Royal Netherlands
Meteorological Institute
Ministry of Infrastructure and the Environment

Evaluation of radiation measurements for solar panel yield

J.-T. Brethouwer, W. van Harten, S, Klootwijk and S. Visser

KNMI Internal Report IR-2016-11

Evaluation of radiation measurements for solar panel yield

March 2016

Jan-Tino Brethouwer, Wouter van Harten,
Stefan Klootwijk and Sander Visser
(University of Twente.)

Supervised by:

Jan Fokke Meirink (KNMI)

1 Introduction

One of the greatest problems of today's world is the lack of renewable energy. Fossil fuels are running out and scientists everywhere are looking for efficient solutions. One of the most promising among the renewable energy sources is solar energy. Numerous analyses predict solar energy to be the most used energy source by 2050. Since solar energy will play such a major role in the world of tomorrow, it's important to use it as efficiently as possible. The placing of solar panels can make a big difference, so it is important to orientate the panels using the optimal tilt and azimuth for its location.

Determining the optimal orientation of a solar panel requires a lot of data. The data used in this report is provided by the World Radiation Monitoring Center - Baseline Surface Radiation Network (WRMC-BSRN). It consists of the data of collected in 2014 by fourteen stations from around the world. In this report, we will briefly describe this used data (section 2), read in the data using Matlab (appendix B.1) and check its quality and consistency (section 3 and appendix A). Then, we will describe briefly the model that is being used for calculating the incoming solar radiation on a tilted plane (section 4). We will calculate the incoming solar radiation for tilted planes with different orientations using Matlab (appendix B.2), visualize the results and determine the optimal orientation on each site (section 5 and appendix C). Finally, we draw some conclusions and discuss the results (sections 6 and 7).

2 Data sets

The used data sets are provided by the WRMC-BSRN. It consists of the readings in 2014 of fourteen different stations around the world, see figure 1 for an overview of their locations. The used data sets are so-called station-to-archive files and contain each all data from one month and one station. These station-to-archive files have a specific format, as described in [5].

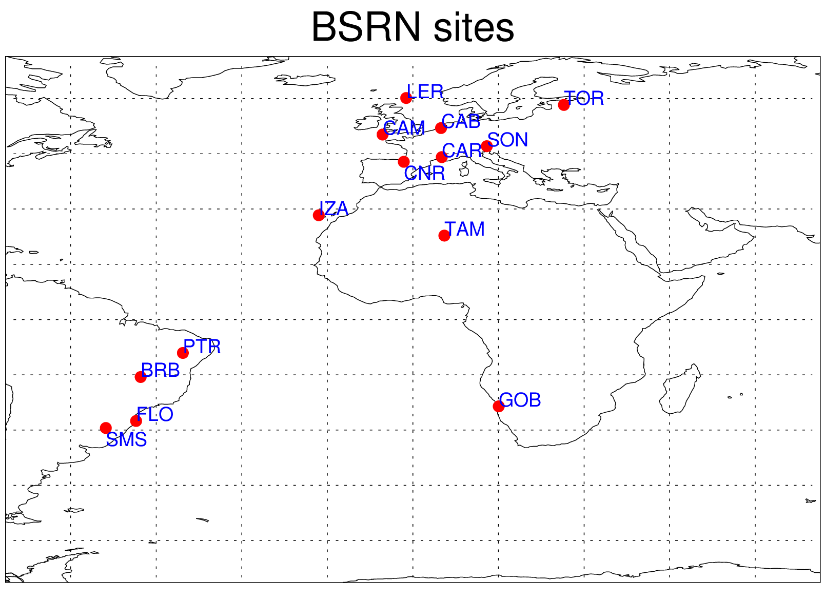


Figure 1: Locations of the 14 used BSRN sites.

The files start out with some general information about the station, of which only the ground type and location of the station will be used. The ground type is necessary for the determining the albedo and the location is necessary for computing the solar azimuth and zenith angles. Apart from the general information the files contain the measurements of the direct solar radiation, the diffuse solar radiation and the global solar irradiance for every minute of the month. Even though the stations are located in different time zones, all measurements are processed using Coordinated Universal Time (UTC 0).

Not all measurement data can be used since some of the data may be invalid. Most invalid data is indicated in the station-to-archive files by the use of large negative numbers. These large negative numbers will be referred to as 'corruption indicators' in this report. Furthermore, since negative radiation is physically impossible, these values are clear indicators that the corresponding data is invalid.

Even though the station-to-archive files are required to contain the measurements for each minute of the month, some station-to-archive files are still missing some minutes. This issue was resolved by manually adding the corruption indicators for these minutes.

Calculating the global solar irradiance requires, besides the information provided in the data sets from WRMC-BSRN, also the solar zenith and azimuth angles corresponding to each measurement. In order to obtain these angles, an implementation of the Solar Position Algorithm [4] as provided in the PV_LIB Toolbox for Matlab will be used.

Furthermore, based on the surface type of each BSRN site, for each site estimations of the corresponding albedo, which influences the amount of reflected irradiance, were made. These estimated values can be found in table 1.

Site	**Surface type**	**Albedo**
BRB	Concrete	0.55
CAB	Grass	0.25
CAM	Grass	0.25
CAR	Cultivated	0.20
CNR	Asphalt	0.10
FLO	Concrete	0.55
GOB	Desert gravel	0.30
IZA	Rock	0.20
LER	Grass	0.25
PTR	Concrete	0.55
SMS	Concrete	0.55
SON	Rock	0.20
TAM	Desert rock	0.25
TOR	Grass	0.25

Table **1**: Albedo estimations based on the surface type of the different BSRN sites.

3 Quality control

As mentioned before, some of the measurement data may be invalid. This can be caused, for example, by birds disrupting the equipment or by equipment which was not calibrated correctly at that time. To find out which measurement data is reliable and which is not, the data is subjected to a quality check.

Every minute three different components of the solar radiation are measured: the direct (normal) irradiance (B_n), the diffuse (horizontal) irradiance (D_h) and the global (horizontal) irradiance (G_h). These are subject to the following formula:

$$G_h = B_n \cos(\theta_z) + D_h, \tag{1}$$

where θ_z denotes the solar zenith angle. Theoretically this formula must hold for all correctly measured data points. However, because of measurement errors, in almost every case this formula does not hold exactly. Therefore, a margin of five percent is being used:

$$0.95G_h \leq B_n \cos(\theta_z) + D_h \leq 1.05G_h. \tag{2}$$

This margin is based on formula (23) in [1]. For each station, the values of $B_n \cos(\theta_z) + D_h$ (SumSW) are plotted against the corresponding values of G_h (SWD). The resulting figures can be found in appendix A. In these figures, also the margin lines are added. Only measurement data which satisfies formula (2) is accepted as reliable.

To check how well the reliable data follows equation (1), some basic statistical analysis was performed on the percentage deviation X defined as

$$X = \frac{(B_n \cos(\theta_z) + D_h) - G_h}{G_h} \cdot 100\%. \tag{3}$$

In table 2 the mean and standard deviation of X and the percentage of approved data points are listed per BSRN-site. The unapproved data point consists of data point that are during the night, corrupted or missing. Whenever the mean of X is not close to zero, this indicates a structural error in the measurements. For example, since PTR has a mean of 3.01%, this indicates that with high probability either the global irradiation measurement equipment structurally gives readings which are too low, or the direct and diffuse irradiation measurement equipment structurally gives readings which are too high. The opposite holds, for example, for SMS and TAM (with a mean of -3.20% and -2.02% respectively). These structural deviations are also clearly visible in the figures in appendix A.

4 Model

For calculating the global solar irradiance on a solar panel, two similar models for predicting the global solar radiation on tilted surfaces as described in [1] are being used. These models are summarized below.

The global irradiance on a tilted surface (G_φ) consists of three basic components, as shown in figure 2: direct (B_ϕ), sky diffuse (D_φ), and reflected (R_φ) solar irradiance component:

$$G_\varphi = B_\varphi + D_\varphi + R_\varphi. \tag{4}$$

Here $\phi = [\beta, \gamma]$ represents the tilted surface orientation, namely tilted surface tilt β and azimuth angle γ.

Site	Mean (μ) in %	Standard deviation (σ) in %	Used data points in %
BRB	-1.27	2.16	17
CAB	-0.43	1.09	49
CAM	0.07	1.00	50
CAR	0.12	1.02	50
CNR	-0.53	1.17	48
FLO	-0.23	1.78	48
GOB	-0.84	0.93	50
IZA	0.13	1.58	45
LER	-0.19	1.16	50
PTR	3.01	1.38	30
SMS	-3.20	1.47	21
SON	-0.05	2.17	36
TAM	-2.02	2.24	46
TOR	-0.67	1.56	50

Table **2**: Basic statistical facts about the percentage deviation X as defined in equation (3).

Each of the three irradiance components can be calculated using its own formulas.
First there is the direct irradiance B_φ, which is easiest to compute. Using basic geometry, it follows that

$$B_\varphi = B_n \cos(\theta) = B_n \cos(\theta_z) \cdot r_b, \tag{5}$$

where B_n is the normal component the direct irradiance, which is being measured, θ_z is the solar zenith angle, and r_b is the direct irradiance conversion factor, defined as:

$$r_b = \max\left(0, \frac{\cos(\theta)}{\cos(\theta_z)}\right). \tag{6}$$

Finally, θ is the angle of incidence, which is the angle between the sun direction and the normal direction of the tilted surface. θ can be computed as follows:

$$\cos(\theta) = \cos(\theta_z)\cos(\beta) + \sin(\theta_z)\sin(\beta)\cos(\gamma_s - \gamma), \tag{7}$$

where γ_s is the solar azimuth angle.

Next is the sky diffuse irradiance D_φ. This depends on the horizontal component of the diffusion D_h, which is being measured, and the diffuse transposition factor R_d:

$$D_\varphi = D_h R_d. \tag{8}$$

There are many models for the factor R_d. Generally, these models can be split into two categories, isotropic models and anisotropic models. To be able to compare these categories, an isotropic as well as a anisotropic model is being used. As isotropic model the Liu Jordan model [3] is chosen, and as anisotropic model the Klucher model [2] is chosen.

The Liu Jordan model defines R_d as follows:

$$R_d = \frac{1}{2}(1 + \cos(\beta)), \tag{9}$$

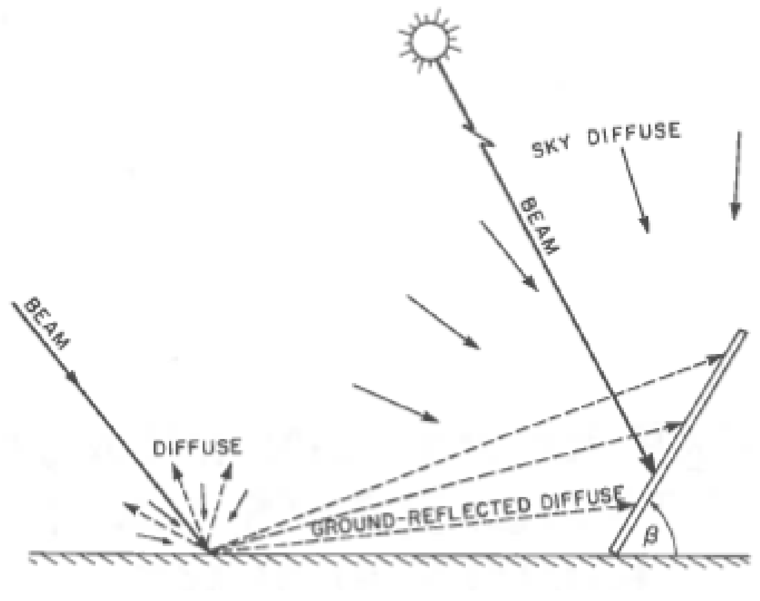


Figure **2**: Solar irradiance components.

while the Klucher model defines R_d as follows:

$$R_d = \left(\frac{1}{2}(1+\cos(\beta))\right)\left(1 + f_K\cos^2(\theta)\sin^3(\theta_z)\right)\left(1 + f_K\sin^3\left(\frac{\beta}{2}\right)\right), \tag{10}$$

where f_K is the Kluchers' conversion factor, defined as:

$$f_K = 1 - \left(\frac{D_h}{G_h}\right)^2, \tag{11}$$

where G_h is the horizontal component of the global solar irradiance, which is also being measured.

Lastly the ground-reflected diffuse irradiance R_φ is computed as

$$R_\varphi = \rho G_h R_h, \tag{12}$$

where ρ is the albedo of the ground around the solar panel and R_h is the transposition factor for ground reflection, defined as follows:

$$R_h = \frac{1}{2}(1-\cos(\beta)). \tag{13}$$

5 Results

In this section, both models (Liu-Jordan and Klucher) will be used to make a prediction for the maximum irradiance yield a tilted solar panel can achieve (and for which tilt and azimuth angles this maximum can be achieved). In order to find the maximum, the irradiance for every possible position of the solar panel has to be calculated. Every position can be evaluated with a tilt from 0° (flat) to 180° (upside down) degrees and an azimuth from 0° (pointed north) to 360° (again pointed north) degrees. The azimuth goes around clockwise, so 90° is pointed east.

For each of the 65341 tilt-azimuth combinations, both models are used to predict the amount of aggregated solar irradiance for the specific tilt-azimuth combination. In order to get a clear difference between the possible yield, the aggregated irradiance for each tilt-azimuth combination will be calculated as a percentage of the aggregated irradiance for a non-tilted surface.

Both Liu-Jordan and Klucher were applied to all fourteen BSRN-sites in figure 1. Contourplots were made for each site to show the optimum and the effect of a slight deviation from the optimum. These plots can be found in appendix C. The left column shows the results of then Liu-Jordan model and the right column shows the results of the Klucher model.

Since the aggregated irradiance was calculated for every tilt-azimuth combination, it is known exactly for which tilt and azimuth we would achieve the optimum yield. These results can be found in tables 3 (Liu-Jordan model) and 4 (Klucher model).

Site	**Yield (%)**	**Tilt (°)**	**Azimuth (°)**
BRB	101,1489	12	67
CAB	110,1925	32	180
CAM	108,5884	30	189
CAR	113,3712	33	178
CNR	108,9873	27	183
FLO	105,4929	26	1
GOB	106,9204	24	348
IZA	106,2589	22	170
LER	109,7170	33	178
PTR	100,5657	8	37
SMS	104,4312	22	13
SON	108,7995	30	167
TAM	115,1685	35	167
TOR	116,5075	38	178

Table **3**: Maximum achievable irradiance yield a tilted solar panel can obtain compared to a non-tilted solar panel (isotropic Liu-Jordan model). The tilt and azimuth were discretized on 1 degree.

Notice there is hardly any difference between the results from the two models. The Klucher model usually results in a slightly higher yield percentage and a slightly higher tilt. This is logical, since the Klucher model assumes more diffuse irradiance from the horizon than the Liu-Jordan model.

Site	**Yield (%)**	**Tilt (°)**	**Azimuth (°)**
BRB	101,2509	12	68
CAB	112,2357	34	180
CAM	110,2500	32	190
CAR	114,7119	34	178
CNR	110,1117	29	183
FLO	106,1898	28	1
GOB	107,3360	25	348
IZA	106,5870	23	169
LER	111,9572	36	178
PTR	100,6147	8	41
SMS	104,6406	23	14
SON	109,8900	32	167
TAM	116,0833	36	166
TOR	118,4535	40	178

Table **4**: Maximal achievable irradiance yield a tilted solar panel can obtain compared to a non-tilted solar panel (anisotropic Klucher model).

6 Discussion

A combination of tables 3 and 4, figure 1 and the figures in appendix C is needed to draw some conclusions about the optimal position of solar panels all around the world. Firstly it can be observed that most stations above the equator reach an optimum when the solar panel is directed to the south. Stations below the equator on the other hand reach an optimum when the solar panel is directed to the north. A rough conclusion is to say that solar panels should always be pointed towards the equator.

Furthermore it can be observed that stations which are further away from the equator reach an optimum with a larger tilt angle than other stations. This is also explainable. After all, the sun moves close to the equator and thus less tilt is required at the equator to direct a panel towards the sun. Because of this, the irradiance yield of a tilted solar panel compared to a non-tilted solar panel is relatively larger for sites further away from the equator.

Even though these two conclusions seem to apply in general, there are a few sites that seem to deviate from our conclusions. BRB and PTR both have an optimal percentage close to 100%. Because of this, a slight noise in the data can result in a completely wrong optimum tilt and azimuth. Luckily this is not a big problem, since hardly any improvement can be achieved by tilting a solar panel at these sites.

The slight difference between the optimum azimuth angles and the expected optimum azimuth angles at the different sites could be explained by the horizon at these different sites. If there are buildings or trees at the horizon, these influence the irradiation measurements. Dependent on the location of these buildings and trees, the optimum might divert a little bit from the optimum without any buildings and trees. Other explanations include difference in overcast between morning and afternoon and dust in desert sites.

7 Conclusion

In this report we have used the data from fourteen BSRN-sites to predict optimum tilt and azimuth degrees to achieve a maximum incoming solar irradiance. The BSRN-sites have measured direct and diffuse irradiance during a whole year. Using the models of Liu-Jordan and Klucher, we were able to predict the optimum tilt and azimuth angles.

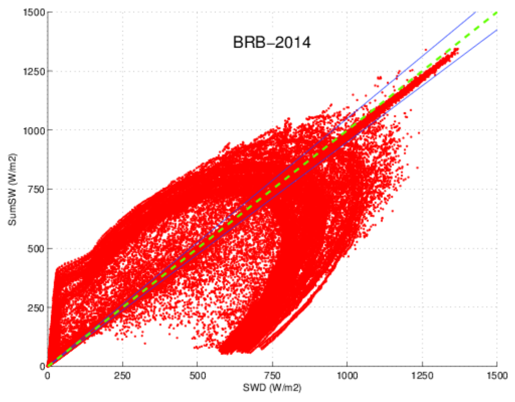
Our results have shown a few patterns, all within expectations. The optimal orientation for the solar panels is to be be directed towards the equator and a larger tilt is required when the site is at a larger distance from the equator. Furthermore, the proportional profit with respect to a flat solar panel is larger if the distance to the equator is larger.

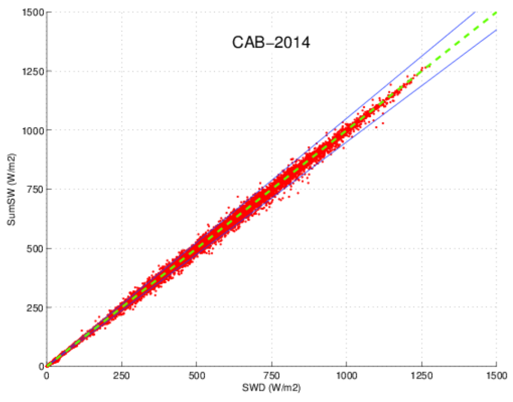
This research needs to be further extended. Firstly, the accuracy of our results can be tested by placing several solar panels and measuring the incoming solar irradiance. Furthermore, the BSRN-data seems to be inaccurate (containing a lot of unreliable data) at several sites. If more reliable data can be collected, better predictions can be made.

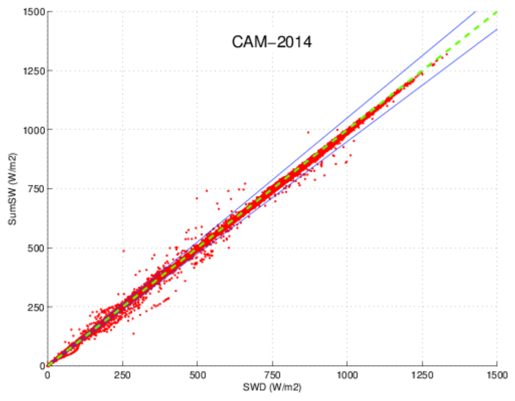
Acknowledgment

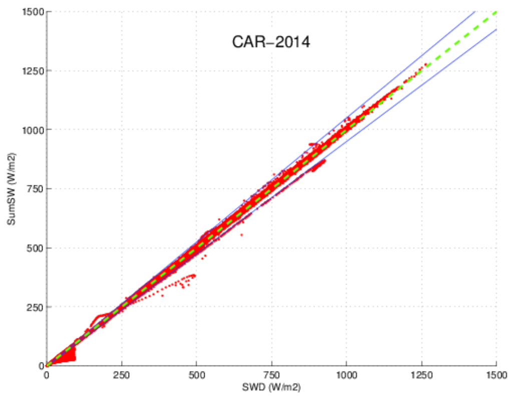
This work has been financially supported by the Royal Netherlands Meteorological Institute (KNMI). This support is gratefully acknowledged.

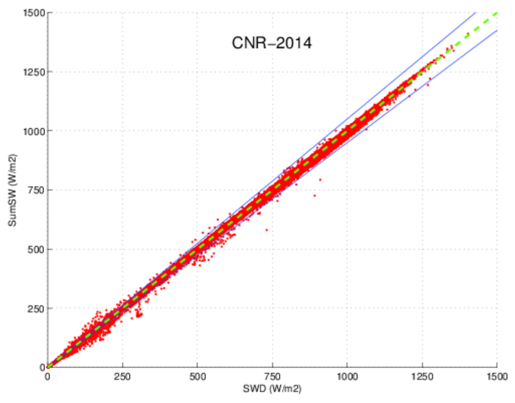
A Quality control

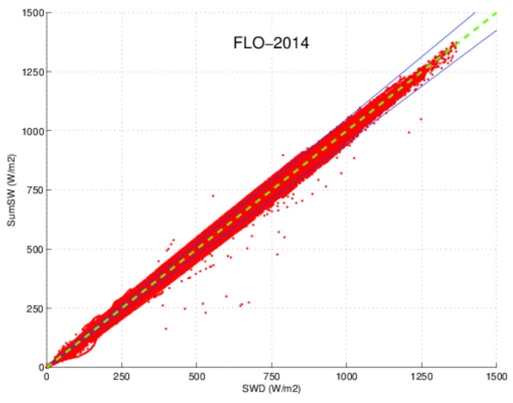


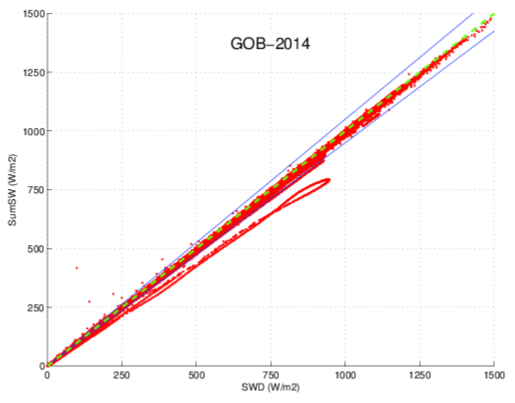


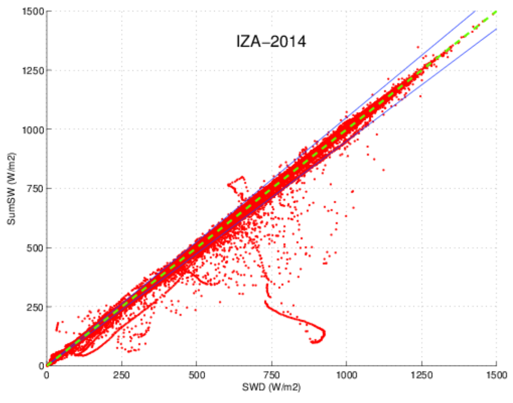


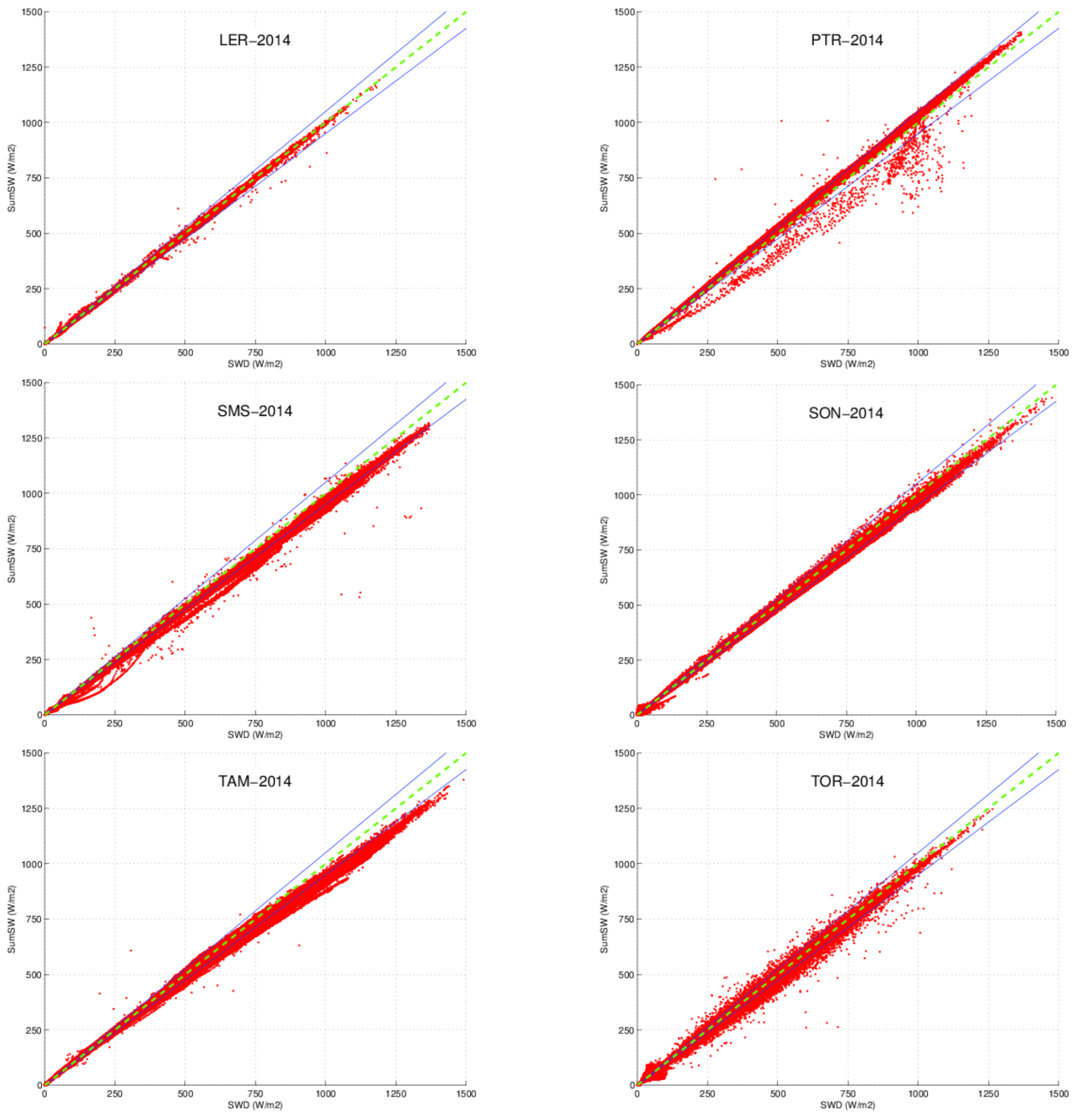











LER–2014
PTR–2014
SMS–2014
SON–2014
TAM–2014
TOR–2014
SumSW (W/m2)
SWD (W/m2)
0
250
500
750
1000
1250
1500

B Matlab

B.1 Reading the BSRN-files

The following Matlab functions were used to read the necessary data from the BSRN-files and convert this data in such a way that we can handle it relatively easy.

```matlab
function [] = creatematfile(location,month,year)
% CREATEMATFILE Read all BSRN archive file for a BSRN-station for a certain
% month and store all measurements of global irradiance, direct radiation,
% diffuse sky radiation, longwave downward radiation, air temperature,
% relative humidity and pressure in a .mat-file
%
%   CREATEMATFILE('LOCATION',MONTH,YEAR) Reads the BSRN-file for the
%   location with Event label LOCATION in the month MONTH of the year YEAR
%   and stores its measurements in a .mat-file
%
%   EXAMPLE: creatematfile('cab',1,2014)
%       This reads 'cab0114.dat' and generates the file 'cab0114.mat'

%% Initialize variables
if (month<10)
    filename=strcat(location,num2str(0),num2str(month),num2str(year-2000));
else
    filename=strcat(location,num2str(month),num2str(year-2000));
end

%% Find the rows of the BSRN-file that contain the basic measurements
[startrow,endrow]=findstartrow(strcat(filename,'.dat'),month,year);

%% Import and convert the basic measurement data from the BSRN-file
matrix=importbsrnfile(strcat(filename,'.dat'),startrow,endrow);

[gl,gldev,glmin,glmax,fdir,fdirdev,fdirmin,fdirmax,dfs,dfsdev,dfsmin,...
    dfsmax,lw,lwdev,lwmin,lwmax,temp,rh,pres]=matrixtoarrays(matrix);

%% Save the basic measurement data in a .mat-file
save(strcat(filename,'.mat'),'gl','gldev','glmin','glmax','fdir',...
    'fdirdev','fdirmin','fdirmax','dfs','dfsdev','dfsmin','dfsmax','lw',...
    'lwdev','lwmin','lwmax','temp','rh','pres');
```

```
function [StartRow,EndRow] = findstartrow(filename,month,year)
%FINDSTARTROW Find the row number in a BSRN-file at which the measurements
%start and the row number at which the basic measurements end
%
%   [STARTROW,ENDROW]=FINDSTARTROW('FILENAME',MONTH,YEAR) Finds the row
%   number in the BSRN-file FILENAME at which the measurements start (i.e.
%   the first row after the row which contains either '*U0100' or '*C0100')
%   and the row number at which the basic measurements end (i.e. '2*1440
%   times the number of days in that month' rows further)
%
%   EXAMPLE: [startrow,endrow]=findstartrow('cab0114.dat',1,2014);

%% Find the first row that contains measurement data
targetdate = '*U0100';
alllines = strsplit(fileread(filename), '\n');
StartRow = find(strncmp(alllines, targetdate, numel(targetdate)))+1;
if (numel(StartRow)==0)
    targetdate = '*C0100';
    alllines = strsplit(fileread(filename), '\n');
    StartRow = find(strncmp(alllines, targetdate, numel(targetdate)))+1;
end

%% Find the last row that contains basic measurement data
numberofdays=eomday(year,month);
EndRow=StartRow+numberofdays*1440*2-1;
```

```
function matrix = importbsrnfile(filename, startRow, endRow)
%IMPORTBSRNFILE Import numeric data from a text file as a matrix.
%
%   MATRIX = IMPORTBSRNFILE('FILENAME', STARTROW, ENDROW) Reads data from rows
%   STARTROW through ENDROW of text file FILENAME.
%
%   EXAMPLE: matrix = importbsrnfile('cab0114.dat', 120, 89399);
%
% Auto-generated by MATLAB on 2015/10/16 12:13:41

%% Initialize variables.
delimiter = ' ';

%% Format string for each line of text:
formatSpec = '%f%f%f%f%f%f%f%f%f%f%f%*s%*s%*s%*s%*s%*s%*s%*s%*s%*s%*s%[^\n\r]';

%% Open the text file.
fileID = fopen(filename,'r');

%% Read columns of data according to format string.
dataArray = textscan(fileID, formatSpec, endRow(1)-startRow(1)+1,...
    'Delimiter', delimiter, 'MultipleDelimsAsOne', true, 'EmptyValue' ,...
    NaN,'HeaderLines', startRow(1)-1, 'ReturnOnError', false);

for block=2:length(startRow)
    frewind(fileID);
    dataArrayBlock = textscan(fileID, formatSpec, ...
        endRow(block)-startRow(block)+1, 'Delimiter', delimiter, ...
        'MultipleDelimsAsOne', true, 'EmptyValue' ,NaN,'HeaderLines', ...
        startRow(block)-1, 'ReturnOnError', false);
    for col=1:length(dataArray)
        dataArray{col} = [dataArray{col};dataArrayBlock{col}];
    end
end

%% Close the text file.
fclose(fileID);

%% Create output variable
matrix = [dataArray{1:end-1}];
```

```
function [gl,gldev,glmin,glmax,fdir,fdirdev,fdirmin,fdirmax,dfs,dfsdev,...
    dfsmin,dfsmax,lw,lwdev,lwmin,lwmax,temp,rh,pres] = matrixtoarrays(matrix)
%MATRIXTOARRAYS Split the matrix obtained from the logical record '0100' of
%a BSRN-file into different matrices (1 matrix for each type of data)

%% Initialize variables
ndays=size(matrix,1)/(2*1440);
nminute=1440;

gl=zeros(ndays,nminute);
gldev=zeros(ndays,nminute);
glmin=zeros(ndays,nminute);
glmax=zeros(ndays,nminute);
fdir=zeros(ndays,nminute);
fdirdev=zeros(ndays,nminute);
fdirmin=zeros(ndays,nminute);
fdirmax=zeros(ndays,nminute);
dfs=zeros(ndays,nminute);
dfsdev=zeros(ndays,nminute);
dfsmin=zeros(ndays,nminute);
dfsmax=zeros(ndays,nminute);
lw=zeros(ndays,nminute);
lwdev=zeros(ndays,nminute);
lwmin=zeros(ndays,nminute);
lwmax=zeros(ndays,nminute);
temp=zeros(ndays,nminute);
rh=zeros(ndays,nminute);
pres=zeros(ndays,nminute);

%% Fill the different matrices with the correct data
for i=1:ndays
    for j=1:nminute
        x=1440*(i-1)+j;
        gl(i,j)=matrix(2*x-1,3);
        gldev(i,j)=matrix(2*x-1,4);
        glmin(i,j)=matrix(2*x-1,5);
        glmax(i,j)=matrix(2*x-1,6);
        fdir(i,j)=matrix(2*x-1,7);
        fdirdev(i,j)=matrix(2*x-1,8);
        fdirmin(i,j)=matrix(2*x-1,9);
        fdirmax(i,j)=matrix(2*x-1,10);
        dfs(i,j)=matrix(2*x,1);
        dfsdev(i,j)=matrix(2*x,2);
        dfsmin(i,j)=matrix(2*x,3);
        dfsmax(i,j)=matrix(2*x,4);
        lw(i,j)=matrix(2*x,5);
        lwdev(i,j)=matrix(2*x,6);
        lwmin(i,j)=matrix(2*x,7);
        lwmax(i,j)=matrix(2*x,8);
        temp(i,j)=matrix(2*x,9);
        rh(i,j)=matrix(2*x,10);
        pres(i,j)=matrix(2*x,11);
    end
end
```

```
function [] = createyearfile(location,year)
% CREATEYEARFILE Collect data from 12 monthly .mat-files into 1 .mat-file
% which contains data for an entire year
%
%   CREATEYEARFILE(LOCATION,YEAR) Collects the data from the 12 monthly
%   .mat-files belonging to the BSRN-station with Event label LOCATION and
%   the year YEAR, and stores all the data from those files into one .mat-file
%
%   EXAMPLE: createyearfile('cab',2014)
%       This collects all data from 'cab0114.mat' till 'cab1214.mat' and
%       stores all that data into 'cab2014.mat'

%% Initialize variables
ndays=sum(eomday(year,1:12));
nminute=1440;
gl2=zeros(ndays,nminute);
gldev2=zeros(ndays,nminute);
glmin2=zeros(ndays,nminute);
glmax2=zeros(ndays,nminute);
fdir2=zeros(ndays,nminute);
fdirdev2=zeros(ndays,nminute);
fdirmin2=zeros(ndays,nminute);
fdirmax2=zeros(ndays,nminute);
dfs2=zeros(ndays,nminute);
dfsdev2=zeros(ndays,nminute);
dfsmin2=zeros(ndays,nminute);
dfsmax2=zeros(ndays,nminute);
lw2=zeros(ndays,nminute);
lwdev2=zeros(ndays,nminute);
lwmin2=zeros(ndays,nminute);
lwmax2=zeros(ndays,nminute);
temp2=zeros(ndays,nminute);
rh2=zeros(ndays,nminute);
pres2=zeros(ndays,nminute);
counter=0;

%% Read the 12 monthly files and store its data
for month=1:12
    if (month<10)
        filename=strcat(location,num2str(0),num2str(month),...
            num2str(year-2000),'.mat');
    else
        filename=strcat(location,num2str(month),num2str(year-2000),'.mat');
    end
    load(filename)
    counternew=counter+size(gl,1);
    gl2(counter+1:counternew,:)=gl(:,:);
    gldev2(counter+1:counternew,:)=gldev(:,:);
    glmin2(counter+1:counternew,:)=glmin(:,:);
    glmax2(counter+1:counternew,:)=glmax(:,:);
    fdir2(counter+1:counternew,:)=fdir(:,:);
    fdirdev2(counter+1:counternew,:)=fdirdev(:,:);
    fdirmin2(counter+1:counternew,:)=fdirmin(:,:);
    fdirmax2(counter+1:counternew,:)=fdirmax(:,:);
    dfs2(counter+1:counternew,:)=dfs(:,:);
    dfsdev2(counter+1:counternew,:)=dfsdev(:,:);
    dfsmin2(counter+1:counternew,:)=dfsmin(:,:);
    dfsmax2(counter+1:counternew,:)=dfsmax(:,:);
    lw2(counter+1:counternew,:)=lw(:,:);
    lwdev2(counter+1:counternew,:)=lwdev(:,:);
    lwmin2(counter+1:counternew,:)=lwmin(:,:);
    lwmax2(counter+1:counternew,:)=lwmax(:,:);
    temp2(counter+1:counternew,:)=temp(:,:);
    rh2(counter+1:counternew,:)=rh(:,:);
    pres2(counter+1:counternew,:)=pres(:,:);
    counter=counternew;
end
```

```
%% Save all basic measurement data for 2014 in one .mat-file
gl=gl2;
gldev=gldev2;
glmin=glmin2;
glmax=glmax2;
fdir=fdir2;
fdirdev=fdirdev2;
fdirmin=fdirmin2;
fdirmax=fdirmax2;
dfs=dfs2;
dfsdev=dfsdev2;
dfsmin=dfsmin2;
dfsmax=dfsmax2;
lw=lw2;
lwdev=lwdev2;
lwmin=lwmin2;
lwmax=lwmax2;
temp=temp2;
rh=rh2;
pres=pres2;

filename=strcat(location,num2str(2014));

save(strcat(filename,'.mat'),'gl','gldev','glmin','glmax','fdir',...
    'fdirdev','fdirmin','fdirmax','dfs','dfsdev','dfsmin','dfsmax','lw',...
    'lwdev','lwmin','lwmax','temp','rh','pres');
```

NB: After running the function `createyearfile.m` for each site, the constants `albedo`, `altitude`, `latitude`, `longitude` and `UTC` were added manually to the created .mat-files.

B.2 Computing G_φ

The Matlab functions in this section were used to compute and aggregate the values of G_φ. The first paragraph contains a Matlab function for the computations of the necessary solar zenith and azimuth angles. The second and third paragraph contain Matlab functions for the computation and aggregation of G_φ itself using the Liu-Jordan and Klucher model, respectively.

B.2.1 Solar zenith and azimuth angles

The Matlab function on the following page was used to compute the necessary solar zenith and azimuth angles. It makes use of the subroutine `pvl_maketimestruct.m` which is available in the PV_LIB Toolbox for Matlab.

```matlab
function [] = SolarAngles(location,year)
%SOLARANGLES Compute solar zenith and azimuth angles for an entire year
%
%   SOLARANGLES('LOCATION',YEAR) Computes the solar zenith and azimuth
%   angles for every minute of the year YEAR on the location of the
%   BSRN-station with Event label LOCATION and stores these angles together
%   with the basic measurement data and the station characteristics of that
%   station into a .mat-file
%
%   EXAMPLE: SolarAngles('cab',2014)
%       This computes the solar zenith and azimuth angles in 2014 for
%       Cabauw and stores this together with the measurement data from
%       'cab2014.mat' into 'cab2014+zon.mat'

%% Load the measurement data and station characteristics
filename=strcat(location,num2str(year),'.mat');
load(filename);

%% Create the needed location structure using the PV_LIB Toolbox
Location=pvl_makelocationstruct(latitude-90,longitude-180,altitude);

%% Create the needed time structure using the PV_LIB Toolbox
% for every minute, we take the midpoint of that minute for the solar angle
% computations
daysofmonth=eomday(year,1:12);
dates=zeros(sum(daysofmonth)*1440,1);
for day=1:sum(daysofmonth)
    datum=datevec(datenum(year,1,day));
    for m=1:1440
        uur=floor((m-1)/60);
        minuut=m-60*uur-1;
        dates(1440*(day-1)+m,1)=datenum(year,datum(2),datum(3),uur,minuut,30);
    end
end
Times=pvl_maketimestruct(dates,UTC);

%% Create the needed pressure and temperature vector
Pressure=zeros(sum(daysofmonth)*1440,1);
Temperature=zeros(sum(daysofmonth)*1440,1);
for day=1:sum(daysofmonth)
    Pressure((day-1)*1440+1:day*1440,1)=100*pres(day,:)'.*(pres(day,:)>=0)';
    Temperature((day-1)*1440+1:day*1440,1)=temp(day,:)';
end

%% Compute solar elevation and azimuth angles using the SPA method and the PV_LIB
     Toolbox
[SunAz2,SunEl2,ApparentSunEl2]=pvl_spa(Times,Location,Pressure,Temperature);

%% Convert elevation and azimuth angle vectors into correct matrix form
SunAz=zeros(sum(daysofmonth),1440);
SunEl=zeros(sum(daysofmonth),1440);
ApparentSunEl=zeros(sum(daysofmonth),1440);
for day=1:sum(daysofmonth)
    SunAz(day,:)=SunAz2((day-1)*1440+1:day*1440)';
    SunEl(day,:)=SunEl2((day-1)*1440+1:day*1440)';
    ApparentSunEl(day,:)=ApparentSunEl2((day-1)*1440+1:day*1440)';
end

%% Convert solar elevation angles into solar zenith angles
SunZe=90-SunEl;

%% Save all data in a .mat-file
filename=strcat(location,num2str(year),'+zon.mat');
save(filename,'gl','gldev','glmin','glmax','fdir','fdirdev', ...
    'fdirmin','fdirmax','dfs','dfsdev','dfsmin','dfsmax','lw','lwdev', ...
    'lwmin','lwmax','temp','rh','pres','SunAz','SunZe','ApparentSunEl', ...
    'albedo','altitude','latitude','longitude','UTC');
```

B.2.2 Liu-Jordan

The following functions were used to compute and aggregate the values of G_φ for different values of $\varphi = [\beta, \gamma]$, the orientation of the solar panel, using the Liu-Jordan model for determining the diffuse irradiance D_φ.

```
function [] = CompleteRun(nBeta,nGamma)
%COMPLETERUN Aggregate incoming radiation for tilted plane to yearly yield
%of solar panel, for all 14 locations, and for different combinations of
%panel tilt and orientation using the Liu-Jordan model for diffuse irradiance
%
%   COMPLETERUN(NBETA,NGAMMA) Aggregates incoming radiation for NBETA
%   different equally spaced values of the panel tilt and NGAMMA different
%   equally spaced values of the panel orientation, for all 14 locations,
%   using Liu-Jordan for diffuse irradiance, and saves all data in 14
%   different .mat-files (1 file for each location)
%
%   EXAMPLE: CompleteRun(180,360)

%% Initialize the 14 locations
Locations=['brb';'cab';'cam';'car';'cnr';'flo';'gob';'iza';'ler';'ptr';...
    'sms';'son';'tam';'tor'];

%% Compute the aggregated radiation for 2014
for l=1:size(Locations,1)
    location=Locations(l,:);
    ComputeG(location,2014,nBeta,nGamma);
end
```

```
function [] = ComputeG(location,year,nBeta,nGamma)
%COMPUTEG Aggregate incoming radiation for tilted plane to yearly yield of
%solar panel, i.e. aggregate Gφ for an year, for different combinations of
%panel tilt and orientation, using the Liu-Jordan model for diffuse irradiance
%
%   COMPUTEG('LOCATION',YEAR,NBETA,NGAMMA) Aggregates incoming radiation
%   for NBETA different equally spaced values of the panel tilt and NGAMMA
%   different equally spaced values of the panel orientation, for the
%   location with Event label LOCATION in the year YEAR, using Liu-Jordan
%   for diffuse irradiance, and saves all data in a .mat-file
%
%   EXAMPLE: ComputeG('cab',2014,180,360)

%% Load the measurements including solar angle data
filename=strcat(location,num2str(year),'+zon.mat');
load(filename);

%% Compute Gφ using Liu-Jordan
G_phi2=calculateSumG_phi(dfs,fdir,gl,SunZe,SunAz,nBeta,nGamma,albedo);

%% Convert Gφ into percentages (100% corresponds to a flat solar panel)
G_phi=G_phi2./G_phi2(1,1)*100;

%% Save all data in a new .mat-file
filename=strcat(location,num2str(year),'+zon+g_phi.mat');
save(filename,'gl','gldev','glmin','glmax','fdir','fdirdev', ...
    'fdirmin','fdirmax','dfs','dfsdev','dfsmin','dfsmax','lw','lwdev', ...
    'lwmin','lwmax','temp','rh','pres','SunAz','SunZe','ApparentSunEl', ...
    'albedo','altitude','latitude','longitude','UTC','G_phi','G_phi2');
```

```matlab
function G_phiSum = calculateSumG_phi(D_h,B_n,G_h,theta_z,gamma_s,nBeta,nGamma,
    albedo)
%CALCULATESUMG_PHI Calculate the sum of the G_φ-values over all
%non-corrupted data points, using the Liu-Jordan model for diffuse irradiance
%
%   G_PHISUM =
%   CALCULATESUMG_PHI(D_H,B_N,G_H,THETA_Z,GAMMA_S,NBETA,NGAMMA,ALBEDO)
%   Calculates the sum of the G_φ-values over all non-corrupted data
%   points, for NBETA and NGAMMA different values of the solar panel tilt
%   and orientation angles respectively, and using the data provided in the
%   matrices D_H (diffuse irradiance), B_N (direct normal irradiance), G_H
%   (global irradiance), THETA_Z (solar zenith) and GAMMA_S (solar azimuth),
%   and the constant ALBEDO
%
%   EXAMPLE:
%   G_phi=calculateSumG_phi(dfs,fdir,gl,SunZe,SunAz,nBeta,nGamma,albedo);

%% Initialize variables
maxGamma = 360;
maxBeta = 180;
G_phiSum = zeros(nBeta+1, nGamma+1);

%% Preconvert some angles from degrees to radians
gamma_s = gamma_s ./ 360 * 2 * pi;
theta_z = theta_z ./ 360 * 2 * pi;

%% Check which data is corrupted
% Data is called corrupted when either an irradiance measurement is
% negative, or the sun is located below the horizon, or the irradiance
% measurements do not satisfy the quality control
% (0.95G_h ≤ B_n cos(θ_z) + D_h ≤ 1.05G_h)
corrupted = ((D_h < 0) + (B_n < 0) + (G_h < 0) + (theta_z > pi/2)) + ...
    (0.95*G_h>B_n.*cos(theta_z)+D_h) + (1.05*G_h<B_n.*cos(theta_z)+D_h) > 0;

%% Compute the sum of the G_φ-values for the non-corrupted data points
for n=1:nBeta+1
    for m=1:nGamma+1
        G_phi = calculateG_phi(D_h, B_n, G_h, theta_z, gamma_s,...
            (n-1)*maxBeta/nBeta, (m-1)*maxGamma/nGamma, albedo, corrupted);
        G_phiSum(n, m) = sum(G_phi(:));
    end
end
```

```matlab
function G_phi = calculateG_phi(D_h,B_n,G_h,theta_z,gamma_s,beta,gamma,albedo,
    corrupted)
%CALCULATEG_PHI Calculate all Gφ-values for a specific combination of
%solar panel tilt and orientation, using the Liu-Jordan model for diffuse
%irradiance
%
%  G_PHI =
%  CALCULATEG_PHI(D_H,B_N,G_H,THETA_Z,GAMMA_S,BETA,GAMMA,ALBEDO,CORRUPTED)
%  Calculates for every minute of the year the correponding value of Gφ,
%  where the solar panel has tilt angle BETA and orientation angle GAMMA,
%  using the data from the matrices D_H, B_N, G_H, THETA_Z and GAMMA_S, the
%  constant ALBEDO, and the precomputed values of the matrix CORRUPTED. If
%  the data for a certain minute is called corrupted, then G_PHI is set to
%  zero for this minute.
%
%   EXAMPLE:
%   G_phi=calculateG_phi(dfs,fdir,gl,SunZe,SunAz,0,0,albedo,corrupted);

%% Check whether all data matrices have the same size
if(numel(D_h) ~= numel(B_n) || numel(D_h) ~= numel(G_h) || ...
        numel(D_h) ~= numel(theta_z) || numel(D_h) ~= numel(gamma_s))
    error('not all variables have the same size.');
end

%% Convert beta and gamma from degrees to radians
beta = beta / 360 * 2 * pi;
gamma = gamma / 360 * 2 * pi;

%% Compute Bφ (by equations (7), (5) and (6))
cos_theta = cos(theta_z)*cos(beta)+sin(theta_z).*cos(gamma_s-gamma)*sin(beta);
B_phi = B_n.*cos(theta_z).*max(0, cos_theta./cos(theta_z));

%% Compute Dφ for the Liu-Jordan model (by equations (9) and (8))
R_d = 0.5*(1+cos(beta));
D_phi = R_d .* D_h;

%% Compute Rφ (by equations (13) and (12))
R_h = 0.5*(1-cos(beta));
R_phi = albedo * G_h .* R_h;

%% Compute Gφ (by equation (4)) and correct for 'corrupted' data
G_phi = (B_phi + D_phi + R_phi) .* (1 - corrupted);
```

B.2.3 Klucher

The following functions were used to compute and aggregate the values of G_φ for different values of $\varphi = [\beta, \gamma]$, the orientation of the solar panel, using the Klucher model for determining the diffuse irradiance D_φ.

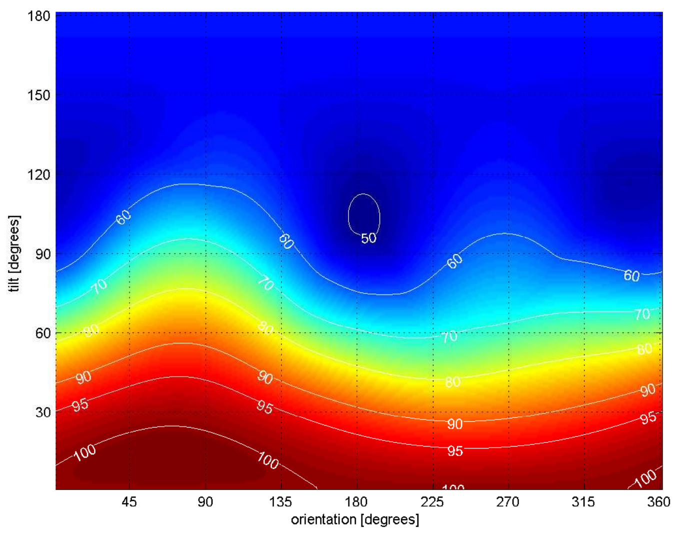
```
function [] = CompleteRun_Kl(nBeta,nGamma)
%COMPLETERUN_KL Aggregate incoming radiation for tilted plane to yearly yield
%of solar panel, for all 14 locations, and for different combinations of
%panel tilt and orientation using the Klucher model for diffuse irradiance
%
%   COMPLETERUN_KL(NBETA,NGAMMA) Aggregates incoming radiation for NBETA
%   different equally spaced values of the panel tilt and NGAMMA different
%   equally spaced values of the panel orientation, for all 14 locations,
%   using Klucher for diffuse irradiance, and saves all data in 14
%   different .mat-files (1 file for each location)
%
%   EXAMPLE: CompleteRun_Kl(180,360)

%% Initialize the 14 locations
Locations=['brb';'cab';'cam';'car';'cnr';'flo';'gob';'iza';'ler';'ptr';...
    'sms';'son';'tam';'tor'];

%% Compute the aggregated radiation for 2014
for l=1:size(Locations,1)
    location=Locations(l,:);
    ComputeG_Kl(location,2014,nBeta,nGamma);
end
```

```
function [] = ComputeG_Kl(location,year,nBeta,nGamma)
%COMPUTEG_KL Aggregate incoming radiation for tilted plane to yearly yield of
%solar panel, i.e. aggregate Gφ for an year, for different combinations of
%panel tilt and orientation, using the Klucher model for diffuse irradiance
%
%   COMPUTEG_KL('LOCATION',YEAR,NBETA,NGAMMA) Aggregates incoming radiation
%   for NBETA different equally spaced values of the panel tilt and NGAMMA
%   different equally spaced values of the panel orientation, for the
%   location with Event label LOCATION in the year YEAR, using Klucher
%   for diffuse irradiance, and saves all data in a .mat-file
%
%   EXAMPLE: ComputeG_Kl('cab',2014,180,360)

%% Load the measurements including solar angle data
filename=strcat(location,num2str(year),'+zon.mat');
load(filename);

%% Compute Gφ using Klucher
G_phi2=calculateSumG_phi_Kl(dfs,fdir,gl,SunZe,SunAz,nBeta,nGamma,albedo);

%% Convert Gφ into percentages (100% corresponds to a flat solar panel)
G_phi=G_phi2./G_phi2(1,1)*100;

%% Save all data in a new .mat-file
filename=strcat(location,num2str(year),'+zon+g_phi_Kl.mat');
save(filename,'gl','gldev','glmin','glmax','fdir','fdirdev', ...
    'fdirmin','fdirmax','dfs','dfsdev','dfsmin','dfsmax','lw','lwdev', ...
    'lwmin','lwmax','temp','rh','pres','SunAz','SunZe','ApparentSunEl', ...
    'albedo','altitude','latitude','longitude','UTC','G_phi','G_phi2');
```

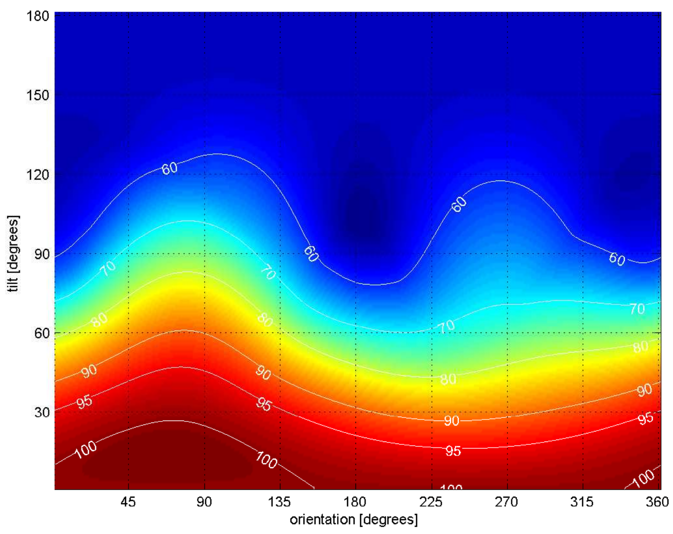
```matlab
function G_phiSum = calculateSumG_phi_Kl(D_h,B_n,G_h,theta_z,gamma_s,nBeta,nGamma
    ,albedo)
%CALCULATESUMG_PHI_KL Calculate the sum of the G_φ-values over all
%non-corrupted data points, using the Klucher model for diffuse irradiance
%
%   G_PHISUM =
%   CALCULATESUMG_PHI_KL(D_H,B_N,G_H,THETA_Z,GAMMA_S,NBETA,NGAMMA,ALBEDO)
%   Calculates the sum of the G_φ-values over all non-corrupted data
%   points, for NBETA and NGAMMA different values of the solar panel tilt
%   and orientation angles respectively, and using the data provided in the
%   matrices D_H (diffuse irradiance), B_N (direct normal irradiance), G_H
%   (global irradiance), THETA_Z (solar zenith) and GAMMA_S (solar azimuth),
%   and the constant ALBEDO
%
%   EXAMPLE:
%   G_phi=calculateSumG_phi_Kl(dfs,fdir,gl,SunZe,SunAz,nBeta,nGamma,albedo);

%% Initialize variables
maxGamma = 360;
maxBeta = 180;
G_phiSum = zeros(nBeta+1, nGamma+1);

%% Preconvert some angles from degrees to radians
gamma_s = gamma_s ./ 360 * 2 * pi;
theta_z = theta_z ./ 360 * 2 * pi;

%% Check which data is corrupted
% Data is called corrupted when either an irradiance measurement is
% negative, or the sun is located below the horizon, or the irradiance
% measurements do not satisfy the quality control
% (0.95G_h ≤ B_n cos(θ_z) + D_h ≤ 1.05G_h)
corrupted = ((D_h < 0) + (B_n < 0) + (G_h < 0) + (theta_z > pi/2)) + ...
    (0.95*G_h>B_n.*cos(theta_z)+D_h) + (1.05*G_h<B_n.*cos(theta_z)+D_h) > 0;

%% Do some more precomputations (Kluchers' conversion factor and sin^3(θ_z) )
% Kluchers' conversion factor (11) is defined as 0 in case G_h=0
f_K = (G_h~=0).*(1 - (D_h./(G_h+(G_h==0))).^2);
sin3theta_z = sin(theta_z).^3;

%% Compute the sum of the G_φ-values for the non-corrupted data points
for n=1:nBeta+1
    for m=1:nGamma+1
        G_phi = calculateG_phi_Kl(D_h, B_n, G_h, theta_z, gamma_s,...
            (n-1)*maxBeta/nBeta, (m-1)*maxGamma/nGamma, albedo, corrupted,...
            f_K, sin3theta_z);
        G_phiSum(n, m) = sum(G_phi(:));
    end
end
```

```
function G_phi = calculateG_phi_Kl(D_h,B_n,G_h,theta_z,gamma_s,beta,gamma,albedo,
    corrupted,f_K,sin3theta_z)
%CALCULATEG_PHI_KL Calculate all Gφ-values for a specific combination of
%solar panel tilt and orientation, using the Klucher model for diffuse
%irradiance
%
%   G_PHI =
%   CALCULATEG_PHI_KL(D_H,B_N,G_H,THETA_Z,GAMMA_S,BETA,GAMMA,ALBEDO,CORRUPTED,
%   F_K,SIN3THETA_Z) Calculates for every minute of the year the correponding
%   value of Gφ, where the solar panel has tilt angle BETA and orientation
%   angle GAMMA, using the data from the matrices D_H, B_N, G_H, THETA_Z and
%   GAMMA_S, the constant ALBEDO, and the precomputed values of the matrices
%   CORRUPTED, F_K and SIN3THETA_Z. If the data for a certain minute is called
%   corrupted, then G_PHI is set to zero for this minute.
%
%    EXAMPLE: G_phi=calculateG_phi_Kl(dfs,fdir,gl,SunZe,SunAz,0,0,albedo,
%             corrupted,f_K,sin3theta_z);

%% Check whether all data matrices have the same size
if(numel(D_h) ~= numel(B_n) || numel(D_h) ~= numel(G_h) || ...
        numel(D_h) ~= numel(theta_z) || numel(D_h) ~= numel(gamma_s))
    error('not all variables have the same size.');
end

%% Convert beta and gamma from degrees to radians
beta = beta / 360 * 2 * pi;
gamma = gamma / 360 * 2 * pi;

%% Compute Bφ (by equations (7), (5) and (6))
cos_theta = cos(theta_z)*cos(beta)+sin(theta_z).*cos(gamma_s-gamma)*sin(beta);
B_phi = B_n.*cos(theta_z).*max(0, cos_theta./cos(theta_z));

%% Compute Dφ for the Klucher model (by equations (10) and (8))
R_d = 0.5*(1+cos(beta)).*(1 + f_K .* cos_theta.^2 .* sin3theta_z).* ...
    (1 + f_K.*sin(beta/2)^3);
D_phi = R_d .* D_h;

%% Compute Rφ (by equations (13) and (12))
R_h = 0.5*(1-cos(beta));
R_phi = albedo * G_h .* R_h;

%% Compute Gφ (by equation (4)) and correct for 'corrupted' data
G_phi = (B_phi + D_phi + R_phi) .* (1 - corrupted);
```

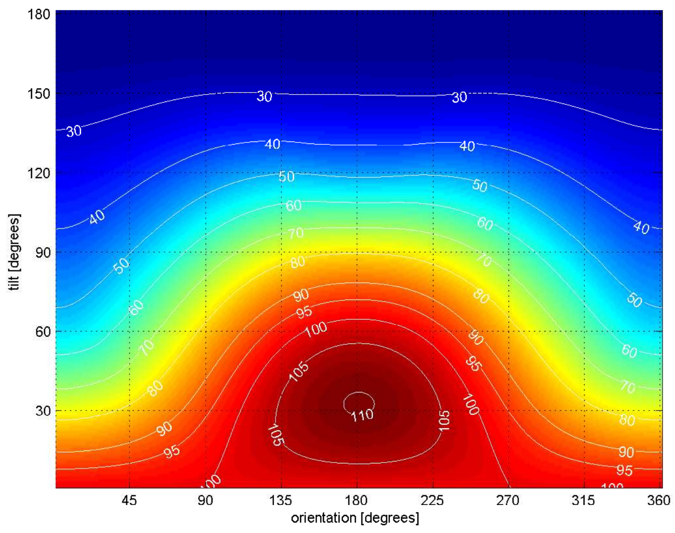
C Figures



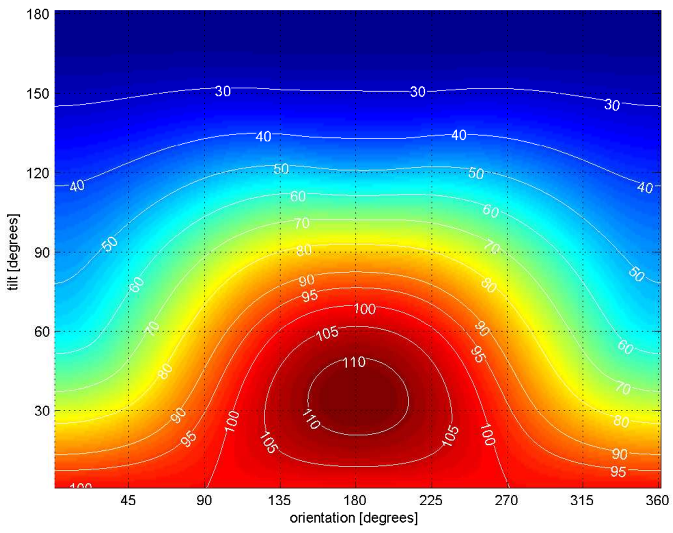
BRB (Liu-Jordan)



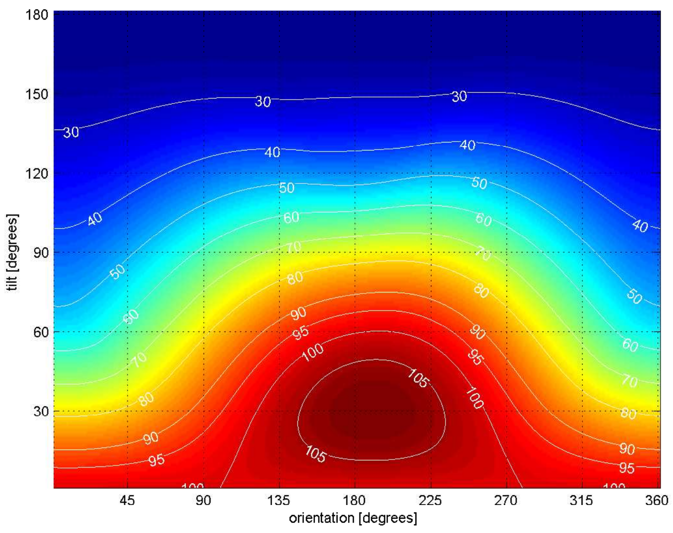
BRB (Klucher)



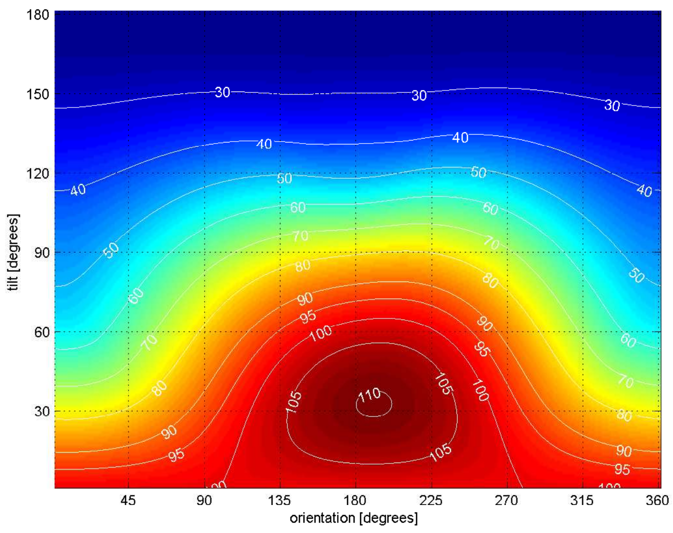
CAB (Liu-Jordan)



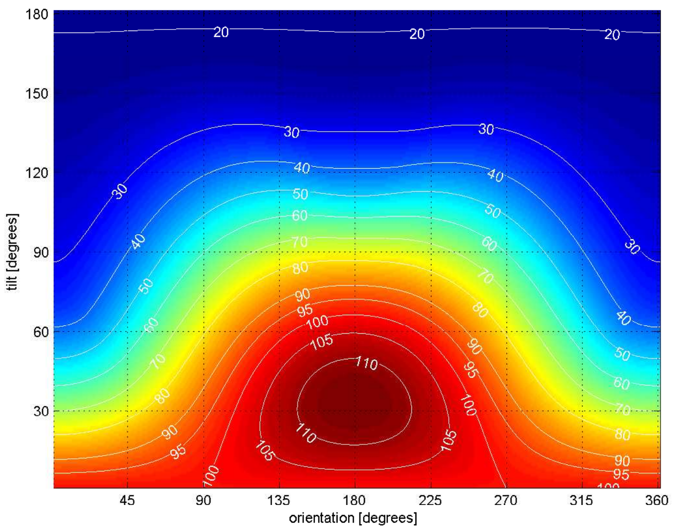
CAB (Klucher)



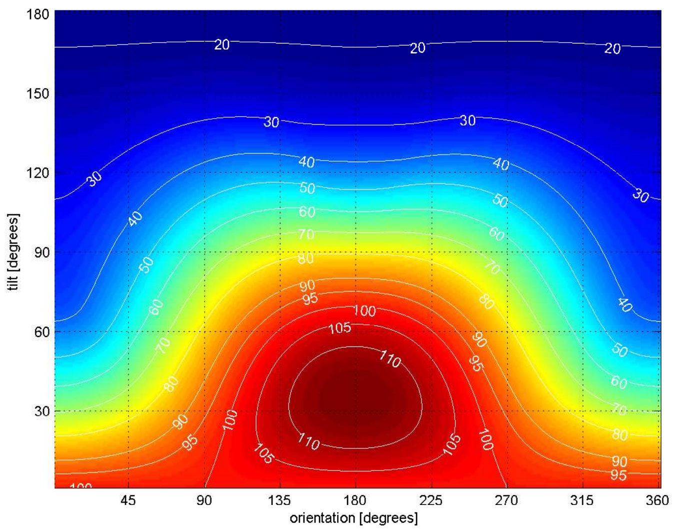
CAM (Liu-Jordan)



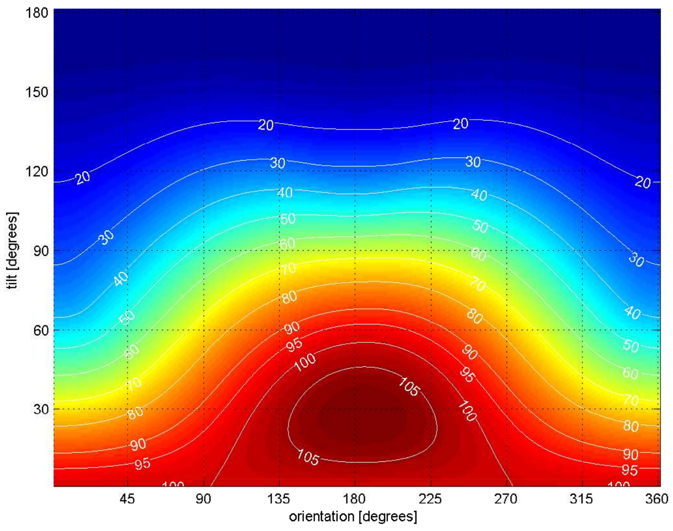
CAM (Klucher)



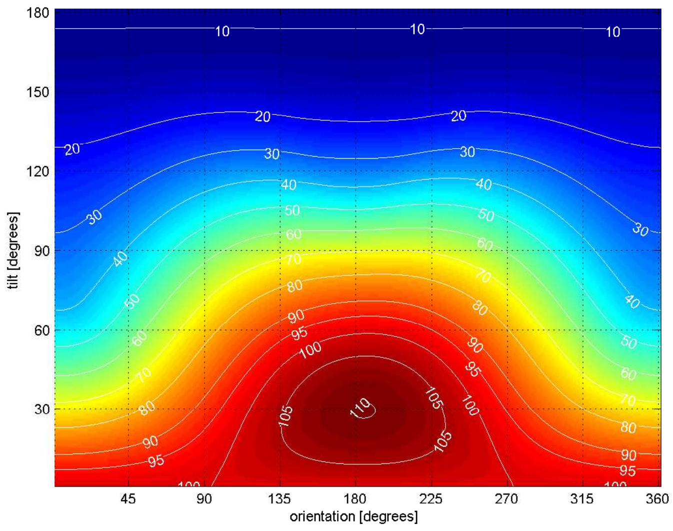
CAR (Liu-Jordan)



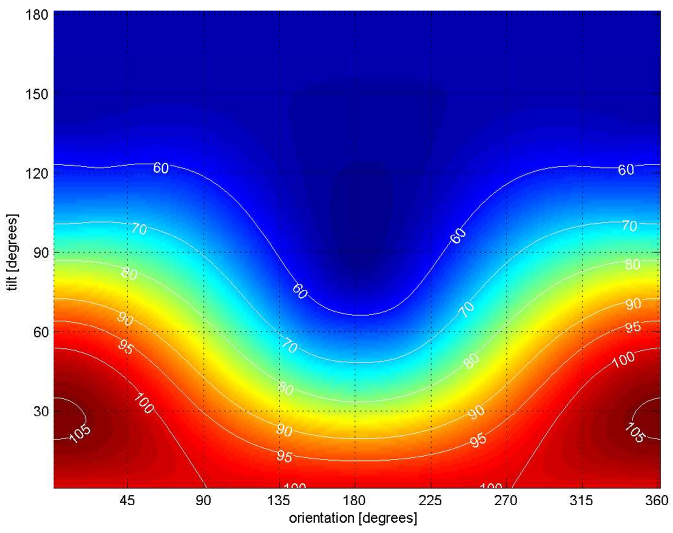
CAR (Klucher)

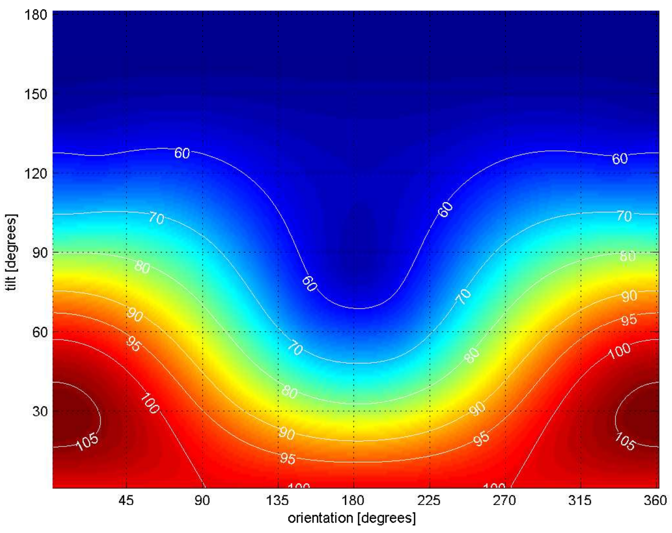


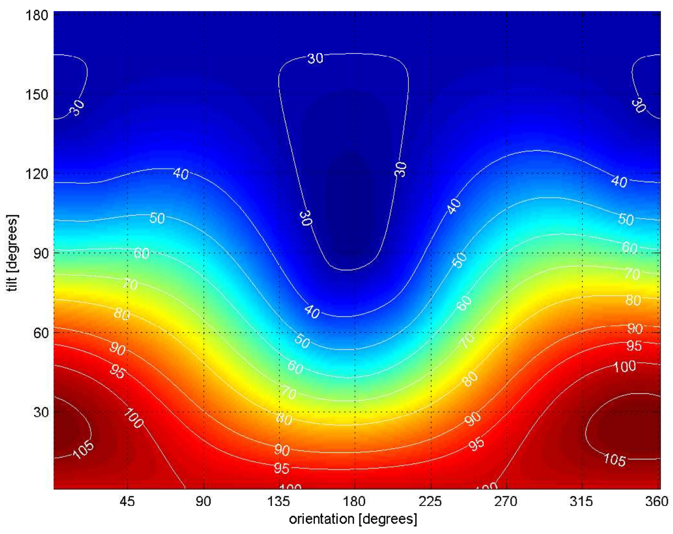
CNR (Liu-Jordan)

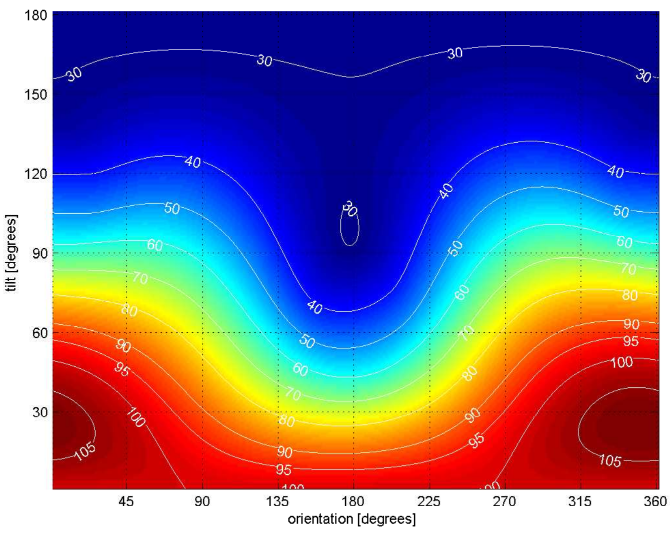


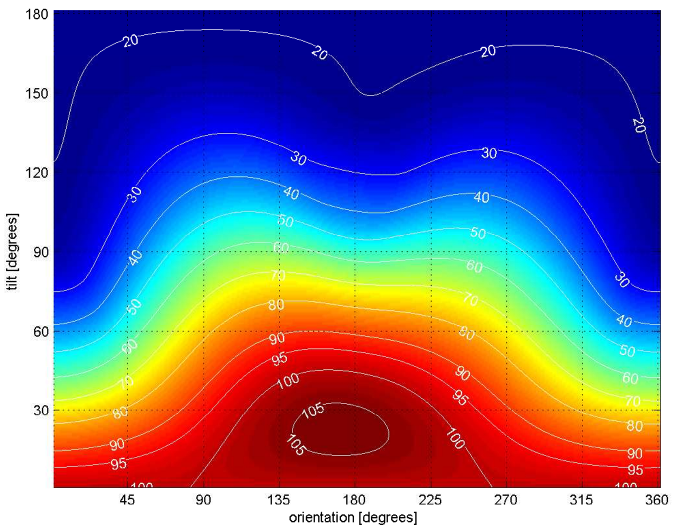
CNR (Klucher)

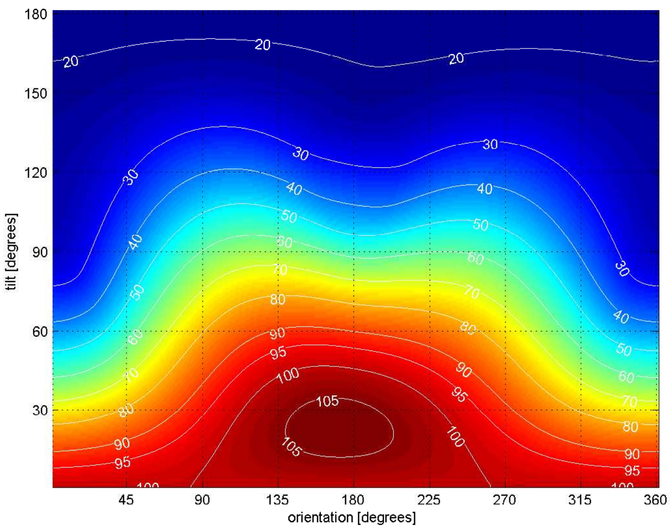

FLO (Liu-Jordan)

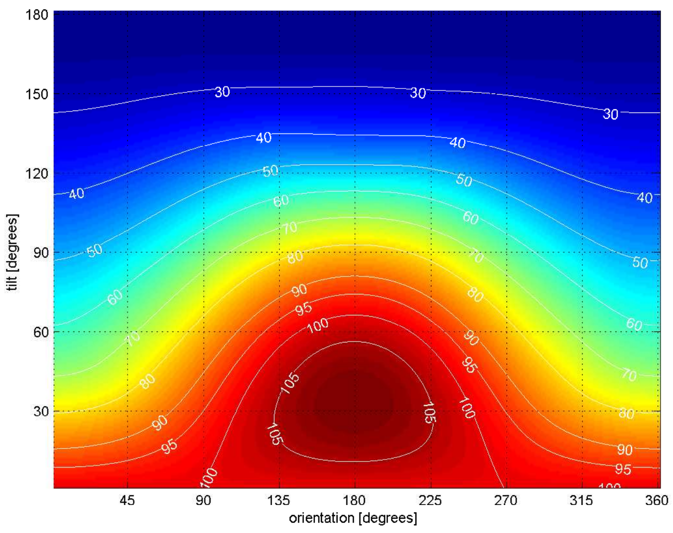

FLO (Klucher)


GOB (Liu-Jordan)

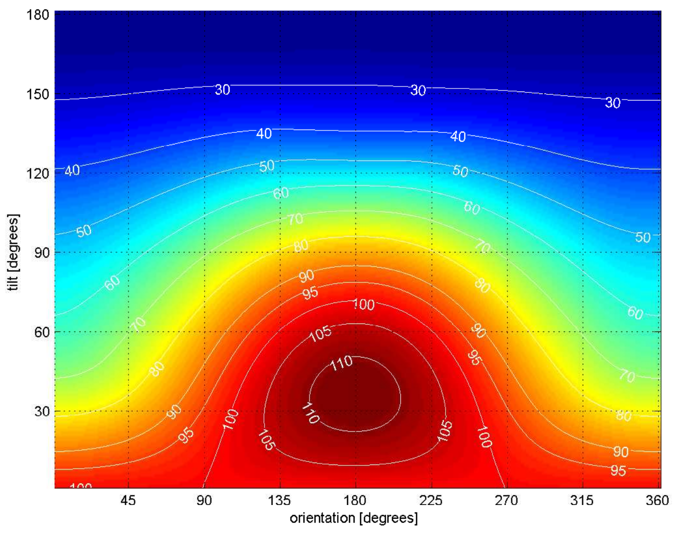

GOB (Klucher)


IZA (Liu-Jordan)

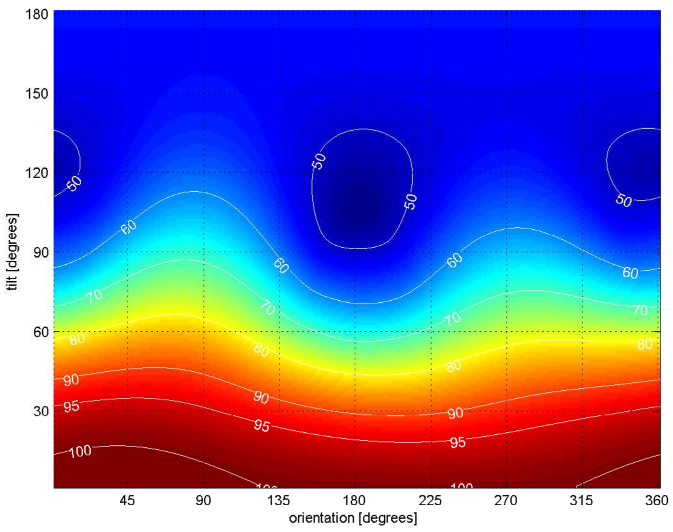

IZA (Klucher)



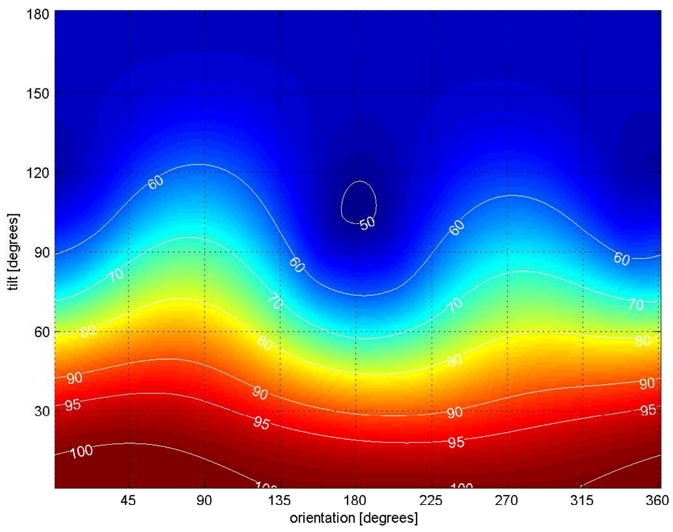
LER (Liu-Jordan)



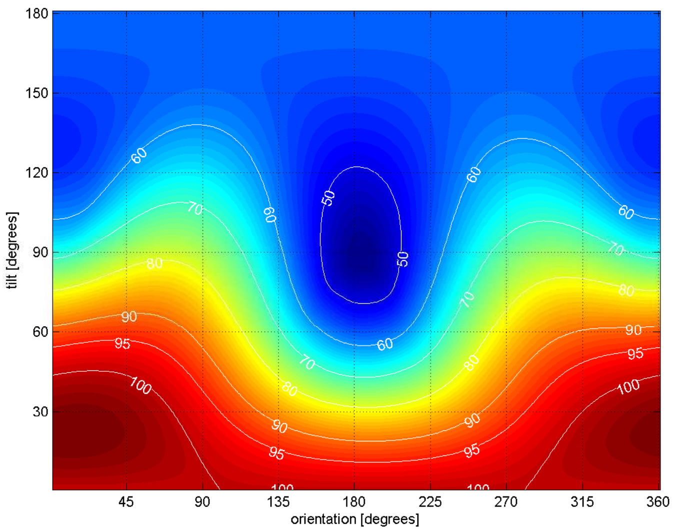
LER (Klucher)



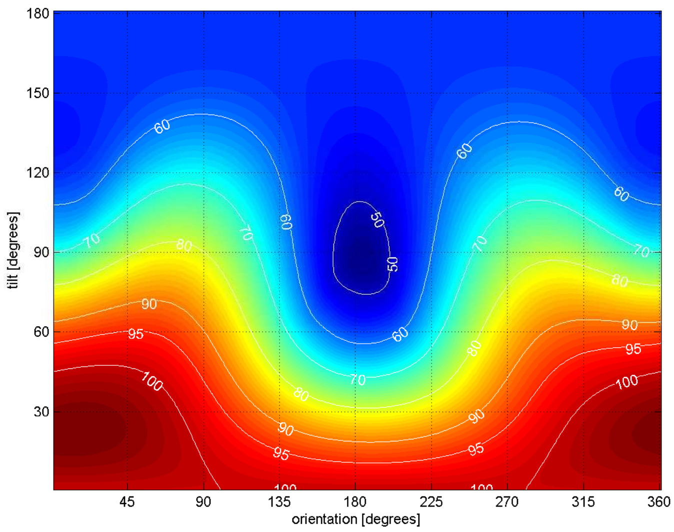
PTR (Liu-Jordan)



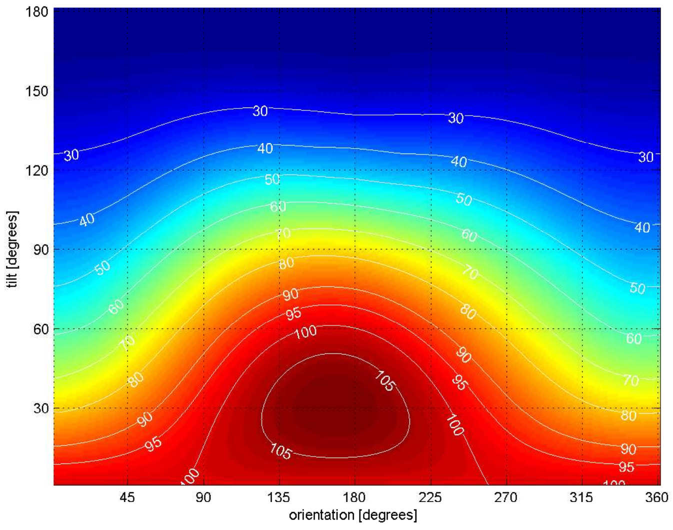
PTR (Klucher)



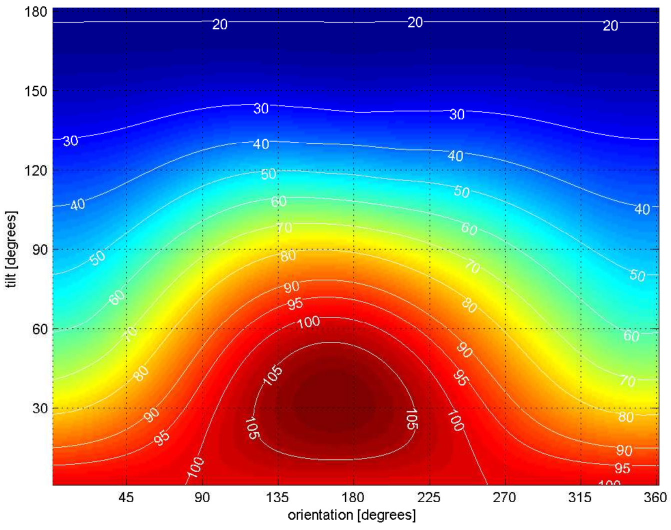
SMS (Liu-Jordan)



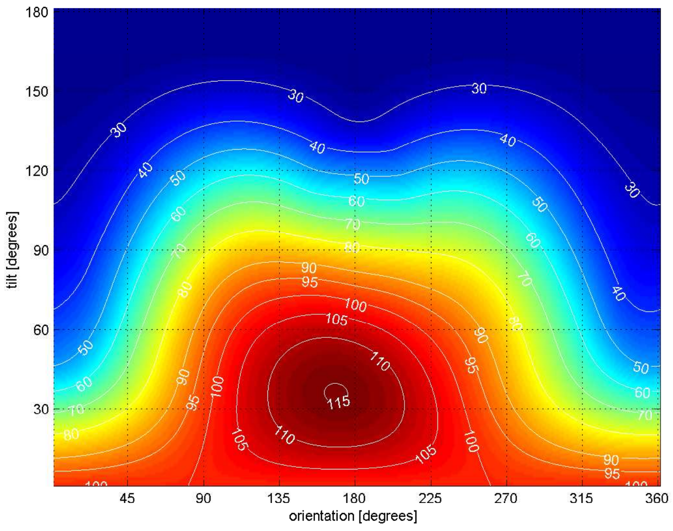
SMS (Klucher)



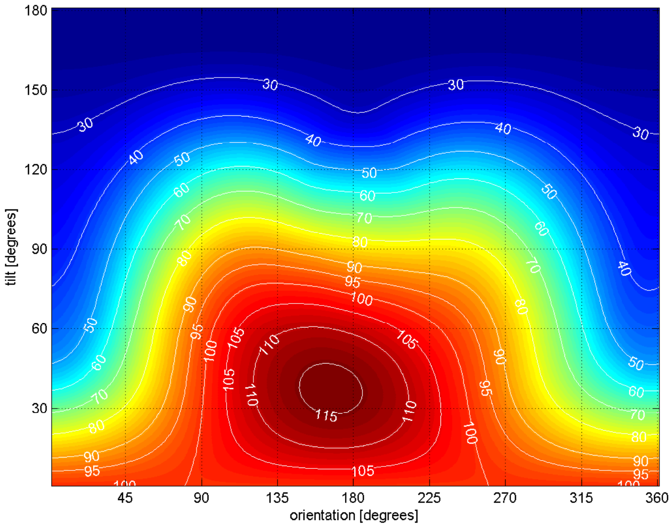
SON (Liu-Jordan)



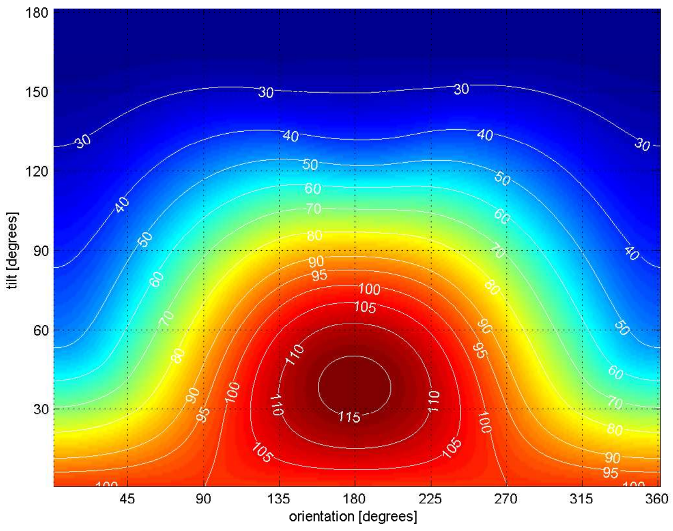
SON (Klucher)



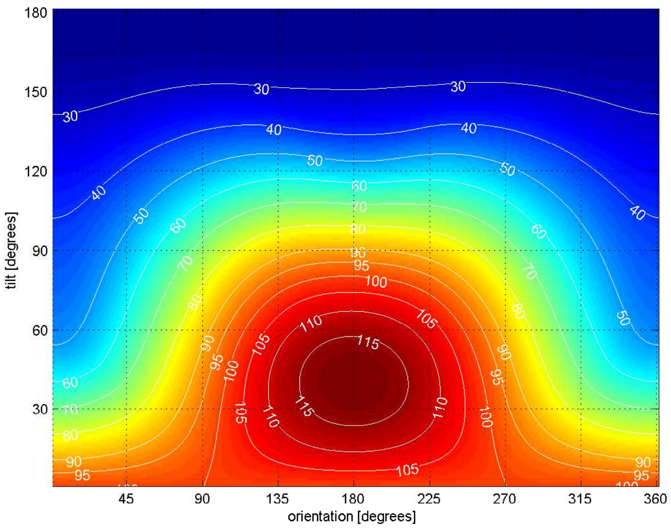
TAM (Liu-Jordan)



TAM (Klucher)



TOR (Liu-Jordan)



TOR (Klucher)

Bibliography

[1] M. Gulin, M. Vašak, and M. Baotić. Estimation of the global solar irradiance on tilted surfaces. In *Proceedings of the 17th International Conference on Electrical Drives and Power Electronics (EDPE 2013)*, pages 334–339, Dubrovnik, Croatia, October 2-4, 2013.

[2] T.M. Klucher. Evaluation of models to predict insolation on tilted surfaces. *Solar Energy*, 36(4):111–114, 1979.

[3] B.Y.H. Liu and R.C. Jordan. Daily insolation on surfaces tilted towards the equator. *ASHRAE Journal*, 3(0):53–59, 1961.

[4] I. Reda and A. Andreas. Solar Position Algorithm for Solar Radiation Applications. Technical Report NREL/TP-560-34302, National Renewable Energy Laboratory (NREL), Revised January 2008.

[5] World Meteorological Organization (WMO). Baseline Surface Radiation Network (BSRN) - Update of the Technical Plan for BSRN Data Management, October 2013.