energies

Article

Radiation View Factor for Building Applications: Comparison of Computation Environments

Marzia Alam [1,*], Mehreen Saleem Gul [1] and Tariq Muneer [2]

1 School of Energy, Geoscience, Infrastructure and Society, Heriot-Watt University, Boundary Rd N, Edinburgh EH14 4AS, UK; m.gul@hw.ac.uk

2 School of Engineering and The Built Environment, Edinburgh Napier University, 10 Colinton Rd, Edinburgh EH10 5DT, UK; T.Muneer@napier.ac.uk

* Correspondence: ma410@hw.ac.uk; Tel.: +44-07544930047

Received: 24 September 2019; Accepted: 8 October 2019; Published: 10 October 2019

Abstract: Computation of view factors is required in several building engineering applications where radiative exchange takes place between surfaces such as ground and vertical walls or ground and sloping thermal or photovoltaics collectors. In this paper, view factor computations are performed for bifacial solar photovoltaic (PV) collectors based on the finite element method (FEM) using two programming languages known as Microsoft Excel-Visual Basic for Applications (VBA) and Python. The aim is to determine the computer response time as well as the performance of the two languages in terms of accuracy and convergence of the numerical solution. To run the simulations in Python, an open source just-in-time (JIT) compiler called Numba was used and the same program was also run as a macro in VBA. It was observed that the simulation response time significantly decreased in Python when compared to VBA. This decrease in time was due to the increase in the total number of iterations from 400 million to 250 billion for a given case. Results demonstrated that Python was 71–180 times faster than VBA and, therefore, offers a better programming platform for the view factor analysis and modelling of bifacial solar PV where computation time is a significant modelling challenge.

Keywords: building energy exchange; view factor; Python programming language; bifacial solar photovoltaic (PV)

1. Introduction

Buildings consume a great deal of energy to maintain comfortable conditions within their enclosures. The design of heating and cooling systems for buildings require a detailed assessment of the temperature of the constituting envelope. Thus, the temperature of the walls, roofs, floors, and glazing surfaces are required. Buildings within urban areas inevitably have radiative exchanges with other buildings, sky, and ground. Radiation heat transfer has many applications within the building services sector. Some of the building engineering applications of radiative exchange between surfaces are:

- Ground and vertical walls or windows (Figure 1).
- A light-pipe illuminating the floor or desk spaces (Figure 2).
- Ground and sloping thermal or photovoltaic (PV) collectors.
- Canyon space in densely populated urban areas with radiative exchange between parallel walls and/or windows.
- Radiative exchange between orthogonal walls and/or windows.
- Wet heating system's radiator and indoor surface of the walls.

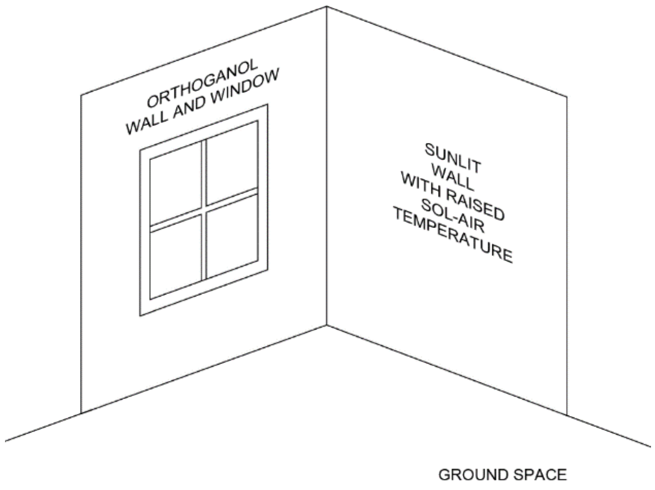

ORTHOGANOL WALL AND WINDOW
SUNLIT WALL WITH RAISED SOL-AIR TEMPERATURE
GROUND SPACE

Figure 1. Ground and vertical walls or windows.

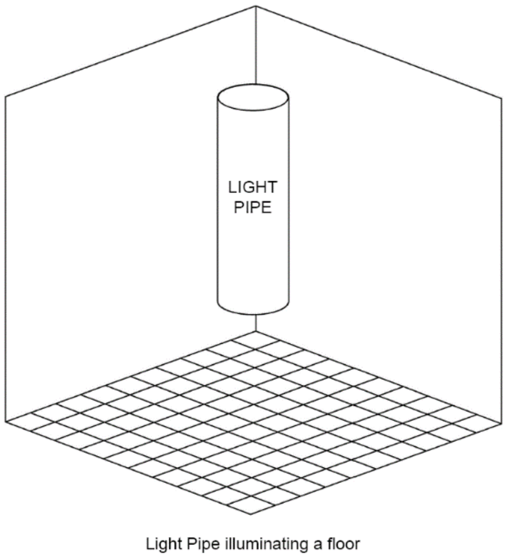


Figure 2. A light-pipe illuminating floor or desk spaces.

In all the above applications' computation of the radiation view factor is required. There have been significant research studies related to energy transfer in building applications such as quantification of available façade areas for installation of building integrated solar PV [1], determination of solar heat gain for daylighting studies [2], theoretical modelling of the view factor to determine the diffused components of solar irradiance [3], and predictive simulation tool to determine the radiance value of a façade in the street canyon [4].

In this article, we have considered the radiative energy exchange of solar photovoltaic collectors such as bifacial solar photovoltaics (PV), which is gaining popularity both for the field level and rooftop building applications [5]. It is a promising technology that increases the production of electricity per square meter of the PV module using light absorption from the albedo [6]. Bifacial solar cells simultaneously collect photons from incident and albedo radiation reaching both the front side and the back side of a solar module whereas traditional monofacial solar cells only collect photons reaching the front side of the device. Cuevas et al. [7] showed that an increase of 50% in electric power generation can be obtained by simultaneously collecting direct and albedo radiation from the rooftop and surroundings around a module. Consequently, it was established that bifacial solar cells can increase the power density of PV modules compared to mono-facial cells while reducing area-related costs for PV systems [8]. Currently, there are two main challenges this technology is facing, which are

(1) a lack of an adequate simulation tool that can help understand how the rear side of the PV interact with the reflected sunlight and (2) how to make the technology bankable. The success of the technology will depend on how well the bifacial PV is understood to the research community. Therefore, it would be easier to make the technology commercially profitable. Hence, the need for developing a simulation tool to understand the energy yield of bifacial solar PV is inevitable. There are different simulation approaches to determine the ground reflected radiation that is incident upon tilted bifacial solar photovoltaics (PV) such as the view factor model, the ray tracing model, and empirical modelling [9]. There has been ongoing research by various scientific communities. For instance, National Renewable Energy Laboratory (NREL) in collaboration with Sandia National Laboratories have developed and compared the view factor model and the ray tracing model to evaluate the back-surface irradiance [10]. However, computation time remain an issue, which is still a modelling challenge that requires further investigation. Conceptually, the back-surface irradiance is composed of direct irradiance, structure reflected irradiance, sky diffused irradiance, and ground reflected irradiance. In this article, we have followed the view factor modelling-based approach to evaluate the radiative energy transfer between the ground (reflecting surface) and the bifacial solar PV (collecting surface).

The present research team had developed a computer code for view factor analysis using a finite-element grid, which can handle an irregular horizon [11]. The software was developed based on a numerical solution of the view factor integral within the Microsoft Excel-Visual Basic for Applications (VBA) environment considering mono-facial solar PV collectors as an example. However, in this work, an improved, simplified brute force algorithm is adopted to determine the view factor for uniform surfaces by utilizing the finite element method (FEM). FEM is one of the most powerful tools for analyzing energy exchange between surfaces. It can be applied for view factor (VF) evaluation. This method is very robust and, yet, it may require excessive computer time. For finite element analysis, the object-oriented computing environment is already proven to be efficient in various areas such as Structural Engineering [12], Chemical Engineering, and Mathematics. In this article, we have implemented the view factor (VF) code in VBA and the Python environment and have compared the computation speed of the simulations in both platforms.

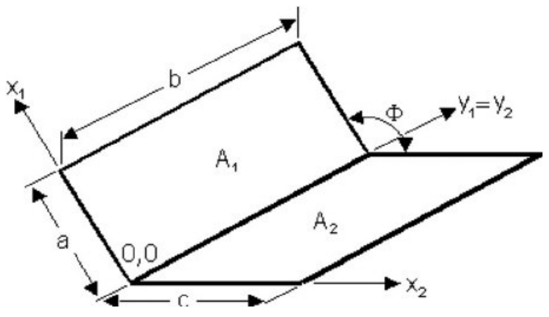
Python is chosen because it has already been accepted by the research community for its ease of use and an open access user-friendly platform. Availability of the open source library has made it eligible for numerous scientific applications such as power systems [13], energy analysis [14] mathematics, and fluid dynamics [15]. VBA has also been deployed as a user-friendly tool in various applications such as in heat transfer, in solar PV applications [16,17], and in the field of agriculture. For example, a milk-producing dairy model was implemented in VBA to control the operation of the dairy farm, which has a significant benefit to the farmers and researchers in the field. However, it takes seven days to run the simulation [18]. Therefore, despite its effectiveness as a design tool, computation time has always been an issue for VBA [19] whereas Python, which is a high-level, benchmark programming language, can outperform VBA in terms of computation speed.

In this paper, we have compared the potentials of the simulation platform to develop the code further for bifacial solar PV collectors. Two simulation platforms are used to quantify the amount of ground reflected radiation received by the solar collectors. We focused on the modelling of computationally intensive view factor analysis between reflecting (ground) and collecting (bifacial solar PV) surface and determined the duration of simulation response time. The paper is organized as follows. The concept of view factor modelling is briefly discussed in Section 2. Section 3 provides the overview of the simulation model. The computation performance of two simulation platforms, Python and Microsoft Excel-Visual Basic for Applications (VBA), is demonstrated in Section 4. We have presented an application related to generation of solar PV electricity and its enhancement by using reflective films placed near the horizon of PV modules. The object is to obtain a numerical procedure for the radiative exchange between the foreground and PV modules. The technique can be applied to a bifacial solar PV cell, which is an emerging technology. The novelty of this work is that we have provided quantitative data that demonstrates faster convergence and accuracy offered by the Python software environment in the context of modelling bifacial solar PV's energy yield where computation time is a significant modelling challenge. Lastly, the paper ends with concluding remarks in Section 5.

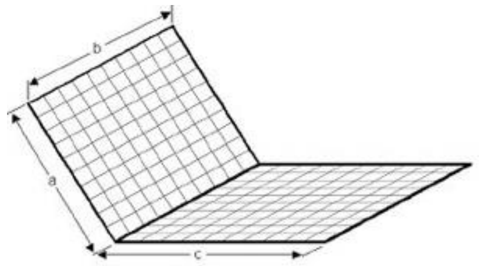
2. View Factor Modelling Concept

In our work, we simulated the mathematical model of the radiation view factor as a discrete model in the computing platform by applying finite element method. To understand the concept of view factor modelling, let us consider two rectangular surfaces A_1 and A_2 with surface dimensions $a \times b$ and $c \times b$ and Φ is the tilt angle between the surfaces (Figure 3a) and $\beta = \pi - \Phi$. In radiative heat transfer, the view factor can be defined as the proportion of irradiance, which leaves the emitting surface A_2 and strikes the receiving/collector surfaces A_1 denoted by F_{A2-A1}. From the mathematical equation of the view factor [11], we may write:

$$F_{A_2-A_1} = \frac{1}{bc}\int_{x_1}^{c}\int_{y_1}^{b}\int_{x_2}^{a}\int_{y_2}^{b} \frac{x_1x_2\, sin^2\beta}{\pi\left[{x_1}^2 + {x_2}^2 + 2x_1x_2\, cos\beta + (y_1 - y_2)^2\right]^2}\, dy_2dx_2dy_1dx_1. \quad (1)$$



(a)



(b)

Figure 3. Radiation view factor analysis: (**a**) Two rectangles with one common edge and (**b**) the reflecting and receiving surface with uniform grids [11].

In the present work, Equation (1) has been used to compute the view factor for different geometries of the receiving (solar collectors) and reflecting (ground) surface, which share a common edge. Both the emitting and receiving surfaces are considered as uniform grids where all the cells within the surface are of the same dimensions and aspect ratios (Figure 3b). The tilt angle between the two surfaces varied from 30°–150°. The height of the receiving and emitting surface is denoted by 'a' and 'c', respectively, and the common edge length is denoted by 'b'.

One of the approaches to examine the solution of the discrete model is to observe the convergence of the computed solution toward the analytical solution (if it exists) of the mathematical model. This approach is known as verification. For our paper, the analytical solution provided by Feingold [20] is considered as the benchmark solution for the purpose of verification. However, any physical model when interpreted by a discrete model, the solution error, or computation error is an important phenomenon that needs to be properly understood. For a uniform grid, the difference between the analytical solution and the computed solution represents the error. In this case, we have set up the finite element procedure to solve the problem repeatedly with uniform meshes designed to determine the view factor and the error.

There are different adaptive finite element techniques available to estimate the error such as local refinement or *h*-refinement, relocating or *r*-refinement, and locally varying the polynomial degree known as *p*-refinement. For the purpose of verification of our model, an *h*-adaptive refinement technique is applied where the *h* denotes the element size or resolution of the grid, which we used to control the error. Decreasing *h* leads to reduction of the error. As *h* approaches zero, the numerical solution converges toward its analytical value. The error can be derived from Equation (2) where error (E_r) is a function of element size h_i. VF_{ex} represents the analytical value of the view factor and VF_{hi} is the simulated value of the view factor at element size h_i. C is an unknown constant with the leading term ${h_i}^{\beta}$ and exponent β is the rate of convergence [21].

$$\lim_{h_{i=1,2,3,4..} \to 0} Er(h_i) = VF_{ex} - VF_{h_i} \approx C{h_i}^{\beta}\ for\ \beta > 0 \tag{2}$$

If the condition $\frac{h_1}{h_2} = \frac{h_2}{h_3}$ holds, Equation (2) can be further derived as:

$$\frac{VF_{ex} - VF_{h1}}{VF_{ex} - VF_{h2}} = \frac{h_1^{\beta}}{h_2^{\beta}}\ and\ \frac{VF_{ex} - VF_{h2}}{VF_{ex} - VF_{h3}} = \frac{h_2^{\beta}}{h_3^{\beta}} \tag{3}$$

The element size determines the number of iterations the simulation must run. The parameters considered for view factor models are presented in Table 1 below.

Table 1. View factor modeling parameters.

Surface Dimensions (m)	Element Size h_i (m)	Φ (°)
a = 2 m, b = 1 m, c = 2 m	0.01	30
a = 1 m, b = 1 m, c = 1 m	0.008	60
a = 0.6 m, b = 1 m, c = 0.4 m	0.004	90
a = 0.4 m, b = 1 m, c = 0.6 m	0.002	120
a = 0.4 m, b = 0.4 m, c = 0.4 m	—	150

3. Overview of the Simulation Model

The numerical view factor model has been executed both in Python and VBA environments. The simulation utilizes one of the most efficient Python libraries for numerically intensive computing named Numba, which can be loaded by a program as the CPython interpreter. The code is written based on Numba just-in-time (JIT) compiler, which creates a specialized loop in the machine code. The just-in-time or @ JIT is a decorator that is utilized as a function. When this function is called, the decorator analyzes the argument and creates a specialized version of the function [22]. This code is run on the 'nopython' mode, which compiles the code without accessing the Python C-API (application program interface). Python version 3.7 with Spyder integrated development environment (IDE) is used for the simulation. On the other hand, VBA adopts the run time library of the Visual Basic, which is compiled in a Microsoft packet code and the MS-Excel works as a host application that saves the code in a separate file such as .xlsx or .xlsm. A machine hosted by Excel run the intermediate code and the data is saved in a text file in XML format, which is readable by the user [23]. The simulations were run on Intel®Core ™ i7-7500U CPU @ 2.7 GHz-2.9 GHz Laptop.

Python and VBA both used a Microsoft Excel worksheet as an input and output file. The user can enter the input parameters, run the simulation, and calculate the simulation response time as well as observe the output, which is saved in an Excel file. Different simulation parameters can be set such as the length and height of the surfaces. For example, referred to Figure 3b, the angle between two surfaces sharing one common edge can be selected and varied for the same element size, h_i. The simulations were run at different resolutions of the element size defined as the iteration step in the program, which is varied as: h_1 = 0.01 m, h_2 = 0.008 m, h_3 = 0.004 m, and h_4 = 0.002 m. This iteration step defines the number of iterations the simulation needs to run, which can be calculated from Equation (4).

$$Iteration\ size = \frac{c}{h_i} \times \frac{b}{h_i} \times \frac{a}{h_i} \times \frac{b}{h_i} \tag{4}$$

Thus, iteration ranges of the simulations vary from 4.00×10^8 to 2.50×10^{11}. In this study, a brute force problem-solving approach is applied within a mathematical equation of the view factor (given by Equation (1) in Section 3), which aims to check all the possible interactions between the surfaces using four nested loops bounded by the surface dimensions [24]. The program flow chart is presented in Figure 4.

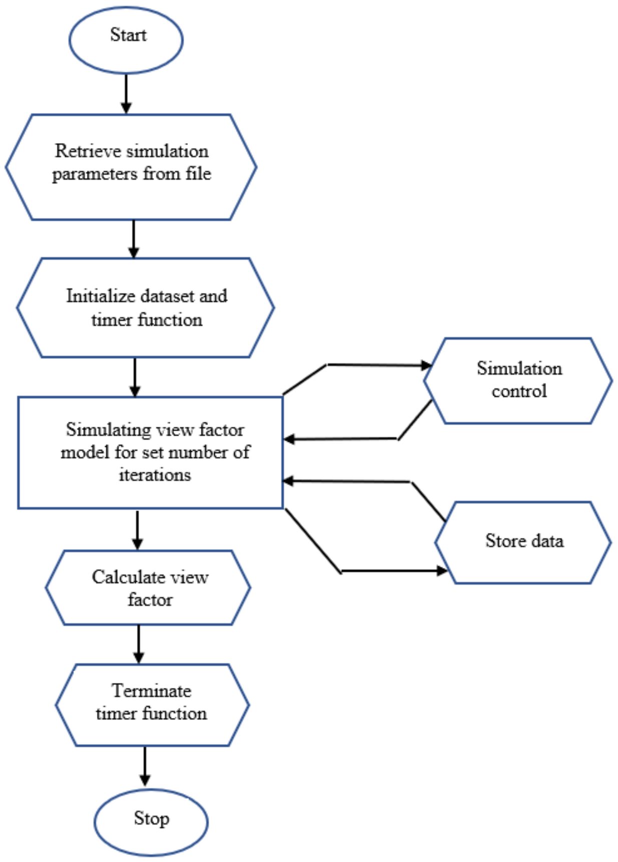

Start
Retrieve simulation parameters from file
Initialize dataset and timer function
Simulating view factor model for set number of iterations
Simulation control
Store data
Calculate view factor
Terminate timer function
Stop

Figure 4. Flow chart of the simulation model.

4. Results and Discussion

The view factor simulation outputs are examined from different approaches, as explained below.

- The simulated view factor output is tested at different computation angles between collectors and reflective surfaces to verify the numerical model with the existing analytical solution.
- The reduction in the percentage (%) error of the view factor output is evaluated with an increasing number of iterations. An accuracy versus computation time dependency is shown in Section 4.2.
- Followed by Section 4.2, the convergence rate of the simulated output to the actual output is examined to determine the type of convergence (linear/quadratic/cubic).
- Next, the computation time versus the number of iterations are tested for specific surface dimensions to compare the improvement of the simulation response time in Python over VBA.
- Lastly, computation time variations with different surface dimensions are determined. Detailed results and analysis are discussed in the following sections. All simulations parameters and outputs are referred to Appendix A.

4.1. View Factor at Different Computation Angles Between Two Surfaces

This section attempts to verify the proposed view factor model at different tilt angles between the surfaces. The simulated results show a minor deviation from the existing analytical solutions. It has been observed that view factors are considerably influenced by the angle between the surfaces, as illustrated in Figure 5. Keeping the dimensions and iteration size constant, as the angle between two surface increases, a significant decrease in the view factor from a 0.51 maximum value (for this study) to the minimum value of 0.015 is observed. Parameters for this analysis are presented in Table 2 below.

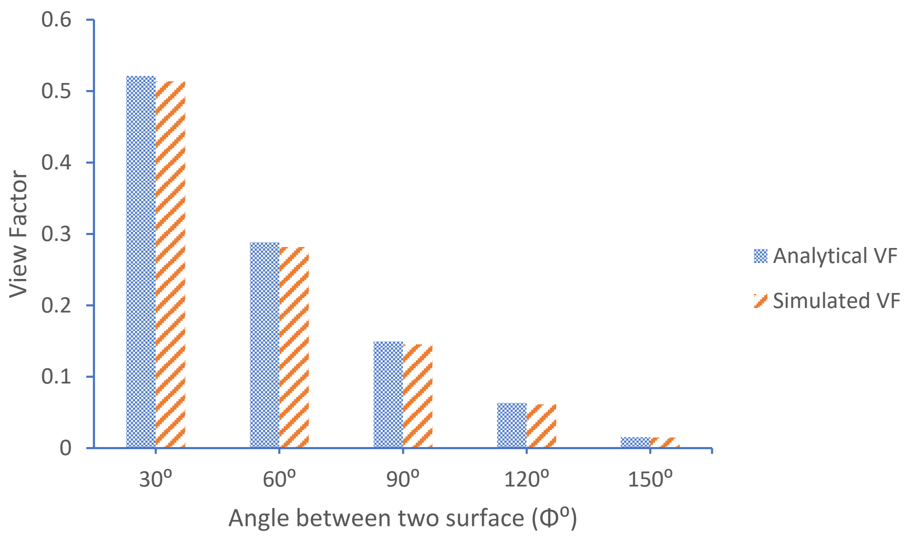


Figure 5. View factor at different angles between surfaces for a = 2 m, b = 1 m, and c = 1 m, and h_i = 0.01 m.

Table 2. View factor simulation parameters for element size h_i = 0.01 m and number of iterations = 4.0×10^8.

a (m)	b (m)	c (m)	Element Size h_i (m)	Φ (°)	Analytical VF	Simulated VF	Error (%)
2	1	2	0.01	30	0.521308	0.513849	1.4
2	1	2	0.01	60	0.288274	0.281650	2.3
2	1	2	0.01	90	0.149300	0.145292	2.7
2	1	2	0.01	120	0.063248	0.061372	3.0
2	1	2	0.01	150	0.015415	0.014937	3.1

Table 2 illustrates that, for element size h_i = 0.01 m, the percentage error varies from a minimum 1.4% to a maximum 3.1% depending on the value of angle Φ. The iteration size for this simulation is set to 4.0×10^8. The next section presents, how this error can be further reduced by decreasing the element size h_i, which, in turn, increases the number of iterations.

4.2. Accuracy Versus the Number of Iterations

The accuracy of the view factor output significantly increased with the number of iterations the simulation runs, which can be seen in Figure 6. Both Python and VBA achieved the same level of accuracy. For surface dimensions: a = 2 m, b = 1 m, c = 2 m, and Φ = 120°, the computation accuracy is 97% with 4.0×10^8 iterations. The numerical results approached the analytical solution at a maximum accuracy of 99.4% for 2.5×10^{11} iterations. To gain this, the iteration size is to be increased by a factor of 625. The simulation parameters for this study are outlined in Table 3 below.

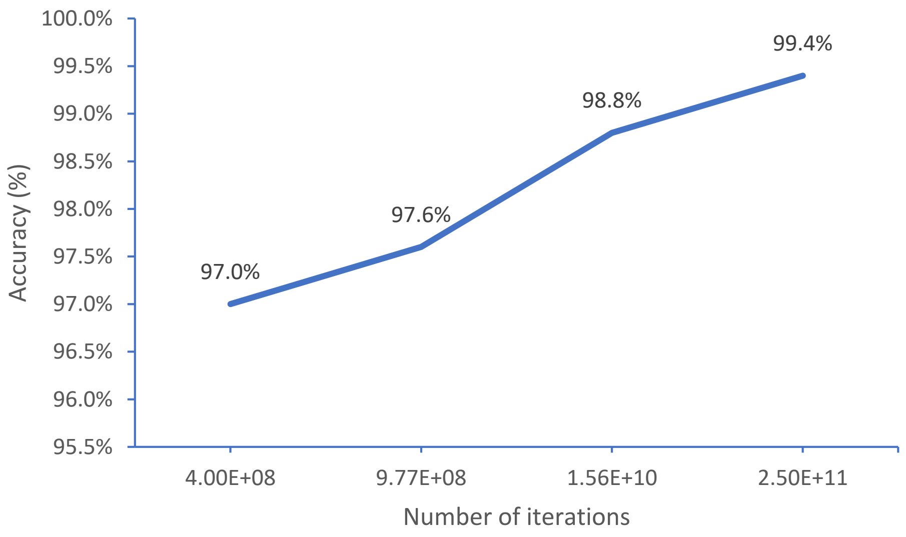


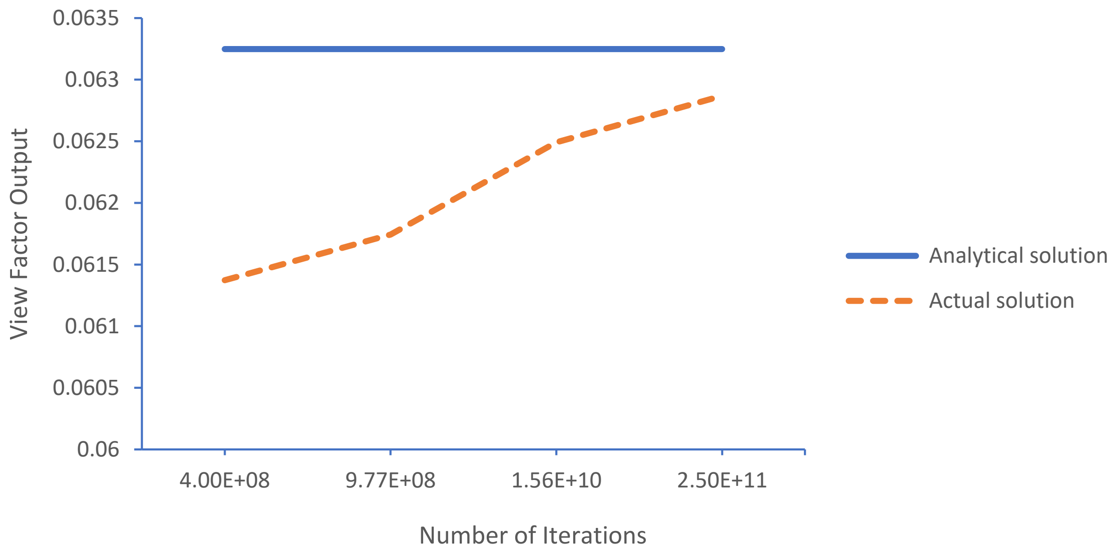
Figure 6. Accuracy versus the number of iterations for a = 2 m, b = 1 m and c = 2 m and Φ = 120°.

Table 3. View factor simulation parameters at various element sizes.

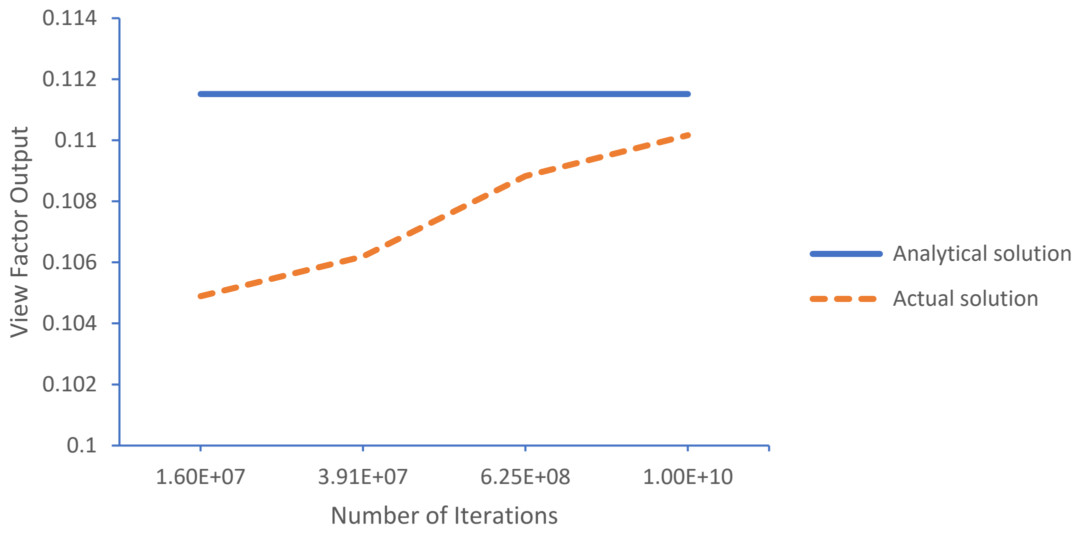
a (m)	b (m)	c (m)	Φ (°)	Element Size, h_i (m)	Analytical VF	Simulated VF	Error (%)
2	1	2	120	0.01	0.063248	0.061372	3.0
2	1	2	120	0.008	0.063248	0.061743	2.4
2	1	2	120	0.004	0.063248	0.062491	1.2
2	1	2	120	0.002	0.063248	0.062868	0.59

4.3. Convergence to an Analytical Solution

The convergence performance of the view factor simulation to the analytical solutions are studied by comparing the output between the highest: a = 2 m, b = 1 m, c = 2 m, Φ = 120° and the lowest: a = 0.4 m, b = 1 m and c = 0.4 m, Φ = 120° surface dimensions. For the same number of iterations, maximum accuracy achieved for Figure 7a was up to 99.4% whereas, for Figure 7b, the accuracy was 98.79%. Comparing both simulation responses, it can be inferred that, to reach the same level of convergence, the element size needs to be further reduced for surface dimensions: a = 0.4 m, b = 1 m, and c = 0.4 m; Φ = 120°. In addition, the convergence type of the model is found to be linear with the β value of approximately 1. The rate of convergence is derived from Equation (3) considering the element size of 0.004 m and 0.002 m. Table 4 summarizes the results.



(**a**)



(**b**)

Figure 7. (**a**). Convergence to an analytical solution for a = 2 m, b = 1 m, c = 2 m, and Φ = 120°. (**b**). Convergence to an analytical solution for a = 0.4 m, b = 1 m, c = 0.4 m, and Φ = 120°.

Table 4. Convergence of the view factor output.

Surface Dimension (m)	Analytical VF	Simulated VF h_i = 0.004 m	Simulated VF h_i = 0.002 m	Convergence Rate (β)
a = 0.4, b = 1, c=0.4	0.111512	0.108828	0.110164	0.993564
a = 0.4, b = 1, c = 0.6	0.085512	0.083668	0.084586	0.993755
a = 0.6, b = 1, c = 0.4	0.128269	0.125502	0.126879	0.993238
a = 1, b = 1, c = 1	0.086615	0.085348	0.085979	0.994318
a = 2, b = 1, c = 2	0.063248	0.062491	0.062868	0.994294

4.4. Computation Time versus Number of Iterations

The preceding sections have shown, although both Python and VBA gained the same accuracy with an increasing number of iterations, VBA simulation took more time than Python to reach the same level of accuracy. Figure 8 presents the computation time in seconds for both programs. We see that Python took 3.52 s to run 4.0×10^8 iterations whereas VBA required 454 s. For all other iterations, Python outperformed VBA in terms of computation speed, and it was about 129–180 times faster than VBA. Simulation parameters for these computations can be found in Table 5.

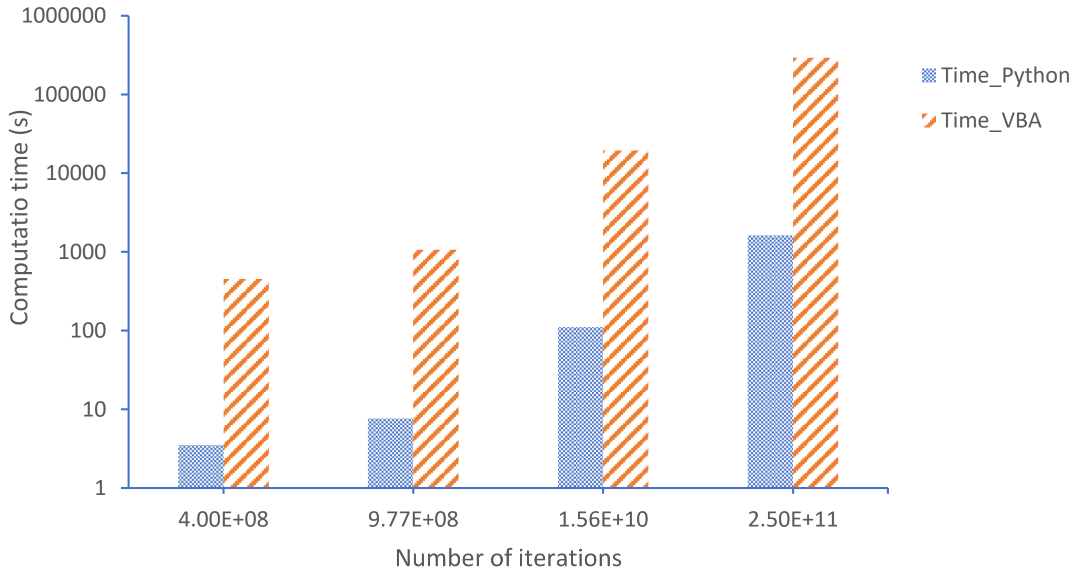


Figure 8. Computation time versus the number of iterations for a = 2 m, b = 1 m, and c = 1 m and Φ = 120°.

Table 5. Simulation parameters for computation time at different element sizes.

a (m)	b (m)	c (m)	Φ (°)	Element Size h_i (m)
2	1	2	120	0.01
2	1	2	120	0.008
2	1	2	120	0.004
2	1	2	120	0.002

4.5. Computation Time for Different Surface Dimensions

The view factor simulation response time increased with the increasing length and height of surface dimensions. Table 6 summarizes results of computation time variations among different surface dimensions. The results show, in VBA, the lowest computation time is 24 s for the surface dimensions: a = 0.4 m, b = 1 m, and c = 0.4 m. This computation time increased by a factor of 19 for the maximum dimension considered in the simulation: a = 2 m, b = 1 m, and c = 2 m. Nevertheless, Python appears to be 71–129 times faster than VBA at a varying surface length and height.

Table 6. Computation time at different surface lengths and heights for Φ = 120° and h_i = 0.01 m.

Surface Dimension (m)	Element Size (m)	Analytical VF	Simulated VF	Time_Python (s)	Time_VBA (s)
a = 0.4, b = 1, c = 0.4	0.01	0.111512	0.104891	0.34	24
a = 0.4, b = 1, c = 0.6	0.01	0.085512	0.080957	0.41	43
a = 0.6, b = 1, c = 0.4	0.01	0.128269	0.121436	0.45	43
a = 1, b = 1, c = 1	0.01	0.087615	0.083481	0.92	222
a = 2, b = 1, c = 2	0.01	0.063248	0.061372	3.52	454

A finite element computation of the view factor is a time critical application. The present view factor model offers a quantitative computational advantage, i.e., grid surface non-uniformity can be handled quite easily. Other advantages offered by this work are: (a) the view factor code is much faster for geometries encountered in most solar energy and building energy exchange applications, and (b) the view factor approach allows for much shorter computation time, particularly if handled in the Python environment. Note that implementation of other alternative method for instance, ray tracing as a modelling tool is more complex compared to the view factor concept.

The computing performance not only depends on the type of computer used but also on the simulation algorithm and its implementation platform, i.e., programming language. Though short response time is one of the benchmark criteria of the performance efficiency, there are additional performance matrixes, which can be achieved with fast computation speed. These include high throughput, minimum utilization of resources [25], higher reliability [26], low power consumption, scalability, and opportunity of performance tuning. For our study, one downside of VBA was that it took about five consecutive days to run 250 billion iterations, which can have substantial damage on computer health. Moreover, utilization of computer resources has been very high, which required lots of power consumption. In addition to this, the scope of performance tuning in VBA is also limited, which reduces the scalability of the number of iterations the simulation can run. In all these aspects, Python can outperform VBA, which make it a suitable option for present day scientific and numeric computing. The existing work demonstrates the benefits of using Python for view factor analysis for solar PV applications, which is an improvement over our previous work where we used VBA for such an analysis.

5. Conclusions

In this paper, we have computed the value of the radiation view factor to determine the reflected solar irradiance reaching the rear side of the bifacial solar PV. We have verified the results with the existing analytical solutions. In this scenario, we focus on the computing performance to examine the improvement in computation speed of Python as compared to VBA. It has been shown that, Python can be used more effectively than VBA for radiation view factor analysis between two surfaces. With the utilization of an appropriate mathematical library, computation time was significantly reduced by 71–180 times for Python when compared with VBA. This improvement in computation speed not only saves time, but also provides an optimized design tool for the research. An important finding of the view factor simulation is that, as the element size of the finite element grid decreased, the computed output converged to the analytical view factor value. Thus, the simulation accuracy could be achieved up to 99.4% for the maximum number of iterations considered for this paper, i.e., 250 billion and the response time of the simulation in Python and VBA was 1628.51 s and 292,714 s, respectively. The application presently considered in this article are for relatively small areas of the reflecting and the receiving surfaces. In an actual industrial environment where the designer will deal with multi-gigawatt solar PV farms, that may employ enhanced reflections near the horizon. In that case, to model such large-scale systems, the number of iterations the computer simulations have to run will increase to the order of a few quadrillion or more. Therefore, the importance of faster code written in a computation environment such as Python will be of great benefit to the PV system designers. Hence, it is concluded that, Python can be utilized as a reliable simulation tool to develop the code further for bifacial solar PV research.

Author Contributions: Conceptualization: T.M., M.S.G., and M.A. Methodology: M.A., T.M., and M.S.G. Formal analysis: M.S.G., T.M., and M.A. Coding: T.M., M.A. Resources: T.M., M.S.G., and M.A. Data curation: M.A., T.M., and M.S.G. Writing—original draft preparation: M.A., M.S.G., and T.M. Writing—review and editing: T.M., M.S.G., and M.A. Supervision: M.S.G. and T.M.

Funding: This project is being run in collaboration with four partners: Energy Technology Partnership (ETP-Project 161) Scotland, Wood Group-Clean Energy, Heriot-Watt University and Edinburgh Napier University, UK.

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

Appendix A View Factor Calculation Tables at Different Element Sizes

Table A1. View factor simulation data for element size h_i = 0.01 m.

a (m)	b (m)	c (m)	Φ (°)	Iteration	Analytical VF	Simulated VF	Error (%)	Time_VBA (s)	Time_Python (s)
2	1	2	30	4.0×10^8	0.521308	0.513849	1.4	415	2.77
2	1	2	60	4.0×10^8	0.288274	0.281650	2.3	410	2.75
2	1	2	90	4.0×10^8	0.149300	0.145292	2.7	408	2.87
2	1	2	120	4.0×10^8	0.063248	0.061372	3.0	454	2.74
2	1	2	150	4.0×10^8	0.015415	0.014937	3.1	410	2.80
1	1	1	90	1.0×10^8	0.200044	0.193529	3.3	143	0.87
0.4	1	0.4	120	1.6×10^7	0.111512	0.104891	5.9	24	0.34
1	1	1	120	1.0×10^8	0.087615	0.083481	3.6	222	0.92
0.6	1	0.4	120	2.4×10^7	0.128269	0.121436	5.3	43	0.45
0.4	1	0.6	120	2.4×10^7	0.085512	0.080957	5.3	43	0.41
0.4	1	0.6	30	2.4×10^7	0.518407	0.508234	2.0	44	0.38
0.6	1	0.4	30	2.4×10^7	0.777610	0.762351	2.0	35	0.39

Table A2. View factor simulation data for element size h_i = 0.008 m.

a (m)	b (m)	c (m)	Φ (°)	Iteration	Analytical VF	Simulated VF	Error (%)	Time_VBA (s)	Time_Python (s)
2	1	2	30	9.76×10^8	0.521308	0.515332	1.1	779	7.98
2	1	2	60	9.76×10^8	0.288274	0.282965	1.8	780	7.35
2	1	2	90	9.76×10^8	0.149300	0.146086	2.2	891	7.67
2	1	2	120	9.76×10^8	0.063248	0.061743	2.4	773	7.35
2	1	2	150	9.76×10^8	0.015415	0.015026	2.5	790	7.40
1	1	1	90	2.44×10^8	0.200044	0.194818	2.6	428	1.98
0.4	1	0.4	120	3.96×10^7	0.111512	0.106192	4.8	55	0.48
1	1	1	120	2.44×10^8	0.087615	0.084099	2.9	429	1.95
0.6	1	0.4	120	5.86×10^7	0.128269	0.122781	4.3	90	0.61
0.4	1	0.6	120	5.86×10^7	0.085512	0.081854	4.3	91	0.61
0.4	1	0.6	30	5.86×10^7	0.518407	0.510242	1.6	92	0.60
0.6	1	0.4	30	5.86×10^7	0.777610	0.765363	1.6	77	1.17

Table A3. View factor for element size h_i = 0.004 m.

a (m)	b (m)	c (m)	Φ (°)	Iteration	Analytical VF	Simulated VF	Error (%)	Time_VBA (s)	Time_Python (s)
2	1	2	30	1.56×10^{10}	0.521308	0.518310	0.58	18971	111.14
2	1	2	60	1.56×10^{10}	0.288274	0.285609	0.92	17010	110.85
2	1	2	90	1.56×10^{10}	0.149300	0.147685	1.08	14997	110.8
2	1	2	120	1.56×10^{10}	0.063248	0.062491	1.20	19407	110.78
2	1	2	150	1.56×10^{10}	0.015415	0.015219	1.27	17493	110.79
1	1	1	90	3.90×10^9	0.200044	0.197415	1.31	7180	27.74
0.4	1	0.4	120	6.25×10^8	0.111512	0.108828	2.41	1120	4.55
1	1	1	120	3.90×10^9	0.087615	0.085348	1.46	4649	27.74
0.6	1	0.4	120	9.37×10^8	0.128269	0.125502	2.16	737	6.66
0.4	1	0.6	120	9.37×10^8	0.085512	0.083668	2.16	738	6.71
0.4	1	0.6	30	9.37×10^8	0.518407	0.514296	0.79	751	6.65
0.6	1	0.4	30	9.37×10^8	0.777610	0.771444	0.79	818	6.71

Table A4. View factor for element size h_i = 0.002 m.

a (m)	b (m)	c (m)	Φ (°)	Iteration	Analytical VF	Simulated VF	Error (%)	Time_VBA (s)	Time_Python (s)
2	1	2	30	2.50×10^{11}	0.521308	0.519806	0.28	322099	1732.83
2	1	2	60	2.50×10^{11}	0.288274	0.286939	0.46	257484	1726.24
2	1	2	90	2.50×10^{11}	0.149300	0.148490	0.54	270788	1632.94
2	1	2	120	2.50×10^{11}	0.063248	0.062868	0.59	292714	1628.51
2	1	2	150	2.50×10^{11}	0.015415	0.015317	0.63	309316	1627.28
1	1	1	90	6.25×10^{10}	0.200044	0.198725	0.65	74904	405.54
0.4	1	0.4	120	1.10×10^{10}	0.111512	0.110164	1.20	23990	65.5
1	1	1	120	6.25×10^{10}	0.087615	0.085979	0.73	78338	407.8
0.6	1	0.4	120	1.50×10^{10}	0.128269	0.126879	1.08	73302	103.16
0.4	1	0.6	120	1.50×10^{10}	0.085512	0.084586	1.082	70983	102.52
0.4	1	0.6	30	1.50×10^{10}	0.518407	0.516343	0.39	45392	103.33
0.6	1	0.4	30	1.50×10^{10}	0.777610	0.774515	0.39	31975	103.42

References

1. Costanzo, V.; Yao, R.; Essah, E.; Shao, L.; Shahrestani, M.; Oliveira, A.C.; Araz, M.; Hepbasli, A.; Biyik, E. A Method of Strategic Evaluation of Energy Performance of Building Integrated Photovoltaic in the Urban Context. *J. Clean. Prod.* **2018**, *184*, 82–91. [CrossRef]
2. Zheng, C.; Wu, P.; Costanzo, V.; Wang, Y.; Yang, X. Establishment and Verification of Solar Radiation Calculation Model of Glass Daylighting Roof in Hot Summer and Warm Winter Zone in China. *Procedia Eng.* **2017**, *205*, 2903–2909. [CrossRef]
3. Maor, T.; Appelbaum, J. View Factors of Photovoltaic Collector Systems. *Sol. Energy* **2012**, *86*, 1701–1708. [CrossRef]
4. Rubio-Bellido, C.; Pulido-Arcas, J.A.; Sánchez-Montañés, B. A Simplified Simulation Model for Predicting Radiative Transfer in Long Street Canyons under High Solar Radiation Conditions. *Energies* **2015**, *8*, 13540–13558. [CrossRef]
5. Cha, H.L.; Bhang, B.G.; Park, S.Y.; Choi, J.H.; Ahn, H.K. Power Prediction of Bifacial Si PV Module with Different Reflection Conditions on Rooftop. *Appl. Sci.* **2018**, *8*, 1752. [CrossRef]
6. Guerrero-Lemus, R.; Vega, R.; Kim, T.; Kimm, A.; Shephard, L.E. Bifacial Solar Photovoltaics—A Technology Review. *Renew. Sustain. Energy Rev.* **2016**, *60*, 1533–1549. [CrossRef]
7. Cuevas, A.; Luque, A.; Eguren, J.; del Alamo, J. 50 Per Cent More Output Power from an Albedo-Collecting Flat Panel Using Bifacial Solar Cells. *Sol. Energy* **1982**, *29*, 419–420. [CrossRef]
8. Kreinin, L.; Bordin, N.; Karsenty, A.; Drori, A.; Eisenberg, N. Experimental Analysis of the Increases in Energy Generation of Bifacial over Mono-Facial PV Modules. In Proceedings of the 26th Europe Photovoltaic Solar Energy Conference Exhibition, Hambourg, Genmany, 5–9 September 2011; pp. 3140–3143.
9. Shoukry, I.; Berrian, D.; Libal, J.; Haffner, F. Simulation Models for Energy Yield Prediction of Bifacial Systems. In *Bifacial Photovoltaics: Technology, Applications and Economics*; Institution of Engineering and Technology (IET): Stevenage, UK, 2018. [CrossRef]
10. Hansen, C.W.; Stein, J.S.; Deline, C.; Macalpine, S.; Marion, B.; Asgharzadeh, A.; Toor, F. Analysis of Irradiance Models for Bifacial PV Modules. In Proceedings of the 2017 IEEE 44th Photovoltaic Specialist Conference, PVSC, Washington, DC, USA, 25–30 June 2017. [CrossRef]
11. Muneer, T.; Ivanova, S.; Kotak, Y.; Gul, M. Finite-Element View-Factor Computations for Radiant Energy Exchanges. *J. Renew. Sustain. Energy* **2015**, *7*, 033108. [CrossRef]
12. Rypl, D.; Patzák, B. From the Finite Element Analysis to the Isogeometric Analysis in an Object-Oriented Computing Environment. *Adv. Eng. Softw.* **2012**, *44*, 116–125. [CrossRef]
13. Milano, F. A Python-Based Software Tool for Power System Analysis. In Proceedings of the IEEE Power and Energy Society General Meeting, Vancouver, BC, Canada, 21–25 July 2013. [CrossRef]
14. Zoder, M.; Balke, J.; Hofmann, M.; Tsatsaronis, G. Simulation and Exergy Analysis of Energy Conversion Processes Using a Free and Open-Source Framework—Python-Based Object-Oriented Programming for Gas- and Steam Turbine Cycles. *Energies* **2018**, *11*, 2609. [CrossRef]

15. Vincent, P.; Witherden, F.D.; Farrington, A.M.; Ntemos, G.; Vermeire, B.C.; Park, J.S.; Iyer, A.S. PyFR: Next-Generation High-Order Computational Fluid Dynamics on Many-Core Hardware (Invited). In Proceedings of the 22nd AIAA Computational Fluid Dynamics Conference, Dallas, TX, USA, 22–26 June 2015. [CrossRef]
16. Richardson, I.; Thomson, M. Integrated Simulation of Photovoltaic Micro-Generation and Domestic Electricity Demand: A One-Minute Resolution Open-Source Model. *Proc. Inst. Mech. Eng. Part A J. Power Energy* **2013**. [CrossRef]
17. Lo Brano, V.; Orioli, A.; Ciulla, G.; Di Gangi, A. An Improved Five-Parameter Model for Photovoltaic Modules. *Sol. Energy Mater. Sol. Cells* **2010**, *94*, 1358–1370. [CrossRef]
18. Ahmadi, A.; Robinson, P.H.; Elizondo, F.; Chilibroste, P. Implementation of CTR Dairy Model Using the Visual Basic for Application Language of Microsoft Excel. *Int. J. Agric. Environ. Inf. Syst.* **2018**, *9*, 74–86. [CrossRef]
19. Oxley, R.L.; Mays, L.W. Optimization-Simulation Model for Detention Basin System Design. *Water Resour. Manag.* **2014**, *28*, 1157–1171. [CrossRef]
20. Feingold, A. Radiant-Interchange Configuration Factors Between Various Selected Plane Surfaces. *Proc. R. Soc. Lond. A Math. Phys. Eng. Sci.* **1966**, *292*, 51–60.
21. Krysl, P. *Finite Element Modelling with Abaqus and Python for Thermal and Stress Analysis*, 1st ed.; Pressure Cooker Press: San Diego, CA, USA, 2017; pp. 189–198.
22. Lam, S.K.; Pitrou, A.; Seibert, S. Numba: A LLVM-Based Python JIT Compiler. In Proceedings of the Second Workshop on the LLVM Compiler Infrastructure in HPC-LLVM'15, Austin, TX, USA, 15 November 2015. [CrossRef]
23. Alexander, M.; Kusleika, D. *Excel® 2019 Power Programming with VBA*; Wiley: Hoboken, NJ, USA, 2019. [CrossRef]
24. Parlier, G.; Guéguen, H.; Hu, F. Smart Brute-Force Approach for Distribution Feeder Reconfiguration Problem. *Electr. Power Syst. Res.* **2019**, *174*. [CrossRef]
25. Mangan, T. *White paper on Perceived Performance–Tuning a System for What Really Matters*; T Murgent Technologies: Canton, MA, USA, 2003.
26. Lilja, D.J. *Measuring Computer Performance-A Practitioners Guide*, 1st ed.; Cambridge University Press: Cambridge, UK, 2008.