SMARTI-

Solar Matlab Ray Tracing Implementation

*Manual:*

Inhalt

[1 Future tasks 1](#_Toc496790037)

[1.1 Transmission through all layers 1](#_Toc496790038)

[1.2 Implementing random pyramides 1](#_Toc496790039)

[1.3 Implementing other geometries 1](#_Toc496790040)

[1.3.1 Isotexture 1](#_Toc496790041)

[1.3.2 Gaussian 1](#_Toc496790042)

[1.3.3 Corner cube 1](#_Toc496790043)

[1.4 Warnings for wrong input 1](#_Toc496790044)

[1.5 Creating GUI 1](#_Toc496790045)

[1.6 Calculating Error 1](#_Toc496790046)

[1.7 Implementing Error as stop 1](#_Toc496790047)

[1.8 Implementing stop buttom 1](#_Toc496790048)

[1.9 Speeding up the simulation 1](#_Toc496790049)

[1.10 Easy layer stacking => as many layers as needed 1](#_Toc496790050)

[2 Description of elementary algorythms 1](#_Toc496790051)

[2.1 Intersection of ray with triangles 1](#_Toc496790052)

[2.2 Skipping side intersections => “Booster” 2](#_Toc496790053)

[2.3 Calculating solution path for generation profiles 3](#_Toc496790054)

[2.4 Implementation of finger and busbar reflection and absorption 3](#_Toc496790055)

[2.5 Creation of the Generation profile 5](#_Toc496790056)

[2.5.1 Calculating the right depth scale 5](#_Toc496790057)

[2.6 Random pyramids algorithm 7](#_Toc496790058)

[2.7 Errors and interrupts 8](#_Toc496790059)

[3 Documented bugs 8](#_Toc496790060)

[4 History 8](#_Toc496790061)

# Getting started

## Workflow: Performing a simulation

1. Modify parameters in file **input\_parameters.m**
   1. There are six batch variable (**batch\_1** to **batch\_6**) that can be used to run batch simulations of up to six different parameters at a time. For each of these parameters as many inputs can be given as desired.   
      *Example*: Simulating a solar cell with two different thicknesses of AR-coating (e.g. 75nm and 100nm) at two different angles of incidence (0° and 60°) can be done by setting **batch\_1=[75 100]** and **batch\_2=[0 60]**. Now the two variables have to be written in the corresponding space for **d\_AR** and **theta**.
   2. The number of layers is set to four as standard. Adding or reducing the number of layers can be done by adding or subtracting a value to the array of the following parameters:  
      **- layer\_mat  
      - AR\_mat  
      - d\_AR  
      - S\_geom  
      - p\_bot  
      - d\_layer  
      - lambert  
      - grid\_pos  
      - detect\_bot  
      - detect\_top**
   3. If one of the desired materials is not in the list of materials that are loaded regularly in the **Mat** variable the excel sheet **All\_data\_PV-Lighthouse.xlsx** in the data folder can be opened to check if the desired material can be found there. To include the material either an unused material can be replaced or additional Mat inputs can be created. The specification for the new material needs to follow the form: **{'GeneralName', 'SpecificName', ‘[Source]’}**
2. Run the **SMARTI.m** function
3. Use the corresponding functions in the **Analyzing-tools** folder to plot the simulated data
   1. Plot the distribution of the reflection, absorption and transmission processes
   2. Generate the EQE and the IQE curves using the function **from\_G\_to\_j.m**

# Description of elementary algorythms

## Intersection of ray with triangles

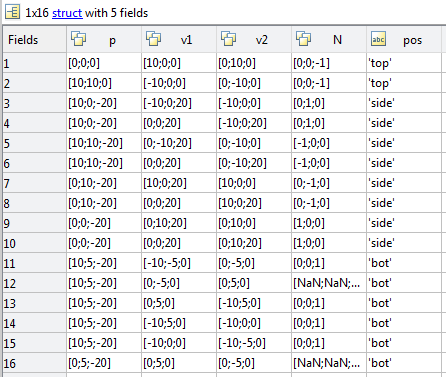
[rst\_struct hit]=intersect(p,P,S,rst\_struct\_init)

Where:

p: Vector indicating start position of ray

P: Vector indicating direction of ray

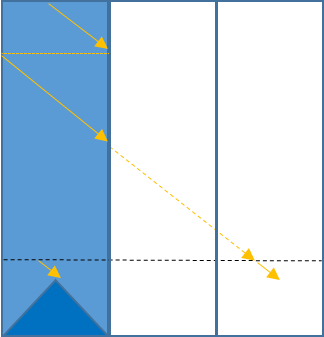
S: Struct including fields for triangles (corner point: p, triangle side 1: v1, triangle side 2: v2, normal vectors N, positions: pos)



## Supported geometries

## Skipping side intersections => “Booster”

* It can be switched on or off in layer.mat => Booster=1 or 0;
* The algorithm starts when there is a second side hit (side\_hit==2)
  + side\_hit is reset at every hit of the top or bottom
* A layer is defined, that is located 0.1 µm above the top of the pyramid
  + For the inverse case (ray pointing upwards) the layer is defined to be 0.1 µm beneath the top pyramide



* The rst values for the ray/layer intersection are then calculated
* It is then checked if the ray is absorbed and if so the z position of the absorption is stored
* If the ray is not absorbed, the hit point is projected back into the unit cell

## Calculating solution path for generation profiles

* The solution path is different from the dimension z for 1/6 the pyramide hight
* Two different regions for path calculation are distinguished

|  |  |  |
| --- | --- | --- |
|  |  | () |

|  |  |  |
| --- | --- | --- |
|  |  | () |

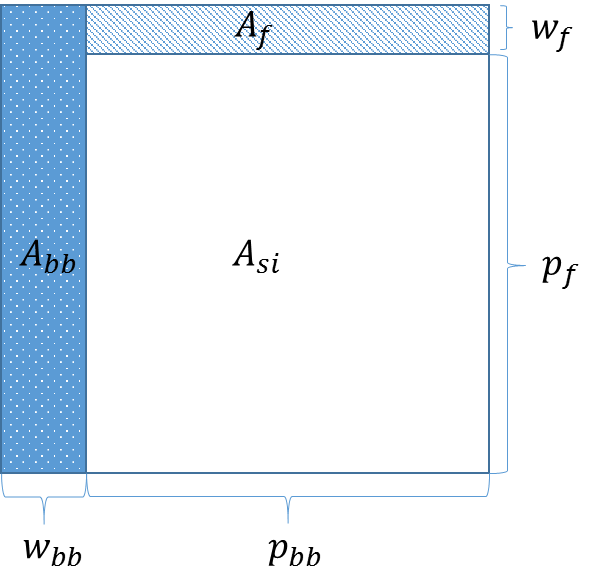
|  |  |  |
| --- | --- | --- |
|  |  | () |

|  |  |  |
| --- | --- | --- |
|  |  | () |

|  |  |  |
| --- | --- | --- |
|  |  | () |

## Implementation of finger and busbar reflection and absorption

* Calculation of the finger and bus bar share of the total module area



Share of busbars:

Share of fingers:

Storing grid position, grid share and grid material:

grid.pos={0 0 1 0}; %On top of the layer

grid.share={Mbb Mf};

grid.mat={Ag Al};

In the layer function at every top and bottom intersection the same algorithm is applied

Two random numbers are created.

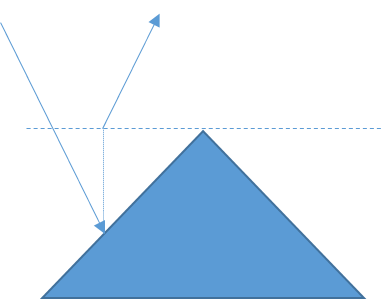
The first is used to decide if the ray hits the finger, the busbar or the cell.

The second is then used in the case of a finger or the busbar being hitten, to decide if the ray is reflected or absorbed in the material.

The cases of hitting the fingers or busbars from above or from beneath is differentiated.

Case ‘top’: The reflection angle is calculated by ‘angle of reflection equals angle of incidence’

Case ‘bot’: The reflected ray is projected on the horizontal plane at the peak of the texture. There the angle of reflection is calculated as if the ray were hitting a horizontal plane.



It is checked if the ray after hitting the texture is reflected again onto the same texture. If this is the case, the grid algorithm is inactivated.

## Programming the Lambertian radiation produced by a surface

## Algorythm for the intercection

## Creation of the Generation profile

### Calculating the right depth scale

The generation profile is created to be used in the semiconductor simulation tool PC1D. There a cumulative profile is needed. Here the correct creation of the profile at the front side is of great importance as it strongly effects the front side recombination and thus the IQE. This is especially true for short wavelengths. The creation of the e/h-pairs takes place close to the surface, while in the bulk there is no more generation so that the slope of the cumulative generation profile is very small.

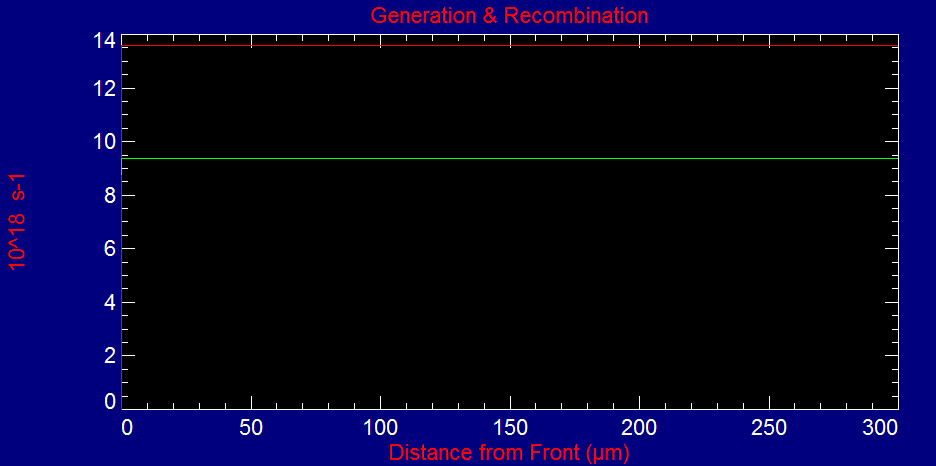


Abbildung 1: Generation profile for 300nm (red), The green line ist he cumulative recombination.

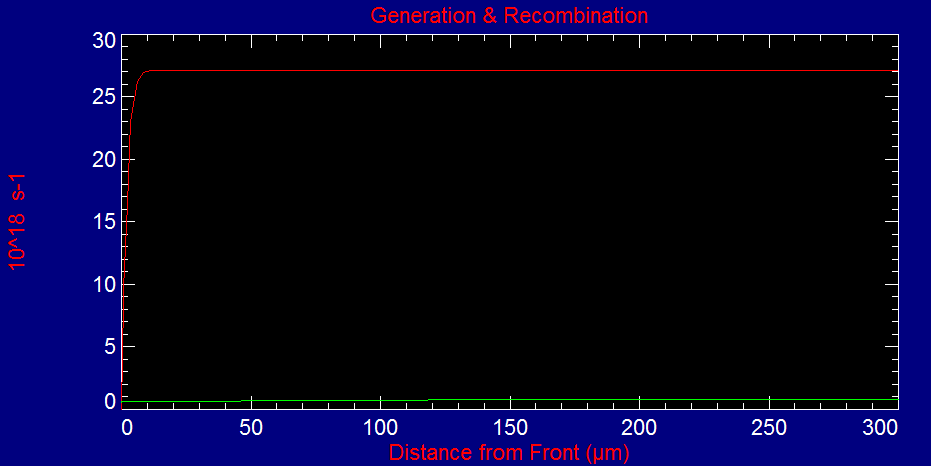


Abbildung : 600nm

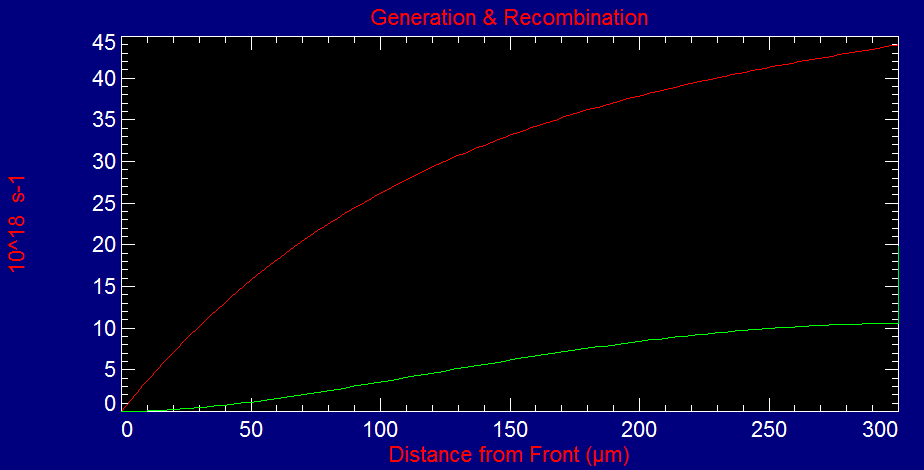


Abbildung : 1000nm

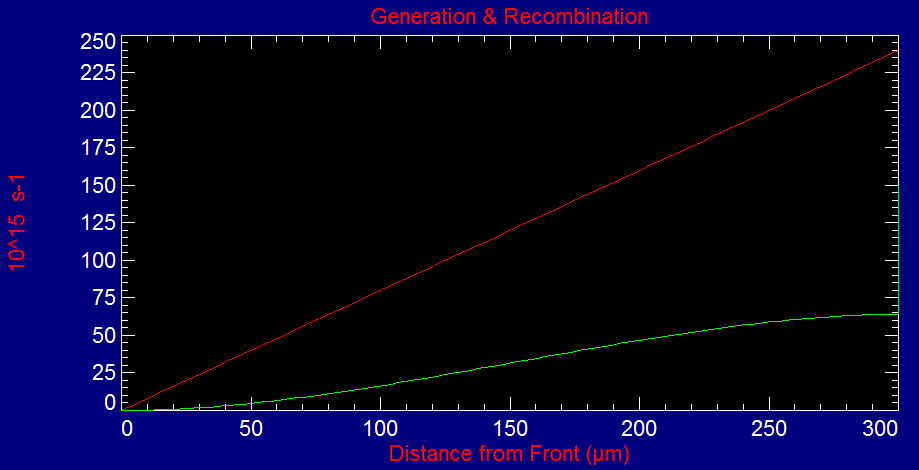


Abbildung 4: 1200nm

This is why we can choose another depth scale for the generation profile for wavelength >900nm.

*Importing the profile to PC1D:*

“The Disk File option allows you to enter the photogeneration rate as a function of position within the device directly. These files are standard ASCII files having filenames of the form \*.GEN. Each line of the file contains two values, separated by one or more spaces. The first value is the distance from the front surface, in cm. The second value is the cumulative photogeneration between the front surface and this position, in units of electron-hole pairs per second. Both the position and photogeneration values must be monotonically increasing functions within the file, and both must start with a value of 0.0 on the first line.

Ideally, the photogeneration values provided in the file should be scaled to correspond to 1 W/cm total input area power density, because the file values will be scaled by the specified light intensity (Section 3.17) when PC-ID sets up the photogeneration profile. If the range of positions included in the file is less than the device thickness, the final photogeneration value is extended for the rest of the thickness, in effect providing no additional photogeneration beyond that position. Only part of the generation-file information is used if the device thickness is less than the position range provided in the file.”

Two depth definitions are separated (higher and lower 960nm).

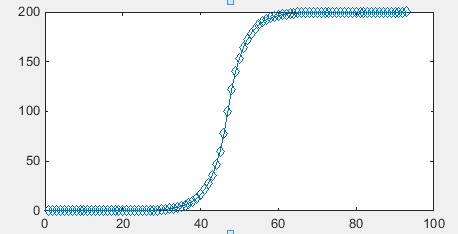


Abbildung 5: depth steps for the 90 steps for a thickness of 200nm

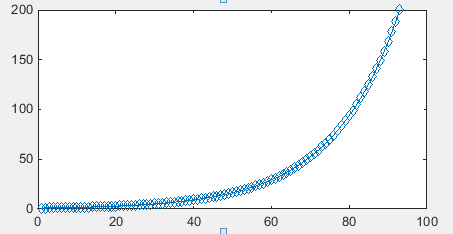


Abbildung 6: depth steps of the 90 steps for wavelengths > 960 nm

The step resolution for wavelengths <960 nm is calculated as follows:

This obligation needs to be read: “Find a factor f that 45 times multiplied with itself and a starting value equals half of the layer thickness.”

## Random pyramids algorithm

## Errors and interrupts

# Documented bugs

* Error\_Struct\_tot => first row is empty
* For the case of an inverted pyramid the Booster function is not correct. The projection layer now is now 0.1 µm beneath the top of the layer for rays pointing upwards und 0.1 µm above the top of the bottom pyramid for rays pointing downward.
* Plotting function in combination with Booster
* 2017\_10\_26: Lambertian behavior is only for reflection not transmission

# Future tasks

## Implementing random pyramides

## Implementing other geometries

### Isotexture

### Gaussian

### Corner cube

## Warnings for wrong input

* Not smaller 0
* Between x and y
* If absorber is at pos x, than

## Creating GUI

## Calculating Error

* Sending batches of 1000 rays comparing to the previous results

## Implementing Error as stop

## Implementing stop buttom

## Speeding up the simulation

* Current Speed on desktop PC: 2.55 min/EQE(30\*1000rays)

## Easy layer stacking => as many layers as needed

# History

2017\_10\_26: Lambertian reflection fixed