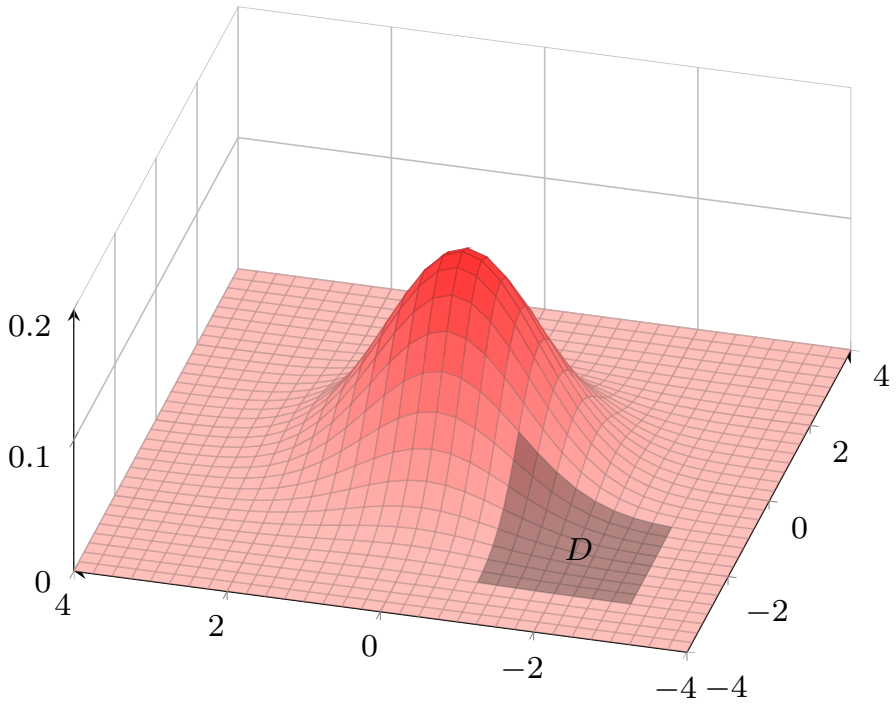


Figure 18: Bivariate Gaussian distribution of Equation (4.2) on the ray image plane with the distance δ and standard deviations $\sigma^{\text{ver}}_{\text{beam}}$, $\sigma^{\text{hor}}_{\text{beam}}$ being exemplary set such that $\sigma^{\text{ver}}_{\text{span}} = \sigma^{\text{hor}}_{\text{span}} \approx 1$ m [33]. D marks a possible region to integrate over.

The bivariate Gaussian distribution on the ray image plane at an exemplary distance of $\delta = 500$ m with standard deviations $\sigma^{\text{ver}}_{\text{beam}} = \sigma^{\text{hor}}_{\text{beam}} = 2$ mrad resulting in $\sigma^{\text{ver}}_{\text{span}} = \sigma^{\text{hor}}_{\text{span}} \approx 1$ m is shown in Figure 18. The integral over a region like D marked in gray then gives the probability $P_{\text{int}}(D)$ that the representative ray will be perturbed so that it intersects this region leading to

$$P_{\text{int}}(D) = \iint_D f(x, y)\ dA, \tag{4.4}$$

with $f(x, y)$ as in Equation (4.2).

Now there are two steps left in order to compute P_{hit}. First the region D for which rays intersecting this region would hit the currently viewed receiver piece has to be computed. Afterwards, an evaluation of the integral in Equation (4.4) is needed since it holds that $P_{\text{hit}} = P_{\text{int}}(D)$.
To get a better understanding of how *HFLCAL* solves these issues a simplified example where the receiver piece lies in the ray image plane is introduced as shown in Figure 19.

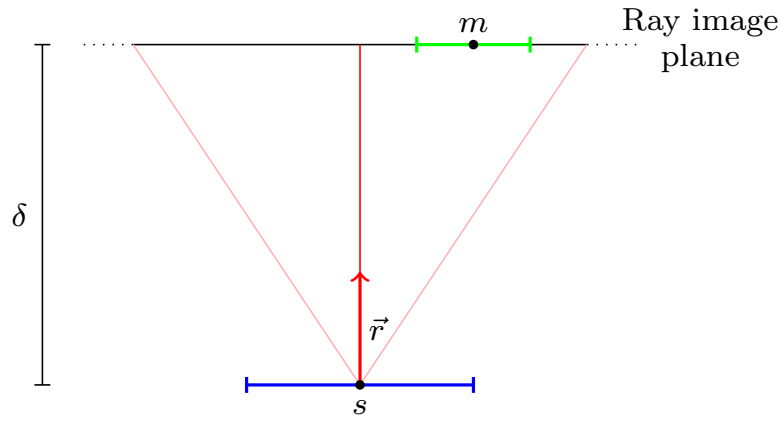


Figure 19: Simplified example where the receiver pieces marked in green lies in the ray image plane.

In this example the region D is already well defined by the area A_{rec} of the receiver piece. To simplify the integral calculation a numerical approach is utilized by *HFLCAL*. Instead of an exact integration the bivariate normal distribution is evaluated at the midpoint m of the receiver piece which gives the relative likelihood L_{hit} to intersect that point. By assuming D to have the same likelihood over its whole area the calculation of P_{hit} is reduced to

$$P_{\text{hit}} = \iint_D f(x, y)\ dA \approx \iint_D f(m'_x, m'_y)\ dA = \iint_D L_{\text{hit}}\ dA = L_{\text{hit}} \cdot A_{\text{rec}}, \tag{4.5}$$

where $(m'_x, m'_y)^T$ are the local coordinates of the midpoint m on the ray image plane.

More formally, the integral is evaluated using a two dimensional midpoint quadrature rule [20]. In order to reduce the error introduced by the assumption in Equation (4.5), *HFLCAL* relies on a fine discretization of the receiver leading to small areas for each receiver piece [16].

So for this example, a problem remaining is to compute the local coordinates $(m'_x, m'_y)^T$. Figure 20 illustrates the situation from the side view, i.e., the viewing direction is parallel to the horizontal direction of the ray image plane. However, an analog figure can be used when viewing parallel to the vertical direction. As shown the local coordinates can be calculated by projecting the vector from the ray origin s to the midpoint m on either of the orthonormal vectors $\vec{r}^{\text{ver}}$, $\vec{r}^{\text{hor}}$ defining the coordinate system of the ray image plane leading to

$$m'_x = \langle \vec{r}^{\text{hor}}, m - s \rangle, \;\; m'_y = \langle \vec{r}^{\text{ver}}, m - s \rangle. \tag{4.6}$$

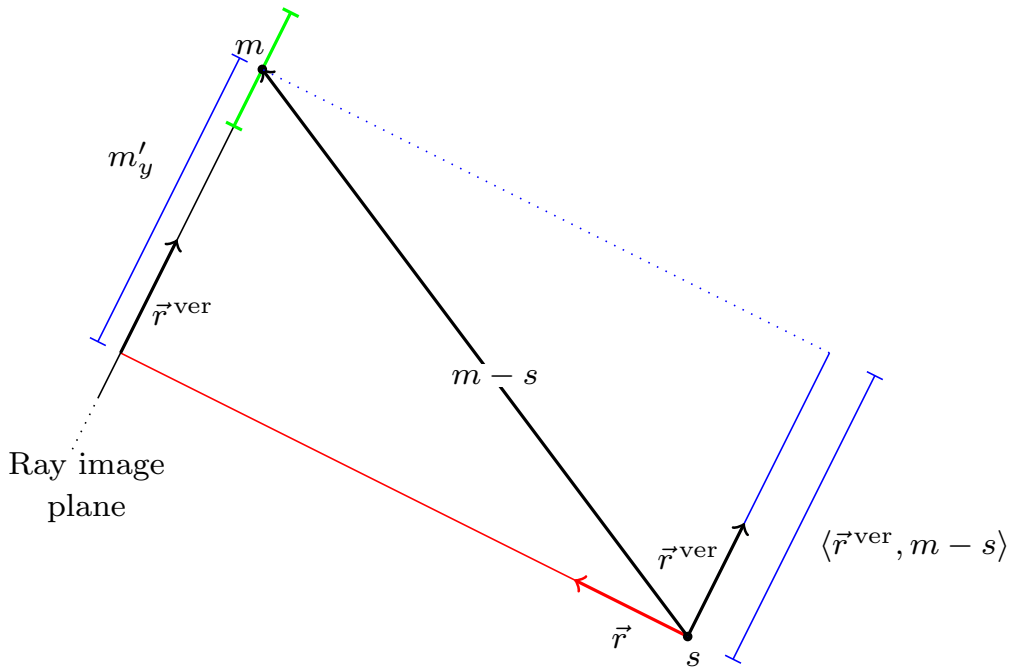


Figure 20: Illustration of the required calculations to compute the local coordinate m'_y of the midpoint m of a receiver piece marked in green. Analogously the local coordinate m'_x can be computed.

As mentioned earlier the transformation in Equation (4.3) which is needed to evaluate $f(m'_x, m'_y)$ also involves the distance δ from ray source to the ray image plane. But as the local coordinates m'_x, m'_y are already known δ can be calculated using the Pythagoras theorem

$$\delta = \left(\sqrt{|m-s|^2 + |(m'_x, m'_y)^T|^2} \right)^2 \tag{4.7}$$

$$= \left(\sqrt{|m-s|^2 + {m'_x}^2 + {m'_y}^2} \right)^2. \tag{4.8}$$

With all that the probability P_{hit} and thus the incoming solar power at the current receiver piece for the simplified example can be calculated. But in the general case the receiver piece does not perfectly lie in the ray image plane. *HFLCAL* addresses this issue by adapting the standard deviation of the flux density function to

$$\sigma_{\text{span,}} = \frac{\sigma_{\text{span}}}{\sqrt{\cos(\theta_{\text{rec}})}}, \tag{4.9}$$

with θ_{rec} as the angle between the vector of the representative ray and the normal of the receiver piece.

The idea behind this approach is that for large incidence angles the flux density gets stretched over the receiver piece area. Therefore, the standard deviation has to increase with the angle of incidence. A larger angle results in a smaller cosine of that angle and therefore in an increased standard deviation. Thus, the flux density function $FD(x, y)$ of *HFLCAL* for a heliostat cell is

$$\begin{aligned} FD(x,y) &= P_{\text{ray}} \cdot f(x,y) \\ &= \frac{P_{\text{ray}}}{2\pi(\sigma_{\text{span,}})^2} \cdot \exp\left(-\frac{x^2+y^2}{2 \cdot (\sigma_{\text{span,}})^2} \right), \end{aligned} \tag{4.10}$$

with $\sigma_{\text{span,}}$ as in Equation (4.9).

In our adaptation of the *HFLCAL* model the general case is handled a bit differently. As we already assume the same relative likelihood and therefore the same flux density over the receiver cell area, a slanted incidence can be modeled by reducing the receiver cell area with a cosine effect as in Section 2.4. Therefore, the calculation of P_{hit} in Equation (4.5) is adapted to the effective area by

$$P_{\text{hit}} \approx L_{\text{hit}} \cdot A_{\text{rec}} \cdot \cos(\theta_{\text{rec}}). \tag{4.11}$$

Doing this keeps the probability density function on the ray image plane unchanged.

However, as already mentioned *HFLCAL* requires small receiver pieces in order to obtain accurate results. Therefore, the flux density function needs to be evaluated a lot

of times even for a single representative ray. To reduce the introduced computational effort, our Gaussian convolution method calculates an exact representation of a receiver piece and utilizes a far more accurate integration method. By doing this the accuracy of the results is independent from the number of receiver pieces.

4.2 Gaussian convolution method

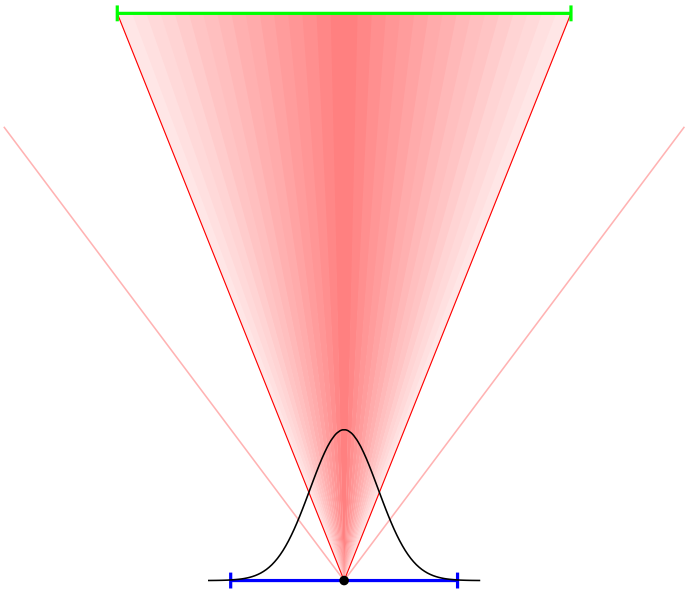


Figure 21: Basic principle of the Gaussian convolution method. The shaded red region illustrates an exact evaluation of the error cone for the current receiver piece.

In extend to the *HFLCAL* model our Gaussian convolution method aims to evaluate the probability P_{hit} as described in the last section with higher precision. For this, the two steps of computing the region representing a receiver piece and evaluating the integral of Equation (4.4) need to be done in more detail. When both steps are performed accurately the evaluation of the error cone is exact. Figure 21 shows the basic idea of the Gaussian convolution method.

4.2.1 Computation of a region representing a receiver piece

As a reminder, the region representing a receiver piece can be in either the angular space or on the ray image plane referred to as D_{ang} and D_{span}, respectively. In the angular space this region gives the required combinations of angles in horizontal and in vertical direction to the perfect reflected vector $\vec{r}$ for which rays having any of these combinations would hit the receiver piece.

Since the corners coordinates c_1, c_2, c_3 and c_4 of the receiver piece are known, a straight forward approach would be to compute the horizontal and vertical angles α_l and β_l for each corner and use them as vertex coordinates of a polygon to describe D_{ang}. The angles are obtained by first projecting the vector from the ray origin s to the corresponding corner c_l onto the horizontal and vertical deviation planes which are orthogonal to $\vec{r}^{\text{ver}}$ and $\vec{r}^{\text{hor}}$, respectively. An exemplary angular region in two dimensions reaching from α_1 to α_2 is shown Figure 22.

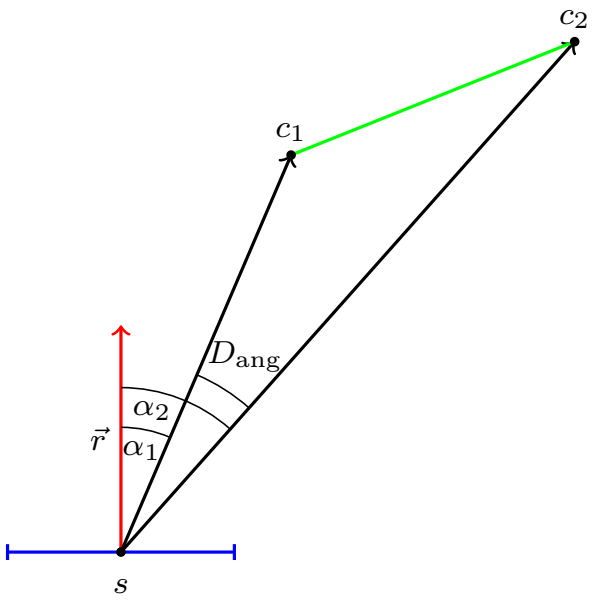


Figure 22: Two dimensional version of the angular region D_{ang} defined by the angles α_1 and α_2.

Figure 23 illustrates the required projection for the corner c_1. The necessary calculations to obtain the projected points c_l^{ver} and c_l^{hor} on the planes orthogonal to either $\vec{r}^{\text{hor}}$ or $\vec{r}^{\text{ver}}$ are as follows

$$c_l^{\text{ver}} = c_l - \vec{r}^{\text{hor}}\langle c_l - s, \vec{r}^{\text{hor}}\rangle,\ c_l^{\text{hor}} = c_l - \vec{r}^{\text{ver}}\langle c_l - s, \vec{r}^{\text{ver}}\rangle,\ l \in \{1,2,3,4\}. \tag{4.12}$$

With this the angles α_l and β_l for the corner c_l are given by

$$\alpha_l = \arccos\langle \frac{c_l^{\text{hor}} - s}{|c_l^{\text{hor}} - s|}, \vec{r}\rangle,\ \beta_l = \arccos\langle \frac{c_l^{\text{ver}} - s}{|c_l^{\text{ver}} - s|}, \vec{r}\rangle, \tag{4.13}$$

where $\vec{r}$ is assumed to be normalized.

However, there is still one problem left for special cases. Figure 24 shows such a case with the viewing direction being parallel to the vertical direction $\vec{r}^{\text{ver}}$. As one can see

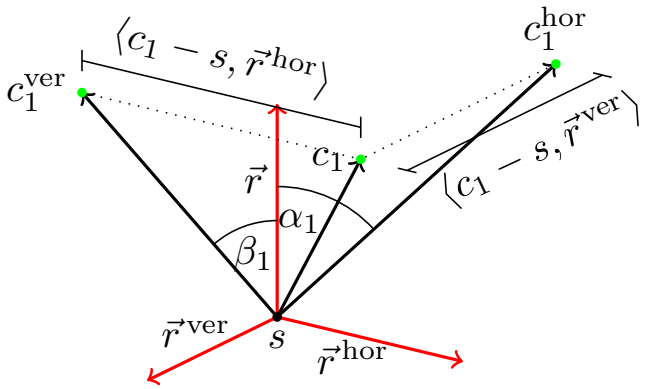


Figure 23: Required projection to obtain the angles α_1 and β_1 of a

the angles α_1 and α_2 are equal. Therefore, an integral from α_1 to α_2 over any function would always be zero. To solve this issue the angles need to receive a sign. Any angle of a projected point differing in the direction of $\vec{r}^{\text{hor}}$ or $\vec{r}^{\text{ver}}$ gets a positive sign and otherwise a negative one. Note that the definition of positive and negative angles can also be switched as the Gaussian distributions are axis-symmetric but we decided to stick to the more intuitive definition. In the example, the sign of the angle is the same as the sign of $\langle c_l - s, \vec{r}^{\text{hor}} \rangle$ with $l \in \{1, 2\}$. Thus, Equation (4.13) is adapted to

$$\begin{aligned} \alpha_l &= \text{sign}(\langle c_l - s, \vec{r}^{\text{hor}} \rangle) \cdot \arccos \langle \frac{c_l^{\text{hor}} - s}{|c_l^{\text{hor}} - s|}, \vec{r} \rangle, \\ \beta_l &= \text{sign}(\langle c_l - s, \vec{r}^{\text{ver}} \rangle) \cdot \arccos \langle \frac{c_l^{\text{ver}} - s}{|c_l^{\text{ver}} - s|}, \vec{r} \rangle, \end{aligned} \tag{4.14}$$

with $\text{sign}(\cdot)$ as the signum function.

The dot products used to determine the sign of the angles α_l and β_l have already been computed in the calculation of c_l^{ver} and c_l^{hor}. By reusing the results, the main computational costs to determine the angles α_l and β_l can be reduced to four dot products and two normalizations. The angles α_l and β_l of each corner are then used as vertex coordinates to define the region D_{ang}

$$\begin{aligned} D_{\text{ang}} = (V, E),\ V = \{a, b, c, d\} = \left\{ \begin{pmatrix} \alpha_1 \\ \beta_1 \end{pmatrix}, \begin{pmatrix} \alpha_2 \\ \beta_2 \end{pmatrix}, \begin{pmatrix} \alpha_3 \\ \beta_3 \end{pmatrix}, \begin{pmatrix} \alpha_4 \\ \beta_4 \end{pmatrix} \right\}, \\ E = \{\{a, b\}, \{b, c\}, \{c, d\}, \{d, a\}\}. \end{aligned} \tag{4.15}$$

However, the computational expense can be reduced even further by computing the region D_{span} on the ray image plane using concepts from the field of computer graphics.

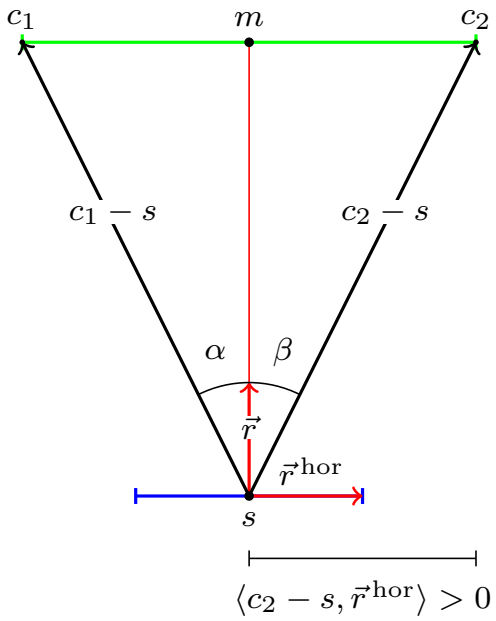


Figure 24: Example to illustrate the need of signed angles as the angles α_1 and α_2 are equal even though they refer to different corners.

In contrast to the calculations in Section 4.1 to compute the local coordinates $(m'_x m'_y)^T$ of m where the distance to the ray image plane varies between different midpoints, now every corner of every receiver piece has to be projected onto the same ray image plane. An illustration of a region D_{span} is shown in Figure 25.

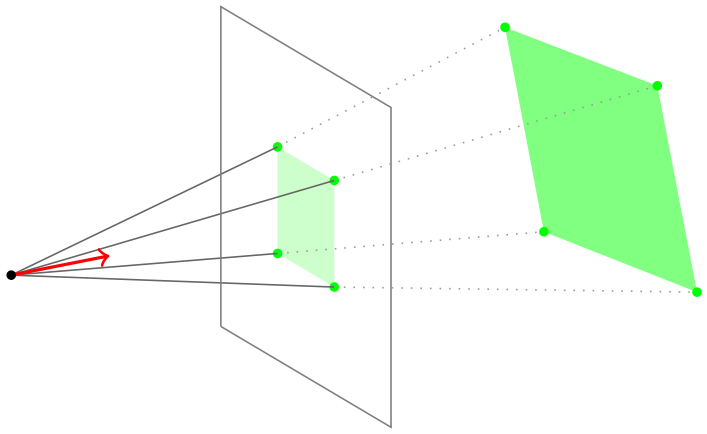


Figure 25: Exemplary representative region D_{span} illustrated in light green on the ray image plane of a receiver piece shown in green.

Another way to think of the region D_{span} is as the image of the receiver piece one would see when looking from the ray source in the direction of the representative ray.

In computer graphics the human eye would be replaced by a conceptional camera with some focal distance δ being equal to the distance to the image plane [38].

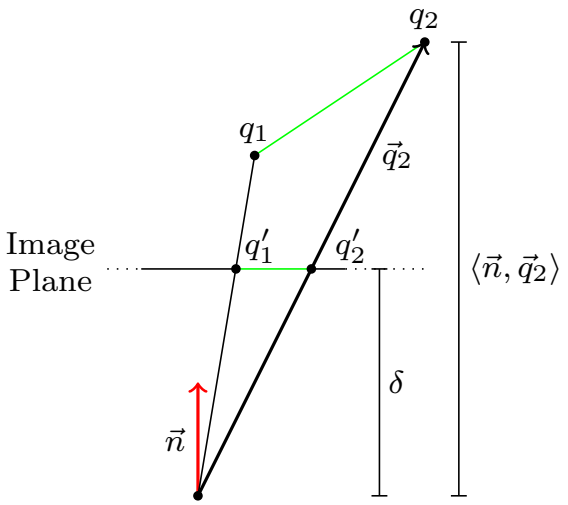


Figure 26: Example of the perspective projection problem with the center of projection lying in the origin. The vector $\vec{n}$ is the normal of the image plane onto which the point q_1 and q_2 are to be projected.

The problem in computer graphics is to compute the projected points q_1' and q_2' of q_1 and q_2 on the image plane orthogonal to $\vec{n}$ at a distance δ as illustrated in Figure 26. This conceptional equality between our problem and the problem in computer graphics is the reason why we refer to the plane orthogonal to the representative ray as the ray image plane. Using the intercept theorem [38], the projection q' of point q can be calculated by

$$q' = \frac{\delta}{\langle \vec{n}, \vec{q} \rangle} q, \tag{4.16}$$

with $\vec{q}$ as the vector from the origin to point q. This projection assumes the camera position, also referred to as the center of projection, to lie in the origin.

The required transformation to obtain the projected points is called perspective projection [38]. To accelerate the computation of transformations in general, a goal in computer graphics is to be able to express them using a matrix [38]. By doing this any combination of transformations can be described as a multiplication of the transformation matrices leading to a single matrix. However, projective transformations like the perspective projection in Equation (4.16) require perspective division [38], i.e., the divisor depends on the point that has to be projected. This operation is non-linear and therefore not expressible by a matrix. But there is a workaround for the problem by using the convention of homogeneous coordinates [38]. The idea is to extend into four

dimensions and use the additional dimension to express perspective division. A point p_{hom} in homogeneous coordinates is transfered to Cartesian coordinates in a process called de-homogenization which divides the first three components by the fourth.

$$p_{\text{hom}} = \begin{pmatrix} p_x \\ p_y \\ p_z \\ w \end{pmatrix} \xrightarrow{\text{de-hom}} \begin{pmatrix} \frac{p_x}{w} \\ \frac{p_y}{w} \\ \frac{p_z}{w} \end{pmatrix} = p_{\text{cart}} \tag{4.17}$$

With this convention the perspective projection in Equation (4.16) can be written as

$$q'_{\text{hom}} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ \frac{n_x}{\delta} & \frac{n_y}{\delta} & \frac{n_z}{\delta} & 0 \end{pmatrix} \cdot \begin{pmatrix} q_x \\ q_y \\ q_z \\ 1 \end{pmatrix} = M_{\text{proj}} \cdot q_{\text{hom}}. \tag{4.18}$$

As stated earlier, the projection is only correct if the camera lies in the origin. Getting back to our problem, this is generally not the case for the ray source. Therefore, the points need to be translated such that the ray source s is equal to the origin and after the projection back again. The following two matrices describe such translations

$$M_{\text{t1}} = \begin{pmatrix} 1 & 0 & 0 & -s_x \\ 0 & 1 & 0 & -s_y \\ 0 & 0 & 1 & -s_z \\ 0 & 0 & 0 & 1 \end{pmatrix}, \; M_{\text{t2}} = \begin{pmatrix} 1 & 0 & 0 & s_x \\ 0 & 1 & 0 & s_y \\ 0 & 0 & 1 & s_z \\ 0 & 0 & 0 & 1 \end{pmatrix}. \tag{4.19}$$

With these transformations the complete projection of a corner c_l onto the ray image plane is given by

$$c'_l = M_1 \cdot c_l = M_{\text{t2}} \; M_{\text{proj}} \; M_{\text{t1}} \cdot c_l \tag{4.20}$$

Thus, a single matrix multiplication is needed to project the corners. A question remaining for our problem is at which distance the ray image plane should be placed. If $\sigma^{\text{hor}}_{\text{span}}$ and $\sigma^{\text{ver}}_{\text{span}}$ are adapted to the distance, the placement can be chosen arbitrary since for all distances a valid representative region of the receiver piece exists. To reduce computational effort we use a distance of $\delta = 1$ in our Gaussian convolution method.

Unfortunately, the projected corners are still given in global coordinates and not in local coordinates on the ray image plane as required. As origin of the local coordinate system the intersection point m_{ray} of the representative ray with the ray image plane is used. In computer graphics this problem is solved by the viewport transformation [38].

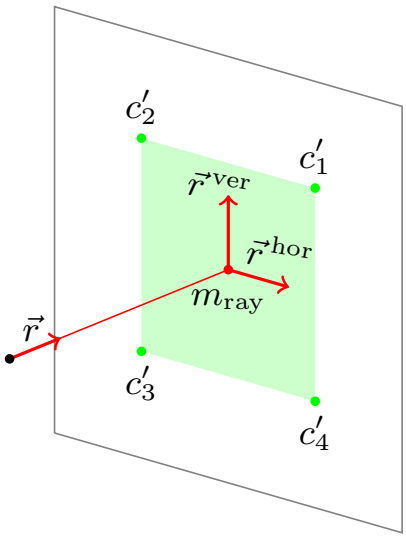


Figure 27: Problem of calculating local coordinates of the receiver corners on the ray image plane. The coordinate system is defined by m_{ray}, $\vec{r}^{\,\mathrm{hor}}$ and $\vec{r}^{\,\mathrm{ver}}$.

An example of this adapted to our problem is shown in Figure 27.

So the task is to find x_l and y_l with $c_l' = m_{\mathrm{ray}} + x_l \cdot \vec{r}^{\,\mathrm{hor}} + y_l \cdot \vec{r}^{\,\mathrm{ver}}$. As the vectors $\vec{r}^{\,\mathrm{hor}}$ and $\vec{r}^{\,\mathrm{ver}}$ are orthonormal, the coordinates can be calculated as follows

$$\begin{aligned} \langle \vec{r}^{\,\mathrm{hor}}, c_l' \rangle &= \langle \vec{r}^{\,\mathrm{hor}}, m_{\mathrm{ray}} \rangle + x_l \cdot 1 + 0 \;\Rightarrow\; x_l = \langle \vec{r}^{\,\mathrm{hor}}, c_l' \rangle - \langle \vec{r}^{\,\mathrm{hor}}, m_{\mathrm{ray}} \rangle, \\ \langle \vec{r}^{\,\mathrm{ver}}, c_l' \rangle &= \langle \vec{r}^{\,\mathrm{ver}}, m_{\mathrm{ray}} \rangle + 0 + y_l \cdot 1 \;\Rightarrow\; y_l = \langle \vec{r}^{\,\mathrm{ver}}, c_l' \rangle - \langle \vec{r}^{\,\mathrm{ver}}, m_{\mathrm{ray}} \rangle. \end{aligned} \tag{4.21}$$

The corresponding matrix for this transformation is [38]

$$M_{\mathrm{view}} = \begin{pmatrix} \vec{r}_x^{\,\mathrm{hor}} & \vec{r}_y^{\,\mathrm{hor}} & \vec{r}_z^{\,\mathrm{hor}} & -\langle \vec{r}^{\,\mathrm{hor}}, m_{\mathrm{ray}} \rangle \\ \vec{r}_x^{\,\mathrm{ver}} & \vec{r}_y^{\,\mathrm{ver}} & \vec{r}_z^{\,\mathrm{ver}} & -\langle \vec{r}^{\,\mathrm{ver}}, m_{\mathrm{ray}} \rangle \\ 0 & 0 & 0 & 1 \end{pmatrix}. \tag{4.22}$$

Therefore, the computation of the local coordinates x_l and y_l of the projected corner c_l can be summarized to

$$M_{\mathrm{view}}\ M_{\mathrm{t2}}\ M_{\mathrm{proj}}\ M_{\mathrm{t1}} \cdot c_l = M \cdot c_l = \begin{pmatrix} x_l w \\ y_l w \\ w \end{pmatrix} \xrightarrow{de-hom} \begin{pmatrix} x_l \\ y_l \end{pmatrix}. \tag{4.23}$$

Note that M can be precomputed and reused for every corner of every receiver piece when evaluating a representative ray. After precomputing M, the coordinates x_l and y_l can be calculated with a single matrix multiplication which comes down to the evaluation of three dot products.

Similar to the definition of the region D_{ang} in Equation (4.15), the coordinates x_l and y_l are computed for every corner c_l and used as vertex coordinates of a polygon

$$\begin{aligned} D_{\text{span}} = (V, E),\ V = \{a, b, c, d\} = \left\{ \begin{pmatrix} x_1 \\ y_1 \end{pmatrix}, \begin{pmatrix} x_2 \\ y_2 \end{pmatrix}, \begin{pmatrix} x_3 \\ y_3 \end{pmatrix}, \begin{pmatrix} x_4 \\ y_4 \end{pmatrix} \right\}, \\ E = \{\{a, b\}, \{b, c\}, \{c, d\}, \{d, a\}\}. \end{aligned} \tag{4.24}$$

Summing up, two methods have been introduced to calculate either of the regions D_{ang} and D_{span}. But by using concepts from the field of computer graphics the derived method to compute D_{span} has shown to be less computationally expensive than the method to compute D_{ang}. For this reason our Gaussian convolution ray tracer uses D_{span} as a representative region of a receiver piece. However, both methods can be used to compute a valid representation of a receiver piece.

Now that the representative region is well defined by a polygon the next problem is to evaluate the integral of the bivariate Gaussian distribution in Equation (4.4) over that polygon.

4.2.2 Integration of the bivariate Gaussian distribution over a polygon

Before getting into details on how to calculate the integral of the bivariate Gaussian distribution over a polygon the problem is recapped. Given a polygon D representing a receiver piece, the problem is to evaluate the following integral

$$\begin{aligned} P_{\text{int}}(D) &= \iint_D f(x, y)\ dA \\ &= \iint_D \frac{1}{2\pi\sigma_{\text{x}}\sigma_{\text{y}}} \cdot \exp\left(-\frac{1}{2} \left(\frac{x^2}{\sigma_{\text{x}}^2} + \frac{y^2}{\sigma_{\text{y}}^2} \right) \right) dxdy, \\ (\sigma_x, \sigma_y) &= \begin{cases} (\sigma_{\text{beam}}^{\text{hor}}, \sigma_{\text{beam}}^{\text{ver}}), & \text{if } D = D_{\text{ang}} \\ (\sigma_{\text{span}}^{\text{hor}}, \sigma_{\text{span}}^{\text{ver}}), & \text{if } D = D_{\text{span}}. \end{cases} \end{aligned} \tag{4.25}$$

The probability $P_{\text{int}}(D)$ is equal to the probability P_{hit} that the representative ray will be perturbed in such a way that it hits the receiver piece represented by D. To solve the integral in Equation (4.25) a method [1] developed in the late 70s is used, a time where computational efficiency was even more important. Their main goal was to derive an efficient algorithm that is also accurate up to a desired degree. The described method works for convex polygons and was extend by [17] to also handle arbitrary polygons efficiently. It utilizes a numerical approach to solve the integral with a proven [1] accuracy of up to 12 decimal digits. In the following, the method in [1] adapted to our problem will be described. Most of the derivations and ideas are from [1] and for further details we refer to the original paper.

Instead of evaluating the integral over the region D the algorithm evaluates the integral over the complementary region $\overline{D}$, i.e., over the outer region of the polygon. As the bivariate Gaussian distribution $f(x, y)$ is a probability density function it holds that

$$\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(x,y)\ dxdy = 1 \Rightarrow P_{\text{int}}(D) = 1 - P_{\text{int}}(\overline{D}). \tag{4.26}$$

To compute $P_{\text{int}}(\overline{D})$ the algorithm divides the region $\overline{D}$ into angular regions A_i as illustrated in Figure 28 and relies on a fast method to evaluate the integral over those regions. For each vertex V_i of the Polygon a corresponding angular region A_i is introduced.

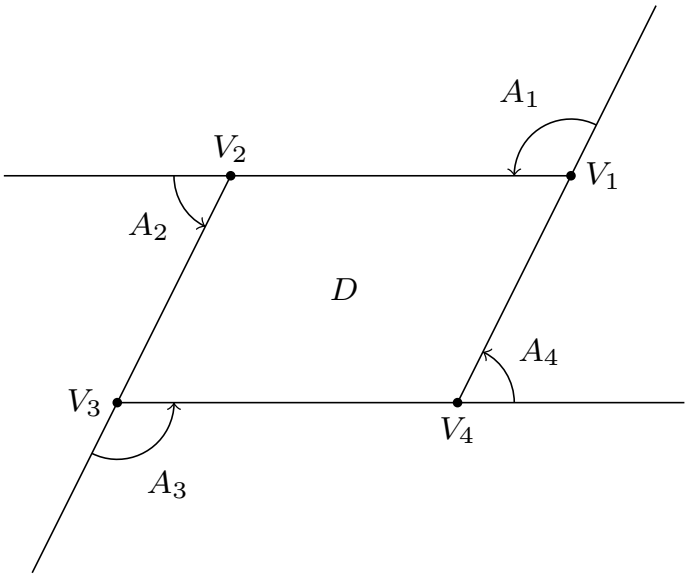


Figure 28: Angular regions of a polygon with four vertices. The figure is derived from [1].

The angular regions A_i are defined as the semi-infinite part bounded by two intersecting lines [1] as shown in Figure 28. Before calculating the integral over these regions the integrand in Equation (4.25) is reduced by substitution to

$$P_{\text{int}}(D) = P'_{\text{int}}(D') = \frac{1}{2\pi}\iint_{D'} \exp\left(-\frac{u^2+v^2}{2}\right) dudv,\ u = \frac{x}{\sigma_x},\ v = \frac{y}{\sigma_y}. \tag{4.27}$$

Note that the polygon D' also has to be given in coordinates of u and v and thus is obtained by using the substitution in Equation (4.27). This transformation can be described by the following matrix

$$M_{\text{sub}} = \begin{pmatrix} \frac{1}{\sigma_x} & 0 & 0 \\ 0 & \frac{1}{\sigma_y} & 0 \\ 0 & 0 & 1 \end{pmatrix}. \tag{4.28}$$

If D was calculated on the ray image plane, as described earlier, M_{sub} can be multiplied to the matrix M of Equation (4.23) and the resulting matrix used to directly calculate the polygon D' in coordinates of u and v.

After the substitution the bivariate Gaussian distribution in Equation (4.27) is circular symmetric. This property is required in order to calculate the integral over an angular region A_i. Using the integration, Equation (4.27) can be written as

$$P'_{\text{int}}(\overline{D}') = \sum_{i=1}^{n} P'(A'_i) = \sum_{i=1}^{n} \frac{1}{2\pi} \iint_{A'_i} \exp\left(-\frac{u^2+v^2}{2}\right) dudv. \tag{4.29}$$

Due to the circular symmetry, a rotation of the axes can be performed without changing the result. Therefore, the axes can be rotated such that the line L of an angular Region as shown in Figure 29 coincide with the positive u-Axis [1]. The line L is defined by the Vertex V and the origin. Its corresponding angular region is described using the distance R from the origin to the vertex V and the angles θ_1 and θ_2. Figure 30 illustrates the resulting angular region. From now on the axes are assumed to be rotated but are still referred to as u and v. Note that these rotations are not actually required when evaluating $P'(A'_i)$ but rather used to simplify the steps necessary to solve the integral.

By introducing polar coordinates centered at the vertex $V_i = (0, R)$ the integral limits in Equation (4.29) can be defined with the angles θ_1 and θ_2. The required transformations are

$$u = R + r\cos\theta,\ v = r\sin\theta,\ dudv = rdrd\theta,\quad -\pi \leq \theta \leq \pi. \tag{4.30}$$

Using this transformation in Equation (4.29) results in the following

$$\begin{aligned} P'(A'_i) &= \frac{1}{2\pi} \int_{\theta_1}^{\theta_2} \int_0^{\infty} \exp\left[-\frac{1}{2}\left(R^2 + 2rR\cos\theta + r^2\right)\right] rdrd\theta \\ &= \frac{1}{2\pi} e^{-\frac{R^2}{2}} \int_{\theta_1}^{\theta_2} \int_0^{\infty} re^{-\frac{r^2}{2}} e^{-pr}\, drd\theta,\ p = R\cos\theta. \end{aligned} \tag{4.31}$$

In our code the angles θ_1 and θ_2 are calculated from the Line L which intersects the origin and the vertex V_i and are positive in counter-clockwise direction. Thus, angle

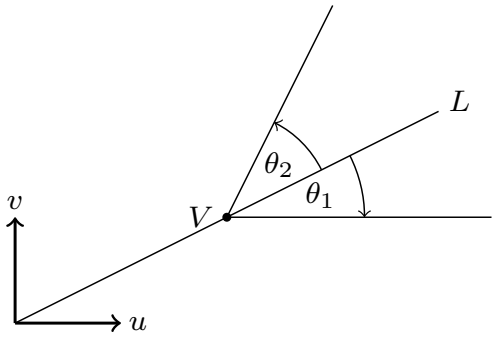


Figure 29: Example of a general angular region before the rotation derived from [1].

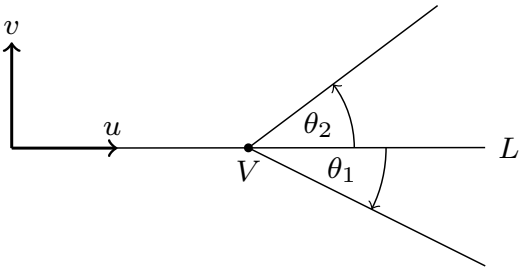


Figure 30: Rotated angular region which is defined by the length R and the angles θ_1 and θ_2 derived from [1].

θ_2 in Figure 30 would have a positive sign and θ_1 a negative one. The sign is determined by the sign of the corresponding determinant. Therefore, the angles are given by

$$\begin{aligned} \theta_1 &= \operatorname{sign}(\det(\vec{v}_1, \vec{l}))\frac{\arccos\langle \vec{v}_1, \vec{l}\rangle}{|\vec{v}_1| \cdot |\vec{l}|}, \\ \theta_2 &= \operatorname{sign}(\det(\vec{v}_2, \vec{l}))\frac{\arccos\langle \vec{v}_2, \vec{l}\rangle}{|\vec{v}_2| \cdot |\vec{l}|}, \end{aligned} \tag{4.32}$$

with $\vec{v}_1$ as the vector from the predecessor of vertex V_i to V_i, $\vec{v}_2$ as the vector from V_i to the successor of V_i, $\vec{l}$ as the vector from the origin to V_i and $\det(\cdot, \cdot)$ as the determinant of two vectors.

Figure 31 illustrates the situation for an arbitrary vertex V_i. By this definition the polygon is required to be given in counter-clockwise order.

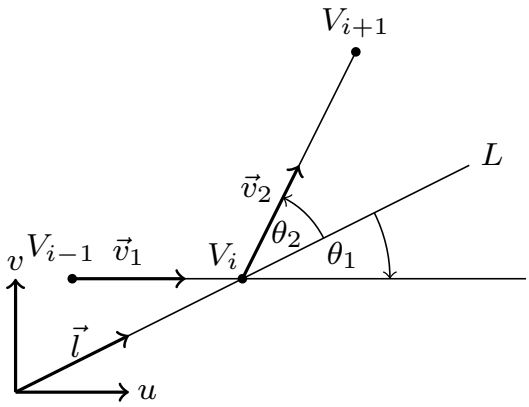


Figure 31: Illustration for the calculation of the angles θ_1 and θ_2.

To solve the integral in Equation (4.31) an integration by parts on the inner integral as well as a substitution is used

$$\begin{aligned}
\int_0^\infty r e^{-\frac{r^2}{2}} e^{-pr}\, dr &= \left[-e^{-\frac{r^2}{2}} e^{-pr}\right]_0^\infty - \int_0^\infty -e^{-\frac{r^2}{2}} \cdot -pe^{-pr}\, dr \\
&= 0 + 1 - p \int_0^\infty e^{-\frac{r^2}{2}} e^{-pr}\, dr \\
&= 1 - p \int_0^\infty e^{-\frac{1}{2}(r^2+2pr+p^2-p^2)}\, dr \\
&= 1 - pe^{\frac{1}{2}p^2} \int_0^\infty e^{-\frac{1}{2}(r+p)^2}\, dr \\
&= 1 - pe^{\frac{1}{2}p^2} \int_0^\infty e^{-(\frac{r}{\sqrt{2}}+\frac{p}{\sqrt{2}})^2} \frac{1}{\sqrt{2}}\sqrt{2}\, dr \\
&= 1 - pe^{\frac{1}{2}p^2}\sqrt{2} \int_{\frac{p}{\sqrt{2}}}^\infty e^{-t^2}\, dt,\ t = \frac{r+p}{\sqrt{2}}.
\end{aligned} \tag{4.33}$$

The remaining integral in Equation (4.33) correlates with the complementary error function being defined as

$$\operatorname{erfc}(x) = \frac{2}{\sqrt{\pi}} \int_x^\infty e^{t^2}\, dt. \tag{4.34}$$

Inserting this into the equation leads to

$$
\begin{aligned}
\int_0^\infty r e^{-\frac{r^2}{2}} e^{-pr}\, dr &= 1 - p e^{\frac{1}{2}p^2}\sqrt{2}\frac{\sqrt{\pi}}{2}\frac{2}{\sqrt{\pi}}\int_{\frac{p}{\sqrt{2}}}^\infty e^{-t^2}\, dt \\
&= 1 - p e^{\frac{1}{2}p^2}\sqrt{2}\frac{\sqrt{\pi}}{2}\mathrm{erfc}\left(\frac{p}{\sqrt{2}}\right) \\
&= 1 - p\sqrt{2}\frac{1}{e^{-\frac{1}{2}p^2}\frac{2}{\sqrt{\pi}}}\mathrm{erfc}\left(\frac{p}{\sqrt{2}}\right) \\
&= 1 - 2w\frac{\mathrm{erfc}(w)}{\mathrm{z}(w)},\ w = \frac{p}{\sqrt{2}} = \frac{R}{\sqrt{2}}\cos\theta,\ \mathrm{z}(w) = \frac{2}{\sqrt{\pi}}e^{-w^2}.
\end{aligned}
\tag{4.35}
$$

This reduces the integral in Equation (4.31) to

$$
\begin{aligned}
P'(A_i') &= \frac{1}{2\pi}e^{-\frac{R^2}{2}}\int_{\theta_1}^{\theta_2}\int_0^\infty r e^{-\frac{r^2}{2}} e^{-pr}\, dr d\theta \\
&= \frac{1}{2\pi}e^{-\frac{R^2}{2}}\left((\theta_2 - \theta_1) - \int_{\theta_1}^{\theta_2} 2w\frac{\mathrm{erfc}(w)}{\mathrm{z}(w)}d\theta\right) \\
&= e^{-\frac{R^2}{2}}\left(\frac{\theta_2 - \theta_1}{2\pi} - \frac{1}{\pi}\int_{\theta_1}^{\theta_2} w\frac{\mathrm{erfc}(w)}{\mathrm{z}(w)}d\theta\right).
\end{aligned}
\tag{4.36}
$$

The issue of evaluating the integral in Equation (4.36) is resolved by using a minmax polynomial fit to $\frac{\mathrm{erfc}(w)}{\mathrm{z}(w)}$, i.e. finding a polynomial of certain degree which has the smallest maximal deviation to the original function. In particular this means to find a set of real numbers a_k for a given $\delta{>}0$ and a least positive integer K such that

$$
\begin{aligned}
&\left|\frac{\mathrm{erfc}(w)}{\mathrm{z}(w)} - \sum_{k=1}^K a_k w^k\right| \le \delta,\ 0 \le w \le c(\delta) \\
\Rightarrow &\left|\mathrm{erfc}(w) - \mathrm{z}(w)\sum_{k=1}^K a_k w^k\right| \le \frac{2}{\sqrt{\pi}}e^{-w^2}\delta \underset{e^{-w^2}\le 1}{\le} \frac{2}{\sqrt{\pi}}\delta,\ 0 \le w \le c(\delta).
\end{aligned}
\tag{4.37}
$$

In [1] it is proven that if the inequality in Equation (4.37) holds then

$$
\left|e^{-\frac{R^2}{2}}\int_{\theta_1}^{\theta_2}\left(w\frac{\mathrm{erfc}(w)}{\mathrm{z}(w)} - \sum_{k=1}^K a_k w^{k+1}d\theta\right)\right| \le \frac{\delta}{\sqrt{\pi}} = \epsilon.
\tag{4.38}
$$

The coefficients a_k as well as K and $c(\delta)$ can be found in [1] for accuracies of $P'(A_i')$ of 3, 6, 9 and 12 decimal-digits. However, the accuracy of $P_{\mathrm{int}}(D)$ can be worse than the accuracy of $P'(A_i')$ used to calculate $P_{\mathrm{int}}(D)$ when the errors sum up in an unlucky way. But in our use case for polygons with a small number of vertices an accuracy loss

of more than one digit is unlikely. With this $P'(A_i')$ is given within $\pm\epsilon$ by [17]

$$\begin{aligned} P'(A_i') &= \frac{e^{-\frac{R^2}{2}}}{\pi}\left(\frac{\theta_2-\theta_1}{2} - \int_{\theta_1}^{\theta_2}\sum_{k=1}^{K} a_k w^{k+1} d\theta\right),\ |\theta_1| \leq \frac{\pi}{2}, |\theta_2| \leq \frac{\pi}{2} \\ &= \frac{e^{-\frac{R^2}{2}}}{\pi}\left(\frac{\theta_2-\theta_1}{2} - \sum_{k=1}^{K} a_k J_{k+1}\right), \end{aligned} \tag{4.39}$$

where

$$J_k = \int_{\theta_1}^{\theta_2} w^k d\theta = \left(\frac{R}{\sqrt{2}}\right)^k \int_{\theta_1}^{\theta_2} \cos^k\theta d\theta. \tag{4.40}$$

The constraints on θ_1 and θ_2 in Equation (4.39) are explained shortly.

Recurrence relations in the integral of Equation (4.40) let us solve it by using a reduction formula [36], i.e., the integral in J_k is reduced to an integral of J_{k-2} for $k \geq 2$. As J_0 and J_1 are given by

$$\begin{aligned} J_0 &= 1[\theta]_{\theta_1}^{\theta_2} = \theta_2 - \theta_1, \\ J_1 &= \frac{R}{\sqrt{2}}[\sin\theta]_{\theta_1}^{\theta_2} = \frac{R}{\sqrt{2}}(\sin\theta_2 - \sin\theta_1), \end{aligned} \tag{4.41}$$

an inductive procedure can be used to solve J_k once the reduction to J_{k-2} is known. Therefore, a reduction formula for the following integral is needed

$$I_k = \int_{\theta_1}^{\theta_2} \cos^k\theta d\theta,\ k \geq 2. \tag{4.42}$$

An integration by parts, the chain rule and the Pythagorean trigonometric identity yield [36]

$$
\begin{aligned}
I_k &= \int_{\theta_1}^{\theta_2} \cos^k \theta d\theta \\
&= \int_{\theta_1}^{\theta_2} \cos^{k-1} \theta \cos\theta d\theta \\
&= [\cos^{k-1} \theta \sin\theta]_{\theta_1}^{\theta_2} - \int_{\theta_1}^{\theta_2} (\cos^{k-1} \theta \frac{d}{d\theta}) \sin\theta d\theta \\
&= [\cos^{k-1} \theta \sin\theta]_{\theta_1}^{\theta_2} + \int_{\theta_1}^{\theta_2} (k-1) \cos^{k-2} \theta \sin\theta \sin\theta d\theta \\
&= [\cos^{k-1} \theta \sin\theta]_{\theta_1}^{\theta_2} + (k-1) \int_{\theta_1}^{\theta_2} \cos^{k-2} \theta \sin^2\theta d\theta \\
&= [\cos^{k-1} \theta \sin\theta]_{\theta_1}^{\theta_2} + (k-1) \int_{\theta_1}^{\theta_2} \cos^{k-2} \theta (1-\cos^2\theta) d\theta \\
&= [\cos^{k-1} \theta \sin\theta]_{\theta_1}^{\theta_2} + (k-1) \left(\int_{\theta_1}^{\theta_2} \cos^{k-2} \theta d\theta - \int_{\theta_1}^{\theta_2} \cos^2\theta d\theta \right) \\
&= [\cos^{k-1} \theta \sin\theta]_{\theta_1}^{\theta_2} + (k-1)(I_{k-2} - I_k).
\end{aligned}
\tag{4.43}
$$

Thus, the reduction formula for Equation (4.42) is

$$
\begin{aligned}
I_k &= [\cos^{k-1} \theta \sin\theta]_{\theta_1}^{\theta_2} + (k-1)(I_{k-2} - I_k) \\
\Leftrightarrow kI_k &= [\cos^{k-1} \theta \sin\theta]_{\theta_1}^{\theta_2} + (k-1) I_{k-2} \\
\Leftrightarrow I_k &= \frac{1}{k} [\cos^{k-1} \theta \sin\theta]_{\theta_1}^{\theta_2} + \frac{k-1}{k} I_{k-2}.
\end{aligned}
\tag{4.44}
$$

Inserting this into Equation (4.40) gives

$$
\begin{aligned}
J_k &= \left(\frac{R}{\sqrt{2}}\right)^k \int_{\theta_1}^{\theta_2} \cos^k \theta d\theta \\
&= \left(\frac{R}{\sqrt{2}}\right)^k \left(\frac{1}{k} [\cos^{k-1} \theta \sin\theta]_{\theta_1}^{\theta_2} + \frac{k-1}{k} \int_{\theta_1}^{\theta_2} \cos^{k-2} \theta d\theta \right) \\
&= \frac{1}{k} \left(\left(\frac{R}{\sqrt{2}}\right)^k [\cos^{k-1} \theta \sin\theta]_{\theta_1}^{\theta_2} + (k-1) \left(\frac{R}{\sqrt{2}}\right)^2 \left(\frac{R}{\sqrt{2}}\right)^{k-2} \int_{\theta_1}^{\theta_2} \cos^{k-2} \theta d\theta \right) \\
&= \frac{1}{k} \left(\left(\frac{R}{\sqrt{2}}\right)^k [\cos^{k-1} \theta \sin\theta]_{\theta_1}^{\theta_2} + (k-1) \frac{R^2}{2} J_{k-2} \right).
\end{aligned}
\tag{4.45}
$$

Finally from Equation (4.39) and 4.45 a procedure to calculate $P'_{\text{int}}(A'_i)$ can be given as follows

$$P'(A_i') = \frac{e^{-\frac{R^2}{2}}}{\pi}\left(\frac{\theta_2-\theta_1}{2} - \sum_{k=1}^{K} a_k J_{k+1}\right),\ |\theta_1| \leq \frac{\pi}{2}, |\theta_2| \leq \frac{\pi}{2}, \tag{4.46}$$

with

$$\begin{cases} J_0 & = \theta_2 - \theta_1 \\ J_1 & = \frac{R}{\sqrt{2}}(\sin\theta_2 - \sin\theta_1) \\ J_{k+1} & = \frac{1}{k+1}\left(\left(\frac{R}{\sqrt{2}}\right)^{k+1}[\cos^k\theta\sin\theta]_{\theta_1}^{\theta_2} + k\frac{R^2}{2}J_{k-1}\right) \\ & = \frac{1}{k+1}\left[(h_2 g_2^k - h_1 g_1^k) + k\frac{R^2}{2}J_{k-1}\right],\ k \geq 2 \end{cases} \tag{4.47}$$

where

$$g_i = \frac{R}{\sqrt{2}}\cos\theta_i,\ h_i = \frac{R}{\sqrt{2}}\sin\theta_i,\ i = 1, 2. \tag{4.48}$$

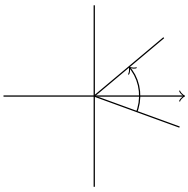
Equation (4.37) holds only if $0 \leq w$ and therefore requires $\cos\theta \geq 0$ as $R \geq 0$ which is the reason for the constraints $|\theta_1| \leq \frac{\pi}{2}$ and $|\theta_2| \leq \frac{\pi}{2}$ in Equation (4.39) and 4.46. For angles outside of the range $|\theta| \leq \frac{\pi}{2}$ the following relation [1] is used

$$P'(a(V,0,\theta)) = \frac{1}{2}\text{erfc}\left(\frac{R}{\sqrt{2}}\sin\theta\right) - P'(a(R,0,\pi-\theta)),\ |\theta| \leq \pi, \tag{4.49}$$

with $a(R,0,\theta)$ as the angular Region at vertex V and angles $\theta_1 = 0$, $\theta_2 = \theta$. Thus, there are different cases when evaluating $P'(A_i')$ in which Equation (4.49) is needed as shown in Figure 32.

With all this $P'(A_i')$ can be evaluated and thus the desired probability $P_{\text{int}}(D)$ as well as P_{hit} calculated. The basic required steps for the computation of $P_{\text{int}}(D)$ are summarized in Algorithm 3 and 4.

$g_1 \geq 0,\ g_2 \geq 0$

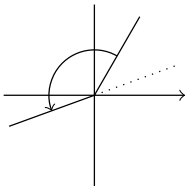


$L \begin{cases} |\theta_1| \leq \frac{\pi}{2} \\ |\theta_2| \leq \frac{\pi}{2} \end{cases}$

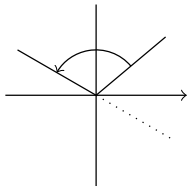
$A^* = a(V, \theta_1, \theta_2)$

$P'(A') = P'(A^*)$

$g_1 \geq 0,\ g_2 < 0 \Rightarrow |\theta_2| > \frac{\pi}{2}$



$L \begin{cases} |\theta_1| \leq \frac{\pi}{2} \\ \theta_2 < 0 \end{cases}$

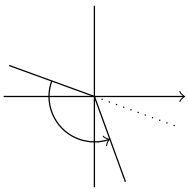


$L \begin{cases} |\theta_1| \leq \frac{\pi}{2} \\ \theta_2 > 0 \end{cases}$

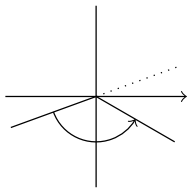
$A^* = a(V, \theta_1, \theta_2 + \pi)$ $\qquad$ $A^* = a(V, \theta_1, \theta_2 - \pi)$

$P'(A) = \frac{1}{2}\text{erfc}(h_2) - P'(A^*)$

$g_1 < 0 \Rightarrow |\theta_1| > \frac{\pi}{2},\ g_2 \geq 0$



$L \begin{cases} \theta_1 > 0 \\ |\theta_2| \leq \frac{\pi}{2} \end{cases}$

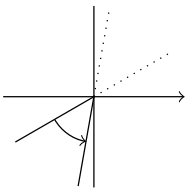


$L \begin{cases} \theta_1 < 0 \\ |\theta_2| \leq \frac{\pi}{2} \end{cases}$

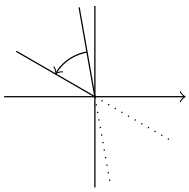
$A^* = a(V, \theta_1 - \pi, \theta_2)$ $\qquad$ $A^* = a(V, \theta_1 + \pi, \theta_2)$

$P'(A) = \frac{1}{2}\text{erfc}(-h_1) - P'(A^*)$

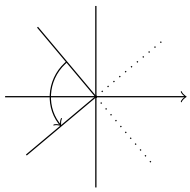
$g_1 < 0 \Rightarrow |\theta_1| > \frac{\pi}{2},\ g_2 < 0 \Rightarrow |\theta_2| > \frac{\pi}{2}$



$L \begin{cases} \theta_1 < 0 \\ \theta_2 < 0 \end{cases}$



$L \begin{cases} \theta_1 > 0 \\ \theta_2 > 0 \end{cases}$



$L \begin{cases} \theta_1 > 0 \\ \theta_2 < 0 \end{cases}$

$A^* = a(V, \theta_1 + \pi, \theta_2 + \pi)$ $\qquad$ $A^* = a(V, \theta_1 - \pi, \theta_2 - \pi)$ $\qquad$ $A^* = a(V, \theta_1 - \pi, \theta_2 + \pi)$

$P'(A) = \frac{1}{2}\text{erfc}(h_2) - \frac{1}{2}\text{erfc}(h_1) + P'(A^*)$

Figure 32: Calculation of angular regions including the cases where the constraints of Equation (4.46) are not given. The figure is derived from [1].

Algorithm 3 Bivariate Polygon Integration

function BIVARIATEPOLYGONINTEGRATION(D, σ_x, σ_y)
  $D' \leftarrow scaleToCircular(D, \sigma_x, \sigma_y)$
  $result \leftarrow 0$
  for each vertex V_i in D' **do**
    $R \leftarrow \text{distance}(R, (0,0))$
    $v_1 \leftarrow \text{vector}(\text{predecessor}(V_i), V_i)$
    $v_2 \leftarrow \text{vector}(V_i, \text{successor}(V_i))$
    if $R == 0$ **then**
      $result \leftarrow result + \arccos\langle v_1, v_2\rangle/(2\pi)$
      continue
    $l \leftarrow \text{vector}((0,0), V_i)$
    $\theta_1 \leftarrow \text{sign}(\det(\vec{v_1}, \vec{l}))\frac{\arccos\langle\vec{v_1},\vec{l}\rangle}{|\vec{v_1}|\cdot|\vec{l}|}$
    $\theta_1 \leftarrow \text{sign}(\det(\vec{v_2}, \vec{l}))\frac{\arccos\langle\vec{v_2},\vec{l}\rangle}{|\vec{v_2}|\cdot|\vec{l}|}$
    $h_2 \leftarrow \sin\theta_2 R/\sqrt{2}$
    $h_1 \leftarrow \sin\theta_1 R/\sqrt{2}$
    $g_2 \leftarrow \cos\theta_2 R/\sqrt{2}$
    $g_1 \leftarrow \cos\theta_1 R/\sqrt{2}$
    if $g_1 \geq 0$ and $g_2 \geq 0$ **then**
      $result \leftarrow result +$ INTEGRATEANGULARREGION$(R, \theta_1, \theta_2, g_1, g_2, h_1, h_2)$
    else
      if $g_1 < 0$ **then**
        if $\theta_1 > 0$ **then**
          $\theta_1 \leftarrow \theta_1 - \pi$
        else
          $\theta_1 \leftarrow \theta_1 + \pi$
      if $g_2 < 0$ **then**
        if $\theta_2 > 0$ **then**
          $\theta_2 \leftarrow \theta_2 - \pi$
        else
          $\theta_2 \leftarrow \theta_2 + \pi$
      $h_2' \leftarrow \sin\theta_2 R/\sqrt{2}$
      $h_1' \leftarrow \sin\theta_1 R/\sqrt{2}$
      $g_2' \leftarrow \cos\theta_2 R/\sqrt{2}$
      $g_1' \leftarrow \cos\theta_1 R/\sqrt{2}$
      if $g_1 \geq 0$ and $g_2 < 0$ **then**
        $result \leftarrow result + \text{erfc}(h_2)/2$
          $-$ INTEGRATEANGULARREGION$(R, \theta_2, \theta_1, g_2', g_1', h_2', h_1')$
      else if $g_1 < 0$ and $g_2 \geq 0$ **then**
        $result \leftarrow result + \text{erfc}(-h1)/2$
          $-$ INTEGRATEANGULARREGION$(R, \theta_2, \theta_1, g_2', g_1', h_2', h_1')$
      else
        $result \leftarrow result + (\text{erfc}(h_2) - \text{erfc}(h_1))/2$
          $-$ INTEGRATEANGULARREGION$(R, \theta_1, \theta_2, g_1', g_2', h_1', h_2')$
  return $1 - result$

Algorithm 4 Angular Region Integration

function INTEGRATEANGULARREGION($R, \theta_1, \theta_2, g_1, g_2, h_1, h_2$)
  $result \leftarrow 0$
  $j_l \leftarrow \theta_2 - \theta_1$
  $j \leftarrow h_2 - h_1$
  for $k = 0, k \leq K, k \mathrel{+}= 1$ **do**
    $result \leftarrow result + a_k j$
    $copy \leftarrow j$
    $j \leftarrow (h_2 g_2^{k+1} - h_1 g_1^{k+1} + j_l(k+1)R^2/2)/(k+2)$
    $j_l \leftarrow copy$
  return $e^{-R^2/2}((\theta_2 - \theta_1)/2 - result)/\pi$

To conclude, an accurate and fast method [1] to evaluate the integral of the bivariate Gaussian distribution over a polygon has been discussed. Furthermore, in the beginning of this section the computation of a representative region for which rays intersecting this region would hit the current receiver piece was given. By combining the two, the probability of the representative ray to be perturbed such that it hits the current receiver piece can be calculated which is the main part of our Gaussian convolution ray tracer. Algorithm 5 sketches the required steps for the total solar power calculation.

Algorithm 5 Gaussian convolution ray tracer

function GAUSSIANCONVOLUTION($s, \vec{r}, \vec{r}^{\,\text{hor}}, \vec{r}^{\,\text{ver}}, \sigma^{\text{hor}}_{\text{beam}}, \sigma^{\text{ver}}_{\text{beam}}$)
  $solarPower \leftarrow 0$
  $\sigma_x \leftarrow \tan \sigma^{\text{hor}}_{\text{beam}}$ ▷ distance to ray image plane $\delta = 1$
  $\sigma_y \leftarrow \tan \sigma^{\text{ver}}_{\text{beam}}$
  $M \leftarrow$ getPerspectiveProjectionMatrix($s, \vec{r}, \vec{r}^{\,\text{hor}}, \vec{r}^{\,\text{ver}}, 1$)
  for each receiver piece P in receiver pieces **do**
    $projectedCorners \leftarrow []$
    for each corner c_i in P **do**
      $projectedCorners.add(Mc_i)$
    $D_{\text{span}} \leftarrow$ polygon($projectedCorners$)
    $solarPower \leftarrow solarPower +$
      BIVARIATEPOLYGONINTEGRATION($D_{\text{span}}, \sigma_x, \sigma_y$)
  return $solarPower$

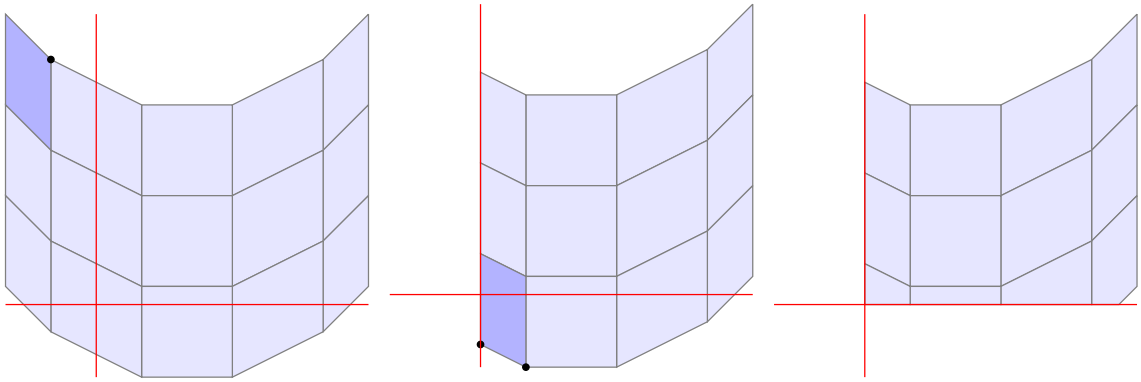
4.2.3 Tower blocking

As described in Section 2.4 it is crucial to be aware of tower blocking in the case of a cavity receiver. Any ray hitting the receiver first has to go through the receiver window. To exclude the rays that would hit the tower before hitting the receiver the receiver pieces have to be cut in such a way that only the visible part remains. The visibility of a point is determined by Equation (3.4) using the two blocking planes of

Equation (3.3), see Section 3.4. Depending on which side of the planes a receiver corner lies it is either visible or blocked. Therefore, a whole receiver piece can be visible, blocked or partly visible which means it has to be cut.

In our Gaussian convolution ray tracer the receiver pieces are first traversed in horizontal direction starting from the topmost receiver piece at the vertical blocking edge. As a reminder, this is the edge used to define the vertical blocking plane B^{ver}, see Equation (3.3). By doing this most of the not visible receiver pieces can directly be excluded from further evaluation. When the right vertical edge of the receiver window is closer to the ray origin, the upper left corner of each receiver piece is checked for vertical visibility otherwise the upper right corner is checked. Figure 33(a) illustrates the first step of a horizontal traverse.

As soon as the checked corner is visible a vertical cut is needed since the other upper corner was not visible or lied on the plane itself. This cut can be omitted if the checked corner of the first receiver piece is visible since the other upper corner lied on the plane itself which implies all receiver pieces to be vertically visible.



(a) First step (b) Vertically cut (c) Fully cut

Figure 33: Simplified example of the cutting process for the cavity receiver to obtain its visible pieces. The currently checked receiver piece is shown in dark blue and the checked corners are marked with a dot.

In order to correctly cut a receiver piece, an intersection of the vertical blocking plane with the line between the two upper corners of the receiver piece is evaluated. The vertically not visible corner is then replaced by the resulting intersection point i. Instead of calculating the same intersection for the line between the two lower corners one can make use of the fact that the receiver pieces are vertically aligned with the vertical blocking plane. The reason for this is the definition of the vertical blocking plane and the fact that all receiver pieces have the same vertical alignment. Therefore, the intersection point of the line between the two lower corners and the vertical blocking plane is the same as the point i plus the vector from one upper corner to the lower one,

see Figure 33(b). By this, the correct vertical cut for all receiver pieces below can also be calculated. Thus, the receiver pieces are now also traversed in vertical direction, vertically cut and the remaining receiver pieces are assumed to be vertically visible. The resulting receiver pieces are shown in Figure 33(b).

After each vertical cut, both lower corners are checked for horizontal visibility. If at least one is not visible another cut is required. As the receiver pieces are not horizontally aligned there are a lot more different cases which require a cut. Figure 34 shows these cases and the resulting receiver piece.

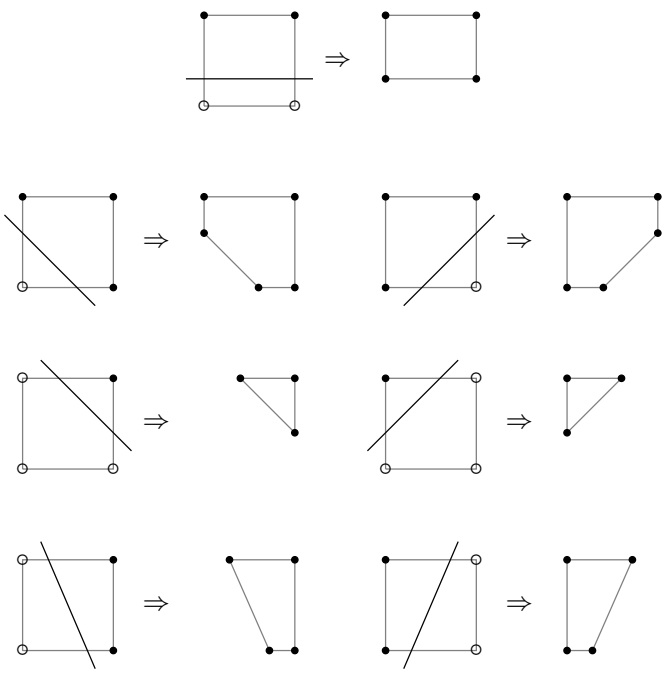


Figure 34: Illustration of all possible cases when cutting a receiver piece horizontally. Not visible corners are marked with a cycle and visible ones with a dot.

When both lower corners are not visible all receiver pieces below are skipped as they are also not visible. Afterwards, the highest receiver piece of the next column is checked in the same way until all receiver pieces are correctly cut. Note that at most one column requires a vertical cut, thus any column after the vertically cut one is only checked for horizontal visibility. An exemplary visible part of a cavity receiver is illustrated in Figure 33(c).

In the case of calculating the representative region D_{span} on the ray image plane this process may also be done on the ray image plane itself. The necessary steps are similar, but the two blocking planes can now be represented by two lines which are defined as the intersection of the ray image plane with either of the blocking plane. Therefore, the intersection points needed in the cutting process are now evaluated by intersecting

two lines in two dimensions instead of a line and a plane in three dimensions.

Finally, all parts of our Gaussian convolution ray tracer are fully specified and we can come to our last ray tracing technique being the integrated convolution method.

4.3 Integrated convolution method

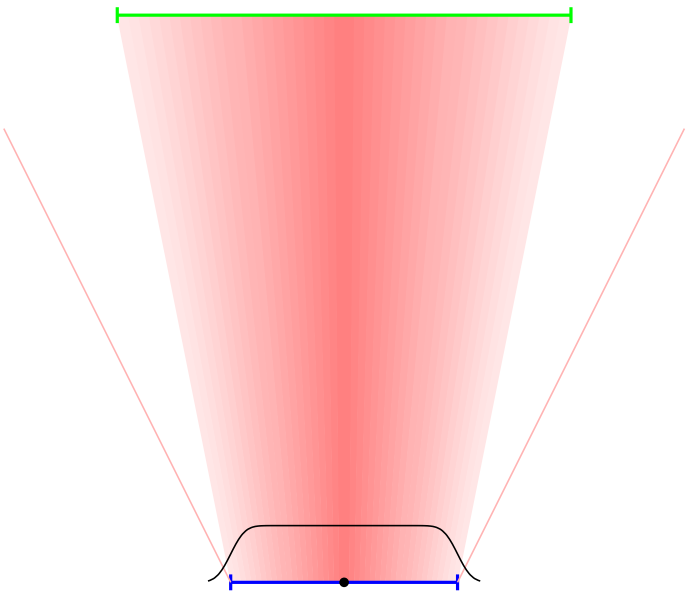
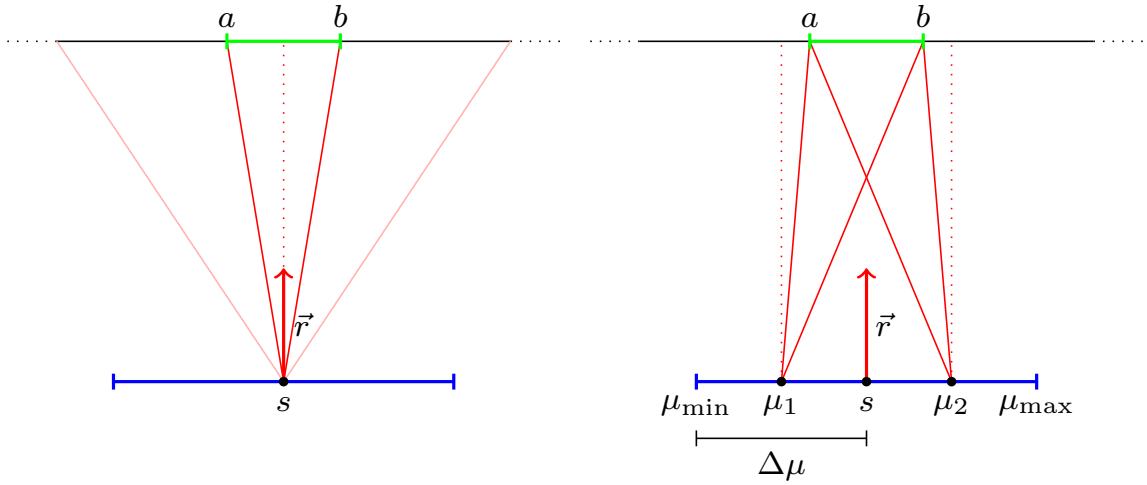


Figure 35: Basic principle of our integrated convolution method. Here, the whole heliostat cell of the representative ray is taken into account which requires a different probability density function, shown in black, for evaluation.

So far, the ray tracing methods have mainly concentrated on the evaluation of the error cone which is one main error source for each ray tracer. However, as discussed in Section 2.5 there is yet another source of error being the fact that one ray is used to represent photon interactions of a whole cell. Previously, this error has been reduced by increasing the number of rays leading to smaller cell areas for each ray. But increasing the number of rays effectively increases the computational costs. Therefore, our integrated convolution ray tracer aims to reduce the error of using one ray for each heliostat cell in a different way, see Figure 35. The method is based on ideas described in Richter et. al [33, 35]. To understand the underlying concept a deeper look into what it means to take more than one ray for a cell is needed. For this, a simplified two dimensional version is introduced before getting to the general case, see Figure 36. When evaluating only the representative ray as in Figure 36(a), the probability P_{hit} that this ray will be perturbed in such a way that it hits the receiver is of interest. In combination with the power of the ray P_{ray}, the solar power P_{rec} at the receiver emitted from the current heliostat cell can be calculated by

$$P_{\text{rec}} = P_{\text{ray}} P_{\text{hit}} = P_{\text{ray}} \int_a^b f(x)\ dx, \tag{4.50}$$

with $f(x)$ as the Gaussian distribution on the ray image line and a and b as the coordinates of the receiver endpoints on that line. Here, the ray image line is the two dimensional equivalent to the ray image plane.



(a) Evaluating only the representative ray.

(b) Evaluting two rays per cell.

Figure 36: Simplified two dimensional examples to illustrate the process of evaluating more than one ray per heliostat cell.

However, when taken two rays for a heliostat cell as in Figure 36(b) the power at the receiver cell for both rays is calculated as follows

$$\begin{aligned} P_{\text{rec}} &= P_{\text{rec},1} + P_{\text{rec},2} = \frac{1}{2} P_{\text{ray}} P_{\text{hit},1} + \frac{1}{2} P_{\text{ray}} P_{\text{hit},2} \\ &= P_{\text{ray}} \frac{1}{2} \left(\int_a^b f(x - \mu_1)\ dx + \int_a^b f(x - \mu_2)\ dx \right) \\ &= P_{\text{ray}} \int_a^b \frac{1}{2} \left(f(x - \mu_1) + f(x - \mu_2) \right)\ dx \\ &= P_{\text{ray}} P_{\text{hit}}^{\text{avg}}, \end{aligned} \tag{4.51}$$

with μ_1 and μ_2 as the origin of the two rays and $P_{\text{hit}}^{\text{avg}}$ as the average probability of a ray to be perturbed such that it hits the receiver.

Comparing Equation (4.51) with Equation (4.50) one can see that evaluating two rays can be modeled by using a different probability density function. In the same way the evaluation of n rays can be modeled which changes $P_{\text{hit}}^{\text{avg}}$ to

$$P_{\text{hit}}^{\text{avg}} = \int_a^b \frac{1}{n} \sum_{i=1}^{n} f(x - \mu_i)\ dx = \int_a^b F(x)\ dx,\ \mu_i \in [\mu_{\min}, \mu_{\max}], \tag{4.52}$$

with $F(x)$ as the new probability density function, l_{cell} as the length of the cell and $\mu_{\min}$ and $\mu_{\max}$ as the endpoints, see Figure 36(b).

Before further evaluation an exact description of μ_i is required. By definition when $n = 1$ then $\mu_1 = 0$, see Figure 36(a). For $n = 2$, μ_1 and μ_2 should be defined as illustrated in Figure 36(b). Therefore, $\mu_1 = \mu_{\min} + \frac{\Delta\mu}{2}$ and $\mu_2 = \mu_{\min} + \frac{\Delta\mu}{2} + \Delta\mu$ with $\Delta\mu = \frac{\mu_{\max} - \mu_{\min}}{2}$. The pattern continues leading to

$$\mu_i = \mu_{\min} + \frac{\Delta\mu}{2} + \Delta\mu(i-1),\ \Delta\mu = \frac{\mu_{\max} - \mu_{\min}}{n}. \tag{4.53}$$

By letting n go to infinity in $F(x)$ meaning that an infinite amount of rays will be evaluated, a Riemann sum [36] is reached. In combination with $f(x)$ being axis-symmetric, this leads to the following integral

$$\begin{aligned} F(x) &= \frac{1}{n}\frac{1}{\Delta\mu} \sum_{i=1}^{n} f(x - \mu_i)\Delta\mu = \frac{1}{\mu_{\max} - \mu_{\min}} \sum_{i=1}^{n} f(x - \mu_i)\Delta\mu \\ &\overset{n\to\infty}{=} \frac{1}{l_{\text{cell}}} \lim_{n\to\infty} \sum_{i=1}^{n} f(x - \mu_i)\Delta\mu = \frac{1}{l_{\text{cell}}} \lim_{\Delta\mu\to 0} \sum_{i=1}^{n} f(x - \mu_i)\Delta\mu \\ &= \frac{1}{l_{\text{cell}}} \int_{\mu_{\min}}^{\mu_{\max}} f(x - \mu)d\mu = \frac{1}{l_{\text{cell}}} \int_{\mu_{\min}}^{\mu_{\max}} f(\mu - x)d\mu. \end{aligned} \tag{4.54}$$

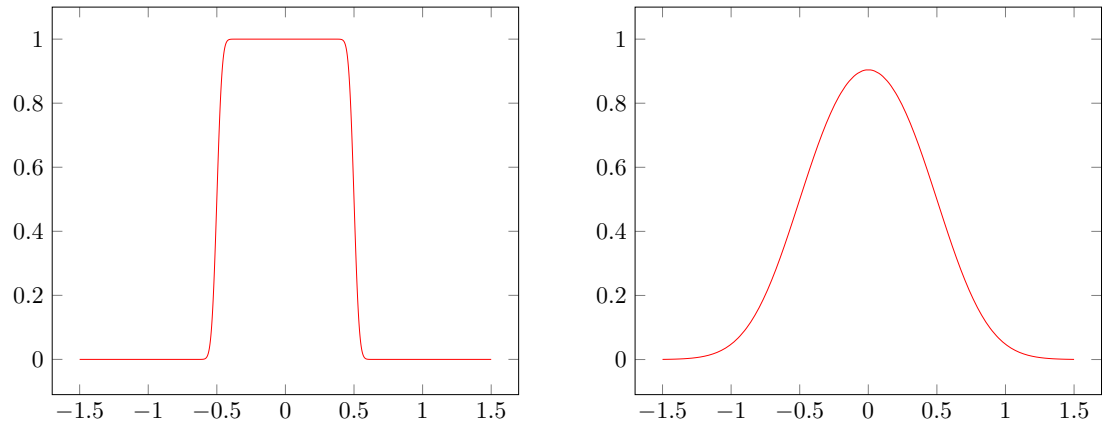
The integral is solved by using the error function which is defined as

$$\text{erf}(x) = \frac{2}{\sqrt{\pi}} \int_0^x e^{-t^2} dt, \tag{4.55}$$

resulting in

$$
\begin{aligned}
F(x) &= \frac{1}{l_{\text{cell}}} \int_{\mu_{\min}}^{\mu_{\max}} f(\mu - x) d\mu \\
&= \frac{1}{l_{\text{cell}}} \frac{1}{\sqrt{2\pi}\sigma} \int_{\mu_{\min}}^{\mu_{\max}} e^{-\left(\frac{\mu - x}{\sqrt{2}\sigma}\right)^2} \frac{1}{\sqrt{2}\sigma} \sqrt{2}\sigma d\mu \\
&= \frac{1}{l_{\text{cell}}} \frac{1}{\sqrt{\pi}} \int_{\frac{\mu_{\min} - x}{\sqrt{2}\sigma}}^{\frac{\mu_{\max} - x}{\sqrt{2}\sigma}} e^{-t^2} dt, \; t = \frac{\mu - x}{\sqrt{2}\sigma} \\
&= \frac{1}{l_{\text{cell}}} \frac{1}{2} \left(\frac{2}{\sqrt{\pi}} \int_0^{\frac{\mu_{\max} - x}{\sqrt{2}\sigma}} e^{-t^2} dt - \frac{2}{\sqrt{\pi}} \int_0^{\frac{\mu_{\min} - x}{\sqrt{2}\sigma}} e^{-t^2} dt \right) \\
&= \frac{1}{2 l_{\text{cell}}} \left(\operatorname{erf}\left(\frac{\mu_{\max} - x}{\sqrt{2}\sigma} \right) - \operatorname{erf}\left(\frac{\mu_{\min} - x}{\sqrt{2}\sigma} \right) \right).
\end{aligned}
\tag{4.56}
$$

An illustration of this probability density function from a heliostat cell of length $l_{\text{cell}} = 1$ m for a distance $\delta = 20$ m and angular standard derivation $\sigma_{\text{beam}} = 3$ mrad resulting in $\sigma = \sigma_{\text{span}} \approx 0.06$ m is shown in Figure 37(a). From now on, $F(x)$ is referred to as the integrated Gaussian distribution.



(a) 20 m distance.

(b) 100 m distance.

Figure 37: Probability density function of the integrated Gaussian distribution from a heliostat cell of length $l_{\text{cell}} = 1$ m at different distances with standard derivations σ_{span} as in Equation (4.3) from $\sigma_{\text{beam}} = 3$ mrad.

Similarly to the definition of the bivariate Gaussian distribution in Section 4.1, the two dimensional equivalent to the integrated Gaussian distribution is defined as the multiplication of two independent integrated Gaussian distributions to

$$
\begin{aligned}
F(x,y) &= F(x)^{\text{hor}} F(y)^{\text{ver}} \\
&= \frac{1}{4 l_{\text{cell}} w_{\text{cell}}} \left(\operatorname{erf}\left(\frac{\mu_{\text{max}}^{\text{hor}} - x}{\sqrt{2}\sigma_{\text{span}}^{\text{hor}}} \right) - \operatorname{erf}\left(\frac{\mu_{\text{min}}^{\text{hor}} - x}{\sqrt{2}\sigma_{\text{span}}^{\text{hor}}} \right) \right) \\
&\quad \left(\operatorname{erf}\left(\frac{\mu_{\text{max}}^{\text{ver}} - y}{\sqrt{2}\sigma_{\text{span}}^{\text{ver}}} \right) - \operatorname{erf}\left(\frac{\mu_{\text{min}}^{\text{ver}} - y}{\sqrt{2}\sigma_{\text{span}}^{\text{ver}}} \right) \right),
\end{aligned}
\tag{4.57}
$$

with w_{cell} as the width of the cell. Note that this is only correct when the ray direction is orthogonal to the heliostat cell.

Figure 38 shows the two dimensional integrated Gaussian distribution for a cell of width $w_{\text{cell}} = 2$ m and length $l_{\text{cell}} = 2$ m where the distance δ and standard deviations $\sigma_{\text{beam}}^{\text{ver}}$ and $\sigma_{\text{beam}}^{\text{hor}}$ were exemplary set such that $\sigma_{\text{span}}^{\text{ver}} = \sigma_{\text{span}}^{\text{hor}} \approx 0.15$ m.

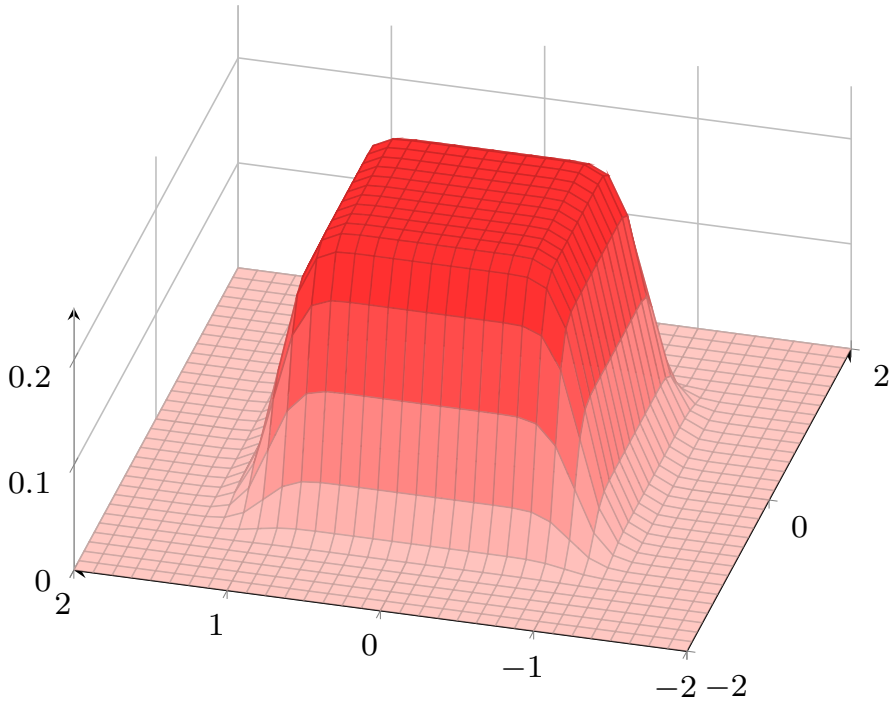


Figure 38: Integrated two dimensional Gaussian distribution for a cell of width $w_{\text{cell}} = 2$ m and length $l_{\text{cell}} = 2$ m at a distance $\delta = 50$ m and standard deviations $\sigma_{\text{beam}}^{\text{ver}} = \sigma_{\text{beam}}^{\text{hor}} = 3$ mrad resulting in $\sigma_{\text{span}}^{\text{ver}} = \sigma_{\text{span}}^{\text{hor}} \approx 0.15$ m [18].

To calculate the desired average probability $P_{\text{hit}}^{\text{avg}}$ of a ray to be perturbed such that it hits the receiver for the three dimensional case the following integral must be evaluated

$$
P_{\text{hit}}^{\text{avg}} = \iint_D F(x,y)\, dA, \tag{4.58}
$$

with D as a representative region of the receiver cell on the ray image plane.

However, even for our two dimensional example the integral is hard to calculate. Moreover, no method to integrate the two dimensional integrated Gaussian distribution over a polygon has been found. The tool *UNIZAR* describes the flux distribution of a cell with a similar function as the two dimensional integrated Gaussian distribution [37]. It solves the integral using a numerical integration similar to the one of *HFLCAL*. Thus, it depends on a fine discretization of the receiver which we want to avoid in order to decrease the computing time.
Therefore, there are two problems left in order to make use of the integrated Gaussian distribution. Firstly, a way to solve the integral in Equation (4.58) is required. Secondly, in the case of the ray direction being non-orthogonal to its cell, a definition of the width and height used in the integrand needs to be given.

4.3.1 Approximating the integrated Gaussian distribution

Again, the simplified case of an orthogonal ray is viewed before getting to the general situation. As already illustrated in Figure 37(b) the integrated Gaussian distribution at a distance has a similar shape as a Gaussian distribution. Thus, the same holds for their two dimensional equivalents. As an integration of the bivariate Gaussian distribution over a polygon has already been given in the last section, the first problem can be solved by approximating the integrated Gaussian distribution with a Gaussian distribution for both deviation directions. Figure 39 shows the result of such an approximation.

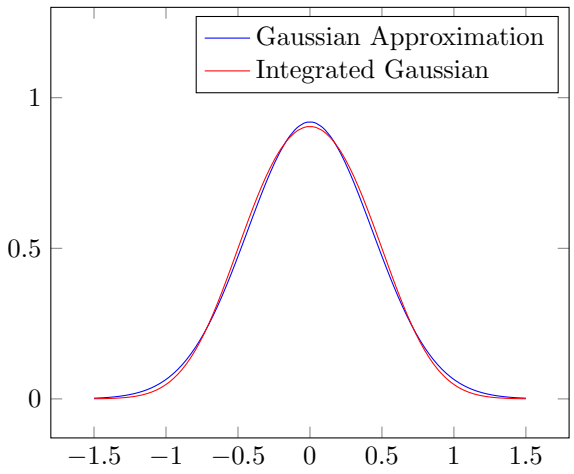


Figure 39: The integrated Gaussian distribution of Figure 37(b) approximated by a Gaussian distribution.

The optimization problem used to obtain an accurate approximation is defined as follows. For a given cell length l_{cell}, distance δ and standard deviation σ_{beam} find a

sigma multiplier σ_{mul} such that

$$\sigma_{\text{mul}} = \underset{\sigma_{\text{mul}} \in \mathbb{R}}{\arg\min} \sum_{i=1}^{n} \left(F(x_i |\ \delta \tan \sigma_{\text{beam}}, l_{\text{cell}}) - f(x_i |\ \delta \tan (\sigma_{\text{beam}} \sigma_{\text{mul}})) \right)^2,\ x_i \in [-b, b]. \tag{4.59}$$

The points x_i are evenly spaced on the interval $[-b, b]$ which is chosen to include most of the values drawn by both distribution functions, thus $b = 3\sigma_{\text{span}} + \frac{l_{\text{cell}}}{2}$. In Equation (4.59) a sigma multiplier is used instead of directly trying to find the optimal sigma, due to its high dependency to the distance. The optimizations were done in python using the scipy libraries[1]. Since this takes too much time to do during the ray tracing process, a function $g(\delta, \sigma_{\text{beam}}, l_{\text{cell}})$ that calculates σ_{mul} is needed. Therefore, the integrated Gaussian distribution is approximated by

$$F(x_i |\ \sigma_{\text{span}}, l_{\text{cell}}) \approx f(x_i |\ \sigma_{\text{span}} \cdot g(d, \sigma_{\text{beam}}, l_{\text{cell}})). \tag{4.60}$$

During the analysis of the function g, two important properties were noticed.

1. $g(\delta, a \cdot \sigma_{\text{beam}}, l_{\text{cell}}) \approx g(\delta, \sigma_{\text{beam}}, \frac{l_{\text{cell}}}{a}),\ a \in \mathbb{R}$
2. $g(\delta, \sigma_{\text{beam}}, a \cdot l_{\text{cell}}) \approx g(\frac{\delta}{a}, \sigma_{\text{beam}}, l_{\text{cell}}),\ a \in \mathbb{R}$

By using these properties, the dependencies of g can be reduced to

$$\begin{aligned} g(\delta, \sigma_{\text{beam}}, l_{\text{cell}}) &= g(\delta, \frac{\sigma_{\text{beam}}}{\sigma_{\text{beam,std}}} \sigma_{\text{beam,std}}, \frac{l_{\text{cell}}}{l_{\text{cell,std}}} l_{\text{cell,std}}) \\ &\overset{1}{\approx} g(\delta, \sigma_{\text{beam,std}}, \frac{\sigma_{\text{beam,std}}}{\sigma_{\text{beam}}} \frac{l_{\text{cell}}}{l_{\text{cell,std}}} l_{\text{cell,std}}) \\ &\overset{2}{\approx} g(\alpha, \sigma_{\text{beam,std}}, l_{\text{cell,std}}) = g_{\text{new}}(\alpha),\ \alpha = \delta \frac{\sigma_{\text{beam}} l_{\text{cell,std}}}{\sigma_{\text{beam,std}} l_{\text{cell}}}, \end{aligned} \tag{4.61}$$

with $\sigma_{\text{beam,std}} = 3$ mrad and $l_{\text{cell,std}} = 1$ m as standard values chosen to reduce the factor α. Here, g_{new} is the new sigma multiplier function which only depends on one variable.

The values of g_{new} have been calculated for various α values and the function was approximated with the help of ZunZun[2], a website for curve and surface fitting, resulting in

$$g_{\text{new}}(\alpha) \approx g_{\text{approx}}(\alpha) = \max\left(a \cdot e^{\frac{b}{x+c}} + d, 1\right). \tag{4.62}$$

[1] https://www.scipy.org/
[2] http://zunzun.com/

Coefficient	Value
a	6.0654123395858638E-03
b	1.0168091727137571E+03
c	1.3522384735784115E+02
d	9.8897960843463073E-01

Table 1: Coefficients of the sigma multiplier function g_{approx} from Equation (4.62).

Table 1 gives the coefficients derived from ZunZun and Figure 40 shows the original values of g_{new} as well as the approximation g_{approx}. As expected, the sigma multiplier approaches one for small cell lengths meaning that the original Gaussian distribution stays unchanged. However, a large α value for g_{approx} is also caused by a far distance δ and a high standard derivation σ_{beam}. Thus, the Gaussian convolution method is presumed to be more accurate in those instances.

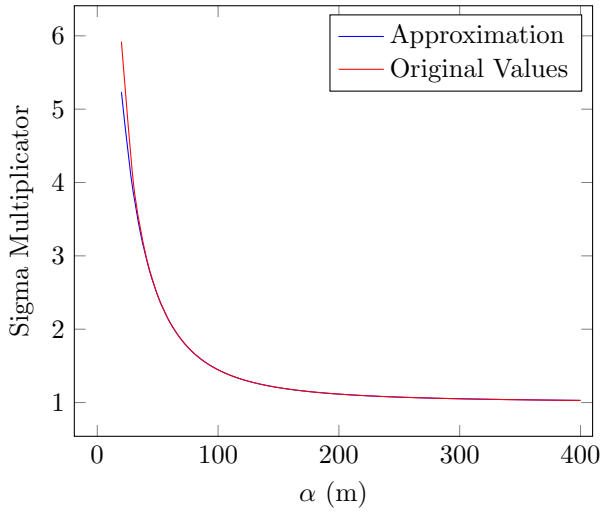
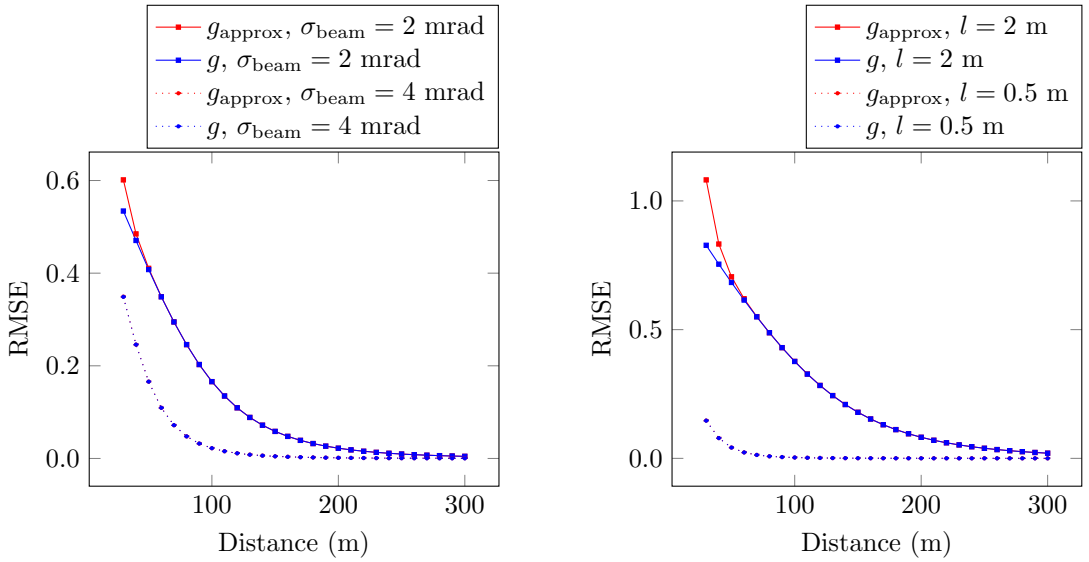


Figure 40: Values of g_{new} and the approximation g_{approx}.

To evaluate the accuracy of the derived approximation in Equation (4.60) the root mean square error (RMSE) between the integrated Gaussian distribution and the Gaussian approximation is used. Besides the approximation using the sigma function also the optimal Gaussian approxomation, derived with the scipy libraries is tested. The results for different cell lengths l_{cell} and standard derivations σ_{beam} in dependence to the distance δ can be found in Figure 41(b) and 41(a).

The figures support the assumption that the integrated Gaussian distribution converges to a Gaussian distribution in the distance. Furthermore, our derived approximation reaches a similar RMSE as the optimal Gaussian approximation, especially for large distances.



(a) RMSE for various standard derivations σ_{beam} and distances with $l_{\text{cell}} = 1$ m.

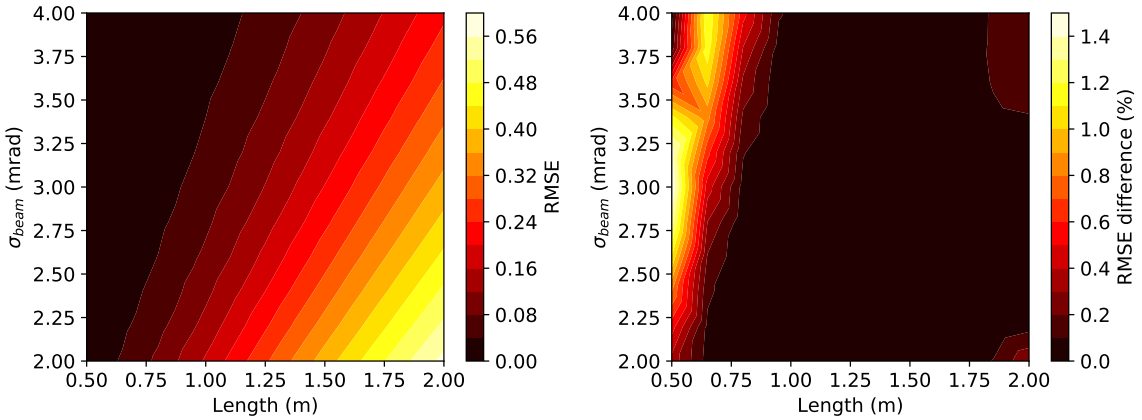
(b) RMSE for various cell lengths l_{cell} and distances with $\sigma_{\text{beam}} = 3$ mrad.

Figure 41: RMSE between the integrated Gaussian distribution and the optimal Gaussian approximation g as well as the approximation using the sigma function g_{approx}.

Additionally, Figure 42(a) compares the RMSE of the integrated Gaussian distribution and the optimal approximation at a fixed distance of $\delta = 100$ m. This is commonly about the smallest distance of a heliostat cell to the tower. As before, the RMSE increases for larger cell length and smaller standard deviations. However, the cell length has a bigger influence on the RMSE than the standard derivation. The relative RMSE difference between the optimal approximated Gaussian distribution and the approximation using the derived function is shown in Figure 42(b). Here, the relative RMSE difference between the derived function and the optimal approximation is always less than 1.5 %.

To conclude the integrated Gaussian distribution function was approximated by a Gaussian distribution with a scaled standard derivation. The definition of an optimal scaling factor as well as a function to approximate it has been given. Furthermore, the derived function was shown to be an accurate approximation.

In order to make use of the derived approximation for the general case, a definition of the cell width and height used in the two dimensional integrated Gaussian distribution is needed.



(a) RMSE of the integrated Gaussian distribution and the optimal approximation.

(b) Relative RMSE difference of the function approximation to the optimal one.

Figure 42: RMSE heatmaps for the approximation steps at a distance of 100 m.

4.3.2 Use of the approximation

For the purpose of developing a reasonable definition of the adapted cell dimensions, the influence of the original cell dimensions in the current ray tracing techniques is reviewed. So far, the only parameter depending on them is the ray power P_{ray}. Moreover, the dependency lies just on the effective cell area, see Section 2.5. This is the area of the heliostat cell projected onto a plane orthogonal to the sun vector. Due to the law of reflection it is the same as the area of the projection onto a plane orthogonal to the ray vector. As the shape of the cell did not influence the ray evaluation so far, it is reasonable to assume the effective cell area to be quadratic and aligned with the deviation directions of the ray, see Figure 43. Therefore, the adapted cell width w'_{cell} and length l'_{cell} are defined as follows

$$l'_{\text{cell}} = w'_{\text{cell}} = \sqrt{A_{\text{cell}}\eta_{\cos}}. \tag{4.63}$$

The resulting integrated Gaussian distributions for both directions are then approximated by Gaussian distributions as described earlier. As a distance δ used in the approximation, the distance between the ray origin and the midpoint of the receiver piece is utilized. From there on, the evaluation is carried out the same way as in the Gaussian convolution method. Note that the perspective projection used to define the area D_{span} representing the receiver piece is not fully correct anymore as the rays do not originate from the same point. However, due to the distances and the small effective areas it is a reasonable assumption for the calculation of D_{span}. Algorithm 6 sums up the required steps of the integrated convolution ray tracer.

A study done in [16] also confirms some approaches of our integrated convolution method. Here, the slope error of *HFLCAL* was fitted such that the resulting flux

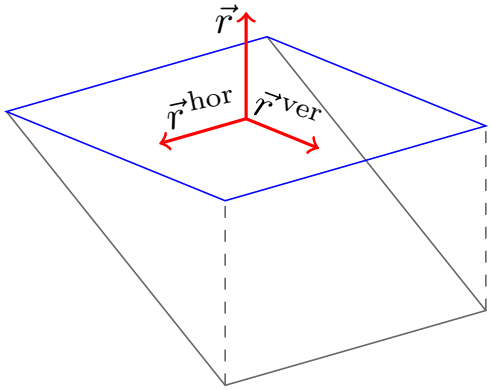


Figure 43: Adapted heliostat cell drawn in blue of the original cell drawn in black.

distribution of the heliostat matches actual measured data. It was done for heliostats with different distances to the tower and varying incident angles of the sun rays. The behavior of the fitted slope errors corresponds to the predictions made by our derived fitting function. They increased for larger cell areas and decreased with the distance. Moreover, the authors were surprised to find that the heliostats closest to the receiver had the worst fittings. But considering the findings of the last section, the flux of theses heliostat is expected to differ most from the actual flux.

Algorithm 6 Integrated convolution ray tracer

function INTEGRATEDCONVOLUTION($s, \vec{r}, \vec{r}^{\text{hor}}, \vec{r}^{\text{ver}}, \sigma^{\text{hor}}_{\text{beam}}, \sigma^{\text{ver}}_{\text{beam}}$)
  $solarPower \leftarrow 0$
  $M \leftarrow \text{getPerspectiveProjectionMatrix}(s, \vec{r}, \vec{r}^{\text{hor}}, \vec{r}^{\text{ver}}, 1)$
  $l'_{\text{cell}} \leftarrow \sqrt{A_{\text{cell}}\eta_{\cos}}$
  for each receiver piece P in receiver pieces **do**
    $\delta \leftarrow |s - P.midpoint|$
    $\alpha^{\text{hor}} \leftarrow (\delta\sigma^{\text{hor}}_{\text{beam}} l_{\text{cell,std}})/(\sigma_{\text{beam,std}} l'_{\text{cell}})$
    $\alpha^{\text{ver}} \leftarrow (\delta\sigma^{\text{ver}}_{\text{beam}} l_{\text{cell,std}})/(\sigma_{\text{beam,std}} l'_{\text{cell}})$
    $\sigma_{\text{x}} \leftarrow \tan\left(\sigma^{\text{hor}}_{\text{beam}} g_{\text{approx}}(\alpha^{\text{hor}})\right)$
    $\sigma_{\text{y}} \leftarrow \tan\left(\sigma^{\text{ver}}_{\text{beam}} g_{\text{approx}}(\alpha^{\text{ver}})\right)$
    $projectedCorners \leftarrow []$
    for each corner c_i in P **do**
      $projectedCorners.add(Mc_i)$
    $D_{\text{span}} \leftarrow \text{polygon}(projectedCorners)$
    $solarPower \leftarrow solarPower +$ BIVARIATEPOLYGONINTEGRATION($D_{\text{span}}, \sigma_x, \sigma_y$)
  return $solarPower$

5 Case study

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* [44]. Afterwards, the optimal settings for the multi- and quasi-Monte Carlo method are derived. Then, the accuracy of the different ray tracers is compared against their corresponding runtime. Lastly, different acceleration strategies for the ray tracing techniques are discussed.

5.1 Validation

In this section, a cross validation of *SunFlower* with *SolTrace* for different test cases is given. The test cases are predicated on the solar tower power plant Planta Solar 10 (PS10) in Spain [29] which utilizes 624 heliostats of the type Sanlúcar 120. Figure 44 displays the positioning of the heliostats. Each heliostat consists of 28 facets with a total mirror area of about 120 m^2 [30]. But as *SolTrace* was not able to handle the total amount of facets, we simplified the heliostas to contain only a single facet.
Two receiver types, namely the cylindrical cavity receiver and the flat receiver were used in the test cases. For each receiver type, three different sun positions are evaluated in order to account for various shading and blocking effects. Table 4 shows the general setup for the test cases that were derived from the settings of a similar test case [18]. The receiver settings in Table 2 and 3 are based on the receiver used in the PS10 plant [30]. 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.

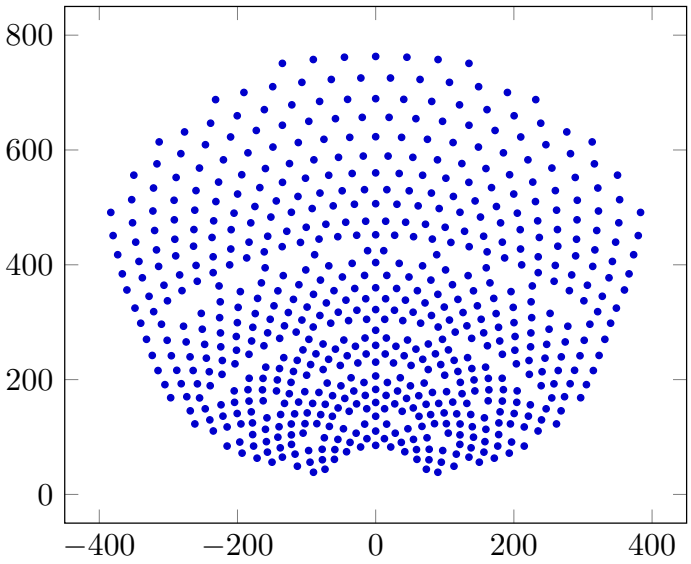


Figure 44: Heliostat field layout of the PS10 plant

Parameter	Value
Height	12 m
Width	13.78 m
Tilt angle	11.5°
Distance to towertop	2.74 m

Table 2: Parameters for the flat tilted receiver

Parameter	Value
Number of panels	4
Panel width	4.8 m
Panel height	12 m
Raise height	2 m
Distance to towertop	2.74 m

Table 3: Parameters for the cylindrical cavity receiver

Parameter	Value
Latitude	37.442400°
Longitude	-6.250188°
Sun Error	2.35 mrad
Global slope error vertical	2.6 mrad
Global slope error horizontal	2.6 mrad
Tracking error horizontal	1.3 mrad
Tracking error vertical	1.3 mrad
Heliostat	Sanlúcar 120
Heliostat reflectivity	88 %
Heliostat facet type	Flat
Tower height	115 m
Tower type	Rectangular tower
Tower length	18 m
Tower width	8 m
Tower position	(0,0)
Canting	None

Table 4: Basic setup for the six validation test cases. The settings are inspired by the PS10 plant and derived from a similar test case [18].

The sun positions as well as the direct normal irradiation (DNI) were inspired by their corresponding values at eight, ten and twelve a.m. on the 21st of June [43]. A definition of each test case is given in Table 5.

SunFlower and *SolTrace* simulated each test case ten times with roughly one million rays. The resulting total optical power was then normalized by the average power of *SunFlower* simulating ten million rays. Figure 45 shows the minimal, maximal and average results of both tools for all test cases. The exact solar powers can be found in Table 6 and 7. 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 is less than 0.07 % and the maximal fluctuations of the results of *SolTrace* is about 0.47 %, whereas for *SunFlower* it is only 0.13 %.

Test case	Receiver type	Azimuth	Altitude	DNI
1	Flat tilted	80°	30°	710 W/m^2
2	Flat tilted	110°	60°	820 W/m^2
3	Flat tilted	180°	70°	850 W/m^2
4	Cylindrical cavity	80°	30°	710 W/m^2
5	Cylindrical cavity	110°	60°	820 W/m^2
6	Cylindrical cavity	180°	70°	850 W/m^2

Table 5: Unique settings for each test case.

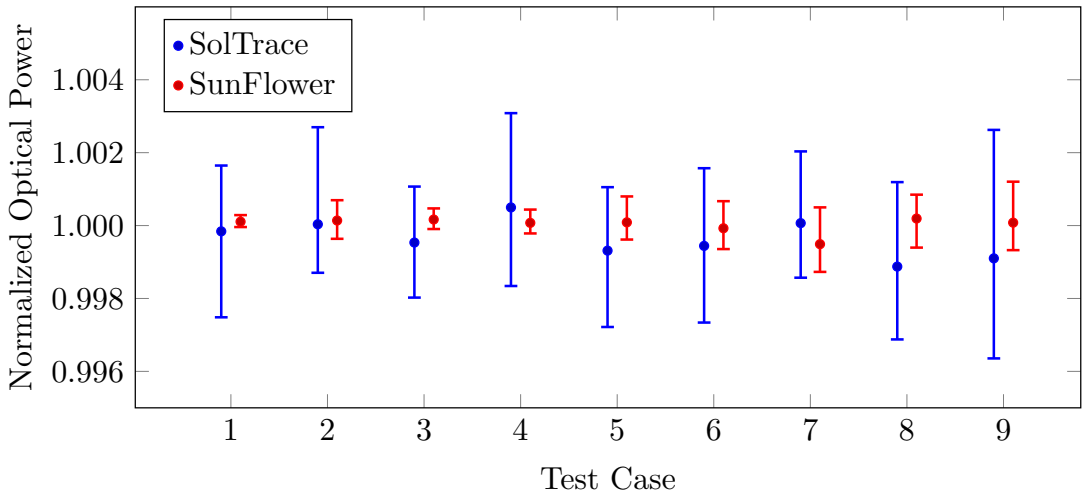
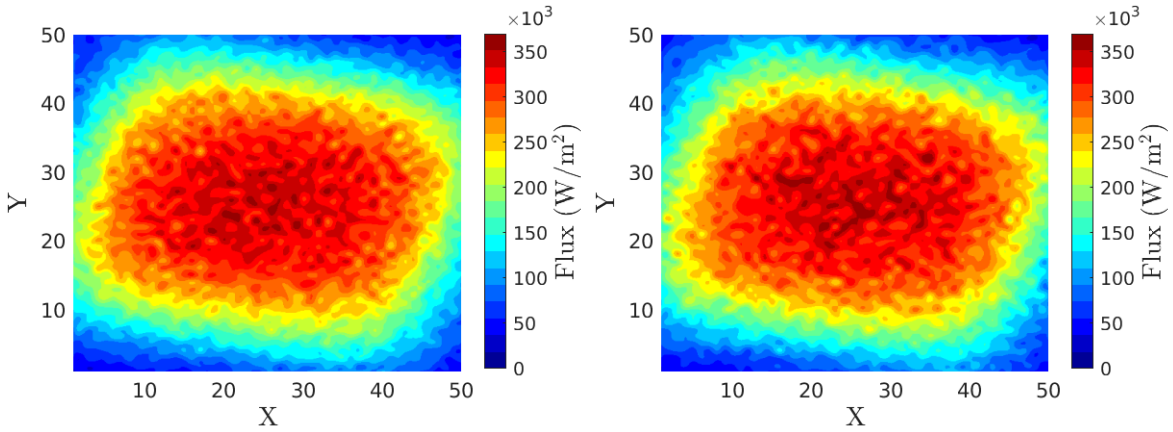


Figure 45: Normalized results of SunFlower and SolTrace for all test cases.



(a) Sunflower

(b) Soltrace

Figure 46: Interpolated fluxmaps of both tools for test case two with a total power difference of less than 0.05 %.

Testcase	**min** (MW)	**avg** (MW)	**max** (MW)
1	27.1412	27.1452	27.1501
2	36.0456	36.0637	36.0838
3	39.0685	39.0787	39.0906
4	25.5098	25.5172	25.5265
5	33.8783	33.8943	33.9184
6	36.5828	36.6037	36.6309

Table 6: Exact results of *SunFlower* for each test case.

Testcase	**min** (MW)	**avg** (MW)	**max** (MW)
1	27.074	27.138	27.187
2	36.012	36.060	36.156
3	38.995	39.054	39.114
4	25.473	25.528	25.594
5	33.797	33.868	33.927
6	36.509	36.586	36.664

Table 7: Exact results of *SolTrace* for each test case.

5.1.1 Fluxmap comparison

So far, the total solar power of both tools for different test cases has been compared. However, in order to get an understanding of how well the solar power is concentrated, not only the total sum is of interest but also the fluxmap. Thus, a short comparison is given in the following.
The fluxmaps of SolTrace and SunFlower for the second test case with 50 by 50 receiver pieces can be found in Figure 46 where X and Y are the horizontal and vertical directions of the receiver, respectively. In order to obtain a clear image, the flux values were interpolated. Figure 47 illustrated the relative difference between both fluxmaps. Except for some peaks, the relative differences stayed below ten percent. The comparatively high differences are mainly due to the very small receiver pieces which therefore have a higher sensibility to fluctuations of the ray disturbance.

5.1.2 Validation of the new integrated convolution method

In [18] the early versions of our convolution methods were tested against the classical-Monte Carlo ray tracer. However, the old Gaussian convolution results still had an error of 2 %. Moreover, at this state of development the results of the integrated Gaussian convolution ray tracer deviated more than 99 % for some test cases. In oder to verify our new developed versions of these methods, the test cases were repeated.

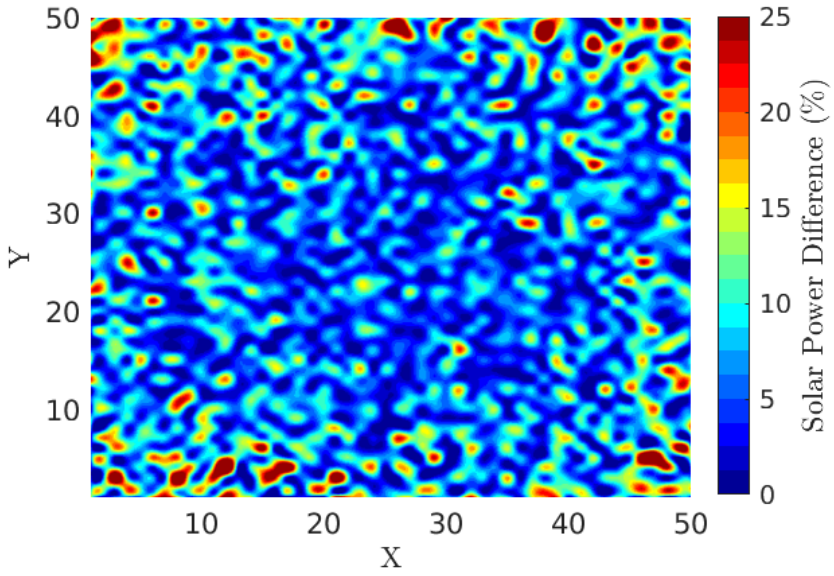


Figure 47: Relative difference between both fluxmaps.

Each test case is based on the PS10 plant and an exact definition can be found in [18]. Table 8 gives a simplified overview over the test cases. The results of our classical-Monte Carlo ray tracer were validated against *SolTrace* [18] and used to normalize the results of each ray tracer. Figure 48 shows the old results and Figure 49 the results of renewed ray tracers. For all test cases the new version of our convolution methods reached an accuracy of more than 99.9 %.

Test case	**Short description**
1'-6'	Single heliostat with different facet types and canting strategies
7'-12'	Two heliostats with different facet types and canting strategies accounting for blocking effects
13'-18'	Two heliostats with different facet types and canting strategies accounting for shading effects
19'-24'	Whole PS10 with different facet types and canting strategies

Table 8: Overview over the test cases in [18].

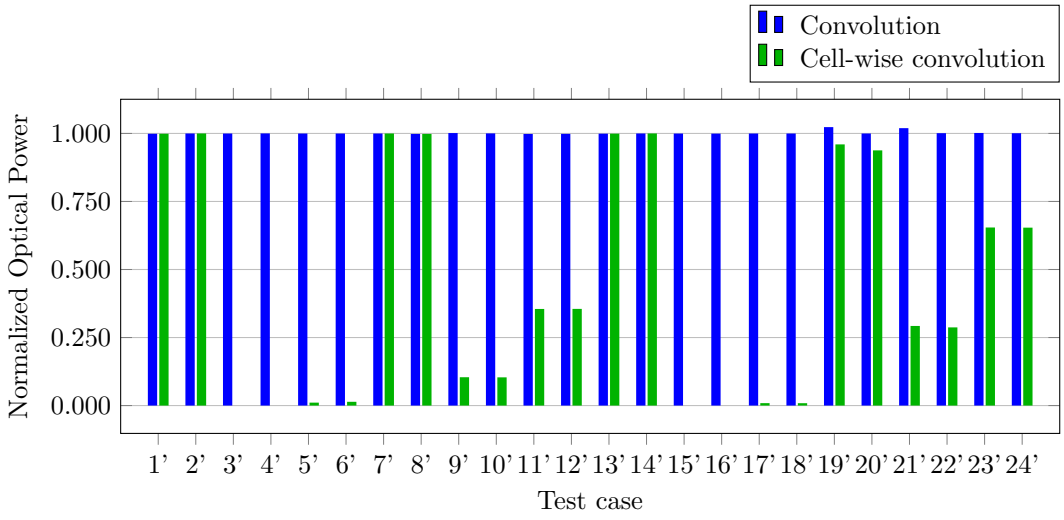


Figure 48: Old results of the convolution methods [18].

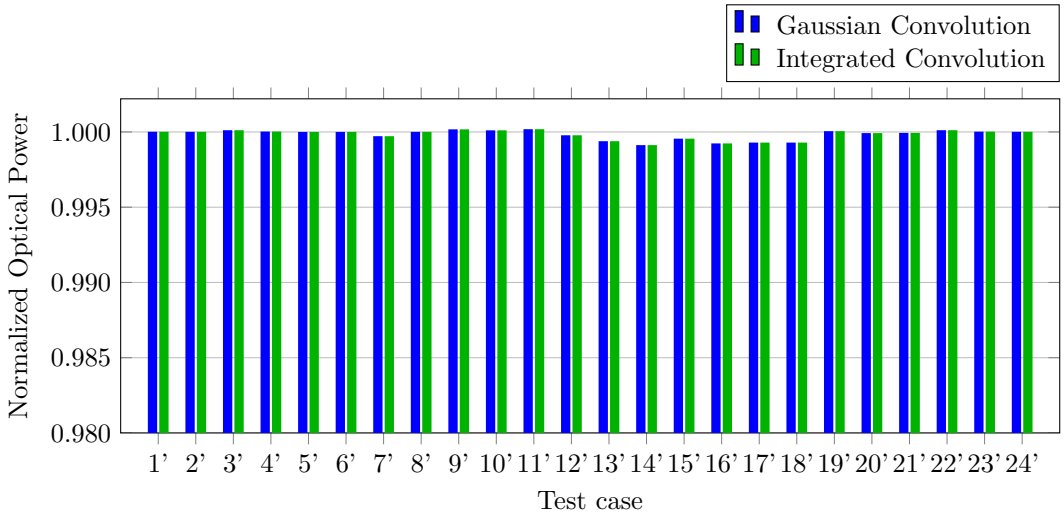


Figure 49: Results of the renewed convolution methods.

5.2 Optimal setting for quasi- and multi-Monte Carlo

As an extension to the classical-Monte Carlo approach, the multi- and quasi-Monte Carlo methods are using multiple samples of the error cone, i.e., several rays are generated when evaluating a representative ray. Therefore, an investigation on the optimal number of samples is needed. For this, the second test case of Section 5.1 is used. However, since *SunFlower* is able to handle the correct number of facets, the test case was adapted and is now referred to as test case 2^*.

To see how the number of rays influences the accuracy of the ray tracers, different

number of rays per facet were simulated. Again, the results got normalized by the average result of the classical-Monte Carlo method simulating ten million rays. Since the facets of Sanlúcar 120 are about twice as wide as they are high, twice as many rays were generated per width than per height. Each simulation was done twenty times to account for fluctuations. A comparison of the classical-Monte Carlo method with the multi-Monte Carlo method which exemplary takes five samples per ray, is given in Figure 50. Here, the shaded regions illustrate the fluctuations of each ray tracer. As shown, the multi-Monte Carlo method has less fluctuations and converges to the same limit as the classical-Monte Carlo method.

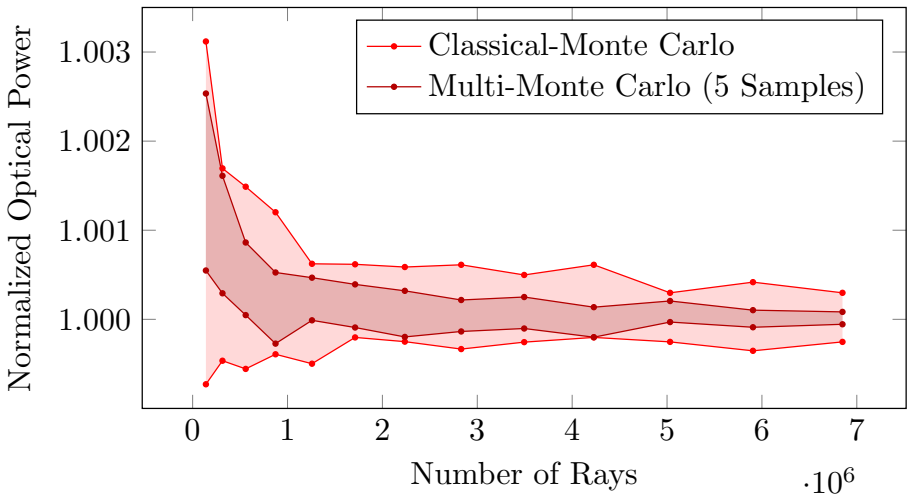


Figure 50: Fluctuations of the classical-Monte Carlo ray tracer and the multi-Monte Carlo ray tracer taking five samples per ray illustrated by the shaded region.

But the evaluation of a sample is almost as computationally expensive as the evaluation of a representative ray. Therefore, the total number of samples needs to be taken into account. In Figure 51 the average results of the multi- and quasi-Monte Carlo ray tracer each using five and one sample per ray is shown. Here, the quasi-Monte Carlo method almost always showed more accurate results than the classical-Monte Carlo method. A comparison of their fluctuations can be found in Figure 52.
However, the improvements are comparatively small.
As illustrated in Figure 51, simulating more rays has a larger impact on the accuracy than taking more samples per ray. A possible explanation is given in the following.

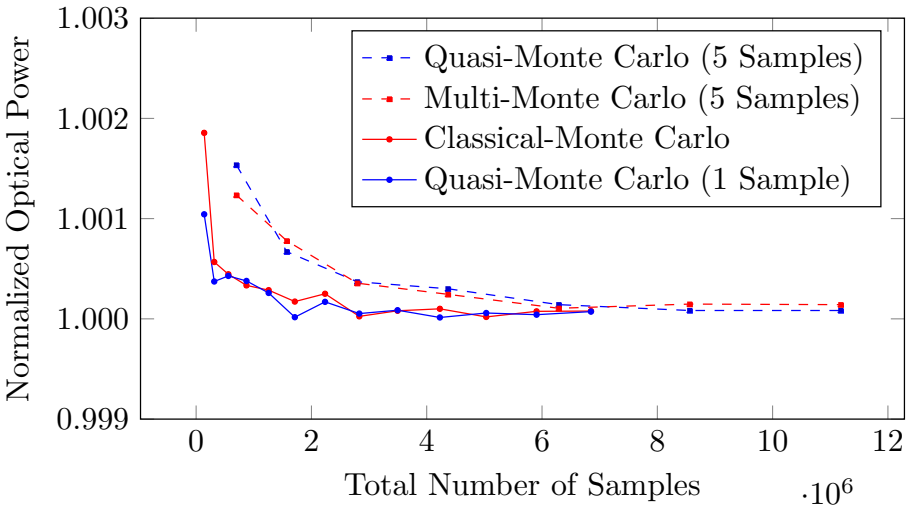


Figure 51: Average results of different Monte Carlo based ray tracers with respect to the total number of samples.

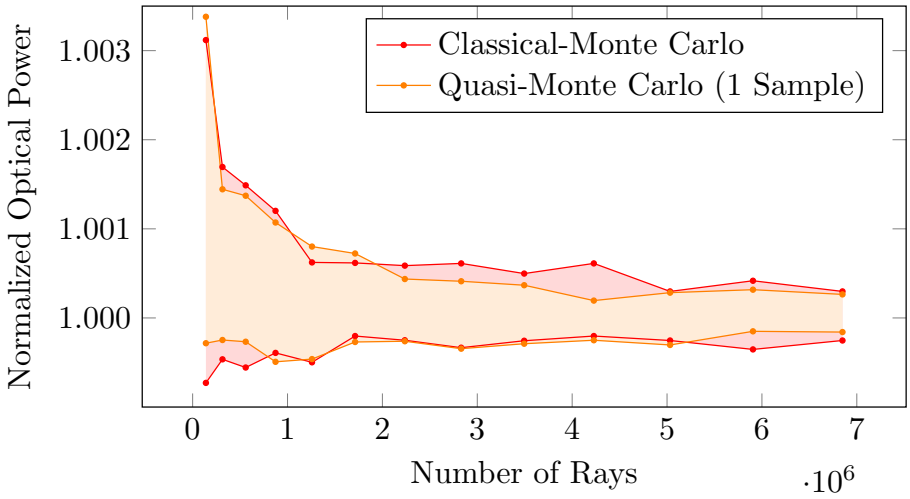


Figure 52: Fluctuations, illustrated by the shaded regions, of the classical- and quasi-Monte Carlo method using one sample per ray.

The question is whether a given heliostat cell should be discretized further or if it would make more sense to take multiple samples from the current representative ray. Taking more samples decreases the evaluation error of the corresponding error cone. On the other hand, when the cell is discretized further the error of using one ray to represent photon interaction of a cell is reduced as each ray has a smaller cell area. But as the cell areas are small and closely packed, their corresponding error cones are similar. Therefore, a reduction of the evaluation error for each error cone is also given. Thus, the total error decreases more when the heliostat is discretized into smaller cells.

In the following sections one sample is taken for each ray.

5.3 Accuracy vs runtime

Since the ray tracing methods are repeatedly used for the optimization of the heliostat field layout, a ray tracer not only needs to be accurate but also fast and precise. Therefore, in this section the runtime and achieved accuracy for each ray tracing technique is investigated. Prior to that, a cross validation of the ray tracing methods is given.

For the validation, the test cases of Section 5.1 with the correct number of facets and the effect of tower blocking will be used. The corresponding test cases are marked with a star. All ray tracers simulated each setup ten times with twenty million rays. In Figure 53 the average results of each ray tracer normalized by the average results of the classical-Monte Carlo method are given. Every ray tracer reached an accuracy of more than 99.99 % in all test cases.

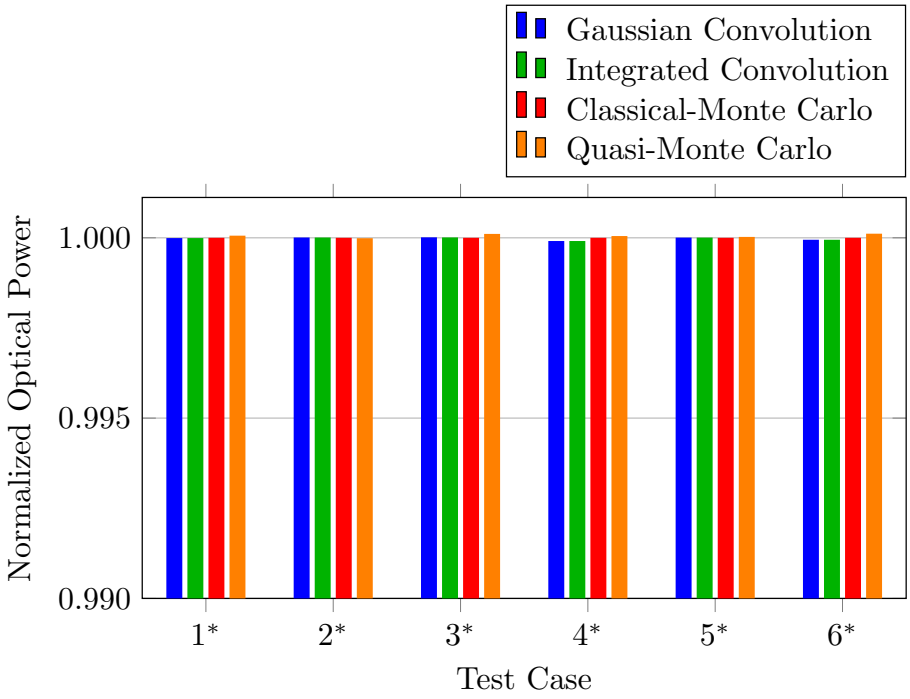


Figure 53: Average results of the different ray tracing methods for the PS10 test cases of Section 5.1 with the correct number of facets.

In order to evaluate the accuracy of the ray tracers against their runtime, different number of rays were simulated. Again, these were chosen such that the cell area of each ray is roughly quadratic. Moreover, the ray tracers ran each setup twenty times and the results were normalized by the average result of the classical-Monte Carlo method

using twenty million rays. As the difference between the classical- and quasi-Monte Carlo method was shown to be comparatively small, only the classical-Monte Carlo ray tracer was evaluated against the convolution methods. Every simulation was done on a commercial laptop with an Intel Core i7-4720HQ CPU using eight cores running at 2.60GHz each.

The normalized results for the test case 2* are given in Figure 54 and a comparison of the accuracy against the runtime in Figure 55. Fluctuations in the results of the classical-Monte Carlo method are illustrated by the shaded region. As shown, the results of the convolution methods are always within the boundaries of the classical-Monte Carlo method. Furthermore, the integrated convolution ray tracer constantly reached an accuracy of more than 99.98 % as illustrated by the dotted lines.
When a certain number of rays is reached, the Gaussian convolution method gives the exact same results as the integrated convolution method. Considering the definition of our sigma multiplier function this is the expected behavior.

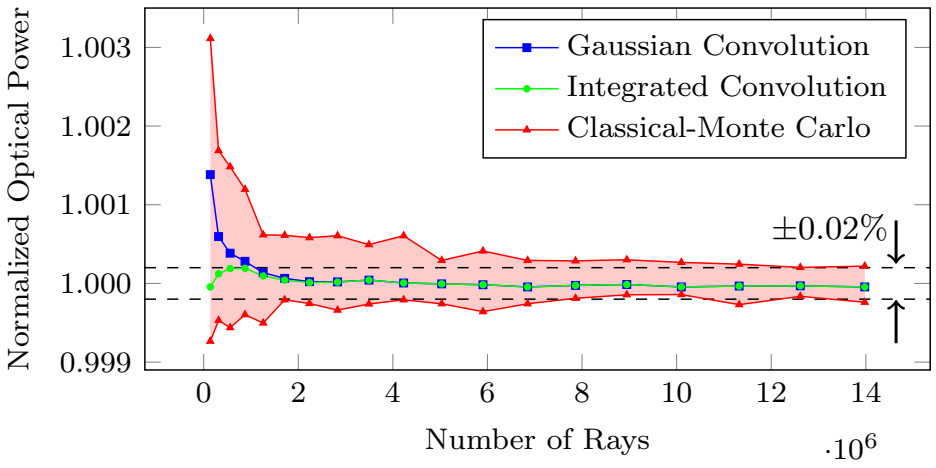


Figure 54: Average results of different ray tracing techniques for test case 2* in dependence on the total number of rays.

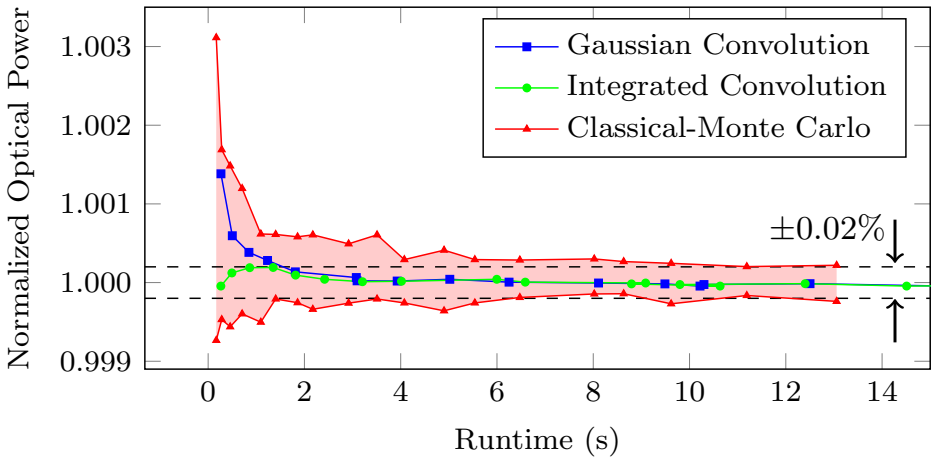


Figure 55: Average results of different ray tracing techniques for test case 2* in dependence on the total runtime.

The same evaluation was done for the test case 6* to account for a cavity receiver as well as a different sun position, see Figure 56 and 57. Here, similar results were obtained but since the cavity receiver has four receiver pieces the runtime of the convolution methods increased.

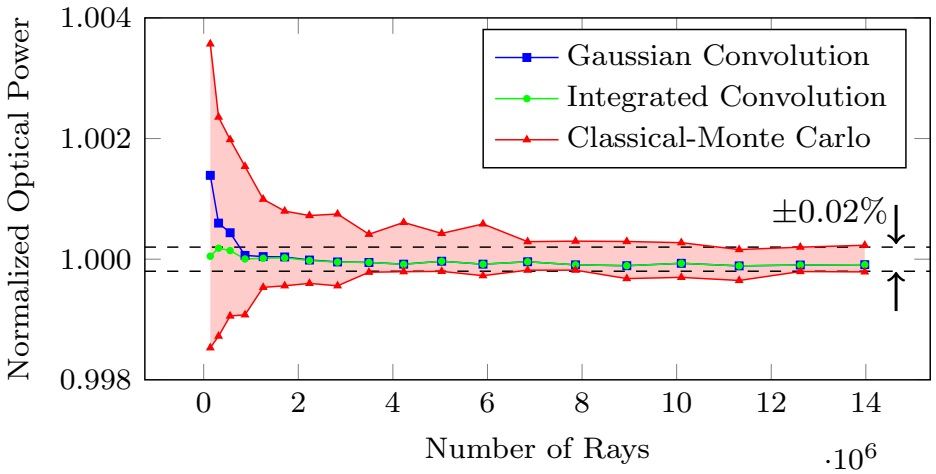


Figure 56: Average results of different ray tracing techniques for test case 6* in dependence on the total number of rays.

In all four figures a major problem of the classical-Monte Carlo method can be seen. Due to its non determinism, even when simulating 14 million rays the result still has an uncertainty of 0.02 %. Therefore, when used in an optimization, an increase or decrease of 0.04 % in solar power could also be caused by fluctuations instead of a better positioning. Furthermore, as a lot of simulations are done within an optimization

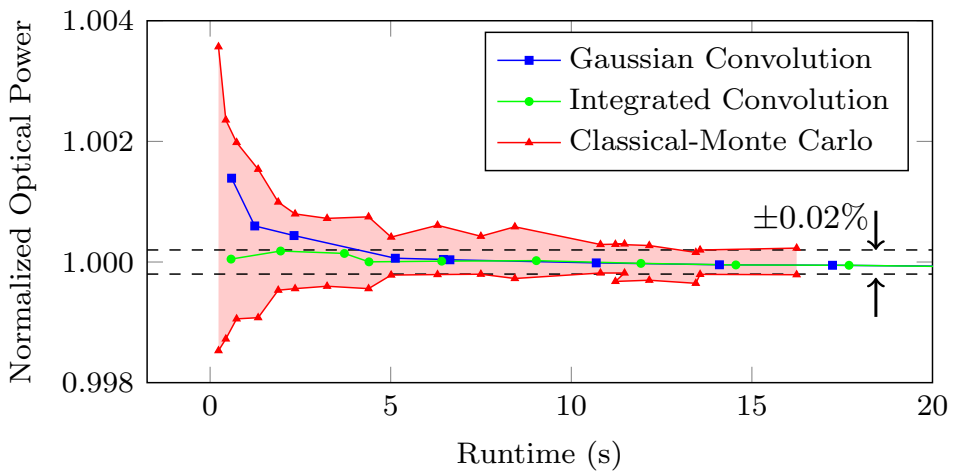


Figure 57: Average results of different ray tracing techniques for test case 6* in dependence on the runtime.

process, usually not more than a million rays are used for each simulation. The corresponding uncertainty for the test cases 2* and 6* averages at around 0.1 % meaning that an increase or decrease of 0.2 % in solar power might only be caused by fluctuations. Therefore, an optimizer interpreting an increase of 0.2 % in solar power as an improvement, can be guided into an incorrect direction. Thus, any optimization within a certain percentage requires the results to be at least twice as accurate which indicates the main advantage of analytical ray tracing methods. Since they are deterministic, the results for a given setup are identical and thus do not fluctuate. Besides the precision, our convolution methods have also been shown to reach a high accuracy in a comparatively small time. For this reason, they are especially useful in an optimization process.

5.4 Characteristics of the convolution methods

In this section different characteristics of the convolution methods are viewed. To do so, one heliostat close and one far from the tower of test case 2* were tested, see Figure 58. Furthermore, each setup was simulated with a varying number of rays and the results were normalized as in the last sections.

Because of the behavior of our derived sigma multiplier function, we would expect the accuracy of both convolution methods to increase with the distance. This assumption is supported by Figure 59.

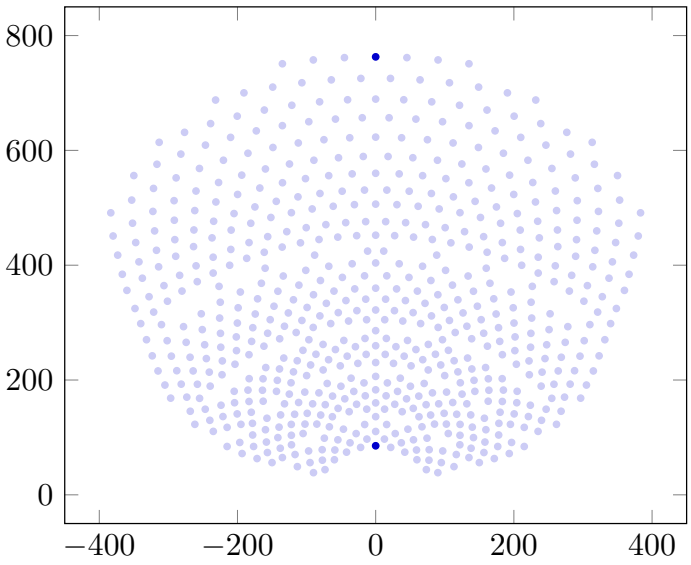


Figure 58: Positions of the close and far heliostats.

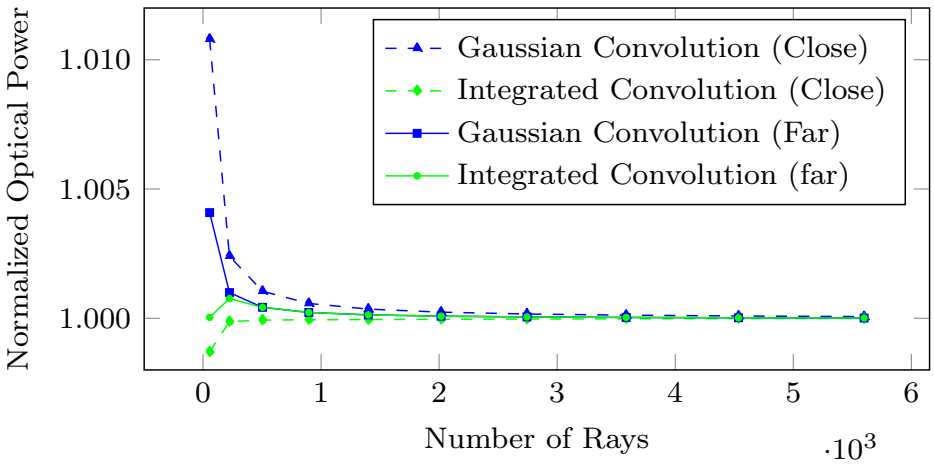


Figure 59: Normalized results of both convolution methods for the far and close heliostat.

As stated in Section 4.2 our Gaussian convolution method is expected to have an exact evaluation of each error cone. In order to validate the statement, the results of the Gaussian convolution method is compared to the average result from ten runs of the multi-Monte Carlo method with one thousand samples per ray. The test was done for the close heliostat to include as much deviation to the actual solar power as possible. For each setup the resulting solar power from both method differed at most 0.025 %, see Figure 60.

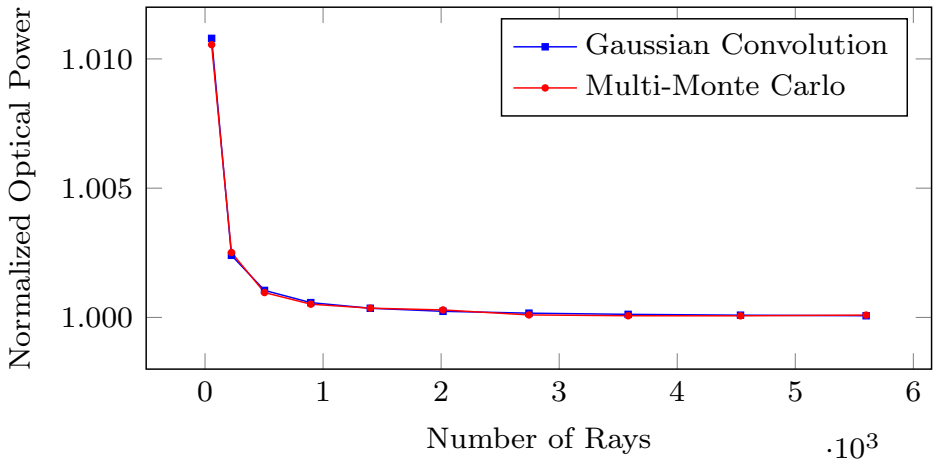


Figure 60: Average results of the multi-Monte Carlo method using thousand samples per ray and the Gaussian convolution method.

5.5 Acceleration of the ray tracers

The ray tracers used in the last sections were already accelerated with different techniques. However, to see the influence of each acceleration they are switched off and gradually back on again in the following. The simulations were done using the classical-Monte Carlo ray tracer as well as the Gaussian convolution ray tracer on test case 2*. Furthermore, the same laptop as in Section 5.3 was utilized.

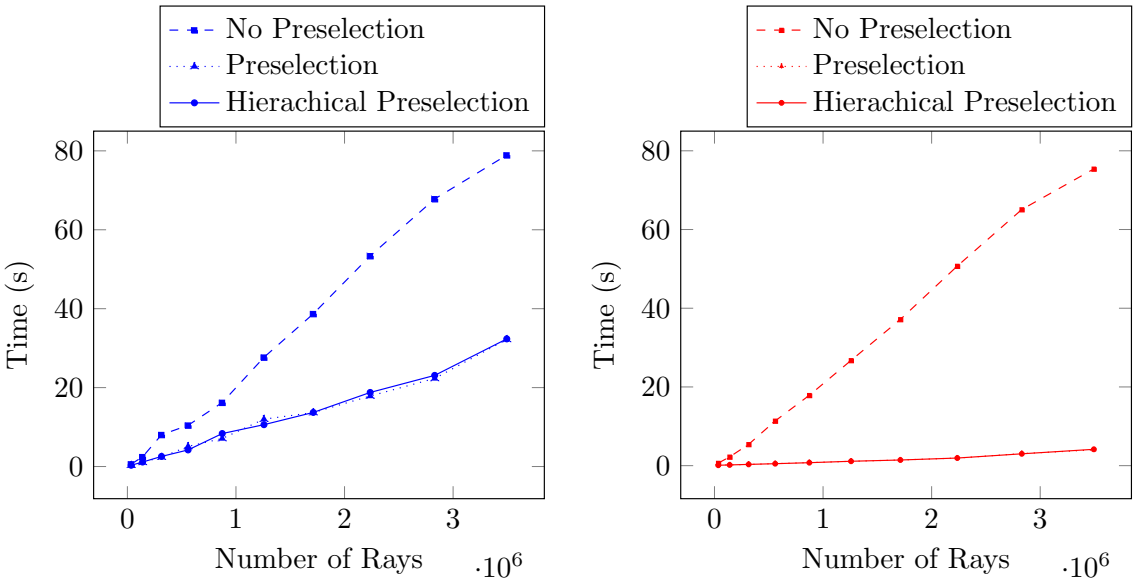
5.5.1 Preselection

In Section 2.4 a method to compute sets of potentially blocking and shading heliostats was discussed. These sets determine for each heliostat the subset of heliostats that might shade or block them. Instead of checking every ray against every heliostat for blocking and shading effects, only the subset is tested. As an extension to this, a hierarchical preselection calculates the subset of potentially blocking and shading heliostats for each facet. Figure 61 shows the runtime improvement of both acceleration strategies.

The classical-Monte Carlo method benefited more from the preselection than the Gaussian convolution method. When simulating roughly 3.5 million rays with the classical-Monte Carlo ray tracer, the preselection decreased runtime by a factor of 18. Both methods were only slightly improved by the hierarchical preselection.

5.5.2 CPU parallelization

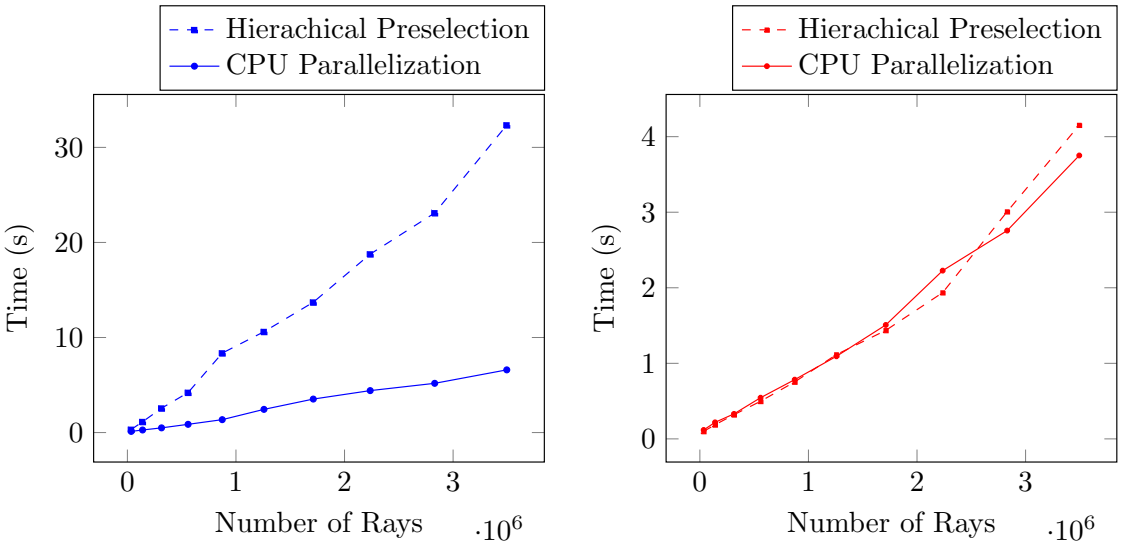
So far, only the generation of representative rays was parallelized on the central processing unit (CPU). The evaluation of each ray was done in a critical section which



(a) Gaussian Convolution (b) Classical-Monte Carlo

Figure 61: Acceleration of a Monte Carlo and a convolution based ray tracer using the two preselection techniques on test case 2*.

especially effects the convolution methods as they put more computational effort into it. By limiting the critical section to the necessary code segments, this disadvantage has been reduced. The results can be found in Figure 62.



(a) Gaussian Convolution (b) Classical-Monte Carlo

Figure 62: Acceleration of a Monte Carlo and a convolution based ray tracer using a CPU parallelization with a minimal critical section on test case 2*.

6 Conclusion and Outlook

6.1 Conclusion

A fast but also accurate ray tracer is a key part for optimizing the heliostat field layout, as a lot of simulations are needed throughout the optimization process. In contrast to other tools, *SunFlower* has the benefit of being able to use a variety of ray tracers. For test cases that were based on the solar power tower plant PS10, the results of our classical-Monte Carlo ray tracer always stayed within the range of the results of *SolTrace*. Moreover, on similar but more realistic test cases all other developed ray tracers showed an accuracy of at least 99.99 % compared to the classical-Monte Carlo results. Furthermore, problems with the early versions of our convolution methods were eliminated. A small improvement to the classical-Monte Carlo method has been made by introducing a quasi-Monte Carlo ray tracer. However, taking multiple samples per ray as done in the multi-Monte Carlo method is not recommended as it increases the processing time more than the accuracy benefits from it.
On the other hand, the multi-Monte Carlo ray tracer helped us to validate that the Gaussian convolution method evaluates each ray perturbation accurately. Besides that, our integrated convolution ray tracer reached an accuracy of more than 99.98 % in less than 0.6 seconds on test case 2^* and 6^* where the classical-Monte Carlo method required at least 13 seconds to overcome their fluctuations. Due to the determinism, high accuracy and comparatively small processing time, the integrated convolution method is a useful alternative to a Monte Carlo ray tracer. Especially for an optimization it is the recommended ray tracer as it produces accurate and reliable results.

6.2 Outlook

Despite the challenges we have overcome during the development of our convolution ray tracers there are still improvements possible. In the following, strategies to accelerate the ray tracers as well as approaches to generalize them will be discussed.

6.2.1 Distant dependent heliostat discretization

As shown in Section 4.3.1 and 5.4, the accuracy of both convolution ray tracers increases with the distance. Currently, for all heliostats the same number of rays is generated per facet. However, by decreasing the number of rays for far distant heliostats, the total number of rays and thus the runtime can potentially be reduced without losing accuracy. To do so, a rule on how each heliostat should be discretized is needed. Our suggestion is to choose the discretization such that certain sigma regions of two neighboring cells just intersect, see Figure 63. Therefore, the cell lengths should fulfill

$$l_{\text{cell}} = 2\delta \tan\left(a\sigma_{\text{beam}}\right), \tag{6.1}$$

for some value $a \in \mathbb{R}$.
However, a closer investigation is needed in order to develop a reasonable distant dependent discretization.

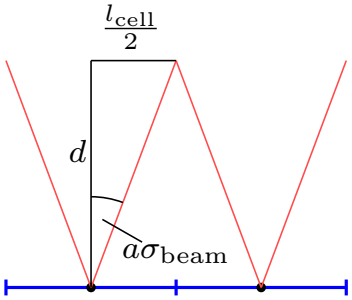


Figure 63: Illustration of the required cell length such that the a-sigma regions of two neighboring cells intersect.

6.2.2 Combined receiver cells

Since the main aim of our convolution methods is the fast and accurate calculation of the total solar power, the receiver pieces of the cylindrical receivers could be combined to an overall receiver piece. So far, the simulation of the cylindrical receivers required the integration over several receiver pieces for each ray. But by combining them to one receiver piece, the computational effort involved in integrating over multiple receiver pieces is reduced to the costs of integrating over a single larger receiver piece. The resulting projection of that piece is not a convex polygon anymore but as described in [17], the polygon integration method can be extended to handle arbitrary polygons.

6.2.3 Reversed convolution methods

In the early version of the integrated convolution method described in Richter et. al [33, 35], the flux was viewed as what the receiver would 'see' from the current heliostat. Based on this idea, a reversed version of each convolution method can also be developed.

Consider the situation drawn in Figure 64. As illustrated, the angle α between the perfect reflected ray and a perturbed ray is the same as the angle between the reversed version of the perfect ray and the corresponding perturbed ray. Here, the reversed version of the perfect reflected ray originates at the receiver piece center and travels in $-\vec{r}$ direction. Similarly, a reversed ray image plane for the reversed ray exists. Therefore, an integration over a representative region D_{heli} on that plane gives the relative average likelihood $L_{\text{hit}}^{\text{avg}}$ of a ray to be perturbed such that it intersects the receiver center. By assuming that the receiver cell has the same likelihood over its area A_{rec} the solar power P_{rec} emitted from the current heliostat cell onto the receiver is

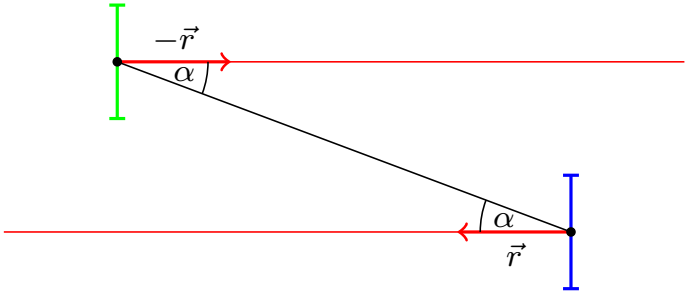


Figure 64: Illustration of a reversed ray and its correlation to the original ray. The reversed ray originates from the receiver piece and travels in $-\vec{r}$ direction.

$$P_{\text{rec}} = P_{\text{ray}} P_{\text{hit}}^{\text{avg}} \approx P_{\text{ray}} A_{\text{rec}} L_{\text{hit}}^{\text{avg}} = P_{\text{ray}} A_{\text{rec}} \iint\limits_{D_{\text{heli}}} f(x, y)\, dA, \tag{6.2}$$

with $f(x, y)$ as the bivariate normal distribution of Section 4.1.

Equation (6.2) is the foundation of the reversed Gaussian convolution method. Applying the same argument as for the integrated convolution method, a reversed version of it can also be formulated

$$P_{\text{rec}} = P_{\text{ray}} P_{\text{hit}}^{\text{avg}} = P_{\text{ray}} \iint\limits_{D_{\text{heli}}} F(x, y)\, dA, \tag{6.3}$$

with $F(x, y)$ as the two dimensional integrated Gaussian distribution of Section 4.3.

Again, the integral in Equation (6.3) can be approximated as described in Section 4.3.1. However, now the dimensions of the receiver piece is used in the approximation process as the reversed rays originate from the receiver. Because the error of our approximation depends on the area from which the rays originate, the reversed convolution methods are especially useful when the heliostat facet area is larger than the receiver piece area. Not only does an increase of the receiver pieces improves the accuracy of the reversed convolution methods, it also gives more information about how the solar power is distributed.

6.2.4 GPU Parallelization

As stated in [10], tools such as TieSol [11] that use the graphic processing unit (GPU) for the ray tracing process are extremely fast. For this reason and as our convolution methods are based on techniques from computer graphics, a parallelization on the GPU would be a useful development. But since our code currently relies on a lot of data structures from CGAL [42], there are some adaptations required to achieve an

appropriate GPU parallelization. First of all, most data structures used in the ray tracers need to be rewritten such that they require as little memory as possible. This is crucial because the GPU has a very limited storage capacity [27]. Furthermore, there are some functions of CGAL which are used during the ray tracing process and thus have to be adapted to the new data structures. Lastly, the parallelization itself can also be done in a lot of different ways. Our suggestion for this is to use OpenACC [14] in combination with the PGI [23] compiler because of its simple implementation and wide application.

6.2.5 Other distributions

Throughout the thesis, we modeled all errors by Gaussian distributions. But at least for the sun shape, it is also common to use other distributions such as Pillbox or Buie [4]. Theoretically, a corresponding convolution and integrated convolution method for those distributions can be developed. However, this requires an accurate and fast method to solve the integral of the convoluted distribution over a polygon.
Another approach would be to approximate the convoluted distribution with a Gaussian distribution based on the Central Limit Theorem [40]. A parameter dependent approximation as used in the integrated convolution ray tracer can also be developed. Moreover, a semi-deterministic version of the convoluted methods might also be a reasonable alternative. Here, the sun vector gets perturbed by the corresponding distribution and the tracking and slope error of the resulting ray are evaluated using the Gaussian or integrated convolution method.

References

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

[2] Milton Abramowitz and Irene A. Stegun. *Handbook of Mathematical Functions with Formulas, Graphs, and Mathematical Tables*. Dover, New York, ninth dover printing, tenth gpo printing edition, 1964.

[3] Nils Ahlbrink, Boris Belhomme, Robert Flesch, Daniel Maldonado Quinto, Amadeus Rong, and Peter 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.

[4] Germain Augsburger and Daniel Favrat. Modelling of the receiver transient flux distribution due to cloud passages on a solar tower thermal power plant. *Solar Energy*, 87:42 – 52, 2013.

[5] Omar Behar, Abdallah Khellaf, and Kamal Mohammedi. A review of studies on central receiver solar thermal power plants. *Renewable and sustainable energy reviews*, 23:12–39, 2013.

[6] Omar Behar, Abdallah Khellaf, and Kamal Mohammedi. A review of studies on central receiver solar thermal power plants. *Renewable and Sustainable Energy Reviews*, 23:12–39, 07 2013.

[7] Boris Belhomme, Robert Pitz-Paal, Peter Schwarzbözl, and Steffen Ulmer. A new fast ray tracing tool for high-precision simulation of heliostat fields. *Journal of Solar Energy Engineering*, 131(3):031002, 2009.

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

[9] Manuel J. 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] Sebastian-James Bode and Paul Gauché. Review of optical software for use in concentrating solar power systems. In *Proceedings of South African Solar Energy Conference*, 2012.

[11] Sebastian-James Bode and Paul Gauché. Review of optical software for use in concentrating solar power systems. In *Proceedings of South African Solar Energy Conference*, 2012.

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

[13] Russel E. Caflisch. Monte carlo and quasi-monte carlo methods. *Acta Numerica*, 7:1–49, 1998. doi: 10.1017/S0962492900002804.

[14] Sunita Chandrasekaran and Guido Juckeland. *OpenACC for Programmers: Concepts and Strategies*. Addison-Wesley Professional, 1st edition, 2017. ISBN 0134694287, 9780134694283.

[15] Hongmei Chi, Peter Beerli, Deidre W. Evans, and Michael Mascagni. On the scrambled soboĺ sequence. pages 775–782, 2005.

[16] Francisco J. 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.

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

[18] Linus Franke. Modelling and optimization of large scale solar tower power plants. Master's thesis, 2018.

[19] Pierre Garcia, Alain Ferriere, and Jean-Jacques Bezian. Codes for solar flux calculation dedicated to central receiver system applications: A comparative review. *Solar Energy*, 82(3):189–197, 2008.

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

[21] Bruce 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.

[22] P.L. Leary and J.D. 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.

[23] B Lebacki, Michael Wolfe, and Douglas Miles. The pgi fortran and c99 openacc compilers. *Cray User Group*, 2012.

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

[25] Jean H. Meeus. *Astronomical Algorithms*. Willmann-Bell, Incorporated, 1991.

[26] Nicholas Metropolis and S. Ulam. The monte carlo method. *Journal of the American Statistical Association*, 44(247):335–341, 1949. doi: 10.1080/01621459.1949.10483310.

[27] Carlos Schmidt Muniz. Gpu ray tracer for solar tower power plants, 2019.

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

[29] Rafael Osunaa, Rafael Olavarriaa, Rafael Morilloa, Marcelino Sancheza, Felipe Canteroa, Valerio Fernandez-Queroa, Pedro Roblesb, Teodoro Lopez del Cerrob, Antonio Estebanb, Francisco Ceronb, Juan Talegonc, Manuel Romerod, Felix Tellezd, MaJesus Marcosd, Diego Martineze, Antonio Valverdee, Rafael Monterreale, Robert Pitz-Paalf, George Brakmanng, and Manuel Silva. Ps10, construction of a 11mw solar thermal tower plant in seville, spain. 01 2006.

[30] R Pitz-Paal, J Dersch, and B Milow. European concentrated solar thermal roadmapping (ecostar): roadmap document. *ECOSTAR, SES6-CT-2003–502578*, 2005.

[31] 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.

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

[33] Pascal 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.

[34] Pascal Richter, Erika Ábrahám, and Martin Frank. Multi-objective optimization of solar tower heliostat fields. 06 2014.

[35] Pascal Richter, Gregor Heiming, Nils Lukas, and Martin 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.

[36] K. F. Riley, M. P. Hobson, and S. J. Bence. Mathematical methods for physics and engineering. *American Journal of Physics*, 67(2):165–169, 1999. doi: 10.1119/1.19216.

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

[38] Mario Botsch Sandip Sar-Dessai. Lecture notes on computer graphics i, May 2005.

[39] 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.

[40] 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.

[41] Peter Schwarzbözl, Robert Pitz-Paal, and Mark Schmitz. Visual hflcal - a software tool for layout and optimisation of heliostat fields. In *SolarPACES Conference*, 2009.

[42] The CGAL Project. *CGAL User and Reference Manual*. CGAL Editorial Board, 4.14 edition, 2019.

[43] Janna Tinnes. Acceleration of ray-tracer models for the simulation of solar tower power plants with real weather data, 2017.

[44] Tim 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.