Solar Panel Output Estimation

Solar PV output is determined by the amount of solar radiation a module receives, influenced by the weather (cloud cover, temperature, shade etc.), and the module's efficiency (See Appendix A: Understanding solar PV calculations).

The Microgeneration Certification Scheme Service (MCS) creates and maintains the standards for low-carbon products and installations used to produce electricity and heat from renewable sources in the UK. It has data on all the solar PV arrays installed in the UK with their models and estimated yearly solar PV output, calculated from a formula using the (1) shading on the roof, (2) roof slope, (3) roof azimuth or orientation towards the sun and (4) number of solar panels that can be installed.

The problem is that determining these four input variables requires a site visit. Instead, the following documentation shows how to derive these values from other data sources and compute the estimated solar PV output. We will focus our scope on residential buildings in the West Midlands.

The three primary functions are performed using QGIS and Python:

1. Roof segmentation to get slope and azimuth
2. Computing average shading from a Digital Surface Model (DSM)
3. Estimate yearly solar panel output by plugging in values to the formula

NOTE: I successfully ran the code on my local Windows machine. However, the licensed Ordinance Survey data required us to use Aridhia (a secure platform) to run the scripts. The scripts get stuck on certain functions without throwing an error, and we could not fix the issue. Therefore, we could not get estimates for all the West Midlands, only one 5km by 5km tile. The method should be applied to a larger area to get a better sense of the viability of this method.

Validation data

We require a validation set for our estimates.

Mapping Solar PV Potential in Ambleside used satellite imagery and Google Street View images to gather test data for the actual area, slope and azimuth of a roof. Then, they input these values to the EU JRC PVGIS online tool to find an estimated true energy output. EU JRC PVGIS calculates solar radiation data from satellite images. It does not validate against any ground truth data, such as getting meter readings from houses on their energy output.

Currently, no ground truth data on solar PV output is available. A more thorough investigation is required to acquire a test data to compare these estimates.

For now, we will take the MCS data as our truth measure since it also uses EU JRC PVGIS data to measure solar radiation. With this, we will validate the roof's slope, aspect, shading and area by plugging in our values to the same formula. We should get a similar order of magnitude of solar PV output in our results.

Data

Ordinance Survey Maps

For more details on the Ordinance Survey data used here, see *Getting Proxies*. This is an overview of the information available.

1. OS MasterMap Topography Layer, which provides the building footprints
2. OS MasterMap Building Height Attribute, which provides the heights of each building
3. AddressBase Premium which provides the address data for each house

If you don't have access to the licensed data which provides the building footprint and height, you can pull building footprints from Open Street Maps, but it will not be as extensive or accurate given that the data is crowdsourced.

Remote Sensing Data (LIDAR) from Defra

With the data from OS, we could already start to estimate the solar PV output, but it would assume that all roofs are flat, which is not a fair assumption to make for most houses. The roof slope and its orientation to the sun play a big role in the amount of sun a solar PV model will receive. As seen in Figure 1, the rooftops of each house can be split into segments which each receive the almost the same amount of light.

Figure 1 Photograph of a residential area in the afternoon where shadows cast on the rooftops are not equal on all sides. [Image from Tom Rumble]

We can use LIDAR (Light Detection and Ranging) data, which is essentially pulses of light used to gauge distance, to get finer-grained information on the height of the roofs. LIDAR works somewhat similarly to echolocation (used by whales and dolphins), if you are familiar with how that works. An aircraft flying at a fixed altitude sends out pulses of lasers (high-frequency light). The lasers reflect on surfaces, and the time it takes for the lasers to return provides the distance.

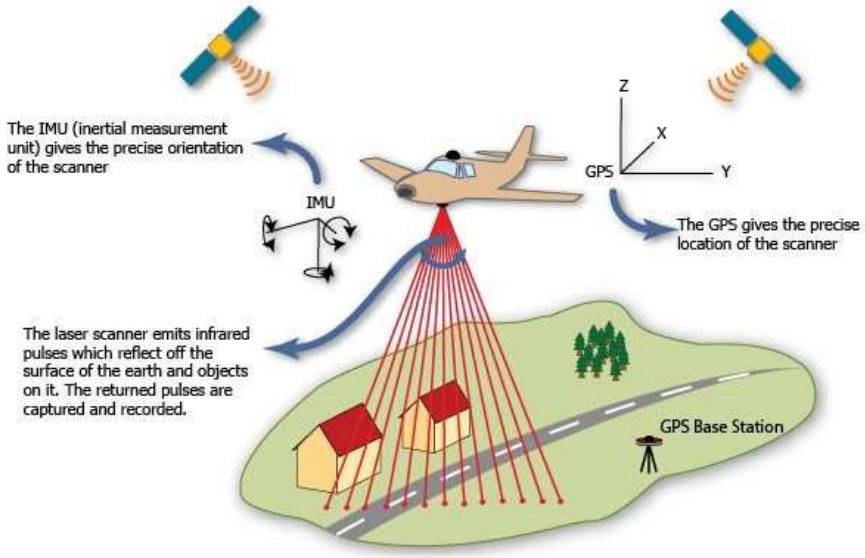


Figure 2 LIDAR data gives information on elevation collected from laser pulses. [Image by from Anthony C. LoBaido]

The final output is an image where the height of each associated location represents each pixel. So, with a resolution of 25cm, the highest resolution available, every pixel will represent 25cm by 25cm of real space. However, a 25cm resolution is only available for a tiny portion of the country, while a resolution of 1m and 2m covers almost everything. We will use a 1m resolution to compromise.

The LIDAR data is provided open source by the UK's Defra (Department for Environment, Food and Rural Affairs). You can download individual tiles from the website, but if you want to bulk download, email Defra. They got back to me in 3-4 business days. I asked for LIDAR Composite DSM[1] at 1m resolution and gave them the shapefile of the West Midlands.[2] The LIDAR data was last updated in 2017, meaning any buildings renovated or built after that will not be captured by LIDAR.

All files were given in .asc format and are single-banded with the height captured. Available tiles and the number of missing ones are given in *osmap_tiles.txt.* Some tiles also look like the sample in Figure 3, where the LIDAR data is patchy. In those cases, we will skip the roof segmentation and assume a flat roof. Overall, 34.76% of the West Midlands has DSM coverage.

[1] Digital Surface Model (DSM): information on the elevation of the Earth's surface including man-made structures (eg: buildings, bridges) and natural features (eg: trees, grass).
[2] You can also ask for the DTM (Digital Terrain Model) to calculate the building height if you don't have access to OS data.

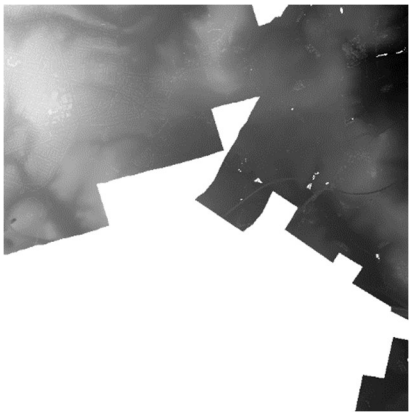


Figure 3 An example of a patchy DSM tile.

LiDAR is just one method of measuring elevation. Researchers use this data to derive the DSM and DTM. You could also use satellite imaging or 3D models, but those are usually harder to find open-source and requires more processing power for the same level of granularity.

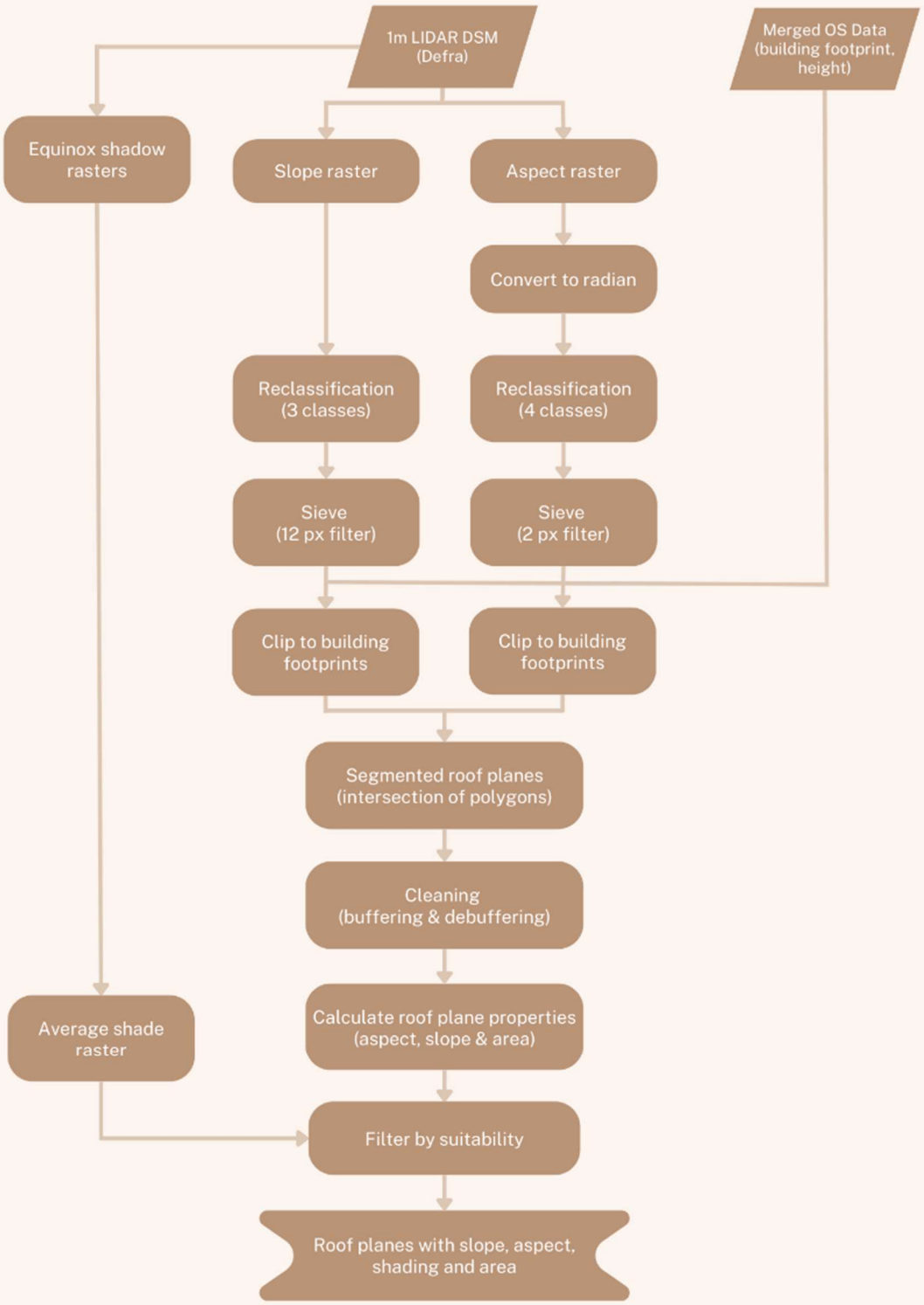
Roof Segmentation

This section aims to improve the estimated solar PV output accuracy by taking a roof by its parts instead of looking at the roof as a single unit. The results of the roof segmentation will be evaluated by eye since no other validation data is available.

Method

The following methodology is adapted from Mapping Solar PV Potential in Ambleside. Our method only requires the DSM layer and building footprints, while the cited paper additionally requires the DTM layer to filter for height. We don't require a DTM because we have the Building Height Attributes from OSMap, but in the absence of this, you can use a DTM to supplement that data.

Roof Segmentation



Validation

The roof segmentation results are dubious even in the paper itself. The roof segments do not look like the segments you would draw yourself (Figure 4). The author only used a test set of 20 rooftops to get an accuracy for slope and aspect within 10 degrees for 75% and 70% of the houses, respectively. These results leave much to be desired, and the poor roof segmentation indicates some of the author's future work recommendations.

Figure 4 Roof segments identified in QGIS (left) and the actual view of roofs from the same area on Google Maps.

Calculating Average Shading

The shading is typically cast by other taller buildings or vegetation. With the DSM layer, we can get a reasonably accurate idea of the amount of shade cast onto a roof (at least accurate to a year ago). Without the DSM, we can still create a pseudo-DSM layer from the building height information.

Pseudo-DSM

Without a DSM, you can still compute the shading as long as you have the building height values and footprint. To create the pseudo-DSM, rasterise the vector layer with the building height as the pixel value and leave all other pixels as '0'.

It will not be as accurate because our data only contains the heights for residential buildings and not all buildings. It also does not consider the height of vegetation.

UMEP: Solar Radiation: Shadow Generator

UMEP (Urban Multi-scale Universal Predictor) is an open-source climate service tool available as a plugin on QGIS. The Shadow Generator creates a model of the shadows produced at different times of the day and year from a DSM. However, it is a computationally expensive process. For scale, it takes 1 hour and 15 minutes to calculate the average shading of a 500 pixel by 500 pixel DSM over two days at two-hour intervals.

I chose the equinoxes to represent the average shading for the year, but given a lower computation time, I would also include the summer and winter solstices.

1. Spring equinox: 20 March 2022
2. Fall equinox: 23 September 2022

As a ballpark estimate, given we have 58 tiles that cover the West Midlands, it will take 2.7 days to complete computing the average shading on each house.

Filtering for solar panel viable houses

At this stage, we should have two datasets which will be filtered based on the following:

Column	Filter	With DSM	Without DSM
UPRN		✓	✓
Roof Area (m2)	>5 m^2	✓	✓
Shading (%)		✓	✓
Height (m)		✓	✓
Aspect (radian)	67.5° and 292.5°	✓	
Slope (degree)	<60°	✓	
parentUPRN	To group	✓	

Estimating Yearly Solar PV Output

We aim to identify solar panel viable houses, which are essentially houses that (1) have a suitable roof to mount a solar panel (i.e. slope and area), (2) produce enough electricity to make the installation costs worth it. The DSM gave us information on the roof slope, area and azimuth to filter houses, and the following will give us an estimation for electricity production.

We compared the MCS estimates with another method of estimation from an open-source Python library -- pvlib[3] with on a single 5km by 5km tile before scaling up the process. Ultimately, we found that the pvlib estimates were within the same order of magnitude as MCS estimates without requiring information on the slope and aspect of the roof. The method can be scaled easily with only the building footprint and height information.

The following two subsections explain the two methods, and the third subsection compares the methods in-depth.

Method 1: MCS Estimates

MCS's calculation for estimated annual output (AC) is

$$AC = kWp \times Kk \times SF$$

where,

- AC is in kWhour/year
- Kk is the solar radiation input factor that can be mapped from the MCS Irradiance Datasets. You will need the slope, azimuth and location area given as different sheets. The kk factor data has been provided by the European Commission, Joint Research Centre. The data is drawn from the Climate-SAF-PVGIS dataset and multiplied by 0.8

[3] There was another package called Solar Energy on Building Envelopes (SEBE) from UMEP which had the most potential for increasing accuracy, but I've had to scrap it because it takes far too long to process and relies heavily on a DSM layer which not all houses have.

- kWp is the array size given by the panel output (W) x number of panels, also known as the installed capacity of the system.
- SF is the shading factor or estimated percentage loss from shade which typically requires a site visit to assess objects which might obstruct the solar PV array. A model for different objects is used to determine the amount of shade.

From the above, we have derived the shading, area, slope and aspect of the roofs. Then, we can map the values to the input.

- Kk
  - Azimuth angle was converted to degrees relative to due South.[4]
  - Then, values were mapped from sheet 'Zone 6 – Birmingham', the central location in the West Midlands.
- $kWp = kW \times N$
  - kWp: capacity of a single (1m2) system which we took as the average peak 2.5kW
  - N: Number of 1m2 systems taken from the roof area after adding a 1m buffer and rounding down the values to get a lower bounds estimate.

For houses without a DSM, we can estimate a shading factor and provide and take an average value for slope and aspect, but it would not be fruitful for the estimation since most of the values required are still unknown. Therefore, we only performed these estimations on areas with a DSM.

Method 2: pvlib

Pvlib is a Python library for simulating the performance of photovoltaic energy systems. I am using the following default settings from System Advisor Model (SAM), a free techno-economic software model for renewable energy.

- Inverter: `'cecinverter': 'ABB__MICRO_0_25_I_OUTD_US_208__208V_'`
  - Clean Energy Commission (CEC)
- Module: `'SandiaMod': 'Canadian_Solar_CS5P_220M___2009_'`
- Temperature model parameters: `'sapm': 'open_rack_glass_glass'`[5]

The model also takes meteorological data from PVGIS based on coordinates and factors in the solar panel module's slope, aspect and elevation. Then, the solar PV output for each roof segment is aggregated to each house.

If the slope and aspect are not available, we will calculate the solar panel output for several acceptable combinations of slopes (0°-60°, every 10°) and aspect (0°-359°, every 10°). Then, we will take the average output as the solar PV output for the house.

The output is the potential solar PV output in kWhr/year/m^2.

[4] See Appendix B – Performance Estimation Method in The Solar Pv Standard (Installation) from MCS.
[5] Other options: `'close_mount_glass_glass', 'open_rack_glass_polymer', 'insulated_back_glass_polymer', 'freestanding', 'insulated'`

Cutting down computing time

Running a loop over all 1.2 million houses will cause even 0.1 seconds of computation to take over a day to run. Therefore, we will trade some precision for computational efficiency:

1. Weather: Using one weather data frame per tile since the weather difference is negligible within a 5km by 5km space is negligible.
2. Solar irradiance: Using one solar irradiance value per tile since the amount of energy received at each point within a 5km by 5km would be impacted by shading or the slope and aspect of the solar module, which we are varying anyway. The difference in solar irradiance within a few 100 meters of altitude difference is also negligible.
3. Slope values: We put the maximum slope at 60° because a tilt at a higher angle would be close to vertical and less efficient. See here for a graphical visualisation of the solar PV output variation with slope. The precision of 10° was chosen arbitrarily.
4. Aspect values: Following the author's methodology, we took aspects between 67.5° and 292.5° for pitched roofs (>10° slope) and did not filter for flat roofs (<10° slope).

Finally, each roof segment needs to be reassigned to a UPRN and matching them takes around 40 minutes for one tile.

Comparing estimates

The houses from the MCS database do not have UPRNs, so we matched them to the houses in our current dataset based on their addresses.

1. Filter houses in the database for matching postcode
2. Merge building number and thoroughfare from the database, and merge the three address lines from MCS database
3. Match filtered houses based on their addresses if they are more than an 80% match

Since we could only run the script on one tile, we found 542 houses with installed solar PV from the MCS database, which matched the tile. We compared these to our estimations for all the buildings in the area with a pseudo-DSM and the calculations from pvlib. Almost all the estimations were within the same order of magnitude, which means that it serves as a good ballpark estimate, especially given that we do not have a value for the roof aspect and slope. However, the differences are such that we can only use these predictions as a rough indication.

Another major drawback is that it took 6.5 hours to run on one tile. We can halve this time by reducing the number of days a year we consider.

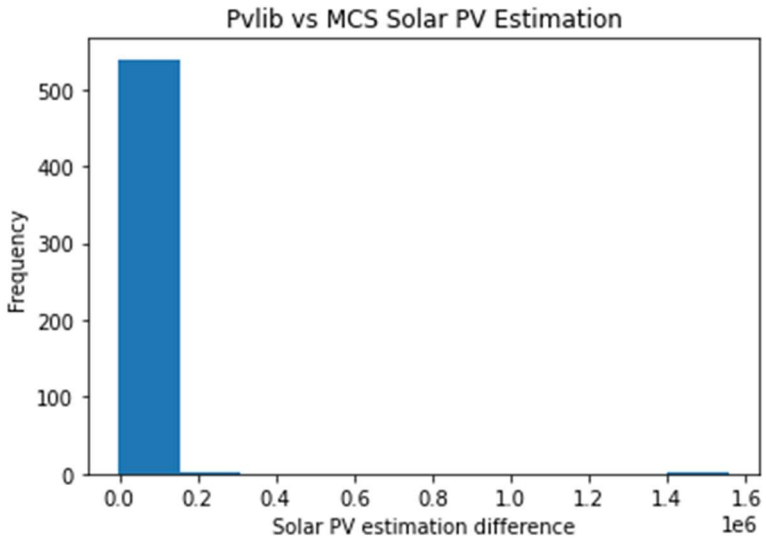


Figure 5 Difference in solar PV output estimation between pvlib and MCS calculations.

After filtering each segment, only five viable roof segments remained, so I tried to use the unfiltered roof segments. But it did not yield a significantly larger sample size. The method may not prove workable because the roof segmentation does not produce visually correct results. This likely degraded the accuracy of future calculations. More work will have to be done to make this method workable.

Final Output

Column Name	Column	Type	Source
uprn	UPRN	Int	AddressBase Premium
roof_area	Total roof area including 0.5m buffer (m^2)	Float	OSMap Topology
height_mean	Building height from sea level (from OSMap AbsHMax)	Float	DSM or OSMap Building Height Attributes
shading_mean	Average proportion of light received	Float	
aspect_mean	Roof aspect (only for houses with DSM) in radian	Float	
slope_mean	Roof slope (only for houses with DSM) in degrees	Float	
lat	Centroid latitude	Float	
lng	Centroid longitude	Float	
pv_output	Total solar PV output *(pv_output x roof_area*) in Whr/year	Float	

full_address	Building number, thoroughfare	String	AddressBase Premium
postcode		String	AddressBase Premium
parentUPRN		String	AddressBase Premium

Future Work

The current methodology leaves much to be desired to get better estimates:

- As mentioned above, Aridhia issues prevented us from running the scripts for *01_calc_shadows* on all the tiles. This issue can be resolved by running the scripts locally.
- Acquire validation data for roof segments, aspect, slope and area
- Improve roof segmentation for each segment to at least look visually correct
- WMCA had concerns about the additional load on the electricity grid when more solar panels are installed. Estimate the self-consumption to determine the amount of electricity pushed back into the grid.
- Acquire validation data for actual solar output to better compare between estimates: (1) pvlib, (2) MCS, and (3) MCS with our inputs.
- Determine the number of solar panel modules that can fit on a roof based on its shape
- The current shading method does not account for shadows cast by houses that lie beyond the border of the tile. This means there is a higher rate of error for all the houses along the border of each tile. I had planned to solve for this by breaking each tile into overlapping smaller tiles. Tiling also prevents QGIS from crashing with too big a tile.
    - For example, say the following table represents a 1000 pixel by 1000 pixel tile.
    - The idea is to split it into 500 pixel by 500pixel tiles with a 20pixel overlap, making 9 files (4 with 500x500px, 2 with 40x500px, 2 with 500x40px, 1 with 40x40px). Then, merge the tiles up to half of the overlap cells (10px) to capture the shading from buildings beyond the tile edge.
    - However, because the shading algorithm is a time intensive process we will need to determine if this method is worth the wait.

			O	O			O	O	
			O	O			O	O	
			O	O			O	O	
O	O	O	O	O	O	O	O	O	O
O	O	O	O	O	O	O	O	O	O
			O	O			O	O	
			O	O			O	O	
O	O	O	O	O	O	O	O	O	O
O	O	O	O	O	O	O	O	O	O
			O	O			O	O	

Appendices

Appendix A: Understanding solar PV calculations

The following is an overview of how solar PV output is calculated. For more detailed information read: https://www.pveducation.org

Typical Meteorological Year (TMY) data provides the information on the weather. Meteorological centers constantly take measurements on an arbitrary oriented and tilted surface and aggregate them for each hour. For example, the total direct normal irradiance at 8am is the total captured from 7am to 8am. All this information is open source.

The amount of solar radiation received by a module is calculated by:

$$Total\ radiation\ (G)\ =\ Direct\ beam\ (B)\ and\ diffuse\ (D)\ components$$

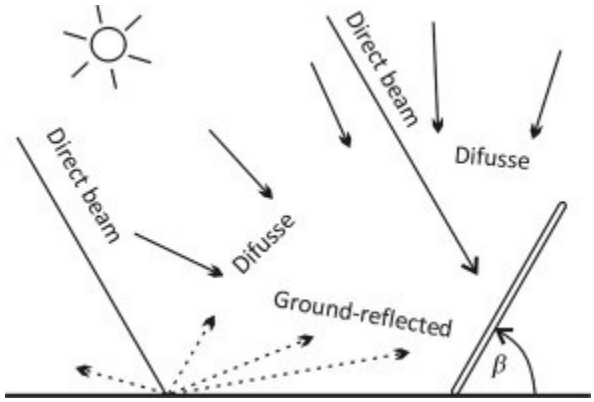


Figure 6 Diagram showing direct vs diffuse radiation. [Image from Martinez-Garcia, Arauzo & Uche]

Direct beam is the amount of sun rays a surface receives perpendicularly. In TMY, this is labelled as Direct Normal Irradiance (DNI), measured by having a sensor track arc of sun. A fixed place solar PV only gets a portion of the rays, calculated by:

$$\begin{aligned} B = DNI\,(&\sin(\delta)\sin(\varphi)\cos(\beta) - \\ &- \sin(\delta)\cos(\varphi)\sin(\beta)\cos(\psi) + \\ &+ \cos(\delta)\cos(\varphi)\cos(\beta)\cos(HRA) + \\ &+ \cos(\delta)\sin(\varphi)\sin(\beta)\cos(\psi)\cos(HRA) + \\ &+ \cos(\delta)\sin(\psi)\sin(HRA)\sin(\beta)) \end{aligned}$$

where, **δ** is the Declination Angle (tilt of Earth on its axis), **ϕ** is the latitude of the location, **β** is module tilt , **ψ** is module azimuth (orientation measured from South to West) and **HRA** is hour angle, number of degrees which the sun moves across the sky which is 0° at solar noon.

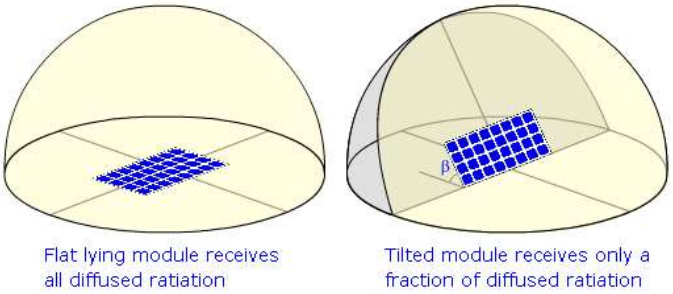


Figure 7 Diagram showing how the proportion of diffused radiation received changes with tilt. [Image from C.B.Honsberg and S.G.Bowden, "Photovoltaics Education Website," 2019.]

Diffuse component are the sun rays that don't reach the solar module perpendicularly. In TMY, the total a module would receive if it was flat is called the diffuse horizontal irradiation (DHI). The proportion the module receives depends on its tilt β, given by:

$$D = DHI\left(\frac{180-\beta}{180}\right)$$

Appendix B: SEBE (Solar Energy on Building Envelopes) on UMEP

SEBE is a function under the UMEP plugin on QGIS, which uses a shadow casting algorithm with a digital surface model (DSM) and the solar position to generate pixel-level information of shadow or sunlit areas. It calculates the raw solar potential of a surface without accounting for any energy loss or transformation efficiency from solar PV.

SEBE requires an additional dataset: Typical Meteorological Year (TMY) .txt, which can be created by running an EnergyPlus Weather file through the MetPreprocessor (another plugin under UMEP). Unfortunately, no processing tool is available for the meteorological files, so each file will have to be processed manually.

Using SEBE would be ideal since it automatically takes shading and weather into account when calculating solar PV output, but the program is too slow and relies heavily on the DSM. Using a pseudo-DSM for 65% of the area will create too large an error. **SEBE is ideal if you work with a relatively small area and have the DSM layer coverage.**

Other Resources

MCS Shading Procedure provides details on how MCS determines the shading factor from site visits.

SAP calculation of electricity generated by solar PV systems provides a comparison between different methods of estimating solar PV output. This gives a better idea of the types of inputs we might use.

Solar photovoltaic (PV) cost data provides data that might be useful to strategise incentives or subsidies for installing solar panels.