energies

MDPI

Article

The Use of Evolutionary Algorithms in the Modelling of Diffuse Radiation in Terms of Simulating the Energy Efficiency of Photovoltaic Systems

Wiktor Olchowik [1], Jędrzej Gajek [2] and Andrzej Michalski [2,*]

[1] Institute of Electronic Systems, Faculty of Electronics, Military University of Technology, 00-908 Warsaw, Poland
[2] Institute of Theory of Electrical Engineering, Measurement and Information Systems, Faculty of Electrical Engineering, Warsaw University of Technology, 00-662 Warsaw, Poland
* Correspondence: andrzej.michalski1@pw.edu.pl

Abstract: In light of the rapidly growing number of photovoltaic micro-grids, the modelling of their short-term power yields based on meteorological measurements is increasing in significance. This requires the knowledge of total and diffuse instantaneous solar radiation; however, most meteorological stations conduct actinometric measurements only with regard to total solar radiation, especially on a minute scale. This paper contains an analysis of the currently used PV cell mathematical model and suggests its modification aimed at calculating PV cell power with satisfactory accuracy, without the knowledge of diffuse solar radiation. Three function families were proposed to approximate the relationship between diffuse irradiance and the total and theoretical total irradiance variance for a cloudless sky. A program has been implemented to identify functions from the aforementioned function families. It leverages an evolution strategy algorithm and a fitness function based on the least-squares point method. It was employed to calculate the desired functions based on actual measurement data. The outcome was the sought-after dependence that enables predicting diffuse irradiance based on more frequently available measurement data.

Keywords: photovoltaic micro-grids; mathematical model; diffuse solar radiation; evolutionary algorithms

Citation: Olchowik, W.; Gajek, J.; Michalski, A. The Use of Evolutionary Algorithms in the Modelling of Diffuse Radiation in Terms of Simulating the Energy Efficiency of Photovoltaic Systems. *Energies* **2023**, *16*, 2744. https://doi.org/10.3390/en16062744

Academic Editor: Carlo Renno

Received: 28 February 2023
Revised: 10 March 2023
Accepted: 13 March 2023
Published: 15 March 2023

1. Introduction

The issue of energy efficiency simulation based on meteorological data is an important one with regards to the operation of photovoltaic (PV) systems, and more broadly, power systems [1,2]. A precise PV cell energy efficiency simulation model will enable, based on actual meteorological data, the accurate determination of the waveforms of generated power as a function of time with relation to PV cells tilted at any angle [3]. This can be used to thoroughly analyse energy yields and support PV system engineering on buildings or noise barriers [4]. Such a model would also allow for short-term predictions [5], by monitoring the energy efficiency of chains in PV micro-grids and their diagnosis in real time [6].

The ground for the conducted studies was the analysis of physical and mathematical aspects associated with the impact of solar radiation. PV cell power is directly proportional to solar irradiance incident on their surface [1,7]. Therefore, it depends on the solar radiation G_{PV} incident on a surface of a PV cell. This, in turn, depends on astronomical, geographical, physical and geometrical factors [8]. Solar radiation passing through the Earth's atmosphere is largely dispersed and absorbed and is subject to other physical effects [9,10]; thus, a complete model would contain many components. At the same time, only direct radiation clearly maintains a vector character among total radiation components [11–14]. Other components give approximately uniform irradiance, regardless of the plane orientation. Therefore, it can be assumed that all components, apart from direct radiation, constitute

generalised diffuse radiation. In such a case, the total radiation model will be expressed by a simplified formula, namely Formula (1) [2,8]. Only radiation components commonly measured by weather stations are included in the following formula:

$$G_S = G_B + G_R \quad (1)$$

where G_S—total solar irradiance [W/m^2], G_B—direct solar irradiance [W/m^2], and G_R—general diffuse solar irradiance [W/m^2].

Such a radiation model is often applied because meteorological stations with more than one radiation sensor usually measure total and diffuse radiation. An example of total and diffuse radiation waveforms measured at the Institute of Meteorology and Water Management (IMWG) station in Warsaw [15] is shown in Figure 1.

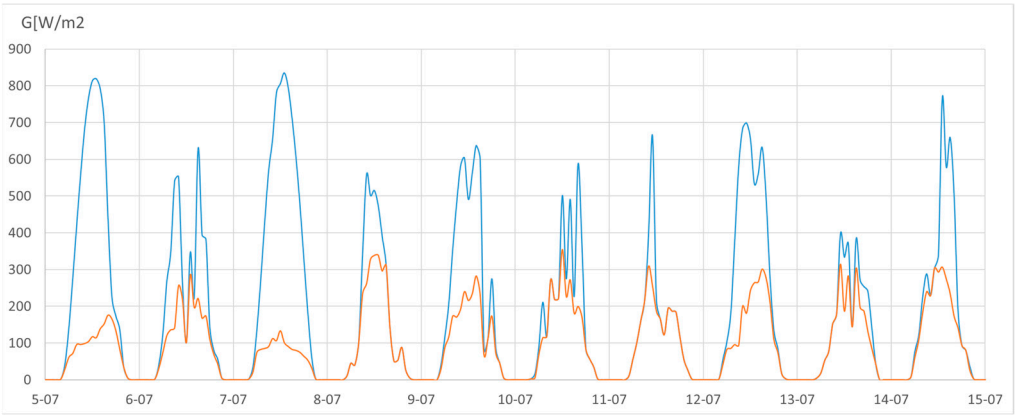

Figure 1. Examples of total (blue) and diffuse (orange) irradiance waveforms for Warsaw.

If we have measurement results for total radiation for a plane parallel to the Earth's surface G_S and diffuse radiation G_R, it is possible to calculate direct radiation on the plane parallel to the Earth's surface G_B (2).

$$G_B = G_S - G_R \quad (2)$$

Next, one can calculate direct irradiance on the plane inclined at any angle β and rotated southward at an angle γ using Formula (3) [1,10], as shown in Figure 2.

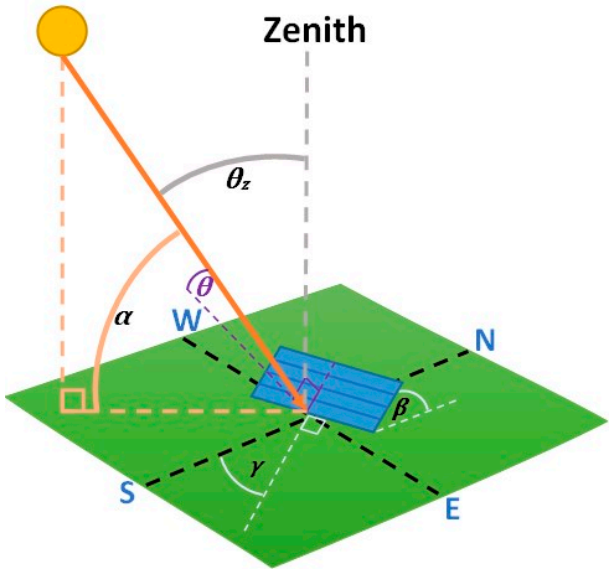

Figure 2. Illustrated relationship between direct solar radiation angle of incidence on a plane inclined at an angle β and rotated within the azimuth by the angle γ.

$$G_{BP} = G_B \cdot \frac{\cos\theta}{\sin\alpha} \tag{3}$$

where G_{BP}—direction irradiance on an inclined and rotated plane [W/m^2], θ—direct solar radiation angle of incidence on a surface with any inclination relative to horizontal and any azimuth [°], and α—Sun's angle of elevation [°].

This enables calculating total irradiance G_{SP} on a plane inclined at an angle β and rotated by an angle γ as per Formula (4).

$$G_{SP} = G_{BP} + G_R \tag{4}$$

By using the Formulas (2)–(4), we obtain Formula (5):

$$G_{SP} = (G_S - G_R) \cdot \frac{\cos\theta}{\sin\alpha} + G_R \tag{5}$$

where: G_{SP}—total irradiance on a plane inclined at an angle β and rotated by an angle γ [W/m^2],

Moreover, sin α is defined by Formula (6) [16,17]:

$$\sin\alpha = (\cos\varphi\cos\delta\cos\omega + \sin\varphi\sin\delta) \tag{6}$$

where φ—latitude [°], δ—solar declination [°], and ω—Sun's hour angle [°].

Whereas cos θ is defined by Formula (7) [3]:

$$\begin{gathered}\cos\theta = \sin\delta\cdot(\sin\varphi\cos\beta - \cos\varphi\sin\beta\cos\gamma) + \cos\delta \\ (\cos\varphi\cos\beta\cos\omega + \sin\beta\sin\varphi\cos\gamma\cos\omega + \sin\beta\sin\gamma\sin\omega)\end{gathered} \tag{7}$$

where β—plane inclination angle (angle between the horizon and receiver) [°], and γ—azimuth angle (angle between the receiver and southward direction) [°].

The formulas make it possible to calculate irradiance on a PV cell plane G_{PV} inclined at an angle β and rotated relative southward by an angle γ, hence simulating instantaneous power of photovoltaic cells for actual meteorological conditions. The presented model is complete if we have total solar irradiance on a plane parallel to the Earth's surface and diffuse irradiance measurement results. However, there are very few meteorological stations that provide diffuse radiation measurement results [15,18]. Therefore, a potential alternative can be calculating its values based on the total radiation on a horizontal plane and other available data.

In this context, the authors analysed the impact of possible diffuse radiation calculation errors on the simulation results of the total radiation incident on the surface of the cell inclined at an angle of 45°. Calculations were performed for sample daily runs of total and scattered radiation values obtained from measurements and then compared with the values of scattered radiation increased or decreased by 1%. The simulation results are presented in Table 1.

Table 1. Mean calculation errors for total radiation depending on mean diffuse radiation determination errors.

Cloud Cover	Diffuse Radiation G_r [W/m^2]		G_{SP} [W/m^2]	Absolute Error	Relative Error
Moderate G_S = 135.69 [W/m^2]	G_R +1%	65.86	282.15	1.64	0.58%
	G_R	65.20	283.79	-	-
	G_R −1%	64.55	285.43	1.64	0.58%
High G_S = 65.92 [W/m^2]	G_R +1%	63.17	72.15	1.35	1.84%
	G_R	62.55	73.50	-	-
	G_R −1%	61.92	74.85	1.35	1.84%

The data in Table 1 indicate that a 1% diffuse radiation calculation error can result in a total radiation calculation error of up to 2%.

The influence of the scatter radiation calculation error on the calculation errors of the total radiation incident on the PV plane can also be observed in the waveforms. Figures 3 and 4 show the waveforms of the calculated values of total radiation on the PV plane inclined at an angle of 45°: G_{SP}—calculated for the values of total radiation to the Earth's surface and diffuse radiation measured by the meteorological station, and G_{SPX}—for diffuse radiation reduced by 20%. The G_{SPX} waveform illustrates the case of a diffuse radiation calculation error of −20%.

Figure 3. G_{SP} and G_{SPX} waveforms for heavy clouds.

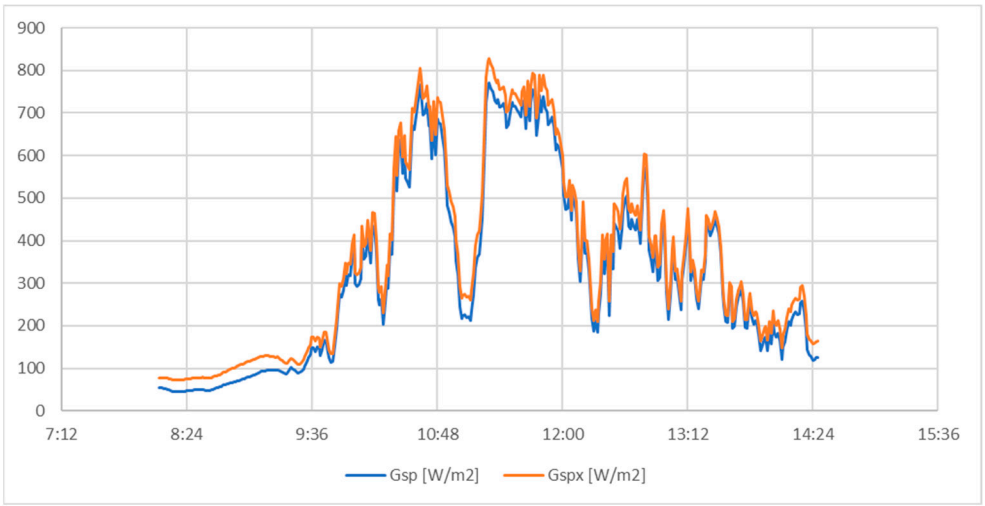

Figure 4. G_{SP} and G_{SPX} waveform for moderate clouds.

Based on both the table and waveforms, it can be seen that the diffuse radiation calculation errors have a significant impact on the simulated total irradiance on a cell plane, and, hence, its yields. Therefore, it is advisable to mitigate this error by developing a precise mathematical model to calculate diffuse and direct radiation [19,20].

Next, the authors analysed the global reference literature in terms of existing and described mathematical models for calculating diffuse radiation. There are numerous empirical dependencies employed to calculate diffuse radiation based on meteorological and environmental data, e.g., models: Perez [21], Liu, Jordan, Klucher, Hay, Skartveit, Olseth, Temps, Coulson and other [22–24]. These dependencies have been generally developed based on data for specific, local conditions [25–27], by using long-term measurements [28].

No general formula for calculating diffuse solar radiation at any location in the world has been developed so far [29]. In addition, existing studies in which the authors compare various diffuse radiation models on a global scale are usually based on average values within an at least 24 h period [30]. In rare cases, studies focused on hourly [31] or minute [32] resolution are present, but they do not describe a precise mathematical relationship.

There are also a number of applications used to analyse the energy efficiency of PV cells. The following can be distinguished, among others: SYSTEM ADVISOR MODEL (SAM) [33], RETscreen [34,35], Solargis Prospect [36–38], PVsyst [39], and Solar Pro [40]. Their descriptions indicated that they enable simulating the energy and economic efficiency for most systems currently on the market. A more thorough analysis indicates that, due to the scarcity of data that take diffuse radiation into account, diffuse radiation is determined based on total radiation and other meteorological or astronomical data. However, the employed mathematical models are usually accurate only for the meteorological data measured with a low temporal resolution. Therefore, it is advisable to develop a proprietary model of scattered radiation, enabling the analysis of dynamic waveforms.

For this purpose, in Section 2, on very large sets of meteorological data (from about 4 years), the relationships between diffuse and total radiation were analysed, and by using data aggregation methods, the approximate nature of the function was initially determined, and the theoretical total radiation with a cloudless sky was adopted as a parameter to refine the results. Then, in Section 3, three types of functions of two variables were defined, and the method of evolutionary algorithms to calculate the detailed parameters of these functions was presented. Section 4 presents the implementation of the author's application, which was created in a low-level programming language, for the effective search for optimal function parameters. Due to the complexity of the functions and the need to use large data sets, it was a demanding programming challenge for the authors. Section 5 presents proprietary functions for calculating solar radiation intensity on the plane of PV cells. Section 6 contains conclusions showing that the developed functions improve the quality of the simulation model of the operation of PV cells, especially for high measurement frequencies (minute resolution). The conclusions are supplemented by Appendix A with examples of use.

The comprehensive aim of the research was to develop a simulation model of the dynamic operation of PV systems based only on basic meteorological parameters (but with high time resolution) that were obtained by simple weather stations. It was assumed that diffuse radiation would not be measured in such meteorological stations. The direct aim of the research described in the article was, therefore, to determine the function that calculates the diffuse radiation on the basis of the total radiation to the Earth's surface with the highest possible accuracy. The specified function will be one of the elements of the simulation model.

2. Diffuse Radiation Modelling Assumptions

Determining the relationship between diffuse and total radiation based on measurements at 1 min intervals is not easy. This is evidenced by a scatter plot based on diffuse and total radiation, which was taken over four years by an IMGW meteorological station in Zakopane [15], as shown in Figure 5.

The plane distribution for approximately one million points does not enable even a preliminary trend determination. The plot analysis indicates that a uni-parameter function that determines the dependence of diffuse radiation on total radiation would be very inaccurate [41]. Developing a function that would precisely map this relationship requires searching also for dependence on other parameters associated with the Sun's position.

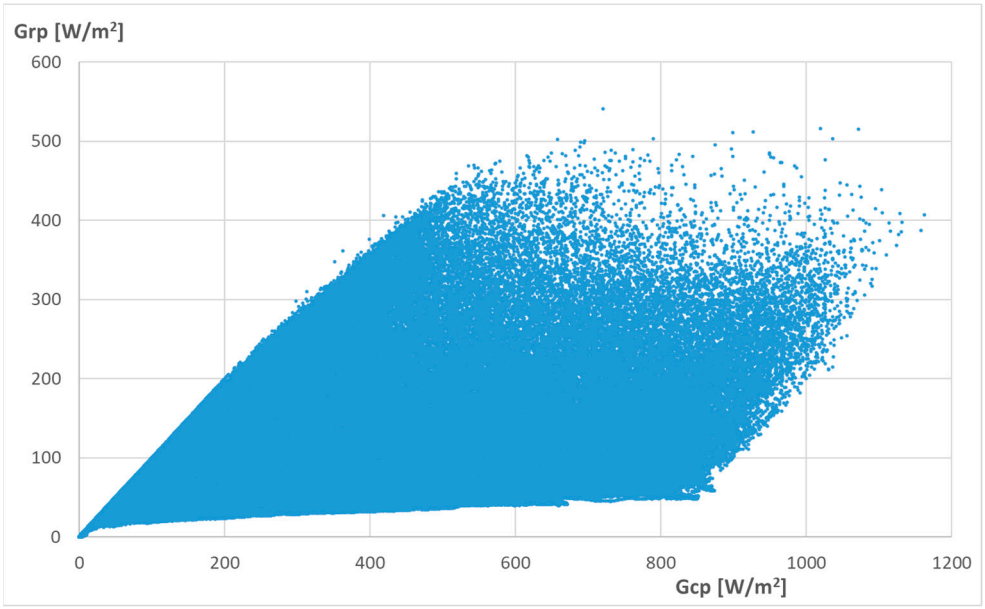

Figure 5. Scatter plot based on diffuse (G_{RP}) and total (G_{CP}) radiation measurements taken by an IMGW station in Zakopane at a 1 min resolution.

2.1. Selection of Diffuse Radiation Modelling Function Parameters

Diffuse radiation modelling function parameters were selected based on available low-resolution diffuse solar radiation mathematical models. An analysis of parameter sets that were measured by meteorological stations enables a conclusion that if it were assumed that diffuse radiation were not measured, the only useful parameter would generally be total irradiance. However, theoretical values calculated for the Sun at a given location can be additionally taken into account. The Sun's elevation angle and total irradiance for a cloudless sky are the most promising in this respect. Based on preliminary studies and analyses, it was assumed that theoretical total irradiance for a cloudless sky was a more prospective, second parameter.

2.2. Cloudless Sky Solar Irradiance Theoretical Model

Determining a cloudless sky solar irradiance theoretical model requires analysing the solar radiation path from the source to a given location on the Earth's surface. Radiation power density decreases with the distance from the Sun. Because the Earth follows an ellipsoidal orbit, the distance from the Sun changes over the course of a year. The lowest distance is recorded between the 2nd and 4th of January, and the greatest is recorded between the 3rd and 6th of July. In light of the above, the intensity of solar radiation reaching the upper layers of the Earth's atmosphere is determined using Formula (8) [2,16].

$$G_{ON} = G_{SC} \cdot \left(1 + 0.033 \cdot \cos \frac{360^\circ \cdot d}{365}\right) \tag{8}$$

where G_{ON}—energy reaching the Earth's atmosphere [W/m^2], d—subsequent day of the year, d for 1 January takes the value 1, and G_{SC}—solar constant equal to 1367 W/m^2.

The radiation reaching the upper atmospheric layers is of an entirely vector nature (directed energy), and thus, irradiance depends on its incidence angle. In the case of the Earth, which is a sphere, the incidence angle is strongly correlated to the latitude φ. Furthermore, due to the Earth's rotating around an axis inclined to the ecliptic (plane of the Sun's orbit around the Earth) at an angle of 66°34′, this angle changes throughout the year, according to the declination Formula (9) [4,8,16].

$$\delta = 23.45^\circ \cdot \sin \frac{360^\circ \cdot (284 + d)}{365} \tag{9}$$

At the same time, the angle of the Sun's elevation above the horizon α is expressed by the dependence (10) [8].

$$\alpha_{max} = 90^\circ - \varphi + \delta \tag{10}$$

where α_{max}—maximum angle of incidence of solar rays on the Earth's plane on a given day.

The angle α_{max} is largest during the day; it represents the highest irradiance value. Due to the rotation of the Earth around its axis, this angle is modulated by the day phase angle ω (or the hour angle), which is determined in degrees as per Formula (11) [8].

$$\omega = 15^\circ \cdot (t - 12) \tag{11}$$

where t—successive hour of the day according to solar time, assuming that $t = 0$ is midnight.

Given the factors above, the intensity of the radiation reaching the Earth's atmosphere at any time G_O can be calculated from Formula (12) [2]:

$$G_O = G_{ON} \cdot \cos\theta_Z = G_{ON} \cdot (\cos\varphi \cos\delta \cos\omega + \sin\varphi \sin\delta) \tag{12}$$

where G_O—solar irradiance on a plane parallel to the Earth's surface at latitude φ [W/m^2].

After passing the atmosphere, solar radiation reaching the atmosphere and determined via Formula (12) is partially attenuated and dispersed, and its components can be presented as a sum of direct and diffuse radiation. Earth's surface theoretical irradiance values for a clear (cloudless) sky can be calculated based on Formulas (13) and (14) [5].

$$G_{BT} = G_O \cdot \tau_B \tag{13}$$

$$G_{RT} = G_O \cdot \tau_R \tag{14}$$

where G_{BT}—direct irradiance of an inclined and rotated plane [W/m^2], G_{RT}—diffuse irradiance [W/m^2], τ_B—coefficient of direct radiation transmission through the Earth's atmosphere, and τ_R—diffuse radiation coefficient.

Through simple transformations, we obtain a dependence (15) for total radiation G_{ST} on the Earth's surface for a clear sky.

$$G_{ST} = G_O \cdot (\tau_R + \tau_B) \tag{15}$$

The τ_B coefficient is expressed by the empirical Formula (16) [6]:

$$\tau_B = a_0 + a_1 e^{-\frac{k}{\cos\theta_z}} \tag{16}$$

where a_0, a_1, k—empirically obtained coefficients.

The values of the a_0, a_1, k coefficients depend on the climatic zone and topography, and they fluctuate to a certain extent, depending on weather conditions.

The τ_R coefficient can be calculated from the empirical Formula (17) [6].

$$\tau_R = 0.271 - 0.294 \cdot \tau_B \tag{17}$$

The formulas above enable calculating theoretical irradiance at any location on the Earth and at any phase of the day and year. This value will be calculated for each measurement and assigned as a third dimension. It will be ultimately employed as a parameter in the sought-after formula for calculating diffuse radiation.

2.3. Experimental Trend Function Search

An analysis of Figure 5 indicates that the direction of searching for the dependence-enabling calculation of diffuse radiation is difficult to determine. One of the reasons for this is a very high number of measurement points. Therefore, the authors aggregated the total calculated and measured irradiance into intervals. Because both values represent irradiance and fall within comparable ranges, data were aggregated using identical intervals. The data

were aggregated for several interval ranges from 10 to 100 W/m^2. Aggregation intervals that were 50 W/m^2 wide were adopted. It is a compromise between mapping accuracy and data representation clarity.

The first aggregation stage involved grouping all measurement points according to theoretical irradiance intervals. For the 0 to 900 W/m^2 range, 18 intervals were defined, and hence, the entire measurement data set was divided into 18 subsets to be further considered independently. During the second stage, data within each of the 18 subsets were grouped by total measured irradiance intervals. For the 0 to 1200 W/m^2 range, 24 intervals were defined, which were then assigned diffuse radiation values falling within a given subset. The outcome was an 18 × 24 matrix of two-dimensional intervals, which were then assigned measurement points. Each of the two-dimensional elements included from several dozen to several dozen thousand measured diffuse radiation values. The third stage involved averaging diffuse radiation within each of the two-dimensional intervals. Aggregation enabled reducing the number of points from approximately a million to less than 400. A family of 18 diffuse radiation graphs based on these points is presented as a function of total measured radiation in Figure 6. Aggregation intervals medians are the parameters of these waveforms.

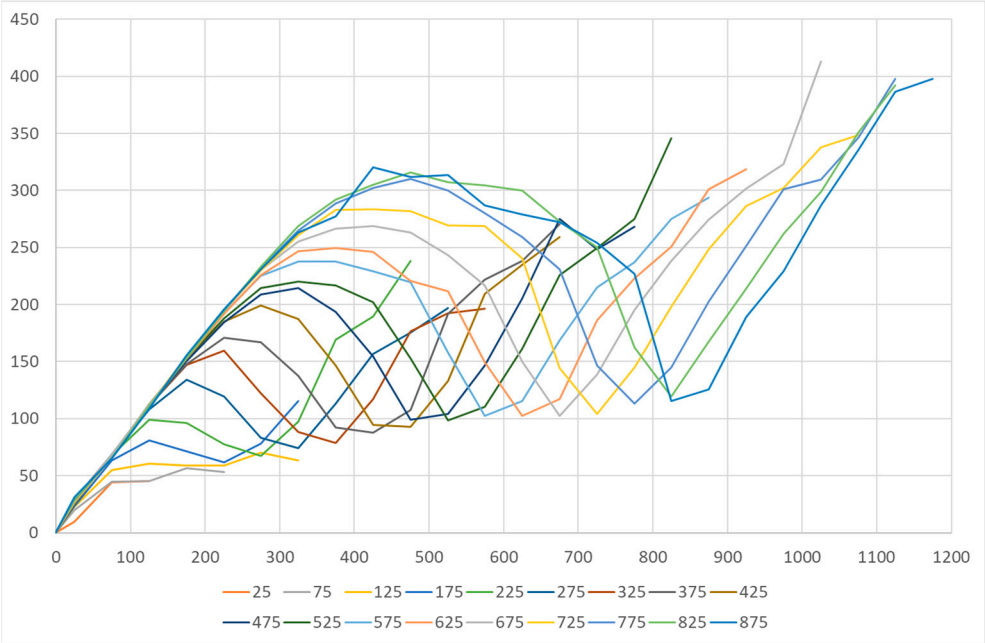

Figure 6. Family of diffuse radiation graphs as a function of total measured radiation.

Data aggregation enabled emphasising existing relationships between diffuse and total radiation. The analysis procedure should also take into account the ratio of total measured radiation to theoretical radiation for a cloudless sky, which can be treated as an atmospheric clearance measure. Certain waveform regularities that arise from the aforementioned impact of the Earth's atmosphere on solar radiation can be noticed in such a case. In the case of low atmospheric clearance (heavy clouds), almost 100% of the radiation is of diffuse character. When atmospheric clearance increases, diffuse radiation in total solar radiation decreases, and the graphs tend to their local minimum for the measured radiation value, approximately equal to the one calculated for a cloudless sky. However, the Central European climate quite often experiences measured instantaneous total radiation values exceeding calculated values by even 20%. This happens because the impact of direct radiation under a partial cloud cover at a given point is the same as for a cloudless sky, and in addition, there is also radiation diffused by neighbouring clouds. This leads to a situation in which diffuse radiation values grow almost directly proportionally with regard to total measured radiation values exceeding theoretical radiation, as calculated for

a cloudless sky. This area of the graphs is the subject of interest as part of this article. Total measured radiation being higher than theoretical radiation for a cloudless sky is practically associated with frequent and dynamic radiation changes. Taking this into account for the development of the mathematical model for diffuse radiation, and then the simulation model for PV cell operation, will enable, among other results, their better diagnostics.

It should be added that, in the case of modelling diffuse radiation based on hourly or longer measurement periods, the aforementioned dynamic phenomena are virtually unnoticeable, and thus, the model described by the sourced literature cannot be applied. This was the main driving factor to determine a proprietary diffuse radiation model which is based on data measured with a minute resolution and taking dynamic changes into account.

Based on the graphs shown in Figure 6 derived from a heuristic analysis, it can be concluded that the function describing the dependence of diffuse radiation on total radiation should be similar to an odd-order polynomial or a modified trigonometric function. In order to enable applying a single function, it should be additionally parametrised relative to the theoretical total radiation for a cloudless sky.

3. Methodology of Determining a Diffuse Radiation Calculation Function

Therefore, this article brings forward a hypothesis that diffuse radiation can be presented with satisfactory accuracy using a bivariate function with appropriately selected coefficients that model the impact of weather phenomena and structure on diffuse radiation content in total radiation. Such a function would take the form of (18).

$$G_{RP} = f(G_{SP}, G_{ST}) \tag{18}$$

where G_{ST}—theoretical total solar irradiance [W/m^2], G_{SP}—measured total solar irradiance [W/m^2], and G_{RP}—calculated total solar irradiance [W/m^2].

3.1. Quality Assessment and Searched Function Optimization Algorithms and Methods

Assuming that measurement data represent a given function but are biased with an error, this issue can be reduced to the problem of surface approximation in a 3D space. The least squares method is commonly used to determine surface approximation match quality [42]. It involves minimising the Sum of Squared Errors (SSE), i.e., the difference between the measured value and the value adopted for given coordinates by an idealised function representing a given physical phenomenon [42]. In the case of the analysed issue, the error can be calculated as per Formula (19), while the SSE can be calculated as per Formula (20).

$$r_i = G_{Ri} - f(G_{Si}, G_{STi}) \tag{19}$$

$$SSE = \sum_{i=1}^{N} {r_i}^2 \tag{20}$$

where r_i—error for the i measurement [W/m^2], N—number of measurements, SSE—error square sum for all measurements [W^2/m^4], G_{STi}—theoretical total solar irradiance for the i measurement [W/m^2], G_{Si}—measured solar irradiance for the i measurement [W/m^2], G_{Ri}—measured general solar diffuse irradiance for the i measurement [W/m^2].

In order to better assess the fitness of a found function relative to measurement data and its physically achievable variables, one can use two linked SSE-based coefficients. The first one is the Mean Squared Error (MSE), which as an SSE mean, renders the assessment of function match quality independent from the number of applied measurements [43]. It enables, e.g., comparing the quality of two different matches found for a different number of samples. The second coefficient is the root mean squared error (RMSE), which can be calculated through MSE root extraction [43]. It can be used as a base to assess a function quality match directly by comparing its values with physically achievable values. A RMSE many times greater than the maximum achievable value suggests that the match is still

biased with a significant error. The method for calculating these coefficients is presented by Formulas (21) and (22).

$$MSE = \frac{SSE}{N} \tag{21}$$

$$RMSE = \sqrt[2]{MSE} \tag{22}$$

where: *MSE*—Mean Squared Error [W^2/m^4], *RMSE*—Root Mean Square Error [W/m^2], *N*—number of measurements.

3.2. Function Determination Method Analysis

There are numerous ways to find a minimum identified through the least squares method. For simple cases, such as a linear or square function matching in $\mathbb{R}^2$, an optimal solution can be found analytically by solving a set of differential equations.

In more complex cases, however, it is possible to find the minimum via numerical calculations. One can apply the Newton–Gauss, the gradient descent method or the more complex Levenberg–Marquardt method [44], which is a combination of the Newton–Gauss and the gradient descent methods. These methods iteratively reach the goal; however, they are susceptible to a very long execution time in the case of iterations with a step that is too large or to small and to stopping at the local minimum for non-linear cases. One way to prevent limiting the searched space to a local minimum is selecting initial parameters of the matched function so that the values are as close to the global minimum as possible.

Due to lack of data that would allow for the identification of initial parameter values, a way to avoid remaining at the local minimum is to change the family of optimal match search methods from numerical to metaheuristic [45]. This family of algorithms is abundant in colourful comparisons. Metaheuristics include particle swarm optimization, taboo search, simulated annealing and an entire class of algorithms based on evolution mechanisms [46].

Metaheuristic algorithms are methods that enable constructing heuristics that enable determining a sufficiently good solution to any problem within an acceptable period of time, provided that the problem is described with notions of given metaheuristics [47]. Each of the algorithms defines a certain method for searching for a problem solution space, and their objective is to search for solutions effectively. This effectiveness should be understood in the concept of both the exploration, which is searching for the largest solution space not to omit a global optimum and operation, which is the effective finding of the best solution for a given local optimum area, which can turn out to be global.

Metaheuristic algorithms are classified according to several parameters [48]. The most important one in the context of this study is the search locality parameter. Local metaheuristic algorithms, similar to numerical methods for a non-linear case, are unable to reach a global optimum if initially set parameters deviate too much from the optimum [49]. For this reason, most metaheuristic algorithms are global algorithms, which have been fitted with a mechanism ensuring that an algorithm can reach any solution within the search space. An example of a local metaheuristic algorithm is the hill-climbing algorithm [50], which operates similarly to the numerical gradient descent method; instead of, however, calculating derivatives, the algorithm browses points adjacent to the current analysed one and selects the best one. Only in this way can the algorithm reach the optimum closest to the starting point in a solution space. This algorithm becomes global by introducing restarts for other, non-random initial parameters after reaching the optimum for the previous execution.

Another criterion is the use of memory by a given metaheuristic algorithm [49]. Some algorithms, such as the hill-climbing and simulated annealing algorithms, are classified as memoryless algorithms since they do not aggregate information on solutions obtained during previous algorithm iterations. The use of a memoryless algorithm can adversely impact its behaviour when encountering a set of solutions forming a cycle. Such an algorithm will then be stuck in this cycle due to the absence of notifications that it is passing through the same search space points. Algorithms with memory, including the taboo search algorithm, ant colony algorithms [51] and evolution algorithms [52], remember a

specific-size set of solutions achieved during previous iterations and use it as a base to modify their operation, avoiding becoming stuck in the cycle.

The last crucial criterion is the criterion of division into path and population algorithms [49]. Path algorithms are so called because they process only one solution, and based on the solution, they generate a further single solution until reaching a stop condition, which is the exhaustion of resources or reaching a satisfactory optimum. In this way, they determine a certain path from the starting point to the final point, which is closer to the optimum. The way they traverse the solution space is similar in behaviour to the aforementioned numerical methods; however, due to the stochastic approach, it is necessary to define adjacency. This means that for every solution from within the space, there must be the possibility of determining adjacent solutions, which the algorithm will try to achieve during the next iteration. These algorithms include taboo search or the hill-climbing ones. In the case of population algorithms, each step involves analysing a set of solutions, which is called a population, which contains (not necessarily interdependent) solutions called individuals. This class of algorithms includes, among others, ant colony, evolution and simulated annealing algorithms [51]. For the power of an analysed set of solutions equal to one, population algorithms can be reduced to the path algorithm. Similarly, population algorithms can be inferred as a generalization of the path algorithms that enables search parallelization for many different solutions. A Venn diagram illustrating a metaheuristic algorithm classification is shown in Figure 7.

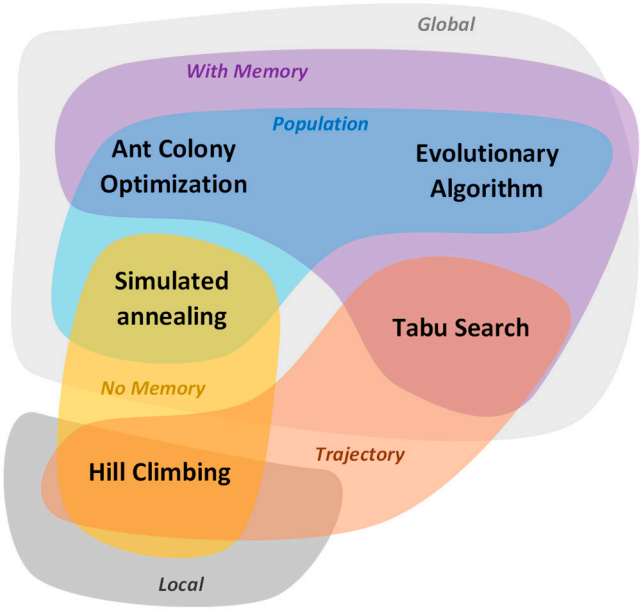

Figure 7. Venn diagram illustrating a metaheuristic algorithm classification.

Evolution algorithms are inspired by the biological behaviour of evolution and apply terminology derived from this field [52]. An analysed search space solution is called an individual, and their set is called a population. Each individual is represented through genes, which are its characterising parameters that can be logical values, numbers and also more complex structures. The quality of a given individual, based on the objective function value calculated for its genes, reflects its adaptation to its environments. The environment shall be understood as the specificity of the problem being solved. Individuals iteratively undergo natural selection as part of the algorithm. Better matched ones are able to survive and have similar, though not identical, offspring, while individuals with worse objective function values are removed from the population. Hence, each new iteration corresponds to a new population, called a generation, which contains individuals generated based on the top individuals from the previous generation.

As the algorithm progresses, individuals with increasingly better parameters are generated.

The evolution strategy algorithm is the one among evolution algorithms that deserves particular attention. It enables an iterative search of the solution space in a manner that promotes approaching an optimal solution, similarly to the Newton–Gauss method [53]. Unlike the Newton–Gauss method, owing to the diverse solution population and searching that are not based on the local function gradient, the algorithm is able to reach any optimum, including a global optimum, regardless of the initial parameters [54]. In addition, the convergence rate of this algorithm can be adjusted based on the diversity level of the entire population, and not based on a single solution [55]. An engineering advantage is the ability to improve the evolution strategy algorithm result quality by increasing the population size and employing parallel calculations at a single iteration level [56]. The possibility of parallelising calculations in a single operation is limited for numerical methods, and their result quality cannot be improved without increasing the number of iterations.

3.3. Selected Algorithm

The selected algorithm is the ES (*Evolution Strategy*) designated as $(\mu + \lambda)$, where μ means the number of parents, and λ is the number of children. This algorithm starts its operation with a randomly generated population of λ individuals, most probably being sub-optimal problem solutions. Next, the algorithm iteratively searches the solution space as follows. By using the objective function, each individual within the population is evaluated in terms of its optimization. In the event of matching a surface to a point cloud, individuals with a lower *RMSE* are considered closer to the optimum. Based on such a qualitative evaluation of the individuals constituting a population, the algorithm records the best solution found so far and removes all individuals from the population, except for the μ best ones. Based on each of the best μ individuals, the λ/μ copies are created, each of which are subject to mutation. This operation changes certain individual genes in a random manner with a pre-set probability distribution. Thus, new created λ individuals join the population [50,53,55]. This population building method involves ES algorithm memory power, since potentially better individuals from the previous iteration are not forgotten until μ number of better children are found. Successive iterations are run until an optimal solution is found or until the iteration limit is reached. Determining whether the global optimum has actually been reached is difficult for many problems. These include the problem of matching a surface to a point cloud. In such a case, this optimum can be defined as an objective function reaching a minimum equal to 0 or an accordingly low ε value, although it will be unattainable for many cases.

The distribution based on which individuals within the initial population of the ES algorithm are generated is not as important as in certain numerical methods, provided the mutation operation is correctly constructed. According to the assumptions, it must enable a global search of the solution space. A mutation in evolution strategies is usually based on normal distribution with the expected value equal to zero and σ^2 variance matched to the domain of numbers operated by the algorithm [55]. Such a distribution enables achieving, with a certain non-zero probability, any point within the search space, as the number of iterations increases. High variance values enable rapid transition to distant points; however, they also mean that it will be challenging to generate similar and optimum-approaching children for already-found near-optimum individuals. In turn, in the event of low variance values, reaching a local optimum for the offspring of a given individual will be faster; however, this process occurs at the cost of an often longer discovery of other optima.

Selecting the variances for the distribution employed within the mutation can be simplified through an adaptive mutation rate. One of the methods for implementing this idea for the evolution strategy algorithm is the so-called Ingo Rechenberg one-fifth rule [53,55], which assumes that the number of mutations leading to an individual being better than its parent should be $1/5$ of all mutations. The application of this method can be described as follows.

1. If more than 20% children are better than their parents, the algorithm is characterized by excessive local optima exploitation, so the variance should be increased.
2. If less than 20% children are better than their parents, the algorithm excessively explores the search space, so the variance should be decreased.
3. If exactly 20% children are better than their parents, equilibrium has been reached, and there is no need to change the variance.

Owing to such an approach, in the case of reaching a local optimum, the variance will grow until any of the newly generated individuals does not encounter an optimum better than the one previously found. Next, according to this rule, the variance is adaptively reduced as part of the growing exploitation of new optimum boundaries.

Increasing and decreasing the variance σ^2 of the normal distribution used within the mutation process can be achieved in many ways. However, the most popular and widely adopted in the sourced literature is the appropriate multiplication or division of the standard deviation σ by a minor constant with a value similar to unity, depending on which condition was met by the current population. For small oscillations of approximately 20%, σ changes in successive iterations will be negligible. In the event of reaching a local optimum, a need to increase the exploration and a drastic increase in variances can already be observed after several dozen iterations, because the exponential rises, and thus, the character of the σ variance multiplication of every iteration rises by a value slightly greater than 1. The method of calculating standard deviation σ is shown by Formula (23).

$$\sigma_{i+1} = \begin{cases} \sigma_i \cdot a, \ dla\ p < \frac{1}{5} \\ \sigma_i \cdot \frac{1}{a}, \ dla\ p > \frac{1}{5} \\ \sigma_i, \ dla\ p = \frac{1}{5} \end{cases}, \ gdzie\ 0,\ 8 \leq a \leq 1 \tag{23}$$

where σ_i—standard deviation value in i iteration, σ_{i+1}—standard deviation value in the $i + 1st$ iteration, a—multiplier value, and p—ratio of children better than their parents to all children in the i iteration.

3.4. Problem Domain Representation

Solving a specific problem using the evolution strategy algorithm requires describing its domain in the perspective of the algorithm. This requires defining an individual and how its quality can be evaluated. For the case of matching bivariate functions to a point cloud in a space $\mathbb{R}^3$, each individual should represent a function formula through its genes. A formula of such a function can be expressed through defining an individual as a vector of real numbers, each of which successively represents a specific parameter of a given function and also constitutes a single gene that can be impacted by mutation. For example, an individual denoting a function from a family function $f(x, y) = ax + by + c$ is represented by a vector of three numbers $[a\ b\ c]$.

Such a representation means that an individual must additionally have the form designation of the function it is to represent, since it would be impossible to calculate its quality otherwise. However, this designation should not be treated as another gene; an algorithm that considers in a given implementation only the individuals denoting functions of the same form will find a potentially satisfactory solution much faster than if it attempted to construct functions of various forms. The function value for a given individual can be calculated by subsequently substituting each of the matched points to the formula, finding the value of the r_i error, and then summing these errors for all points. SSE, MSE and RMSE can all be utilised to compare the quality of individuals; however, as previously established, RMSE is the best parameter owing to the ability to draw further conclusions in terms of the best individual found by the algorithm after completing its operation.

Due to the fact that the accurate relationship between measured and actual radiation is unknown, the analysis should involve several functions that can well reflect the actual dependence between the measured total, theoretical total and measured diffuse radiation.

All analysed functions will be bivariate functions. The selection of these functions will be based on a heuristic approach [57].

The first of the considered functions is a third-order polynomial for two variables of the form shown in Formula (26). Polynomial functions are often employed to approximate physical phenomena. If the actual dependence is n + 1 times differentiable, it can be approximated based on the Taylor theorem as an *n*-order polynomial, which is then a partial sum of the Taylor series. If the searched function can be approximated by a lower-order polynomial, it does not have to be considered separately, since these polynomials are contained within the selected one.

$$f(x,y) = p_{00} + p_{10}x + p_{01}y + p_{20}x^2 + p_{11}xy + p_{02}y^2 + p_{30}x^3 + p_{21}x^2y + p_{12}xy^2 + p_{03}y^3 \quad (24)$$

where x, y—variables adopted by the function; the variable sequence is unimportant.

Better results can be provided by a fifth-order polynomial, which is a longer partial sum of the Taylor series. This function's formula is presented in (25). However, its application is biased with considerably longer calculation times. One third-order polynomial evaluation is 20 multiplication and 9 summation operations. In the case of a fifth-order polynomial, a single calculation of the value requires 70 multiplications and 20 additions, which will extend the time over threefold. A fourth-order polynomial can also be considered; however, if it exists, it will be contained in a fifth-order polynomial. Therefore, it was not subject to analysis as part of this paper.

$$\begin{aligned} f(x,y) = p_{00} + \; & p_{10}x + p_{01}y + p_{20}x^2 + p_{11}xy + p_{02}y^2 + p_{30}x^3 + p_{21}x^2y + p_{12}xy^2 \\ & + p_{03}y^3 + p_{40}x^4 + p_{31}x^3y + p_{22}x^2y^2 + p_{13}xy^3 + p_{04}y^4 + p_{50}x^5 \\ & + p_{41}x^4y + p_{32}x^3y^2 + p_{23}x^2y^3 + p_{14}xy^4 + p_{05}y^5 \end{aligned} \quad (25)$$

Furthermore, based on the waveform shape observations for a point cloud transformed in the data aggregation process, it can be concluded that the functions based on cosine G_{CP}, the period and offset of which are impacted by G_{CT}, is also worth considering. Due to the frequent occurrence of trigonometric functions in calculations associated with breaking down solar irradiance into prime factors, such a heuristic function can be expressed by Formula (26).

$$f(x,y) = a + b \cdot \cos(c \cdot x + d) \cdot \cos(e \cdot y + f) + g \cdot y + h \cdot x \quad (26)$$

where x—G_{CP}, measured total irradiance [W/m^2], and y—G_{CT}, theoretical total irradiance [W/m^2].

The next step required to match and determine the quality of thus-defined functions was designing and implementing a software program.

4. Application Design and Implementation

The authors created an application that implements an evolution algorithm as per the assumptions set out in Section 3. The application was developed in Go, created to offer a programming language with the speed of C or C++ [58] and code readability similar to that of Python and JavaScript [59], and it is one that supports concurrency [60]. Its ease of use, operating and compilation speed, and the fact that it provides multithreading that enables concurrent calculations as part of basic language mechanisms were important factors that lead to this choice. Additionally, it was possible to run the Go language compiler on most modern operating systems, both GNU Linux and Microsoft Windows, owing to the interoperability of this language.

The Go language does not offer ready-made modules that facilitate employing the evolution strategy algorithm. As a result, an application has been implemented for this exact purpose. Standard modules [61], as the Go nomenclature refers to programming libraries, were used when the application was developed.

4.1. User Interface

Due to the batch operating nature of the application, the user interface is limited to setting program parameters and input data when it is executed from the command line and to previewing its results in a result file or command line, depending on how the program is run. The program adopts seven parameters that are represented in Table 2.

Table 2. Parameters adopted by the program.

Parameter Name	Parameter Type	Default Value
mi	int, integer	1
lambda	int, integer	8
iterations	int, integer	100,001
iterationStep	int, integer	10,000
mutationSigma	float64, floating point number	0.0005
functionName	string, character string	Poly3 × 3
filePath	string, character string	test.txt

These parameters take the following roles:

- mi—means the number of parents, i.e., individuals to be chosen from each generation as the best and used to create a next generation;
- lambda—defines the number of individuals generated in each generation as children of μ best individuals;
- iterations—defines the program operation time limit; after reaching this number of iterations, its operation terminates, and the best result found so far is written;
- iterationStep—defines the iteration step after which data on the current program status is written;
- mutationSigma—defines the initial standard deviation value employed in the adaptive distribution used by the mutation operator;
- functionName—defines the implemented function to be matched by the application; permissible values are Poly3 × 3, Poly5 × 5 and Cos, corresponding to the functions from Section 3;
- filePath—specifies the path to the file with input data related to points to be matched to the function; the path can be relative or absolute.

To be correctly read, a file loaded by the program must satisfy specific formatting requirements. The name and extension of the file do not matter; however, it has to be a text file compliant with the TSV (Tab-Separated Values) tabular data storage format. Subsequent rows in the file correspond to rows from the data table that is stored therein, while values from the columns for a given row are separated from each other by using tab characters. Each value is represented as a floating-point number. The program assumes that the file consists of four columns, sequentially denoting values from a given point on the X, Y and Z axes and then the weight of a given point. Point weights have been introduced to reduce the number of calculations for multiple points with the same coordinates, owing to calculating the quality function for a given coordinate triple only once.

Program operation results are presented in a standardised format separated by a pipe symbol. From the left, for every iteration step, the program stipulates:

- timestamp in a format compliant with ISO 8601;
- current number of executed iterations and total number of iterations separated by a slash;
- current standard deviation σ of the distribution used by the mutation operator;
- RMSE value for the currently found best solution;
- age (in iterations) of the currently found best solution;
- best found solution in the form of a row vector.

The program writes operation results via a standard output stream, by using existing solutions typical for the operating system it is run on. The results can be written both in the command line or saved to a file with a specific name. Owing to the use of native operating system mechanisms, the user can monitor the result destination without the need to implement additional parameters [62].

4.2. Software Architecture

The created application was developed in accordance with the object-oriented programming paradigm. Such an approach enables an ordered separation of various algorithms and different abstraction levels. It also increases the possibility of reusing an existing code when developing subsequent modules, without required re-implementation. The developed program consists of 11 classes in total. Their diagram is shown by Figures 8 and 9.

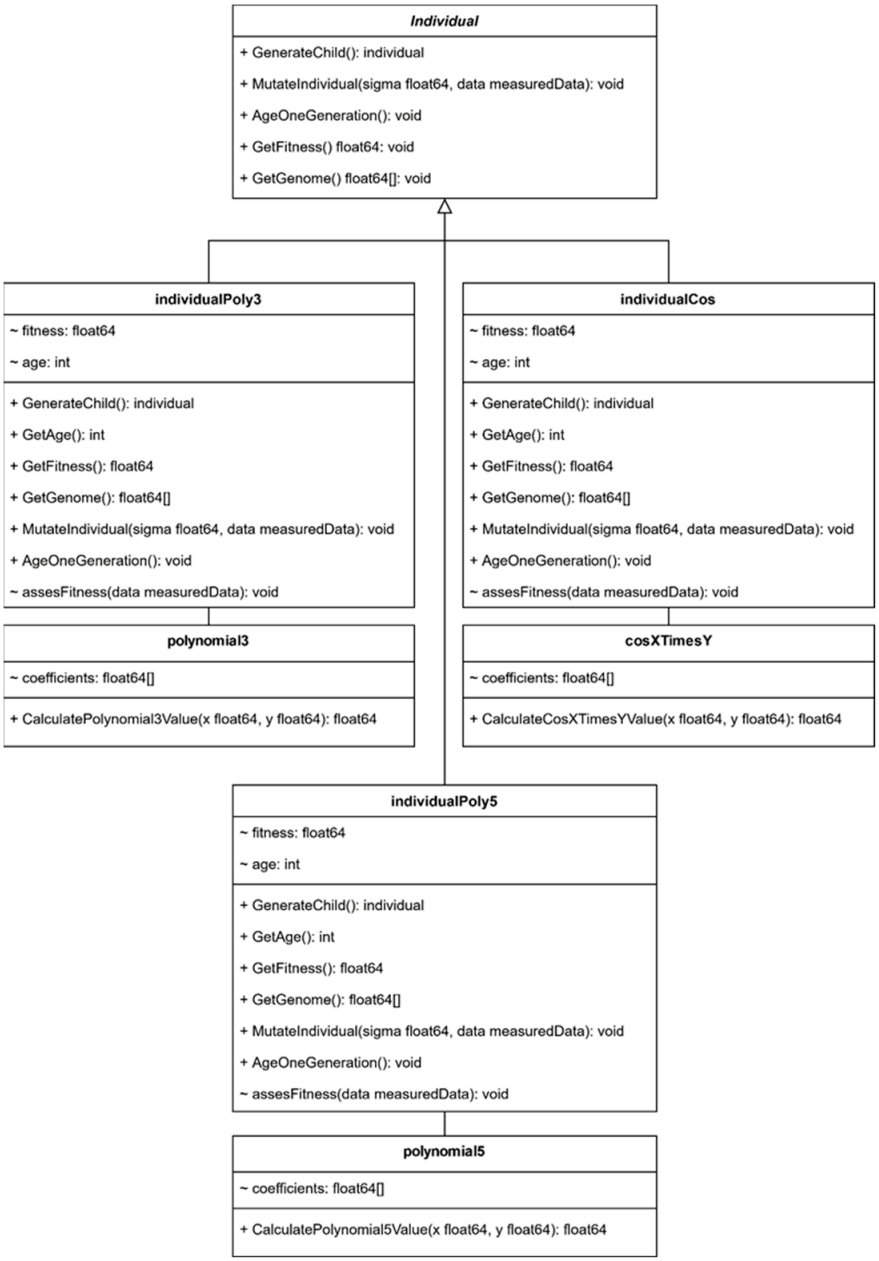

Figure 8. Diagram of implemented classes (1).

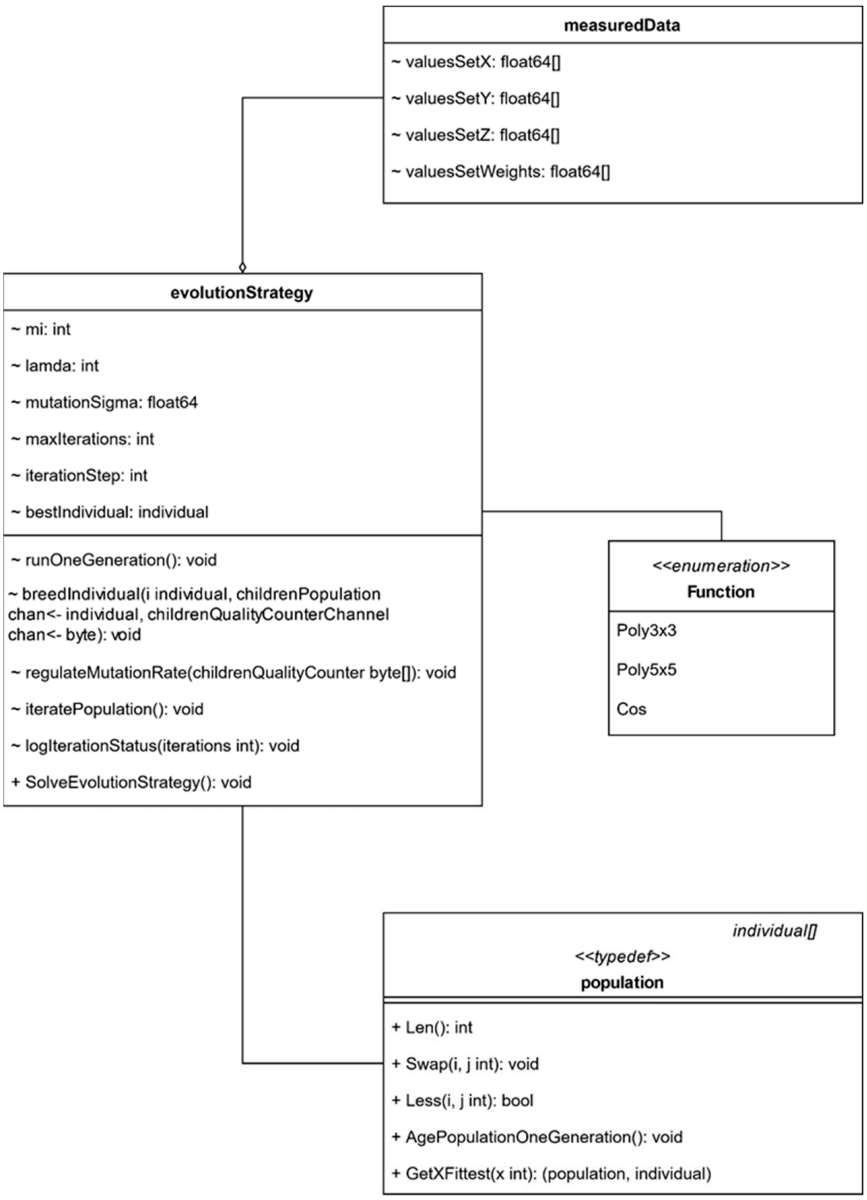

Figure 9. Diagram of implemented classes (2).

There are three basic classes representing functions from different classes, which are then matched to the point cloud loaded with input data. These classes are *polynomial3*, *polynomial5* and *cosXTimesY*, and they represent the successive functions discussed in Section 3.4. Each of the classes contains one field, *coefficients*, which represents successive function parameters as a list of 64-bit floating-point numbers. All three classes apply two methods with a shared nomenclature pattern: *New[ClassName]()* and *Calculate[ClassName]Value()*, where *[ClassName]* is *polynomial3*, *polynomial5* or *cosXTimesY*, respectively. The *New[ClassName]()* method is a class constructor, adopting a list of parameters that are to create a given function and returning an object of that class.

The *Calculate[ClassName]Value()* methods takes two floating-point numbers, called x and y, and based on them calculates the f(x,y) value for an object of this class that represents a specific function. These classes constitute the lowest abstraction level, operating on at most one point simultaneously and solely being an implementation of specific bivariate functions.

An additional baseline element that constitutes the foundation for evolution strategy algorithm implementation is the *measuredData* structure, which is only employed for organ-

ised storage of data loaded by the program. Therefore, it has not methods. This structure contains four fields, *valuesSetX*, *valuesSetY*, *valuesSetZ* and *valuesSetWeights*, which are lists of floating-point numbers. Each item on the list corresponds to a single row of loaded data, i.e., a measurement point understood as coordinates (x, y and z) within the Cartesian space and a point weight, which is the number of points with such coordinates.

Another abstraction level is the evolution strategy algorithm individual level. Polymorphism was used in order to avoid the necessity of repeatedly implementing the same algorithm. A class that represents an individual, *individual*, is only an abstract class, which is an interface that collects a set of methods that should be expected from an individual, regardless of the function it represents. This interface is implemented by a unified constructor *GenerateRandomIndividual()*, which takes the *measuredData* structure at the input and the *Function* enumerator that defines the function type loaded by the program in the *functionName* parameter during start-up. Based on the read enumerator, this constructor calls a constructor appropriate for the non-abstract class representing a specific function individual, *GenerateRandomIndividualPoly3()*, *GenerateRandomIndividualPoly5()* or *GenerateRandomIndividualCos()*, respectively. All constructors take only the *measuredData* structure. Their outcome is a new individual in the *individualPoly3*, *individualPoly5* or *individualCos* class, respectively.

Each of the three classes consists of three identically named fields. The first one is *genome*, which, depending on the class, takes the *polynomial3*, *polynomial5* or *cosXTimesY* type and identifies a given individual's genome. Another one is *fitness*, which is a 64-bit floating-point number representing individual quality. The last one is *age*, which denotes the age of a given individual in generations and is represented by an integer. This field is not required in the ES algorithm but enables drawing additional conclusions with regard to the current state of the individuals.

The *individual* class specifies the requirement for the presence of six public methods. *GenerateChild()* is a constructor corresponding to the *Copy()* method. Mutation is implemented via the *MutateIndividual()* method. This method takes a current standard deviation value σ and the *measuredData* structure, operating only on a given individual. The next three methods are *GetAge()*, *GetFitness()* and *GetGenome()*, which are used to read the age, quality and genome of the individual, respectively. The last of the methods that has to be shared by all classes representing the individuals is *AgeOneGeneration()*, which does not take any parameters, but after being called, it increases the value of the *age* field by one.

In addition to these methods, each of the classes representing an individual also has the private *assessFitness()* method, mapping the *Quality()* method and taking the *measuredData* structure. This method is applied to calculate individual quality during its creation and modification based on all points loaded by the program and an appropriate *Calculate[ClassName]Value()* method. Due to the privacy of this method, it is not visible from the *individual* interface level.

Another abstraction level is the population level and its representing *population* class. This class stores a list of individuals, which are objects of the *individual* class. Such a structure allows it to use methods typical for lists, such as determining collection length and adding elements, without the need to implement them. Its constructor is *InitializePopulation()*, which takes the initial number of individuals in a population, the *measuredData* structure and the *Function* enumerator as the input parameters. In addition to this process, this class implements five methods. Three of those have been implemented to enable using the sort interface from the sort module. These are:

- Len()—argument-free, return the collection length based on the len() function for the list;
- *Swap()*—takes two list items and swaps their positions;
- *Less()*—takes two list items and returns true if the first argument pointed to an individual with a lower value returned by the *GetFitness()* than the second one, or false in any other case.

In addition, this class implements the *AgePopulationOneGeneration()* method, which calls the *AgeOneGeneration()* method for each individual in a population, without taking arguments and returning results. The last method of this class is *GetXFittest()*, which takes an integer, sorts the individual list, and returns a result pair; a population subset is limited to a preset value by an argument of the number of best individuals and the best individual within the population.

The *evolutionStrategy* is the class highest in the hierarchy. It fully implements the ES-evolution strategy algorithm ($\mu + \lambda$). This class contains a total of nine fields, four of which store objects of already-described classes. These are *pop*, which stores objects of the *population* class and represents an algorithm population for a given iteration; *bestIndividual*, which stores an *individual* class object and remembers the best previously found individual; *data*, which is a *measuredData* structure containing data to which a function is matched; and the *function* enumerator of the *Function* type, which defines the algorithm-operating mode in the context of the selected, matched function. In addition, the *evolutionStrategy* class contains five fields directly related to the adjustment of its behaviour. These are *mi*, which denotes the number of selected parents; *lambda*, which is the number of new individuals generated at every iteration; *mutationSigma*, which stores the current initial standard deviation value used in the adaptive distribution applied by the mutation operator; *maxIterations*, which denotes the maximum number of iterations that can be called by an algorithm; and *iterationStep*, which determines the step at which the algorithm reports previous results.

The *evolutionStrategy* class has one constructor, *NewEvolutionStrategy()*, which takes parameters corresponding to the *mi*, *lambda*, *mutationSigma*, *maxIterations*, *iterationStep*, *data* and *function* fields. The *pop* and *bestIndividual* fields are initially empty. Besides the constructor, this class also has one more public method, *SolveEvolutionStrategy()*, which executes an evolution strategy algorithm for the parameters set in the constructor. This method does not take any additional parameters, as it uses the already-existing class fields. It also does not return anything. The outcomes of this method in the form of logs are sent to a standard output stream over the course of its execution.

In order to ensure higher code clarity, the class also implements private auxiliary methods. *iteratePopulation()* is a method of the highest level, which operates on data acquired within an object of this class and executes subsequent iterations of the ES algorithm. This method employs two further lower-level methods: *logIterationStatus()*, which writes previous program results, and *runOneGeneration()*, which is responsible for executing a single algorithm iteration. Two further private methods are used within this single operation. The *breedIndividual()* method takes an individual and two interthread communication channels and implements the *Mutuj(Copy())* mechanism. The *regulateMutationRate()* method takes a list of bytes representing children which are better or worse than their parents and, on this basis, changes the *mutationSigma* pursuant to the Ingo Rechenberg one-fifth rule.

Three methods not assigned to any class have been implemented additionally. Based on the input file path entered as an argument, the *GetDataInput()* method loads the file's data and fills the *measuredData* structure, which it then returns. In the event of encountering an error while reading a file, the method terminates program operation and displays an appropriate message. The *init()* method is a built-in Go feature. It is characterised by the fact that the operations implemented therein are executed prior to any other program tasks. In the case of the discussed application, this method has been used to define user interface input flags and transfer the values of loaded arguments to their corresponding global variables. After executing the *init()* method, the program calls the *main()* method, which is its main function. Based on arguments preset by a user, this method first loads a data file, then creates an object of the *evolutionStrategy* class, which is followed by executing the ES algorithm itself.

4.3. Implementation of a Function Determination Module

The implemented algorithm executes similar steps to those described in Section 3; however, an analysis of the applied solutions is required from an engineering perspective. The *SolveEvolutionStrategy()* method behaves identically to the original ES algorithm by first initialising an initial population of λ individuals, then launching the iteration process via the *iteratePopulation()* method.

The *iteratePopulation()* method is responsible for iteration counting, i.e., an infinite loop, and calls the *runOneGeneration()* method. After it is executed, based on the remainder of dividing the current number of iterations by *iterationStep*, it decides to enter the previous program operation result to the standard output stream. Stop conditions present in the original algorithm have been fine-tuned for the purposes of the Go language as reaching an iteration limit or reaching the best individual quality lower or equal to $\varepsilon = 5 \times 10^{-322}$, which is 100 times the lowest achievable positive value of a 6-bit floating-point number. In both cases of reaching the stop condition, the end result of the program operation is entered into the standard output stream.

The *runOneGeneration()* method implements equivalents of the steps executed in a single iteration of the original algorithm, albeit with minor changes. Due to the fact that individual quality is calculated when it is created, there is no need to recalculate it. Therefore, this method calls the *GetXFittest() population* class method, which simultaneously returns the best *mi* of individuals in the population as its subset, and the best individual. Because the method sorts directly on an object of the *population* class and keeps the best parents, there is no need to verify whether the new best individual is better than the current one. There are only two possibilities: it is better, or it is the same individual. The *bestIndividual* can be overwritten in both cases. Next, the step of ageing a subset of surviving individuals is conducted using the *AgePopulationOneGeneration()* function.

After determining a group of parents to be used to create the next generation of individuals, this method calls the *breedIndividual()* time *lambda*/*mi* methods. In order to benefit from the Go language multithreading, this method is a so-called *goroutine*. This means that each call of this method is executed in a new thread and processed independently of the others. The channel mechanism is applied in order to be able to synchronise the results of the subsequent calls of this method. Channels are named queues, which implement safe multi-threaded access.

The *breedIndividual()* method executes two activities, the results of which are then transferred to appropriate channels. First of all, it copies the parent genome and creates a new individual based on it, including the calculation of its new *fitness* quality value. A child is transferred to a descendant collective channel. If the *fitness* of both the parent and the child is known, it is also possible to determine whether a given child was better. This information will be required to calculate the ratio of children better than parents to all children, which is used in the Ingo Rechenber one-fifth rule. In the case of a child better than the parent, the second channel receives byte 1 (0×31), and if the parent was better, the 0 (0×30) byte is transmitted.

As the next called threads are completed, the *runOneGeneration()* method receives the data they send, adding the population of children to the general population represented by the *pop* field. Bytes denoting a better or worse children quality relative to the parents are sent to the *regulateMutationRate()* method. This method counts awarded ones and compares their total with the *lambda* field, followed by increasing or reducing the *mutationSigma* according to Formula (23). The value of the *a* parameters was adopted as 0.9.

4.4. Testing

The application was tested on a set composed of a thousand randomly selected points determined by the function (27), to which a noise according to the distribution $\mathcal{N}\left(0.25 \times 10^{-4}\right)$ has been applied.

$$f(x, y) = 2 + x^2 + y^3 \tag{27}$$

where x, y—variables adopted by the function.

The following initial program operating parameters were selected:

- *mi*—5;
- *lambda*—40;
- *iterations*—100,000,000;
- *mutationSigma*—0.0005;
- *functionName*—Poly3 × 3.

Figure 10 represents the waveform of the RMSE (*fitness*) value.

Figure 10. Dependence of the mean squared error in the process of matching a surface to a point cloud on the number of iterations on a logarithmic scale.

A value that approximates the wanted function is obtained after completing 75,527,000 iterations. The obtained solution is the following number vector:

[1.998216877812118	−0.00010561199842972524	$-1.7439057315740977 \times 10^{-5}$
1.00000317833339	$9.306751266278393 \times 10^{-7}$	$1.2590903513959131 \times 10^{-6}$
$6.22269483420075 \times 10^{-8}$	$3.4355343136749796 \times 10^{-8}$	$1.3122233646017448 \times 10^{-7}$
0.9999999548969462]		

By approximating these numbers to the second decimal point, one can obtain the following parameter values of the wanted function, according to Formula (26):

$$p_{00} = 2, p_{10} = 0, p_{01} = 0, p_{20} = 1, p_{11} = 0, p_{02} = 0, p_{30} = 0, p_{21} = 0, p_{12} = 0, p_{03} = 1$$

It accurately maps the original function expressed by Formula (29). Furthermore, it can be noticed that the ultimately obtained value of the *fitness* parameter was 0.05, which is exactly the same as the standard deviation. In the case of such a selected noise distribution, this dependence was not accidental. The expected value of the given distribution was zero; hence, the *f(x,y)* parameter value estimator was not biased. In the case of an unbiased estimator, the MSE is equal to its variance, and RMSE is equal to the standard deviation.

The algorithm was executed repeatedly for various pseudo-random number generator seeds, each time achieving a result reflecting the initial function. After such tests of the mechanism, the next step involved calculations on real data and for an unknown function determining the value of diffuse irradiance based on measured and theoretical total irradiance values.

5. Calculation and Analysis of Obtained Functions

Calculating the function with the implemented application requires the appropriate selection of its parameters and input data. After a series of calculations, it was possible to analyse the results of algorithm operations.

5.1. Selection of Parameters for the Evolution Strategy Algorithm

Searching for the solution with the evolution strategy method requires matching its parameters to the specifics of the environment in which it will be deployed [54]. This is necessary due to their impact on the time required for a single duration and the degree to which their increase or decrease improves or deteriorates the achieved results.

The first parameter was *lambda*, which is the number of individuals generated within each generation. Very low values of this parameter may be unable to fully utilise the full potential of multi-core processors, while appropriately large values will fully exhaust the CPU, leading to increased iteration execution time. The impact of the lambda value on the average time of 10,000 iterations on an Intel® Core™ i7-7700HQ processor (4-core, 8 logical processors) is shown in Figure 11.

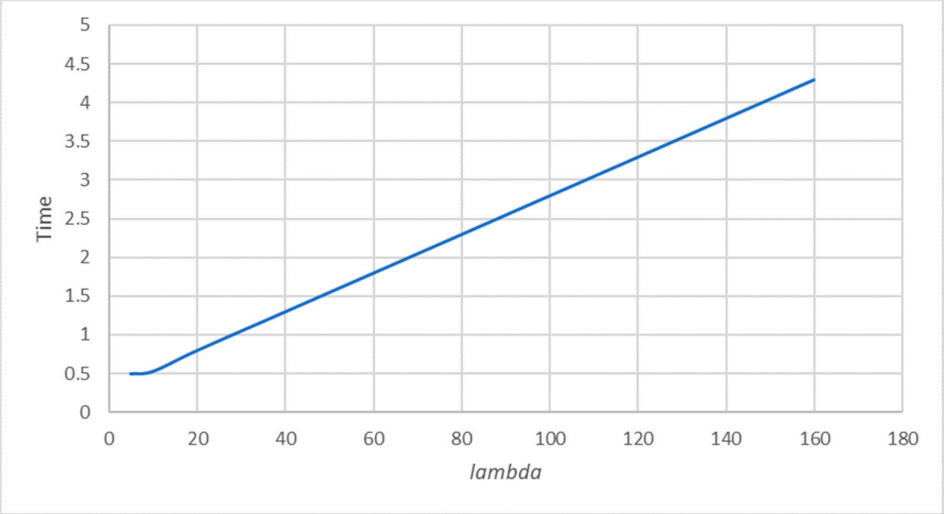

Figure 11. Average time of 10,000 iterations depending on the *lambda* parameter value.

Tests were conducted for the *mi* parameter of 5 and the *lambda* parameter from 5 to 160. It can be seen that for values above 10, this dependence takes a linear character, since the processor is fully used. For smaller values, the duration of 10,000 iterations is constant, since the calculations associated with every subsequent individual are also allocated to unused cores.

Full tests were conducted for *lambda* values equal to 10, 20 and 40. Five tests were executed for each of the three values, and their results were averaged. The dependence of the mean *fitness* value on the number of iterations for different values of the *lambda* parameter is shown in Figure 12.

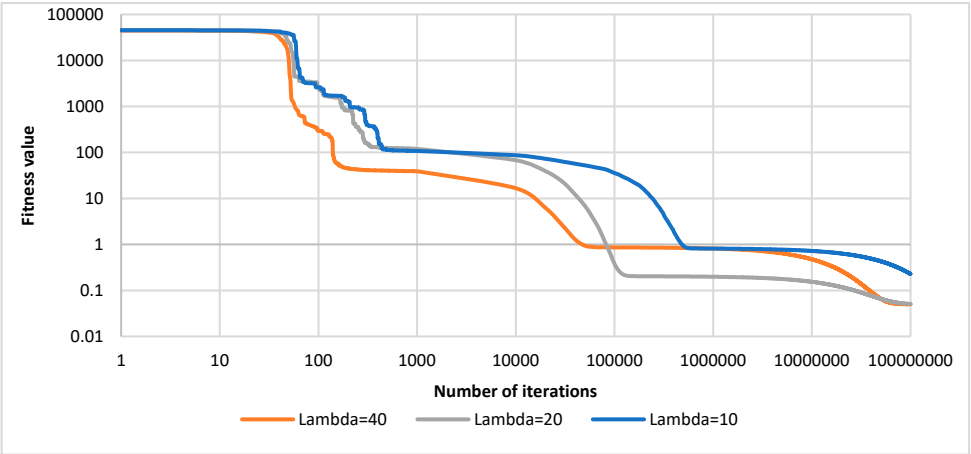

Figure 12. Dependence of the *fitness* value on the number of iterations for different *lamba* values.

The second parameter that impacts the algorithm operation is *mi*, which is the number of parents, i.e., individuals to be chosen from each generation as the best and used to create a next generation. Due to its memory-like nature, the value of this parameter does not noticeably impact program execution time. A single individual in the analysed case is a set of several to several dozen 8-byte numbers. For modern desktop computers equipped with 8–16 Gb RAM, a set of even thousands of such individuals is negligibly small and does not require using a hard drive as part of the paging mechanism. However, this parameter is inherently limited by the value of the *lambda* parameter and should be at least twice as low. Tests were conducted for a *lambda* equal to 40 and *mi* equal to 2, 5, 10 and 20. Five tests were performed for each of the four values, and their results were averaged. Very small *mi* values do not use the memory capacity of the evolution strategy well enough, as such values are very slow in improving the results. Increasing the number of remembered good individuals leads to a more rapid achievement of better results, although the yields become smaller as the maximum *mi* parameter value is approached. A *mi* value four- or eightfold lower than the *lambda* parameter is a good choice for the environment in question.

5.2. Analysis and Selection of Input Data

Determining an accurate relationship between total irradiance for a plane parallel to the Earth's surface and diffuse irradiance requires large data sets with these values. Available data sets should be assessed in terms of representativeness and then validated in order to eliminate erroneous measurements.

Meteorological stations are the basic data source. In Poland, these are primarily the devices of the Institute of Meteorology and Water Management (IMGW), although only certain stations record diffuse radiation. Historical data from the stations in Warsaw, Kołobrzeg, Mikołajki, Legnica and Zakopane (Poland) were available in the course of the study. Most of the data were recorded as an hourly-averaged value, and only the measurements from Zakopane were available with a minute resolution [15]. Minute-resolution measurements are primarily suitable for the purposes of the simulation model, since the objective of the model is to reflect instantaneous capacity and its change dynamics, and not averaged yields.

Next, the data were selected. If certain measurement data subsets are characterised by a drastically different scale of errors to others, they will distort the resultant function [63]. The resultant data were analysed in order to mitigate the impact of these subsets. The authors found measurements covering several days, which were discarded as collected incorrectly given that the obtained values were far from what was expected (and physically possible). Discarding this group of measurement points did not affect the representativeness of the remaining data set. In addition, due to the minor suitability in predicting PV cell yields, the points for which the solar irradiance value was lower than 1 W/m^2 for all three measurement point coordinates were also discarded. Next, the collected data were rounded to two decimal places, reducing the size of the set from 700,000 to 400,000 points. These points were then aggregated by counting the points with identical coordinates, which enabled further cutting of the number of loaded points to 200,000, reducing calculation time but not impacting the results.

5.3. Optimal Function Determination

The program was executed 10 times for each of the 3 analysed function families, each time with a different pseudo-random number generator seed. The initial parameters were set to *mi* = 10 and *lambda* = 40, which were selected in Section 5.1 as being matched to the runtime environment. The number of iterations was limited to 10,000,000.

For the function determined by Formula (26), the best result is a vector of numbers:

[2.9605494704543585	0.5377862858796892	0.06004726482252161
-0.0026910268413485984	0.0019042965793833588	-0.0001775673348808147
$3.04217224869 \times 10^{-6}$	$-3.1786249095923 \times 10^{-6}$	$8.214875913822176 \times 10^{-7}$
$-1.0830525365875006 \times 10^{-7}$]		

for which the RMSE value is 40.95284691653278. This vector translates to the resultant function determined by Formula (28):

$$\begin{aligned} f(x,y) = {} & 2.9605494704543585 + 0.5377862858796892x \\ & +0.06004726482252161y - 0.0026910268413485984x^2 \\ & +0.0019042965793833588xy - 0.00017756733488081473y^2 \\ & +3.042172248692004 \times 10^{-6}x^3 - 3.178624909592273 \times 10^{6}x^2y \\ & +8.214875913822176 \times 10^{7}xy^2 - 1.0830525365875006 \times 10^{7}y^3 \end{aligned} \tag{28}$$

where x—G_{CP}, measured total irradiance [W/m^2], and y—G_{CT}, theoretical total irradiance [W/m^2].

For the function determined by Formula (27), the best result is a vector of numbers:

[1.9659108551573048	0.8188506352759151	−0.0256210937603285
−0.006611032243318028	0.003714565968617965	$5.597695265923985 \times 10^{-5}$
$2.2881860799959824 \times 10^{-6}$	$1.1086528752446855 \times 10^{-5}$	$-6.560944781900792 \times 10^{-6}$
$-8.339455306211456 \times 10^{-7}$	5.615×10^{-8}	-1.418×10^{-7}
1.015×10^{-7}	-2.613×10^{-8}	3.783×10^{-9}
2.273×10^{-11}	-1.215×10^{-11}	2.052×10^{-10}
-1.341×10^{-10}	3.362×10^{-11}	-3.201×10^{-12}

for which the RMSE value is 47.92115946319969. This vector translates to the resultant function determined by Formula (29):

$$\begin{aligned} f(x,y) = {} & 1.9659108551573048 + 0.8188506352759151x - 0.0256210937603285y \\ & -0.006611032243318028x^2 + 0.003714565968617965xy \\ & +5.597695265923985 \times 10^{-5}y^2 + 2.2881860799959824 \times 10^{-6}x^3 \\ & +1.1086528752446855 \times 10^{-5}x^2y \\ & -6.560944781900792 \times 10^{-6}xy^2 - 8.339455306211456 \times 10^{-7}y^3 \\ & +5.615 \times 10^{-8}x^4 - 1.418 \times 10^{-7}x^3y + 1.015 \times 10^{-7}x^2y^2 \\ & -2.613 \times 10^{-8}xy^3 + 3.783 \times 10^{-9}y^4 + 2.273 \times 10^{-11}x^5 \\ & -1.215 \times 10^{-10}x^4y + 2.052 \times 10^{-10}x^3y^2 - 1.341 \times 10^{-10}x^2y^3 \\ & +3.362 \times 10^{-11}xy^4 - 3.201 \times 10^{-12}y^5 \end{aligned} \tag{29}$$

For the function determined by Formula (28), the best result is a vector of numbers:

[22.63909757016813	99.99165999987993	0.00755917293489859
0.492087131216322	0.0034090857597221373	0.5332257196602475
0.12301637049075635	0.15722034811653302]	

for which the RMSE value is 55.244444210453764. This vector translates to the resultant function determined by Formula (30):

$$\begin{aligned} f(x,y) = {} & 22.63909757016813 + 99.99165999987993 \\ & \cdot \cos(0.00755917293489859 \cdot x + 0.492087131216322) \\ & \cdot \cos(0.0034090857597221373 \cdot y + 0.5332257196602475) \\ & +0.12301637049075635 \cdot y + 0.15722034811653302 \cdot x \end{aligned} \tag{30}$$

The search results are summarised in Table 3. Among the three determined functions, the best results are exhibited by the third-order polynomial function. It might seem that since it is possible to represent a third-order polynomial via a fifth-order polynomial, the function expressed by Formula (25) should achieve results not worse than (24). This is true, assuming an infinite number of iterations; however, the iteration limit in this case was the same for all three function families. Searching the solution space for a fifth-order polynomial is executed for a larger number of parameters (21 instead of 10); hence, their determination might take longer. A potential improvement of such a function formula

matching method is using the best result of a third-order polynomial, which is less time-consuming to calculate, as one of the initial starting points for a fifth-order polynomial, with non-present higher-order parameter initially set to 0.

Table 3. Program operation results.

	3rd-Order Polynomial	5th-Order Polynomial	Heuristic Function
RMSE	40.95284691653278	47.92115946319969	55.244444210453764
Average time of 10,000 iterations	2 min 28.9 s	5 min 2.6 s	1 h 32 min

On average, the algorithm iteration execution time for a fifth-order polynomial was twice as long as that for the third-order polynomial, which means slightly better results for the fifth-order polynomial than predicted. The heuristic function based on the cosine product took 18 times as long as the fifth-order polynomial and 36 times as long as the third-order polynomial. It was possible to terminate program functioning as per the iteration limit for both analysed polynomial function families. The heuristic function defined by Formula (26) achieved worse results than did the polynomials.

Referring to the RMSE values presented in the table, it can be seen that they are relatively large (approximately 10%). The reason is primarily a large dispersion of sourced data. This can be seen in Figure 5. For the same total radiation intensity, the diffuse radiation values range from 15% to 95% of the total radiation. At the same time, extreme values, although they occur relatively rarely, have a large impact on the RMSE. It should be added that the reason for the large dispersion of the measured values is not the low accuracy of the sensors. The theoretical measurement uncertainty of the sensors should not exceed 3%. The main reason for such a large dispersion of sourced data in static conditions is changing weather and environmental conditions causing changes in the reflection and scattering of solar radiation. In addition, with a minute measurement resolution, there is a large influence of dynamic states. Different time constants of the sensors and the distance between them cause the situation in which measurements recorded at the same time with variable insolation give significantly different values. Therefore, taking into account the large spread of input data, the obtained RMSE values should be considered good.

The RMS values presented in the table show that the application of the third-order polynomial function, as defined by Formula (29), gives the statistically smallest RMS calculated on the basis of data from the entire four-year period. The highest RMS occurred for the heuristic function, based on the product of cosines. In addition, taking into account the fact that the algorithm is the least computationally complex, the third-order polynomial can be considered the best.

5.4. Analysis of Resultant Functions in Terms of Various Data Acquisition Periods

The obtained functions were then employed to calculate the error between the measured and calculated diffuse irradiance for specific measurement points and to analyse the collected data, taking time measurement into account. RMSE values have been recalculated, but this time, they were broken down into months, to which a given measurement point belonged. The results of this process are depicted in Figure 13.

Figure 13 shows that the RMSE in the summer months is higher than in the winter; in some cases, it is even twice as high. However, under the climatic conditions in which the research was conducted, the average solar radiation intensity in June is at least five times higher than that in December. This means that the relative accuracy of diffuse radiation calculations will be greater in the summer months. In addition, Figure 13 shows that although the third-order polynomial for the whole year gives the smallest RMSE, with low insolation in the winter months (November to February), lower RMSE values are for the fifth-order polynomial function.

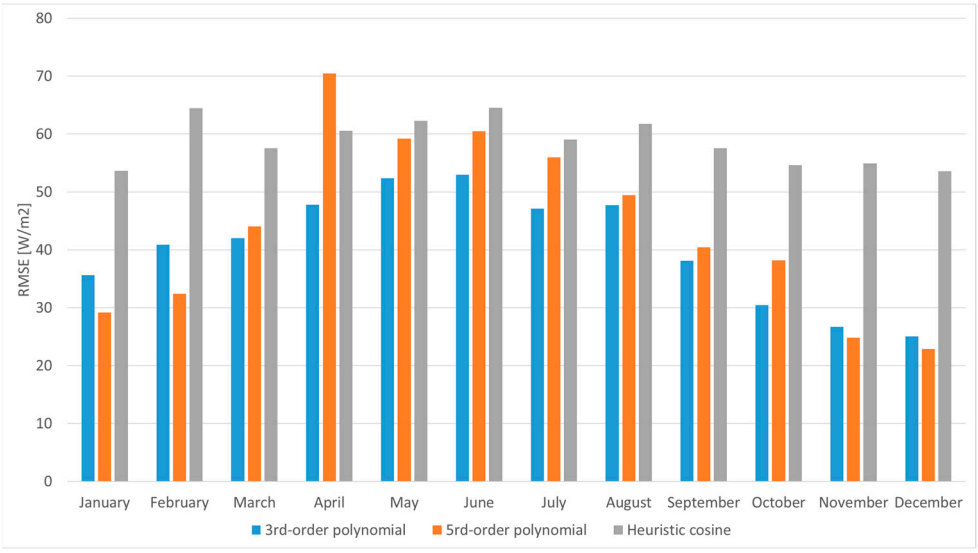

Figure 13. Root-mean-squared error value for resultant bivariate functions, depending on the measurement month.

Since most of the energy is obtained at high angles of the Sun above the horizon, the analysis was repeated for the data, excluding periods when the angle of incidence was 10° or less. The result of the calculations is shown in Figure 14. The exclusion of these data is also justified by the fact that for very small angles of the Sun's angle of elevation ($\alpha < 5°$), additional functions are used to correct the content of direct radiation and prevent tracking errors in calculations using Formula (3).

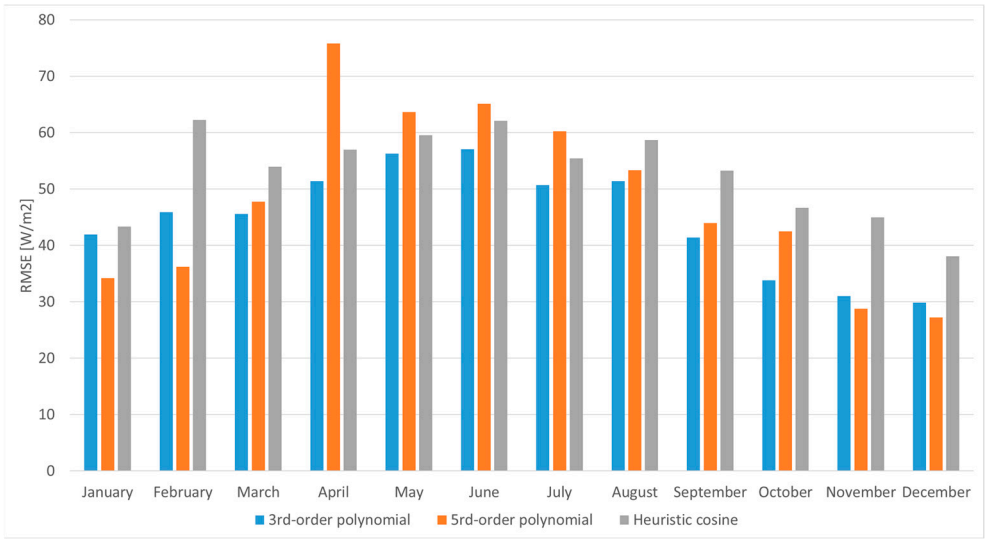

Figure 14. Root-mean-squared errors of resultant functions depending on the measurement month for an incidence angle greater than 10°.

In Figure 14, you can see that the error for the heuristic function based on the cosine product decreased and was performing well for the summer months. This suggests that a polynomial function can be used for small angles of elevation of the Sun, and for large angles, a heuristic function can be used.

Since the aim is to use the developed functions to simulate the waveforms, in addition to the RMSE value, an analysis of the daily waveforms was also performed. Figures 15–17 show the waveforms of scattered radiation measured and calculated for sample days. This

makes it possible to compare the quality of reproducing the values measured with the developed functions that calculate the diffuse radiation intensity.

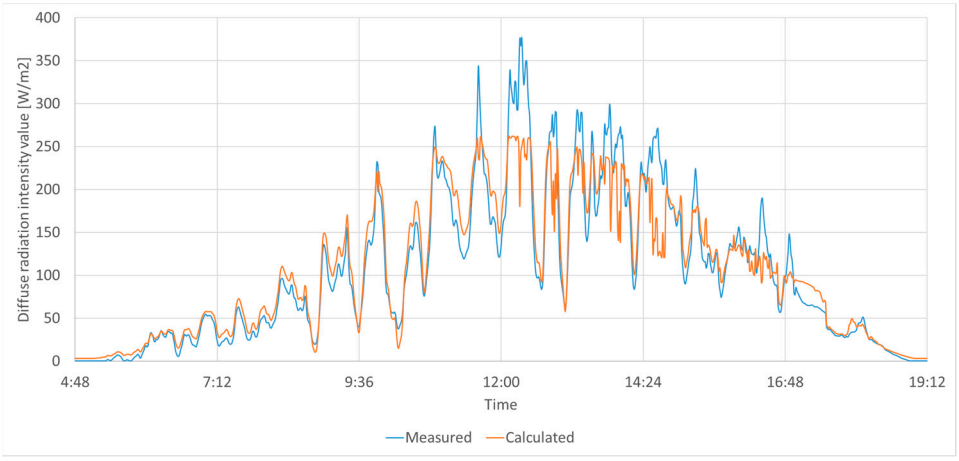

Figure 15. Compared scattered irradiance values measured and calculated using bivariate 3rd-order polynomial.

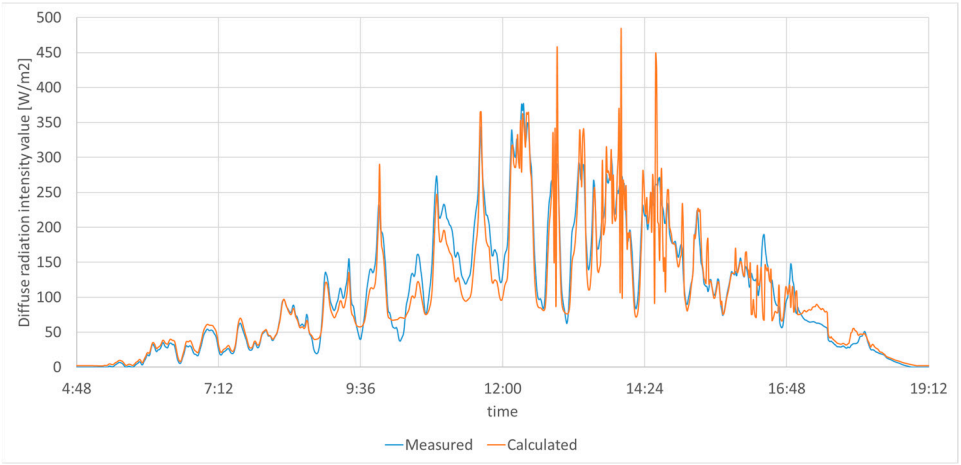

Figure 16. Compared scattered irradiance values measured and calculated using bivariate 5th-order polynomial.

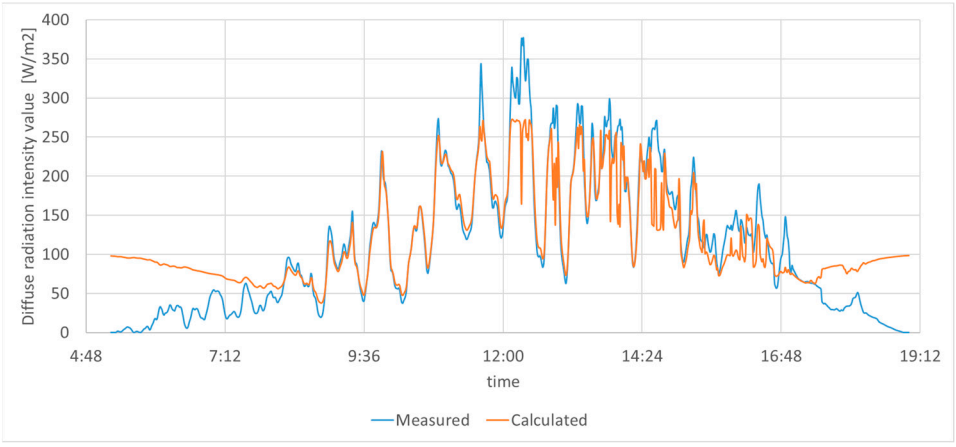

Figure 17. Comparison of measured and calculated diffuse irradiance by using a heuristic function based on the cosine product.

A day with high dynamics of changes in solar radiation intensity was selected for the analysis. The graphs show that for waveforms with high variability, the diffuse radiation values are best reproduced by a fifth-order polynomial function. This suggests that it is worth using it in the analysis of dynamic states, e.g., diagnostics of PV systems or maintaining the maximum power point (MPP).

6. Summary and Conclusions

As presented in the introduction, the comprehensive aim of the research is to develop a simulation model of the dynamic operation of PV systems based only on basic meteorological parameters. The resulting aim of the research described in the article was, therefore, to determine the function that calculates the diffuse radiation on the basis of the total radiation, as measured with a minute time resolution. Since the diffuse radiation models presented in the global literature are mostly based on a larger number of measured parameters and are adapted to a smaller frequency of measurements, own functions were developed, which were optimised by using evolutionary algorithms using a proprietary computer application.

Due to the lack of an analogous function that meets the assumptions presented in the introduction, the authors could not compare the results obtained with other methods in terms of the accuracy of calculating the instantaneous values of scattered radiation based on identical data sets.

However, the functions presented in the article, compared to those previously used by the authors, enable obtaining RMSE from 30% to 50% lower, depending on the set of input data, and better reproducing the waveforms. Previously, the authors used optimised functions of one variable or functions of two variables but without machine optimisation. A detailed comparison with the functions of two machine-optimised variables described in this article would, therefore, not be useful.

According to the authors, the quality of the developed functions is best demonstrated by the accuracy of reproducing the power waveforms of PV modules by using a simulation model containing the third-order polynomial function presented in the article.

This article involved creating and describing a modified mathematical model that was used to simulate the power efficiency of photovoltaic cells. The hypothesis was confirmed by using a developed universal tool to calculate the coefficients of a two-parameter function matching a surface to a point cloud based on an evolution strategy algorithm. Good results were obtained despite the scattered nature of the data. Both the polynomial and heuristic function based on the cosine product enable calculating the sought irradiance with a minor error. The authors developed functions to calculate diffuse solar irradiance based on measured and theoretical total irradiance for a cloudless sky. Such a modified model enables a more accurate simulation of the solar radiation on PV cells located at any azimuth and inclination angle. This consequently leads to a more thorough simulation of photovoltaic cell yields based on minute-by-minute meteorological data. Furthermore, preliminary practical tests presented in Appendix A indicate the versatility of the developed model with regard to several locations in Poland. Therefore, it is advisable to continue the research in order to determine their adequacy for more dispersed locations.

As far as the practical application of this model is concerned, it can be used with regard to both current and historical meteorological data, as well as forecast data. A simulation model for actual measurement data will enable more precise monitoring and faster diagnosing of photovoltaic systems, taking dynamic states into account. Another important aspect will be increasing the accuracy of short-term energy efficiency forecasts and the assessment of its impact on the energy balance. The developed model will also be suitable for an in-depth analysis of designed photovoltaic systems installed in an irregular manner, which will be optimised in terms of adapting the power characteristics to local grid load as a function of time.

Author Contributions: Methodology, W.O.; Software, J.G.; Validation, W.O.; Data curation, J.G.; Writing—review & editing, A.M.; Supervision, A.M. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: The data presented in this study are available on request from the corresponding author.

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

Appendix A

Besides theoretical analyses, the authors conducted preliminary comparative studies of power waveforms that were obtained via a simulation model and based on actual measurements of photovoltaic system examples. The simulation model employed the aforementioned bivariate third-order polynomial function. The studies are treated as preliminary because the model can still be improved with regard to its function of cell temperature under varying weather conditions and the function of relative cell efficiency depending on solar irradiance.

The representative example included waveforms for a photovoltaic system located near Warsaw (geographic coordinates: 52.25, 20.86) with cells oriented along three azimuths: 110°, 200°, 290° relative to the north and inclined at an angle of 37° relative to the Earth's surface. The measurements were taken by using a meteorological station that was located 2 ÷ 10 m from the PV cells and equipped with two solar irradiance sensors, a temperature sensor and an anemometer. Power waveforms for individual cells were recorded using built-in sensors in power optimisers. Graphs with the waveforms for three azimuths are shown in Figures A1–A3.

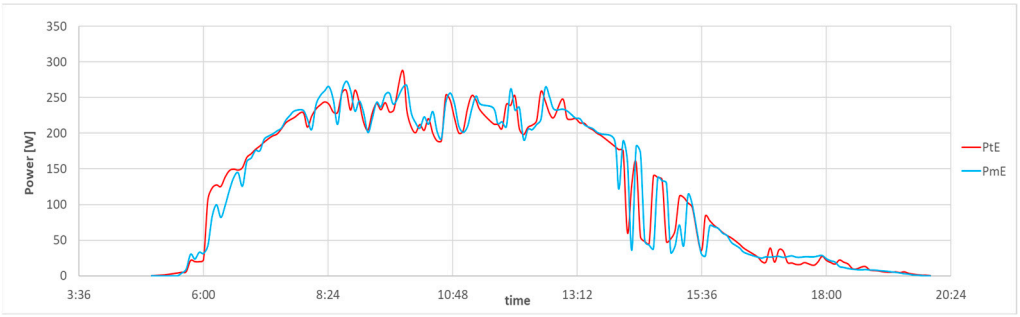

Figure A1. Comparison of simulated and measured power waveforms for PV cells oriented eastward, bearing 110°; PtE—theoretical power, PmE—measured power.

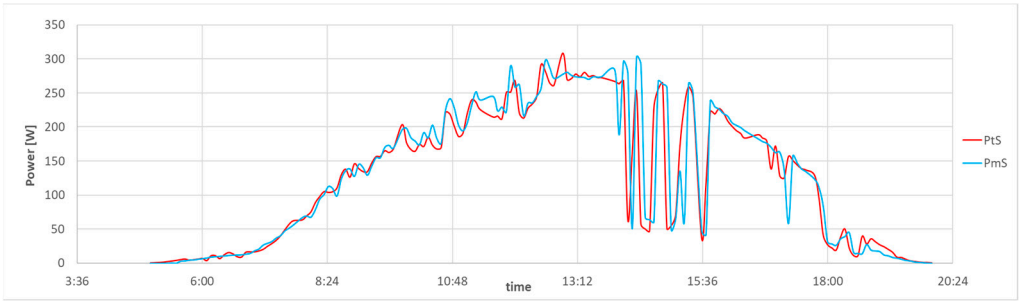

Figure A2. Comparison of simulated and measured power waveforms for PV cells oriented southward, bearing 200°; PtE—theoretical power, PmE—measured power.

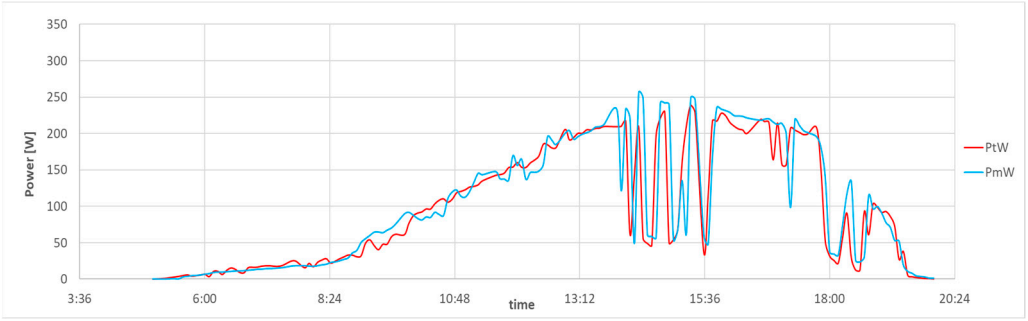

Figure A3. Comparison of simulated and measured power waveforms for PV cells oriented westward, bearing 290°; PtE—theoretical power, PmE—measured power.

An analysis of the figures indicates that simulated power waveforms significantly differ for the same meteorological data. The causes behind the differences are solely the values of calculated direct radiation, which provides considerably varying values after conversion for individual azimuths. It can also be seen that simulated and measured power waveforms for individual azimuths are very similar. Particularly noteworthy is the consistency of the waveforms during intensive insolation changes (14 ÷ 16 h). The obtained results prove the high accuracy in calculating direct and diffuse radiation; hence, the developed function covering the dependence of diffuse radiation on total radiation is accurate. Similar studies were also conducted for photovoltaic systems at three different locations based on meteorological station data and inverter capacity waveforms. The results also confirmed high accuracy of the simulation model.

References

1. Vald, L. *Fundamentals of Solar Radiation*; CRC Press: Boca Raton, FL, USA, 2021.
2. Letcher, T. (Ed.) *Comprehensive Renewable Energy*; Elsevier: Amsterdam, The Netherlands, 2022.
3. Hay, J.E. Calculating solar radiation for inclined surfaces: Practical approaches. *Renew. Energy* **1993**, *3*, 373–380. [CrossRef]
4. Messenger, R.A.; Abtahi, A. *Photovoltaic Systems Engineering*; CRC Press: Boca Raton, FL, USA, 2010. [CrossRef]
5. Piotrowski, P.; Parol, M.; Kapler, P.; Fetliński, B. Advanced Forecasting Methods of 5-Minute Power Generation in a PV System for Microgrid Operation Control. *Energies* **2022**, *15*, 2645. [CrossRef]
6. Pavlovic, T. *The Sun and Photovoltaic Technologies*; Springer Nature: Berlin, Germany, 2019.
7. King, D.L.; Boyson, W.E.; Kratochvil, J.A. *Photovoltaic Array Performance Model*; Sandia National Laboratories: Albuquerque, NM, USA, 2004.
8. Olchowik, W. Simulation of Systems with Solar Collectors in Relation to the Raw Meteorological Data. *Biul. Wojsk. Akad. Tech.* **2017**, *66*, 37–54. (In Polish) [CrossRef]
9. Cooper, P. The absorption of radiation in solar stills. *Sol. Energy* **1969**, *12*, 333–346. [CrossRef]
10. Mohanty, P.; Muneer, T.; Jadraque Gago, E.; Kotak, Y. *Solar Radiation Fundamentals and PV System Components*; Springer: Cham, Switzerland, 2015.
11. Sevilla-Camacho, P.-Y.; Zuniga-Reyes, M.-A.; Robles-Ocampo, J.-B.; Castillo-Palomera, R.; Muniz, J.; Rodriguez-Resendiz, J. A Novel Fault Detection and Location Method for PV Arrays Based on Frequency Analysis. *IEEE Access* **2019**, *7*, 72050–72061. [CrossRef]
12. Martínez-Sánchez, R.A.; Rodriguez-Resendiz, J.; Álvarez-Alvarado, J.M.; Macías-Socarrás, I. Solar Energy-Based Future Perspective for Organic Rankine Cycle Applications. *Micromachines* **2022**, *13*, 944. [CrossRef]
13. Alvarez-Diazcomas, A.; López, H.; Carrillo-Serrano, R.V.; Rodríguez-Reséndiz, J.; Vázquez, N.; Herrera-Ruiz, G. A Novel Integrated Topology to Interface Electric Vehicles and Renewable Energies with the Grid. *Energies* **2019**, *12*, 4091. [CrossRef]
14. Lopez, H.; Resendiz, J.R.; Guo, X.; Vazquez, N.; Carrillo-Serrano, R.V. Transformerless Common-Mode Current-Source Inverter Grid-Connected for PV Applications. *IEEE Access* **2018**, *6*, 62944–62953. [CrossRef]
15. Institute of Meteorology and Water Management, IMGW: Actinometric Data. Available online: danepubliczne.imgw.pl/data/dane_pomiarowo_obserwacyjne/dane_aktynometryczne/ (accessed on 3 June 2022).
16. Jastrzębska, G. *Solar Cells. Construction, Technology and Application*; WKŁ: Warsaw, Poland, 2014. (In Polish)
17. Musta, L.G.; Zhurov, G.N.; Belyaev, V.V. Modeling of a solar radiation flow on an inclined arbitrarily oriented surface. *J. Phys. Conf. Ser.* **2019**, *1333*, 032054. [CrossRef]
18. NASA. NASA EarthData. Available online: https://search.earthdata.nasa.gov/search (accessed on 4 June 2022).
19. Muneer, T.; Younes, S.; Munawwar, S. Discourses on solar radiation modeling. *Renew. Sustain. Energy Rev.* **2007**, *11*, 551–602. [CrossRef]

20. Appelbaum, J.; Massalha, Y.; Aronescu, A. Corrections to anisotropic diffuse radiation model. *Sol. Energy* **2019**, *193*, 523–528. [CrossRef]
21. Perez, R.; Ineichen, P.; Seals, R.; Michalsky, J.; Stewart, R. Modeling daylight availability and irradiance components from direct and global irradiance. *Sol. Energy* **1990**, *44*, 271–289. [CrossRef]
22. Yao, W.; Li, Z.; Zhao, Q.; Lu, Y.; Lu, R. A new anisotropic diffuse radiation model. *Energy Convers. Manag.* **2015**, *95*, 304–313. [CrossRef]
23. Berrizbeitia, S.E.; Gago, E.J.; Muneer, T. Empirical Models for the Estimation of Solar Sky-Diffuse Radiation. A Review and Experimental Analysis. *Energies* **2020**, *13*, 701. [CrossRef]
24. Lave, M.; Hayes, W.; Pohl, A.; Hansen, C.W. Evaluation of Global Horizontal Irradiance to Plane-of-Array Irradiance Models at Locations Across the United States. *IEEE J. Photovolt.* **2015**, *5*, 597–606. [CrossRef]
25. Fan, J.; Wu, L.; Zhang, F.; Cai, H.; Ma, X.; Bai, H. Evaluation and development of empirical models for estimating daily and monthly mean daily diffuse horizontal solar radiation for different climatic regions of China. *Renew. Sustain. Energy Rev.* **2019**, *105*, 168–186. [CrossRef]
26. Zhou, Y.; Wang, D.; Liu, Y.; Liu, J. Diffuse solar radiation models for different climate zones in China: Model evaluation and general model development. *Energy Convers. Manag.* **2019**, *185*, 518–536. [CrossRef]
27. Liu, Y.; Zhou, Y.; Chen, Y.; Wang, D.; Wang, Y.; Zhu, Y. Comparison of support vector machine and copula-based nonlinear quantile regression for estimating the daily diffuse solar radiation: A case study in China. *Renew. Energy* **2020**, *146*, 1101–1112. [CrossRef]
28. Fan, J.; Wu, L.; Ma, X.; Zhou, H.; Zhang, F. Hybrid support vector machines with heuristic algorithms for prediction of daily diffuse solar radiation in air-polluted regions. *Renew. Energy* **2020**, *145*, 2034–2045. [CrossRef]
29. Despotovic, M.; Nedic, V.; Despotovic, D.; Cvetanovic, S. Evaluation of empirical models for predicting monthly mean horizontal diffuse solar radiation. *Renew. Sustain. Energy Rev.* **2016**, *56*, 246–260. [CrossRef]
30. Bamisile, O.; Oluwasanmi, A.; Ejiyi, C.; Yimen, N.; Obiora, S.; Huang, Q. Comparison of machine learning and deep learning algorithms for hourly global/diffuse solar radiation predictions. *Int. J. Energy Res.* **2021**, *46*, 10052–10073. [CrossRef]
31. Liu, P.; Tong, X.; Zhang, J.; Meng, P.; Li, J.; Zhang, J. Estimation of half-hourly diffuse solar radiation over a mixed plantation in north China. *Renew. Energy* **2020**, *149*, 1360–1369. [CrossRef]
32. Engerer, N. Minute resolution estimates of the diffuse fraction of global irradiance for southeastern Australia. *Sol. Energy* **2015**, *116*, 215–237. [CrossRef]
33. Freeman, J.M.; DiOrio, N.A.; Blair, N.J.; Neises, T.W.; Wagner, M.J.; Gilman, P.; Janzou, S. *System Advisor Model (SAM) General Description*; National Renewable Energy Laboratory: Golden, CO, USA, 2018.
34. Government of Canada. RETScreen. Available online: https://www.nrcan.gc.ca/maps-tools-and-publications/tools/modelling-tools/retscreen/7465 (accessed on 13 November 2022).
35. Government of Canada. RETScreen eLearning. 2022. Available online: https://www.youtube.com/channel/UCyFMjG_OXXGtRVnsiTim0IQ (accessed on 13 November 2022).
36. Solargis s.r.o. Solargis Prospect. Available online: https://solargis.com/products/prospect/overview (accessed on 14 November 2022).
37. Solargis s.r.o. Solargis Evaluate. Available online: https://solargis.com/products/evaluate/overview (accessed on 14 November 2022).
38. Solargis s.r.o. Solargis Prospect App. Available online: https://apps.solargis.com/prospect/map (accessed on 14 November 2022).
39. PVsyst SA. PVsyst—Features. Available online: https://www.pvsyst.com/features/ (accessed on 14 November 2022).
40. Laplace System. Laplace System: Solar Pro. Available online: https://www.lapsys.co.jp/english/products/pro.html (accessed on 14 November 2022).
41. Stawowy, M.; Olchowik, W.; Rosiński, A.; Dąbrowski, T. The Analysis and Modelling of the Quality of Information Acquired from Weather Station Sensors. *Remote Sens.* **2021**, *13*, 693. [CrossRef]
42. Ahn, S.J. *Least Squares Orthogonal Distance Fitting of Curves and Surfaces in Space*; Springer: Berlin/Heidelberg, Germany, 2004.
43. Hansen, P.C.; Pereyra, V.; Scherer, G. *Least Squares Data Fitting with Applications*; Johns Hopkins University Press: Baltimore, MD, USA, 2013.
44. Osowski, S.; Siwek, K. Local dynamic integration of ensemble in prediction of time series. *Bull. Pol. Acad. Sci. Tech. Sci.* **2019**, *67*, 517–525.
45. Corriou, J.P. *Numerical Methods and Optimization*; Springer: Cham, Switzerland, 2021.
46. Heiner, Y.; Stier, O.; Türck, V.; Waschull, J.; Sumpf, B.; Ostermeier, A. Evolution strategies applied to least-squares curve fitting of spectroscopic data. *J. Quant. Spectrosc. Radiat. Transf.* **1996**, *56*, 769–782. [CrossRef]
47. Sörensen, K. Metaheuristics-the metaphor exposed. *Int. Trans. Oper. Res.* **2015**, *22*, 3–18. [CrossRef]
48. Cuevas, E.; Zaldivar, D.; Perez-Cisneros, M. *Advances in Metaheuristics Algorithms: Methods and Applications*; Springer: Cham, Switzerland, 2018.
49. Stegherr, H.; Heider, M.; Hähner, J. Classifying Metaheuristics: Towards a unified multi-level classification system. *Nat. Comput.* **2022**, *21*, 155–171. [CrossRef]
50. Birattari, M.; Paquete, L.; Stützle, T.; Varrentrapp, K. *Classification of Metaheuristics and Design of Experiments for the Analysis of Components*; Technical University of Darmstadt: Darmstadt, Germany, 2001.

51. Talbi, E.G. *Metaheuristics. From Design to Implementation*; John Wiley & Sons, Inc.: Hoboken, NJ, USA, 2009.
52. Siarry, P. *Metaheuristics*; Springer: Cham, Switzerland, 2016.
53. Simon, D. Evolutionary Optimization Algorithms. John Wiley & Sons Inc.: Hoboken, NJ, USA, 2013.
54. Rechenberg, I. *Evolutionsstrategie: Optimierung Technischer Systeme nach Prinzipien der Biologischen Evolution*; Frommann-Holzboog: Stuttgart, Germany, 1973.
55. Beyer, H.G. *The Theory of Evolution Strategies*; Springer: Berlin/Heidelberg, Germany, 2001.
56. Luke, S. *Essentials of Metaheuristics*; Lulu: Fairfax, VA, USA, 2013.
57. Eiben, A.E.; Michalewicz, Z.; Hinterding, R. Parameter Setting in Evolutionary Algorithms. *IEEE Trans. Evol. Comput.* **1999**, *3*, 124–141. [CrossRef]
58. Moustakas, C. *Heuristic Research: Design, Methodology, and Applications*; SAGE Publications: London, UK, 1990.
59. Płotka, B. *Efficient Go, Sebastopol*; O'Reilly Media, Inc.: Sebastopol, CA, USA, 2022.
60. Balbaert, I. *The Way To Go*; iUniverse: Bloomington, IN, USA, 2012.
61. Sheehan, L. *Learning Functional Programming in Go*; Packt Publishing: Birmingham, UK, 2017.
62. Google. Go Modules Reference. Available online: https://go.dev/ref/mod (accessed on 1 December 2022).
63. Ritchie, D.M. A Stream Input-Output System. *ATT Bell Lab. Tech. J.* **1984**, *63*, 1897–1910. [CrossRef]