Calculation of the Shading Factors for Solar Modules with MATLAB

Martín Silva, Justo Jose Roberts * and Pedro Osvaldo Prado

Abstract: Shadows severely affect the performance of solar photovoltaic (PV) systems. A proper description of this effect is useful for sizing and simulating PV systems when shadows cannot be avoided. Shading factors represent the basis for simulating the effect of shadows on solar modules. These factors can be used to estimate shading losses, calculate their I-V and P-V curves under shading conditions, or develop new maximum power point tracking (MPPT) techniques. Open-source libraries focused on solar energy have gained popularity in recent years. One of the currently most popular ones is the PV_LIB toolbox initially developed by Sandia Laboratories. PV_LIB significantly facilitates solar energy calculations. However, it currently lacks functions for taking into account shaded conditions. In this paper, a detailed Matlab-based method for calculating the shading factors is provided. The method has been used for elaborating a toolbox for shading calculations. The current work could help extend the functionalities of the PV_LIB toolbox. The results were compared against other currently popular computer programs, namely the System Advisor Model (SAM) and PVsyst. With this method, it is also possible to calculate shading factors with smaller time steps than possible with the mentioned programs. This work also shows the importance of using small time steps and how this can affect the accuracy of the calculated shading factors. The contribution of this work is providing a way of quantifying shadow losses in PV systems with Matlab, allowing for better accuracy, flexibility, and transparency during the calculation. The functions developed in this work can be accessed by contacting the authors.

Keywords: solar; shadows; Matlab; SAM; PVsyst

1. Introduction

With the reduction of solar energy costs, government initiatives in many countries and a general trend toward renewable energies, the use of solar photovoltaic (PV) systems in urban and residential environments has greatly increased [1–7]. Urban environments often include obstacles that could cast shadows on a PV system, badly affecting energy production. In Germany, studies have shown that shading is one of the main causes of a lower energy yield [8]. Results from the German 1000 Roofs Programme in 1990 revealed shading losses of up to 10% in more than half of their PV systems [4,8,9]. More recently, an analysis of 46 residential PV systems in the United States showed that annual shading losses can account for up to 20% [10]. Bayrak et al. [11] conducted an experiment where they applied different shading ratios to a single module. They analyzed three shading configurations: namely, shading of a single cell, horizontal shading of a row of cells, and vertical shading of a column of cells. The study showed that horizontal shading can completely reduce the power yield of a module when all by-pass diodes are affected, while single-cell shading and vertical shading presented reductions of 69.92% and 66.93% respectively. More recently, Numan et al. [12] investigated theoretically and experimentally the impacts of various cases of partial shading (vertical string, horizontal string, and single cell) on the performance of a photovoltaic system. The authors found that at the...
100% shading condition, the maximum power dropped by 99.36%, 43.7%, and 41.15% for horizontal, cellular, and vertical shading at the same solar radiation level compared to their initial state values.

Limited space in urban areas and the inability to avoid or remove obstacles demand a method for including their effects in the PV system design phase. The reduction of global irradiance due to shadows can be computed with shading factors that reduce its components linearly [13]. The electrical effect needs to be computed with mathematical models of the solar module, such as the one-diode or two-diode model [14,15]. This can be computationally expensive, and some authors have proposed using look-up tables with pre-calculated values to speed up calculations [16]. In this paper, the main focus is on the global irradiance reduction and the computation of shading factors.

Even though shading factors have been used for more than 25 years, the number of articles on this topic is limited. In 1995, Quaschning and Hanitsch [13] proposed the use of shading factors and a vectorial approach, where the module, the obstacle, and the sun are represented as vectors. This allowed one to easily calculate the shadows projected on the surface of a PV array. In 2011, Cascone et al. [17] presented a similar procedure for the calculation of the shading factors. Although this work was meant for the study of heat gain in buildings, the principle of the factor is the same as for solar modules. The procedure is very similar to the one proposed by Quaschning. However, it is much more detailed.

Melo et al. [18] developed an add-in for SketchUp that can be used for calculating a table of direct shading factors and the diffuse shading factor as well. Their approach is the same used by Quaschning and Cascone. With SketchUp, they can analyze shading situations with complex geometries. Westbrook et al. [19] modeled diffuse shading losses with an analytical approach based on an isotropic sky. They limited their study to an array being shaded by another array of unlimited length and compared the results to experimental data. Li et al. [20] proposed a pixel-based methodology to assess the annual solar potential of building rooftops. In this work, the authors used SketchUp to extract images depicting a certain shading situation and then processed the images with Matlab.

Currently, many computer programs use shading factors when calculating shadow losses. The most popular ones are SAM [21], PVsyst [22], PV*SOL [23], and Helioscope [24]. Another option currently gaining more popularity is the PV_LIB library provided by Sandia National Laboratories [25]. This library is free and available for both Matlab and Python. It has many functions that greatly facilitate the simulation of solar modules. Despite this, there are currently no functions in this library for the inclusion of shading losses. The present work could help extend the PV_LIB Matlab library functions to include shade losses. Matlab has been widely used for shading calculations [26]; therefore, the idea of including a Matlab-based tool for estimating shading losses is very attractive.

This paper provides a detailed and simple procedure for calculating the shading factors for both the direct and diffuse components. The procedure was implemented in Matlab, and the results were compared with SAM and PVsyst.

2. The Shading Factors

The usual approach for introducing the effect of shadows on the solar modules is including shading factors. These factors work by reducing the irradiance reaching the surface of the module. They take values from 0 (absence of shadow) up to 1 (fully shaded). The factors $f_B$ and $f_D$ are the direct and diffuse shading factors, respectively. The direct shading factor depends on the geometry of the modules, obstacles, and the position of the sun. The diffuse shading factor depends solely on the geometry of the modules and obstacles. Obstacles also reduce the reflected component, and this is usually taken into account by reducing the albedo [13]. However, this effect is out of the scope of this work. The reduced direct and diffuse irradiances can be calculated with Equations (1) and (2) [13].

\[
\text{DNI}_S = \text{DNI} \cdot (1 - f_B)
\]  

(1)
DHI_S = DHI \cdot (1 - f_D) \tag{2}

To calculate these factors, we first calculate $f_B$ with a vectorial approach. Then it is necessary to compute all the $f_B$ into a table where the rows correspond to sun elevation angles and the columns to sun azimuth angles, going from $0^\circ$ to $90^\circ$ and $-180^\circ$ to $180^\circ$, respectively. This table has two main purposes. First, it will serve as a look-up table when calculating factors for all the given sun positions throughout a year. Second, $f_D$ can be calculated by integrating the values in this table. The methodology for creating the shading table is illustrated in Figure 1.

$\begin{align*}
f_B &= \frac{A_S}{A_M}
\end{align*}$

To calculate this factor, it is necessary to calculate the area of the shadow $A_S$. First, the geometry of both the solar module and the obstacle needs to be specified. Second, the vertices of the obstacle are projected onto the plane defined by the solar module. Third, a convex hull procedure is applied to the projected points. Finally, we subtract the part of the projected shadow that falls outside the module to obtain $A_S$ and calculate $f_B$. The final step will be making a shading table [27].

2.1.1. Module Geometry Definition

The module is represented with four Cartesian coordinates. These coordinates correspond to the vertices of each corner of the module. This was implemented as a class that receives a structure with the dimensions of the module, its coordinates, and its orientation angles. The constructor of this class calculates the coordinates for the four vertices that represent the module. Figure 2 shows the four vertices $v_i$ of a module facing north (positioned on the southern hemisphere). Azimuth angles are measured from north and positive clockwise (West = 270°).
Figure 2. Geometric definition of the vertices $v_1$, $v_2$, $v_3$, and $v_4$ for a solar module with tilt angle $\beta$ and azimuth angle $\gamma$. Azimuth angles are measured from north and positive clockwise.

The initial coordinates of the four vertices are given by:

$$v_1 = (0, 0, 0)$$
$$v_2 = (0, \text{width}, 0)$$
$$v_3 = (\text{length}, \text{width}, 0)$$
$$v_4 = (\text{length}, 0, 0)$$

Then, each vertex $v_i$ is placed on its final position performing two rotations and one translational displacement. The two rotations are performed by multiplying each vertex coordinate by a rotation matrix. The first rotation is about the $y$ axis and corresponds to the tilt angle $\beta$ of the module. The second rotation is about the $z$ axis and corresponds to the azimuth angle $\gamma$ of the module. The translational displacement is simply performed by adding the displacement vector $D$ to each vertex coordinate. This calculation is shown in Equation (4).

$$v_i = R_y(\beta) \cdot R_z(\gamma) \cdot \begin{pmatrix} v_x \\ v_y \\ v_z \end{pmatrix} + D \quad (4)$$

where $R_y$ and $R_z$ are the rotation matrices. The expressions for $R_y$ and $R_z$ are shown in Equations (5) and (6), respectively [28].

$$R_y(\beta) = \begin{bmatrix} \cos \beta & 0 & \sin \beta \\ 0 & 1 & 0 \\ -\sin \beta & 0 & \cos \beta \end{bmatrix} \quad (5)$$

$$R_z(\gamma) = \begin{bmatrix} \cos \gamma & -\sin \gamma & 0 \\ \sin \gamma & \cos \gamma & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad (6)$$

2.1.2. Obstacle Geometry Definition

We limited this study to obstacles modeled as rectangular prisms. Other, more complex geometries could be included in future works. In a similar way to the module, the prism is represented with eight Cartesian coordinates, each representing one of its corners. This was done with a class that receives the dimensions of the obstacle, its position, one tilt angle, and one azimuth angle. Figure 3 shows the 3D output generated in Matlab. For more than one obstacle, they must be created and saved in a vector array of obstacle objects.
2.1.3. Shadow Calculation Procedure

The goal of this step is obtaining the coordinates of the projected shadow. This is done by projecting each vertex of the obstacle onto the plane given by the module. The coordinates of the projections can be calculated as the intersection of a line (defined by the obstacle vertex \( p_0 \)) and a plane (defined by the module). Quaschning provides Equation (7) based on this principle [13], where \( a \) is a vector perpendicular to the module, \( p_0 \) is a vertex of the obstacle, \( p_s \) is the projection of \( p_0 \) on the plane defined by the module, and \( s \) is a unit vector indicating the position of the sun. Figure 4 shows one projected vertex of the obstacle on the surface of the module. Vectors \( v_i \) are the corners of the module.

\[
p_s = p_0 - \left( \frac{a \cdot (p_0 - v_1)}{a \cdot s} \right) s
\]  
\[
a = (v_4 - v_1) \times (v_2 - v_1)
\]  

**Figure 3.** Visual output of one module with one obstacle generated in Matlab.

**Figure 4.** A vertex \( p_0 \) of one obstacle is projected on the plane defined by the module. The projected point is called \( p_s \). The vector \( s \) is a vector pointing towards the sun. The vector \( a \) is normal to the plane of the module. \( v_1, v_2, v_3, \) and \( v_4 \) are the positions of the vertices that define the module.
It is important to notice that when $s < 0$, the sun will be behind the surface of the module. Knowing when the sun is behind the surface of the module will be helpful when estimating the diffuse shading factor.

On the other hand, vertices behind the module will also have projections on the plane. This will have no physical sense and therefore, these vertices need to be removed. If part of an obstacle is behind the module, the obstacle will have to be sliced. This procedure can be complex because it will lead to a new shape with new vertices. However, one simple approach that works for not-tilted rectangular prisms is moving the points that are behind the plane vertically onto the plane (see Figure 5).

\begin{equation}
xa_x + ya_y + za_z + d = 0 \tag{9}
\end{equation}

\begin{equation}
p_{0,z} = -\left(a_x p_{0,x} + a_y p_{0,y} + d\right) / a_z \tag{10}
\end{equation}

where $p_{0,x}$, $p_{0,y}$, and $p_{0,z}$ are the $x$, $y$, and $z$ coordinates of the obstacle vertex, respectively.

After projecting all the vertices on the plane, a convex hull procedure needs to be applied. The reason is that not all projected vertices will correspond to shadow vertices (see Figure 6). This is easily done in Matlab with the convhull function [30].

First, we temporarily remove the $z$ coordinate of the projected points. Then, we apply the convhull function, retrieve the points that correspond to the shadow, and finally include the $z$ coordinates again. The result is shown in Figure 7.
2.1.4. Shading Factor Calculation

The final step is to crop the resulting shadow to calculate the portion that falls on the surface of the module. First, we remove the z coordinate from the module vertices and the shadow vertices. This will flatten both the module polygon and the shadow polygon on the x − y plane. Then, we can convert these 2D polygons into polyshape objects [31] and find the intersection of the shadow with the module with the intersect function [32]. The result is another polygon $A_S$ that represents the shadow falling on the module as shown in Figure 8.
Finally, we compute the shadow factor as the ratio between the intersected shadow area $A_s$ and the area of the module polygon $A_M$ with Equation (11).

$$f_B = \frac{A_s}{A_M}$$ (11)

### 2.1.5. Shading Table

The shading table is a matrix of shading factors for the direct component [27]. The columns correspond to sun azimuth angles and the rows to sun elevation angles. The advantage of using a table is calculating a set of shading factors once, and then using this matrix to retrieve the factors as needed. Apart from this, the table is helpful when calculating the diffuse shading factor by the integration of its values. In our approach, we calculated the shading factors for all azimuth angles from $-180^\circ$ to $+180^\circ$ and elevation angles from $0^\circ$ to $90^\circ$. This can be done in 5-degree steps and by later interpolating the results for a 1-degree resolution. The resulting table will have a similar structure to the one shown in Table 1. Figure 9 is a heat plot that illustrates the high-resolution shading table. The white portions in the bottom corners correspond to sun positions behind the module (NaN values). It is very important to include these points and clearly distinguish them from the rest. Blue spots correspond to low shading factor values, while yellow spots correspond to shading factors closer to 1.

| Elevation \ Azimuth | $-150$ | $-100$ | $-50$ | $0$ | $50$ | $100$ | $150$ |
|---------------------|--------|--------|------|----|-----|------|------|
| 0                   | NaN    | NaN    | NaN  | 1.00 | NaN  | NaN  | NaN  |
| 10                  | NaN    | 0.00   | 0.86 | 0.92 | 0.86 | 0.00 | NaN  |
| 20                  | NaN    | 0.20   | 0.74 | 0.82 | 0.74 | 0.20 | NaN  |
| 30                  | 0.00   | 0.30   | 0.66 | 0.74 | 0.66 | 0.30 | 0.00 |
| 40                  | 0.00   | 0.34   | 0.61 | 0.68 | 0.61 | 0.34 | 0.00 |
| 50                  | 0.01   | 0.37   | 0.56 | 0.62 | 0.56 | 0.37 | 0.01 |
| 60                  | 0.19   | 0.39   | 0.52 | 0.57 | 0.52 | 0.39 | 0.19 |
| 70                  | 0.29   | 0.40   | 0.49 | 0.52 | 0.49 | 0.40 | 0.29 |
| 80                  | 0.37   | 0.41   | 0.46 | 0.48 | 0.46 | 0.41 | 0.37 |
| 90                  | 0.42   | 0.42   | 0.42 | 0.42 | 0.42 | 0.42 | 0.42 |

Table 1. Simplified version of the shading table for an arbitrary situation.
Figure 9. Heat map representation of the shading table. Yellow portions correspond to higher shading factors for the direct irradiance. Blue values correspond to low shading factors for the direct irradiance. The white parts on the bottom left and right corners of the heat map correspond to angles where the sun is behind the module (NaN values).

2.2. Diffuse Shading Factor

The diffuse shading factor \( f_D \) is constant throughout the year and for a fixed geometry needs to be determined only once [13]. If we consider an imaginary hemisphere surrounding the module, a nearby obstacle will block the sun on the area \( A_S \). \( f_D \) can be defined as the ratio between the irradiance \( I_S \) that traverses the shaded area \( A_S \) and the irradiance \( I_H \) reaching the portion of the hemisphere \( A_H \) in front of the module as shown in Figure 10. The expression for \( f_D \) is given by Equation (12) [13].

\[
f_D = \frac{I_S}{I_H}
\]

Figure 10. Calculation of the shading factor for the diffuse irradiance. If the solar module is covered by an imaginary hemisphere, then one nearby obstacle will reduce the diffuse irradiance reaching this hemisphere. \( A_S \) is the portion of the hemisphere where the sun is blocked by the obstacle. The diffuse irradiance shading factor is given as the ratio between the irradiance traversing \( A_S \) and the irradiance traversing the hemisphere \( A_H \).

The diffuse shading factor can also be calculated with Equation (13) [17], where \( R \) is the radiance of a sky element and \( AOI \) is the angle of incidence [33]. The angle of incidence is the angle between the vector normal to the array and the sun position. It is calculated
with Equation (14). To account only for the radiance component normal to the array, the cosine of the angle of incidence is applied [19]. The integration region is limited by all the points of the hemisphere that are in front of the module plane [9].

\[
f_D = \iint_{R} f_R \cdot \cos(\text{AOI}) \cdot \cos(\alpha) \cdot da \cdot d\gamma \]

(13)

\[
\text{AOI} = \cos^{-1}\left[ \sin(\alpha) \cos(\beta) + \cos(\alpha) \sin(\beta) \cos(\gamma - \gamma_{array}) \right]
\]

(14)

The \( \cos(\alpha) \) factor takes into account that the hemisphere subdivisions are smaller when the elevation angle is closer to 90\(^\circ\) [34]. If an isotropic sky is considered, the radiance \( R \) can be removed from the integrals and cancelled. Moreover, if the sky is discretized, the values of the previously calculated shading table can be used. The calculated table has a resolution of 1 degree for both the altitude and azimuth angles, going from 0\(^\circ\) to 90\(^\circ\) and -180\(^\circ\) to 180\(^\circ\), respectively. Therefore, the table has 91 \times 361 elements. Finally, Equation (13) is simplified into Equation (15). The NaN values will not add up to each summation.

\[
f_D = \frac{\sum_{i=1}^{91} \sum_{j=1}^{361} f_{BIj} \cdot \cos(\text{AOI}_{ij}) \cdot \cos(\alpha_i)}{\sum_{i=1}^{91} \sum_{j=1}^{361} \cos(\text{AOI}_{ij}) \cdot \cos(\alpha_i)}
\]

(15)

### 2.3. PV Array Class

More complex shading situations can be simulated with the use of a PV array class. The inputs are the previously created module object, number of rows, number of PV modules per row, row spacing in millimeters, and the azimuth angle of the array. This class makes an array of modules and calculates all the coordinates and vertices from each module composing the array. Figure 11 shows the visual output of one array composed by 3 rows of 4 modules each, separated by 2 m and a tilt angle of 30\(^\circ\).

![Figure 11. Visual output of a PV array modeled in Matlab.](image)

This class has its own method for the computation of the shading table. The procedure for calculating the shading table is the same as previously explained, with the only difference that this method takes into account the shading between rows. To add the inter-row shading effect, it is only necessary to calculate \( f_{BI} \) for each module from the second row of the array. We know that the inter-row shading factors will be the same for all the rows behind because the distances of separation between rows are the same. Therefore, after calculating the factors for the second row we only have to replicate these values for the rows behind. The first row will not be affected by inter-row shading. For the inter-row shading calculation, each module is shaded by one rectangular obstacle with the dimensions of the row in front of it. For example, the first module of the second row will be shaded as illustrated in Figure 12. After calculating the shading tables for each module, the final result will be a cell array where each cell contains the shading table for each module. The used class was named createPVarray and its inputs are a module object, the number of rows of the array, number of columns, row spacing, and the azimuth angle of the array.
Figure 12. First module of the second row being shaded by the row in front. The shading factor for the inter-row shading is calculated for each module of the second row, being shaded by an obstacle with the size of the row in front.

3. Validation

The proposed tool was validated against two widely used PV simulation programs, SAM and PVsyst. First, we compare the resulting shading factors for the direct component and then for the diffuse component. To achieve this goal, two different shading situations with one single module and one obstacle are proposed. The first situation is a single module shaded by a rectangular block of $1 \times 1 \times 5$ m with 2 m of separation placed on the east side of the module (situation 1). The second situation consists of a module shaded by an infinite row of modules in front of it separated by 2 m (situation 2). The size of the module is $1 \times 2$ m, its tilt angle 30°, and the azimuth angle 0°. For the validation of the diffuse factor, we compared the results for different distances of separation in both situations. Figures 13 and 14 show the 3D geometry of the proposed situations and the heat plots for the shading tables calculated in Matlab. The geometry parameters are summarized in Table 2. It is important to emphasize that because the shading factors depend only on the time, location, and geometry of the given situation, other parameters are not needed to calculate the shading factors. Moreover, these shading situations were chosen due to their simplicity. After performing this validation with situations 1 and 2, a more complex shading simulation will be performed.
Figure 14. Situation 2, one PV module being shaded by one infinite row in front. The tilt angle of the row is 30° and is positioned at 2 m from the module. The tilt angle of the module is 30° and the azimuth angle is 0°: (a) Visual output; (b) Heat plot representation of the shading table.

Table 2. Geometric parameters of the obstacle and the module for the proposed situations.

| Parameters                                         | Situation 1 (East Obstacle) | Situation 2 (Infinite Row) |
|----------------------------------------------------|----------------------------|----------------------------|
| **Obstacle Geometry**                               |                            |                            |
| Width (m)                                          | 1                          | 30                         |
| Length (m)                                         | 1                          | 0.05                       |
| Height (m)                                         | 5                          | 1                          |
| Tilt angle (°)                                     | 0                          | 60                         |
| Azimuth angle (°)                                  | 0                          | 0                          |
| Distance of separation from module (m)             | 2                          | 2                          |
| **Module Geometry**                                 |                            |                            |
| Width (m)                                          | 2                          |                            |
| Height (m)                                         | 1                          |                            |
| Tilt angle (°)                                     | 30                         |                            |
| Azimuth angle (°)                                  | 0                          |                            |

3.1. Direct Irradiance Shading Factor

We computed the $f_B$ values for an entire year (2020) in Mar del Plata, Argentina (latitude $-38°$, longitude $-57.5°$, UTC-3). Figures 15 and 16 show the results for a single day for situations 1 and 2, respectively. In principle, the results of the three models agree thoroughly. The values of PVsyst had to be retrieved from the 3D geometry interface, where 5-min time steps are available. This is because currently PVsyst is limited to annual outputs with hourly steps, and the only way of retrieving factors with higher resolution is through the 3D geometry interface.
Figure 15. Shading factor $f_B$ for situation 1 calculated for one day (1 January). $f_B$ was calculated with the System Advisor Model (SAM), PVsyst, and the proposed Matlab tool.

![Graph showing shading factor $f_B$ for situation 1 on 1 January.]

Figure 16. Shading factor $f_B$ for situation 2 calculated for one day (1 July). $f_B$ was calculated with the System Advisor Model (SAM), PVsyst, and the proposed Matlab tool.

![Graph showing shading factor $f_B$ for situation 2 on 1 July.]

When comparing the shading factors, hourly values had to be considered. Even though our model and SAM allow for smaller time steps, PVsyst is currently limited to hourly outputs. The effect of using different time steps is illustrated in Figure 17, where we calculate the shading factor with the Matlab tool for situations 1 and 2. The used time steps were 5 min, 30 min, and 1 h. It can be seen that as the magnitude of the steps increases, the shape of the figures deteriorates and loses symmetry.
Figure 17. Effect of using different time steps when calculating the shading factor $f_B$ with Matlab. For situation 1: (a) 5 min; (b) 30 min and (c) 1 h. For situation 2: (d) 5 min; (e) 30 min and (f) 1 h. As the time step increases, the yearly curve of $f_B$ deteriorates and loses symmetry.

The $f_B$ factors calculated with each program and with hourly time steps are shown in Figure 18. The first column corresponds to situation 1 and the second column to situation 2. Small differences can be seen in the shape of each figure. The greatest differences appear to be in the PVsyst results for situation 2.
The monthly mean bias error (MBE) and the root mean square error (RMSE) were calculated. The results for situation 1 are shown in Table 3. Because in this particular condition the shading factor is zero from May to August, these months were excluded. The direct irradiance shading factors calculated with the Matlab tool had good correlation with both SAM and PVsyst. The scatter plots in Figure 19 illustrate the comparison between our model, SAM, and PVsyst.
Table 3. Deviation parameters between the Matlab tool, SAM, and PVsyst for situation 1.

| Month      | East Obstacle |         |         |         | PVsyst |         |         |         |
|------------|---------------|---------|---------|---------|--------|---------|---------|---------|
|            | SAM           | MBE     | MBE (%) | RMSE    | RMSE (%) | MBE     | MBE (%) | RMSE    | RMSE (%) |
| September  | −0.001        | 14      | 0.0107  | 35      | 0.0006  | 17      | 0.0058  | 29      |
| October    | −0.0004       | −17     | 0.0079  | 11      | 0.0000  | 0       | 0.0052  | 7       |
| November   | −0.0002       | 11      | 0.0073  | 9       | −0.0001 | 4       | 0.006   | 7       |
| December   | −0.0004       | −9      | 0.0195  | 22      | 0.0004  | 1       | 0.0062  | 7       |
| January    | 0.0000        | −20     | 0.0073  | 8       | 0.0009  | 1       | 0.0113  | 13      |
| February   | −0.0219       | −20     | 0.1318  | 134     | 0.0011  | 28      | 0.0084  | 14      |
| March      | −0.0095       | −23     | 0.0721  | 149     | −0.001  | 12      | 0.0367  | 126     |
| April      | −0.0016       | −52     | 0.0082  | 239     | 0.0001  | 74      | 0.0013  | 79      |

Figure 19. Scatter plots for the direct irradiance shading factor $f_B$. Comparison of: (a) Matlab with SAM and (b) Matlab with PVsyst. Results are for situation 1, one module shaded by a single obstacle positioned on the east side.

Deviation parameters for situation 2 are shown in Table 4. Because in this situation the shading factors are zero from October to April, these months were excluded from the table. Again, the Matlab tool and SAM present similar results. However, PVsyst results differed significantly. The scatter plot of Figure 20 shows good correlation between Matlab and SAM. In this case, Matlab predicted slightly larger shading factors.

Table 4. Deviation parameters between the Matlab tool, SAM, and PVsyst for situation 2.

| Month    | Infinite Row |         |         |         |         |         |         |         |         |
|----------|--------------|---------|---------|---------|---------|---------|---------|---------|---------|
|          | SAM           | MBE     | MBE (%) | RMSE    | RMSE (%) | MBE     | MBE (%) | RMSE    | RMSE (%) |
| April    | 0.005         | 89      | 0.021   | 118     | 0.002   | 35      | 0.013   | 159     |
| May      | 0.008         | 88      | 0.024   | 57      | 0.004   | 54      | 0.015   | 62      |
| June     | 0.008         | 40      | 0.021   | 54      | 0.005   | 18      | 0.015   | 34      |
| July     | 0.006         | 60      | 0.016   | 49      | 0.004   | 41      | 0.012   | 41      |
| August   | 0.003         | 33      | 0.015   | 38      | 0.004   | 68      | 0.021   | 181     |
| September| −0.002        | −29     | 0.032   | 1005    | 0.000   | 216     | 0.006   | 1768    |
The main reason for the discrepancies of PVsyst was the hourly time steps that PVsyst used to calculate the shading factors. This shows the importance of using smaller steps during calculations. Furthermore, these results were achieved using the “Slow calculation mode” for the lineal shading simulation tool of PVsyst. This mode calculates $f_B$ for each time step and leads to relatively good results. The fast mode uses a pre-calculated shading table and can lead to greater discrepancies.

This is because the table has large steps of the azimuth and elevation angles. These angles are 20° and 10°, respectively.

3.2. Diffuse Irradiance Shading Factor

The diffuse shading factor is constant and needs to be calculated only once. To compare the model with SAM and PVsyst, we used the previously mentioned geometries. The only difference was that in this case, we varied the distance of separation $d$ between the module and the obstacle (previously considered 2 m). This allowed a broader comparison. For the infinite row, $f_D$ was calculated with different distances of separation between the module and the row in front. For the $1 \times 1 \times 5$ block, different distances from the module to the block were considered. These distances are shown in Figure 21.

![Figure 20. Scatter plots for the direct irradiance shading factor $f_B$. Comparison of: (a) Matlab with SAM and (b) Matlab with PVsyst. Results are for situation 2, one module shaded by one infinite row.](image)
Figures 22 and 23 show the diffuse shading factors for the situations described. Small discrepancies are observed. The proposed tool yielded good correlation with the results obtained in PVsyst. This suggests that our model and PVsyst use a similar approach during the calculation of the diffuse shading factor. Surprisingly, in the second situation, the results of SAM diverged significantly from both PVsyst and the Matlab tool for small distances of separation. One possible reason for this is that SAM uses a separate calculation procedure for self-shading losses [34]. Deviation parameters are shown in Tables 5 and 6 for situations 1 and 2, respectively.
Figure 23. Diffuse shading factor for a module shaded by an infinite row, assuming different distances of separation.

Table 5. MBE for the diffuse shading factor $f_D$. East obstacle.

| East Obstacle | Distance (m) | SAM MBE (%) | SAM MBE (%) | PVsyst MBE (%) | PVsyst MBE (%) |
|---------------|--------------|-------------|-------------|----------------|----------------|
|               | 0            | 0.0061      | 3.94        | 0.0056         | 3.61           |
|               | 0.5          | 0.0097      | 11.50       | 0.0003         | 0.35           |
|               | 1            | 0.0066      | 11.36       | −0.0005        | −0.80          |
|               | 2            | 0.0016      | 4.79        | 0.0006         | 1.80           |
|               | 3            | −0.0010     | −4.45       | 0.0058         | 36.16          |

Table 6. MBE for the diffuse shading factor $f_D$. Infinite row.

| Infinite Row | Distance (m) | SAM MBE (%) | SAM MBE (%) | PVsyst MBE (%) | PVsyst MBE (%) |
|--------------|--------------|-------------|-------------|----------------|----------------|
|              | 0.07         | 0.4034      | 75.35       | 0.0098         | 1.06           |
|              | 0.1          | 0.3986      | 77.88       | 0.0094         | 1.04           |
|              | 0.2          | 0.3748      | 85.17       | 0.0098         | 1.21           |
|              | 0.5          | 0.2691      | 98.70       | 0.0057         | 1.06           |
|              | 1            | 0.1073      | 94.73       | 0.0026         | 1.21           |
|              | 1.25         | 0.0692      | 88.56       | 0.0013         | 0.87           |
|              | 1.5          | 0.0487      | 84.03       | 0.0017         | 1.66           |
|              | 1.75         | 0.0372      | 81.71       | 0.0017         | 2.07           |
|              | 2            | 0.0300      | 80.72       | 0.0012         | 1.86           |

4. Results

An array of PV modules was modeled. The proposed situation was an array with 3 rows and 5 columns, i.e., 15 modules, as shown in Figure 24. This situation also had one wall of 3 m height and situated at a distance of 2 m from the east-side column. Modules
had a tilt angle of 30° and an azimuth of 0°. The location was Mar del Plata (38° S, 57.5° W) and the time was the year 2020. With this method, a shading table for each module of the array could be calculated, as shown in Figure 25. Also, Figures 26 and 27 present the calculated direct shading factor $f_B$ as a function of time and the diffuse shading factor $f_D$ for each of the array’s modules, respectively.

Figure 24. Proposed situation for calculating the shading tables of an array.

Figure 25. Shading tables for each of the modules in the array, represented as heat plots.
5. Conclusions

In this work, a Matlab-based procedure and a tool for estimating the shading factors were proposed. This method provides a means of including the effect of shadows in the direct and diffuse irradiances with Matlab. By setting the geometry of the module and the obstacle, a shading table can be calculated. This provides an easy way for retrieving the direct shading factor $f_B$ and the diffuse shading factor $f_D$ by the integration of the shading table.

The calculated shading factor for the direct irradiance $f_B$ showed a good relationship with the values obtained with the software SAM. However, we observed larger discrep-
ancies when compared with the software PVsyst, mainly due to its hourly time steps. In this matter, we have shown the importance of small time steps when calculating accurate shading factors, and 5-min steps seem to achieve good results. In regard to the diffuse shading factor, the results obtained in Matlab correlated well with the results of PVsyst.

A key advantage of this work is that it provided more flexibility, control, and transparency over the shading factor calculation process. These characteristics make this Matlab tool an attractive option for the performance of research. Moreover, this could be a valuable addition to the PV_LIB library of Sandia Laboratories, which currently has no means for including shading effects. Finally, this tool is not limited to photovoltaic systems; it also may be applied to every situation where an irradiance reduction must be considered.

Any reader interested in obtaining the Toolbox source code can contact the corresponding author.

Author Contributions: Conceptualization, methodology, software and writing—original draft preparation, M.S.; writing—review and editing, J.J.R.; supervision, P.O.P. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Acknowledgments: The authors are thankful to the LEyDE laboratory from the UNMdP who facilitated the experimental data from the PV system to be used in the present research.

Conflicts of Interest: The authors declare no conflict of interest.

Nomenclature
The following nomenclature is used in this manuscript:

- $f_B$: Shading factor for direct irradiance
- $f_D$: Shading factor for diffuse irradiance
- DNI: Direct normal Irradiance
- DHI: Diffuse horizontal Irradiance
- $A_S$: Shaded area of the solar module
- $A_M$: Area of the module
- $v_i$: Vertices of each corner of the module
- $\beta$: Tilt angle of the module
- $\gamma$: Azimuth angle of the module
- $R_y$: Rotation matrix about the y axis
- $R_z$: Rotation matrix about the z axis
- $p_0$: Obstacle vertex
- $p_s$: Projection of the obstacle vertex on the plane of the module
- $a$: Vector normal to the module plane
- $s$: Unit vector with the sun position in Cartesian coordinates
- $I_S$: Irradiance traversing the shaded area AS (diffuse)
- $I_H$: Irradiance reaching the area of the hemisphere AH (diffuse)
- $R$: Radiance of a sky element
- AOI: Angle of incidence
- $\alpha$: Elevation angle of the sun

References
1. Green, M.A. Commercial progress and challenges for photovoltaics. Nat. Energy 2016, 1, 15015. [CrossRef]
2. Eigensonne Das 1000 Dächer Programm. Available online: https://www.eigensonne.de/1000-daecher-programm/ (accessed on 22 October 2020).
3. Strahs, G.; Tombari, C. Laying the Foundation for a Solar America: The Million Solar Roofs Initiative; National Renewable Energy Lab. (NREL): Golden, CO, USA, 2006. [CrossRef]
