Surrogate Functions and Probabilistic Bisection for Solar Panel Orientation

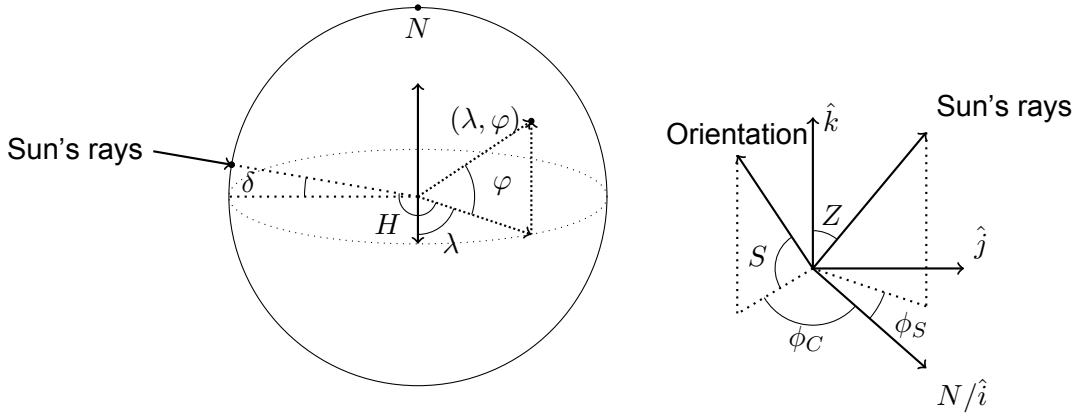
Charlie Matthew Davenport

27th August 2026

Contents

1 Introduction

Solar power is an increasingly important contributor to global energy needs in the era of global warming. This is a rapidly expanding sector. In 2023 China had installed over 570 GW in solar capacity (Kishore, Kumar and Ippili 2025). Additionally, in the UK the planning process for solar panel installation has been streamlined to allow for easier domestic installation of photovoltaic arrays (Energy Security & Net Zero 2025). However, there are still only limited facilities for modelling photovoltaic arrays. A popular library exists on Python for modelling photovoltaic arrays called PVLib (Holmgren et al. 2024). However, PVLib does not have any function to determine annual power output, and Python is much slower than other languages, such as C or Julia (Krishna et al. 2024).



(a) Global View (b) Local Axes

Figure 2: Diagram of angles used in derivation.

To model a solar cell for a given coordinate (λ, φ) on Earth, we consider the position of the sun (ϕ_S, Z) at a given time, as well as the orientation of the solar panel (ϕ_C, S). This is where λ is latitude, φ is longitude, ϕ_S is solar azimuth (measured from North), Z is solar zenith angle, ϕ_C is panel azimuth and S is panel tilt.

This is used with weather data to predict total irradiance on the solar panel. The reflection losses due to the panel are then modelled, and power generated by the panel is calculated.

This is an expensive function to compute- sampling weather data introduces random noise. This makes use of optimisation algorithms prohibitive. When designing a solar array for a suburban house to generate annual energy above a certain threshold (e.g. annual energy consumption of 3 people), minimum panel tilt angle is useful to determine viability of the array. This is made more difficult by noise in the data, which introduces uncertainty.

We propose methods to decrease computational cost of annual energy generation of solar panels, and compare different optimisation algorithms to determine the best (ϕ_C, S) for a fixed (λ, φ). We also explore a method for determining minimum panel tilt angle for fixed cell azimuth.

2 Background

2.1 Derivation of Annual Energy Generation

D. Yang and Kleissl (2024) laid out a framework for an 12-stage function determining power output of a solar cell (Fig. 11.1 in D. Yang and Kleissl (2024)). For this function, it was decided to only use the first 6 stages of this framework. In this section, we outline the equations and theory underlying these stages to derive an objective function.

2.1.1 Solar Position Algorithm

Solar declination angle δ is the angle between the line parallel to the sun's rays and the plane defined by the equator (Figure 2).
Basic Solar Position Algorithms (2026) wrote a formula for solar declination angle δ in degrees as:

$$\delta = 23.45 \sin 2\pi \frac{284 + n}{365} \tag{1}$$

Where n is the number of days through the year ($n \in \mathbb{Z}^+$).
We can defined 2 planes. The meridian plane is defined by the direction of the sun's rays and the poles of the Earth. The hour circle is the plane defined by the position of the observer and the poles of the earth. The hour angle H is the angle between these planes (Editors n.d.).
Solar time, T_{sol}, starts from 0 at solar noon (the highest point of the Sun in the sky), and increases to 1 until the next solar noon. T_{sol} does not change at a consistent rate throughout the year due to the seasons. The variation between T_{sol} with no seasonal variation and the apparent T_{sol} is expressed in the equation of time $E_{qt}(n)$.
A formula for H was derived by *Basic Solar Position Algorithms* (2026) as follows:

$$E_{qt}(n) = \begin{cases} -14.2 \sin \frac{\pi(n+7)}{111} & 1 \leq n \leq 106 \\ 4 \sin \frac{\pi(n-106)}{59} & 107 \leq n \leq 166 \\ 6.5 \sin \frac{\pi(n-166)}{80} & 167 \leq n \leq 246 \\ 16.4 \sin \frac{\pi(n-247)}{113} & 247 \leq n \leq 365 \end{cases} \tag{2}$$

$$T_{sol} = T_{local} + \frac{E_{qt}(n)}{60} + \frac{\lambda_{sm} - \lambda_{loc}}{15} \tag{3}$$

$$H = \pi \frac{12 - T_{sol}}{12} \tag{4}$$

Where λ_{sm} is the longitude of the local standard meridian, and λ_{loc} is the local longitude.
Michalsky (1988) found these formulae for solar elevation angle (α) and azimuth angle ϕ_S, increasing from 0 at North in the eastward direction.

$$\sin \alpha = \cos \lambda \cos \delta \cos H + \sin L \sin \delta \tag{5}$$

$$\sin \phi_S = -\frac{\cos \delta \sin H}{\cos \alpha} \tag{6}$$

Solar zenith angle, Z, is the angle between the observer's vertical and the solar elevation angle. $Z = \frac{\pi}{2} - \alpha$. It is then possible to define the incidence angle θ on the solar array, given the solar panel tilt angle S and the cell azimuth angle ϕ_C.

$$\cos \theta = \cos S \cos Z + \sin S \sin Z \cos \phi_S - \phi_C \tag{7}$$

2.1.2 Annual Energy Modelling

3 quantities are now defined: G_h, B_h and D_h that are related to each other as follows.
Global horizontal irradiance G_h [Wm^{-2}] is the total irradiance onto the horizontal surface. Beam irradiance B_h [Wm^{-2}] is the irradiance from the sun directly. Diffuse irradiance D_h [Wm^{-2}] is the scattered irradiance (e.g. from cloud cover or Rayleigh scattering). The relationship is given below:

$$G_h = B_h \cos \theta + D_h \tag{8}$$

A diagram showing more clearly how these relate to each other is displayed in Figure 3.
These 3 quantities can then be used to determine the Global Tilted Irradiation G_c using the transposition equation (D. Yang and Kleissl 2024). This transposes the irradiance on the horizontal surface onto the solar array.

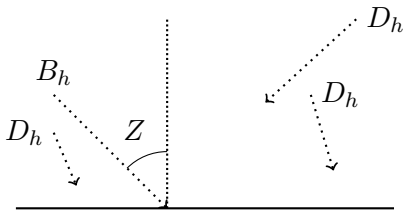


Figure 3: Diagram to show B_h and D_h on a horizontal surface.

$$G_c = B_h \cos\theta + R_d D_h + \rho_g R_r G_h \tag{9}$$

Transposition coefficients calculate the difference between irradiance on the flat ground, and on the tilted solar panel. R_r is the transposition coefficient for the ground's reflection, and R_d is the transposition coefficient for diffuse irradiance.
If we assume an isotropic sky (uniform irradiance across the sky), we can define R_r and R_d as per Gueymard (2009) and D. Yang and Kleissl (2024):

$$R_r = \frac{1 - \cos S}{2} \tag{10}$$

$$R_d = \frac{1 + \cos S}{2} \tag{11}$$

The absorbed irradiance G_c' can be found, given that $B_c = B_h \cos\theta$ and $D_c = R_d D_h$, where ρ_g is local albedo.

$$G_c' = \tau_b B_c + \tau_d D_c + \tau_g \rho_g R_r G_h \tag{12}$$

While it is possible to model the reflection losses from a solar panel using the Fresnel equations, integrating these across a whole solar array has historically proven difficult (King, Kratochvil and Boyson 2004) (Martin and Ruiz 2001). Hence, a more general approach has been pursued, involving relative transmittance of front material of the solar panel. There are distinct values of relative transmittance for B_h, D_h and G_h. These are given as τ_b, τ_d and τ_d.
These are expressed as functions of S, a_r, c_1 and c_2, which are given in Martin and Ruiz (2001). $c_1 = \frac{4}{3\pi}$, and for an air-glass interaction, $a_r = 0.173$ and $c_2 = -0.0675$.
We can use the equations from Martin and Ruiz (2001) to determine τ_b, τ_d and τ_g.

$$\tau_b = 1 - \left\{1 - \left[\frac{1 - \exp -\frac{\cos\theta}{a_r}}{1 - \exp -\frac{1}{a_r}}\right]\right\} \tag{13}$$

$$\tau_d = 1 - \exp\left\{-\frac{1}{a_r}\left[c_1\left(\sin S + \frac{\pi - S - \sin S}{1 + \cos S}\right) + c_2\left(\sin S + \frac{\pi - S - \sin S}{1 + \cos S}\right)^2\right]\right\} \tag{14}$$

$$\tau_g = 1 - \exp\left\{-\frac{1}{a_r}\left[c_1\left(\sin S + \frac{S - \sin S}{1 - \cos S}\right) + c_s\left(\sin S + \frac{S - \sin S}{1 - \cos S}\right)^2\right]\right\} \tag{15}$$

PV cell performance depends on temperature. We model cell temperature using the equations given by King, Kratochvil and Boyson (2004):

$$T_{mod} = G_c \exp\left(a + bv\right) + T_{amb} \tag{16}$$

$$T_{cell} = T_{mod} + \frac{G_c}{1000 W m^{-2}} \Delta T \tag{17}$$

Where v is the windspeed measured at 10m, T_{amb} is the ambient temperature, and ΔT, a and b depend on the configuration of the array. For the purposes of this study, $\Delta T = 3K$, $a = -3.56$, $b = -0.0750 s m^{-1}$ for an open-rack glass/cell/polymer sheet (King, Kratochvil and Boyson 2004).
While there have been successful models proposed for PV cells using different combinations of resistors and diodes to predict power output (Laudani et al. 2013), the formula proposed by Evans and Florschuetz (1977) has been found consistently popular in PV modelling (D. Yang and Kleissl 2024).

Hence, the below formula was used to predict power, where γ_{mpp} is the maximum power temperature coefficient:

$$P_{dc,mpp}^{Evans} = P_{dc,mpp,ref} \frac{G_c'}{1000} \left[1 + \gamma_{P_{mpp}}(T_{cell} - 298.15)\right] \tag{18}$$

The above equations can be used to predict instantaneous power output from t, where t is number of hours through the year, where all other variables (S, ϕ_C et.) are fixed.
The total energy per year (in kWh) is then found by the integral in Equation 19, where A_{mod} is the area of the total solar array, and η_{mpp} is the efficiency:

$$\frac{\eta_{mpp} A_{mod}}{1000} \int_{YEAR} P_{dc,mpp}^{Evans} dt \tag{19}$$

For $S \in (0, \pi), S \neq 0, \pi$ and $\phi_c \in \mathbb{R}, \phi_c \neq n\pi$, where n is an integer.

2.2 Surrogate Functions

For global optimisation problems, objective functions are functions that are used to optimise a certain value and find the best $\vec{x*}$ in a search space $\vec{X}*$. Objective functions map $D \subset \mathbb{R}^n \rightarrow \mathbb{R}$ (Jamil and X.-S. Yang 2013), where D is a bounded domain for the function, and n is the number of independent variables. These can be useful when used to model real-life processes, such as wind turbines (Wilson, Wakes and Mayo 2017). However, owing to the complexity of modelling real-life situations, it is often too expensive to use accurate objective functions for optimisation.
A way to decrease the computational cost of modelling and optimising PV cells could be surrogate modelling - the creation of simple functions to emulate a more complicated system. These have been used previously to model aerodynamics (Bo et al. 2024) (Y. Zhang et al. 2026), as well as in structural engineering (Samadian, Muhit and Dawood 2025) to great effect. There are 2 popular surrogate models that this project will explore. These are Random Forest surrogates (based on decision trees) and neural network surrogates. In particular, multi-layer perceptrons are feed-forward neural networks that take a number of inputs and produce numeric outputs by using weighting and different activation functions.

2.2.1 Neural Networks

Neural networks can be split down into two categories: regression and classification. Regression algorithms produce a numerical output, whereas classification algorithms produce a qualitative output. Multi-layer perceptrons are a type of neural network. They take a number of inputs (n) as n neurons. These are then connected to further neurons in a series of hidden layers. After a number of hidden layers, the neurons feed into output neurons.
Each neuron in the hidden layers has an activation value. The neuron takes inputs, and determines its activation value using an activation function. For each neuron in the next layer the neuron is connected to, the activation value is multiplied by some weighting matrix ω and is an input to that next neuron, with a bias term $\vec{b}$.
These are then summed to get the total input of that neuron. For an MLP with input $\vec{x}$ with $N-1$ intermediates $\{\vec{y_1}, \vec{y_2}, \ldots, \vec{y_{N-1}}\}$ and an output $\vec{Z} = \vec{y_N}$:

$$\vec{y_1} = \vec{x} \tag{20}$$

$$\vec{y_{i+1}} = \sum \omega_i \vec{y_i} + \vec{b_i} \tag{21}$$

$$\vec{Z} = \vec{y_N} \tag{22}$$

Non-linear activation functions are necessary for neural networks to be distinct from linear transformations. The activation function σ is a non-linear function from $\mathbb{R} \rightarrow \mathbb{R}$. Typically used activation functions are tanh , sigmoid functions, step functions and rectified linear unit (Relu) functions. These functions and plots of them are shown in Figure 4 (Kaelbling et al. 2020). For an appropriate activation

function, neural networks can be trained to produce a surrogate function for the mapping between the input and output to a function.

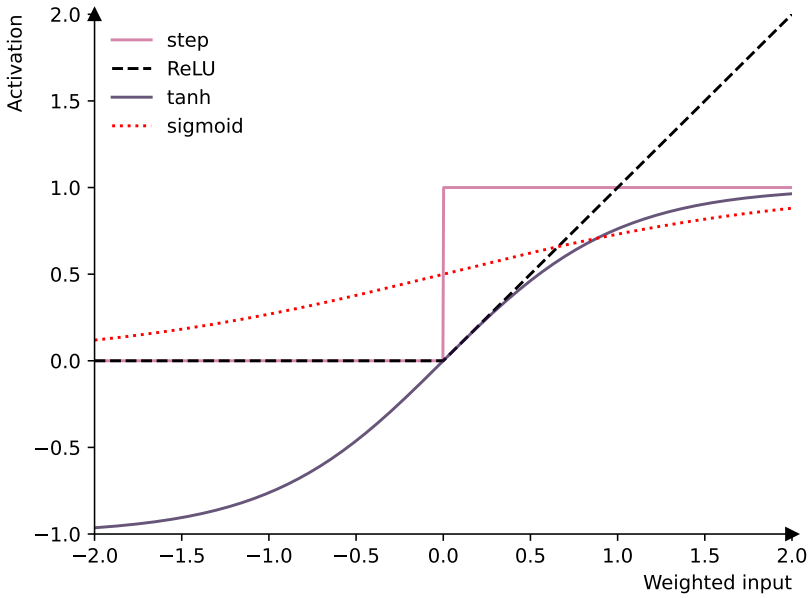


Figure 4: Activation Functions

2.2.2 Random Forest Surrogates

Decision trees are logical operations that can be used for either classification or regression. An example is shown below (Figure 5) to show a decision tree of survival in the Titanic disaster. These can be tuned to make regression or classification decisions with high fidelity (for example predicting weather conditions or determining what a photo is displaying) by varying binary conditions.
Random Forest surrogates are an expansion of decision trees, which are training-efficient and high-fidelity for some problems. A large number of decision trees are trained from different samples of the training data by bootstrapping. Each of these trees then makes a prediction based on an input. The prediction of each tree produces a weighted average output prediction (Pavlov 2000).

2.2.3 Measures of fidelity

The input data to develop the model is typically split into 2 groups: training data and validation data. To check the fidelity of the model, we compare predictions made by the surrogate to actual values in the validation data. We can compute 3 values as we do this: mean absolute error, mean squared error and R^2 value.
Mean absolute error (MAE) is given by summing the differences between the test value and the predicted value for all test inputs, and taking the mean. This is useful if all values are above or below the surrogate, but if values can be both positive and negative, they may cancel out to give a low value that doesn't account for the data's variance. If y_i is the output predicted by the surrogate, and $\hat{y}_i$ is the true output for a given input i of n inputs, this is given by the formula (Amor et al. 2026b):

$$MAE(y, \hat{y}) = \frac{1}{n} \sum_{i=0}^{n-1} |y_i - \hat{y}_i| \tag{23}$$

Mean square error squares the differences between the test and predicted values before taking the average. This is a good way to account for variance in the data if the differences are both positive and negative (Amor et al. 2026b).

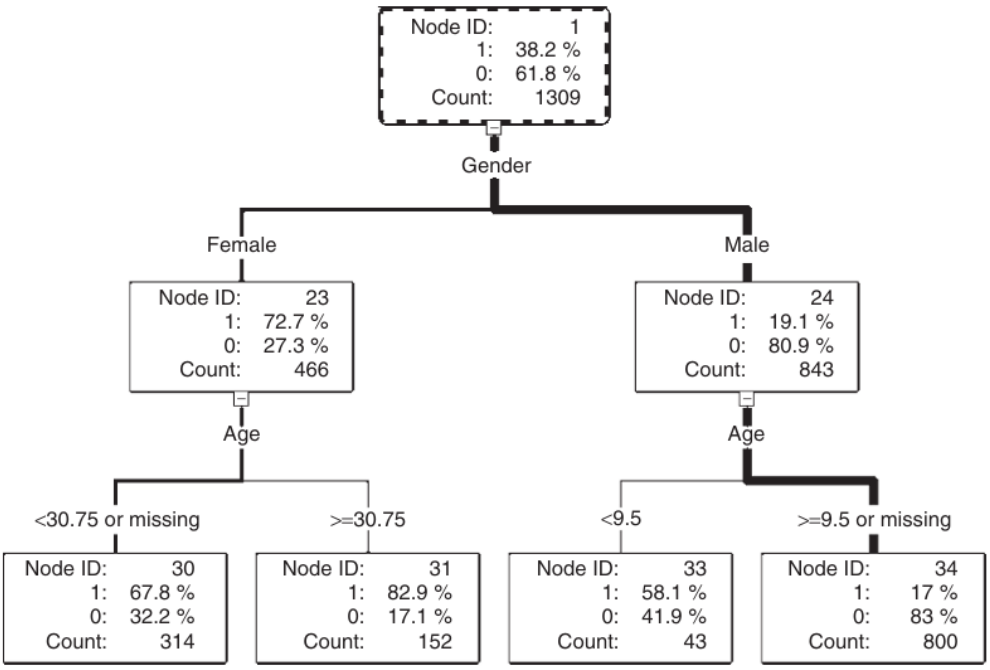


Figure 5: Figure 1 from De Ville (2013)

$$MAE(y, \hat{y}) = \frac{1}{n}\sum_{i=0}^{n-1}(y_i - \hat{y}_i)^2 \tag{24}$$

R^2 is the coefficient of determination, and measures how well the surrogate model matches the data. If $R^2 = 1$, the surrogate predicts the model perfectly. If the surrogate always predicts the average output of the function, $R^2 = 0$. For assessing surrogate model fidelity, R^2 is not actually the square of a quantity, so can take on negative values. The formula for R^2 is given as (Amor et al. 2026a):

$$R^2(y_i, \hat{y_i}) = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y_i})}{\sum_{i=1}^{n}(y_i - \bar{y})^2} \tag{25}$$

2.3 Optimisation Methods

Optimisation methods are an active area of research, important for many engineering, physics and mathematical research applications, for example Faraggiana et al. (2022) conducted a review of optimisation methods for offshore wind turbine support, while Bala et al. (2023) looked at applying Particle Swarm Optimisation (PSO) to Fisher's equation, used to define the relationship between interest rates and inflation.
In Bonyadi and Michalewicz (2017) a minimisation problem is defined as:

$$\text{find } \vec{x} \in S \subseteq \mathbb{R}^d \text{s.t.} \forall \vec{y} \in S,\ f(\vec{x}) \leq f(\vec{y}) \tag{26}$$

Where d is the number of dimensions, $f(\cdot)$ is the fitness function, and S is the search space, bounded by $l_j \leq y_j \leq u_j$, where j is the dimension of the vector $\vec{y}$.
A number of popular optimisation methods have been proposed for solving engineering problems. 2 of the most popular of these are Particle Swarm Optimisation and Genetic Optimisation Algorithms (GOAs). These are both derivative-free optimisation methods. Additionally, a number of other derivative-free optimisation methods have been proposed, including constrained optimisation by linear approximation (COBYLA) (Powell 1994). It was found by Brockhoff and Villain (2025) that COBYLA was the worst of the 5 derivative-free optimisation methods Powell proposed, so it is a useful benchmark to compare Particle Swarm Optimisation and Genetic Algorithms against.

2.3.1 Particle Swarm Optimisation

PSO (introduced in Kennedy and R. Eberhart (1995)) was inspired by the flight of birds. It involves particles going through the search space to minimise or maximise a quantity (e.g. wait time at traffic lights) (Abualigah 2025). The particle's position (expressed as a vector), is put through a fitness function assessing its quality, and this is used to update the particle's position for the next iteration. However, issues with convergence have been found (Freitas, Lopes and Morgado-Dias 2020). Each particle has 3 quantities: current position $\vec{x}_t^i$, velocity $\vec{V}_t^i$ and best position $\vec{p}_t^i$, for the ith particle at the tth iteration (Bonyadi and Michalewicz 2017).
Velocity of a particle i at iteration $t+1$ is given by Shi and R. C. Eberhart (1999):

$$\vec{V}_{t+1}^i = \omega\vec{V}_t^i + \varphi_1 R_{1t}^i(\vec{p}_i^t - \vec{x}_t^i) + \varphi_2 R_{2t}^i(\vec{g}_t - \vec{x}_t^i) \tag{27}$$

Where φ_1 and φ_2 are 2 real numbers called the cognitive and social weights, $\vec{p}_i$ and $\vec{g}_t$ are the particle's best solution and the global best solution respectively at iteration t. R_{1t}^i and R_{2t}^i are diagonal matrices of random numbers $\in [0,1]$, which are generated at the start of each iteration (Bonyadi and Michalewicz 2017). The swarm of particles continues until convergence or a maximum number of iterations, returning $\vec{g}_t$.

2.4 Probabilistic Bisection

The probabilistic bisection algorithm is a stochastic root-finding function based on the bisection method (Waeber 2013) which works in one dimension over [0,1]. While there has been extensive research done on the probabilistic bisection algorithm where the domain is separated into discrete intervals (Karp and Kleinberg (2007) explored a "binary search with noisy responses"). This method is guaranteed to converge exponentially (Wang, Cheng and Xu 2026) where the domain is broken into discrete regions.
There has been very limited research into probabilistic bisection where [0,1] is non-discrete. Waeber (2013) and Frazier, Henderson and Waeber (2019) have proven that the method is guaranteed to converge for the continous case, but not how quickly. There is no extant literature on probabilistic bisection being used on a practical problem, so this report will seek to explore the algorithm's use for solar panel orientation.

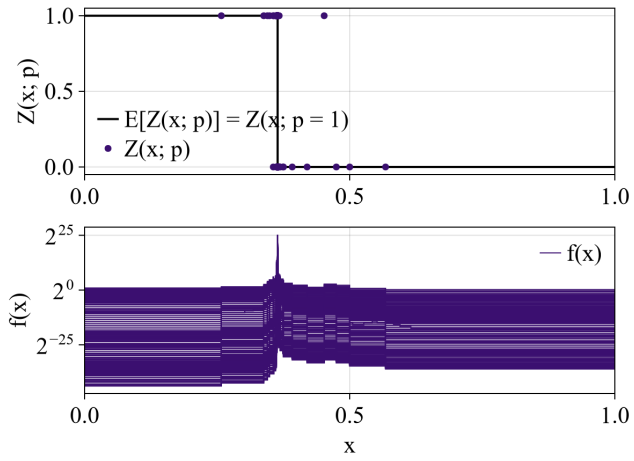


Figure 6: Probabilistic Bisection Example

The Probabilistic Bisection Algorithm (PBA) works like the traditional bisection method (Waeber, Frazier and Henderson 2013). For an input variable x on a noisy function g, an oracle Z is queried to find if the root of the function is greater or less than x. However, there is only a probability $p_c \in (0.5, 1]$ that the oracle gives an accurate assessment of the outcome. p_c is not necessarily fixed, as Frazier, Henderson and Waeber (2019) relaxed this assumption. Waeber (2013) published a PhD thesis

on the algorithm, while Frazier, Henderson and Waeber (2019) also found that this result converges almost as fast as stochastic approximation.
Waeber (2013) assumed that there exists a unique $x* \in [0,1]$ such that $g(x) < 0$ for all $x < x*$, and $g(x) > 0$ for all $x > x*$.
If we take a prior density f_0 that is positive on [0,1] and a constant $p_c \in (0.5, 1)$ as input parameters, Waeber (2013) found the steps of the Probabilistic Bisection Algorithm to be as follows, where $q_c = 1 - p_c$. "For $n = 0, 1, 2, \ldots$, [the PBA] iterates as follows:

1. Determine the next measurement point: $X_n = F_n^{-1}(\frac{1}{2})$ where F_n is the edf of f_n. Note that X_n is uniquely defined since f_n is a density with domain [0,1].

2. Query the function g at the point X_n, to obtain the random variable $Z_n = \text{sgn}\, Y_n(X_n) \in \{-1, +1\}$, and define $Z_n = +1$ if $Y_n(X_n) = 0$."

3. Then update f_n using the equations below (Waeber 2013):

$$\text{if } Z_n(X_n) = +1, \text{ then } f_{n+1}(y) = \begin{cases} 2p_c f_n(y), & \text{if } y \geq X_n \\ 2q_c f_n(y), & \text{if } y < X_n, \end{cases} \tag{28}$$

$$\text{if } Z_n(X_n) = -1, \text{ then} f_{n+1}(y) = \begin{cases} 2q_c f_n(y), & \text{if } y \geq X_n, \\ 2p_c f_n(y), & \text{if } y < X_n, \end{cases} \tag{29}$$

The PBA (Waeber 2013) is implemented in Julia at Marcotte (2025). The convergence test below is used to halt the iterations:

$$|b_i - a_i| < |b - a| \cdot \epsilon_{rel} + \epsilon_{abs} \tag{30}$$

Where b and a are the upper and lower limits of the sparse distribution (1 and 0 respectively), and b_i and a_i are the lower and upper bounds of the region the oracle believes X could lie in at iteration i. The probability density is constant in this region (Figure 6). When the inequality is true, the algorithm stops and the median of the probability density function f_i is returned as the output.

3 Methodology

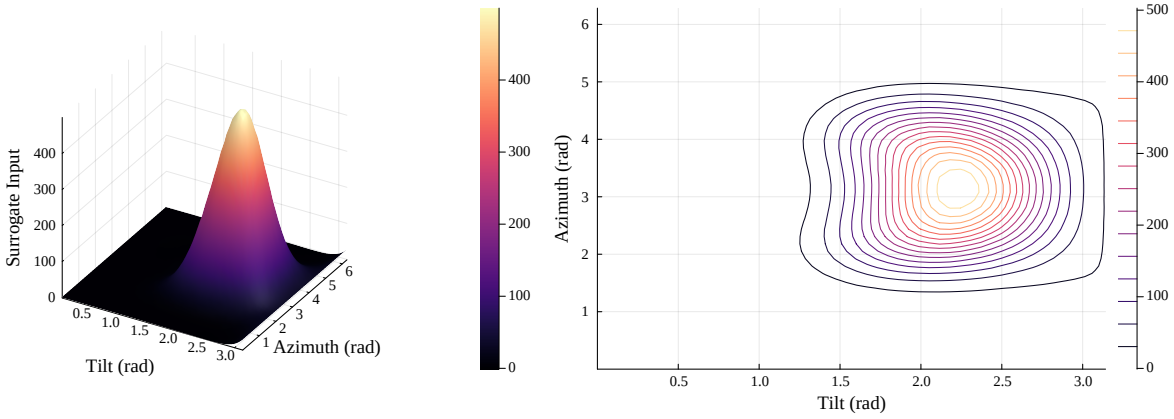
The function derived in Section 2.1 was run using parameters from the Adani Solar ASP-7-305 (Solar 2026). $\eta_{mpp,ref} = 15.67\%$, $V_{mpp} = 35.55$V, $I_{mpp} = 8.58$A, $\gamma_{mpp} = -0.4\%\text{K}^{-1}$ and $A_{mod} = 20\text{m}^2$. This A_{mod} is usual for commercial solar arrays. Local albedo ρ_g=0.3. For determining τ_g, τ_b and τ_d, $c_1 = \frac{3}{4\pi}$, $c_2 = -0.0675$ and $a_r = 0.173$.
Norway is leading the green energy transition. Oslo Port recently installed solar panels on top of warehouses (Havn 2023). They also committed to decreasing CO_2 emissions by 85% from 2018 levels by 2030 and becoming the world's first emissions free port (Havn 2026). Weather data was taken at hourly resolution between 2000-01-01 to 2025-01-01 from the Open-Meteo API (Zippenfenig 2024). However, it was not possible to extract data for more than one location, so the data was taken for the coordinates of Oslo, Norway ($\lambda = 52.54833°$, $\varphi = 13.407822°$).
The values recorded were temperature at 2m, cloud cover as a percentage (%), wind speed at 10m elevation (v) (ms^{-1}), wind gusts at 10m elevation (ms^{-1}), wind direction at 10m elevation, diffuse radiation (D_h) (Wm^{-2}) and direct normal irradiance (B_h) (Wm^{-2}).
Means and standard deviations of B_h (μ_{bh}, σ_{bh}), D_h (μ_{dh}, σ_{dh}) and v (μ_v, σ_v) for each hour of the year (excluding 29th February) were taken, as well as standard deviations. Hence, for each hour of the year, it is assumed that B_h, D_h and v take on the distributions:

$$B_h \sim N(\mu_{bh}, \sigma_{bh}^2) \tag{31}$$

$$D_h \sim N(\mu_{bh}, \sigma_{bh}^2) \tag{32}$$

$$v \sim N(\mu_v, \sigma_v^2) \tag{33}$$

3.1 Surrogate Functions and Optimisation



(a) Surface plot

(b) Contour plot

Figure 7: Plot of surrogate function training data (Equations 1-19) ($A_{mod} = 20\text{m}^2$).

The value of annual energy predicted by the objective function was too large for neural network surrogate functions to model it successfully without prohibitive amounts of training. Thus, for training the surrogate model, the objective function prediction was divided by $P_{mpp,ref} \times 10^3$. This is shown in Figure 7.

Using scikit-learn (Pedregosa et al. 2011), one Random Forest surrogate and one Neural Network surrogate were trained. The random forest surrogate had 1000 trees and was trained on bootstrapped data, with a minimum sample split of 2. The neural network surrogate was a Multi-Layer Perceptron, with 3 hidden layers of sizes 32, 128 and 128 respectively. The activation function was sigmoid, and the model was trained to 5000 epochs.

A third surrogate was a 2D Gaussian surrogate, given by the equation:

$$f(x, y) = A \exp\left(-\frac{(x-x_0)^2}{2\sigma_X} - \frac{(y-y_0)^2}{2\sigma_Y}\right) \tag{34}$$

Where A is the amplitude, x_0 and y_0 are the centres of the Gaussian along the x and y axes respectively. σ_X is the standard deviation of the Gaussian along the x axis, and σ_Y is the standard deviation along the y axis. The values selected were $A = 500, S_0 = 2.1, \sigma_S = 0.4, \phi_C = \pi, \sigma_\phi = 1.6$.

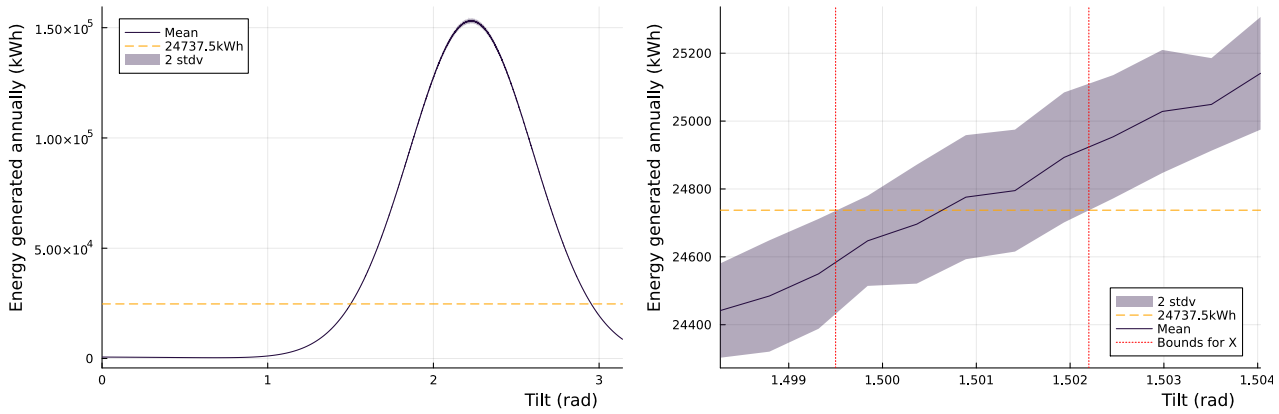
For each of these surrogate functions, 3 optimisation methods were used to find the best solution. These were GOA, using the PyGAD library (Gad 2024); a PSO algorithm, modified from Geeks (2026), and COBYLA from Virtanen et al. (2020), Z. Zhang (2023) and Powell (1994). The GOA ran with 7 parents mating, and 50 solutions per population for 30 generations. The PSO had 30 particles. All methods were bounded to $S \in [0.0001, \pi - 0.0001]$, and $\phi_C \in [0.0001, 2\pi - 0.0001]$.

Position and fitness of the best solution from each iteration of the optimisation methods were recorded, and shown on contour plots of each surrogate. Fitness of each optimisation method against number of iterations was also plotted.

3.2 Probabilistic Bisection

As discussed in Section 2.4, the Probabilistic Bisection algorithm has only been explored in the 1 dimensional case. From Norway (2026), it was found that Norway's average annual energy consumption in dwelling and holiday cottages from 2020-2025 (inclusive) was 46402.67GWh. Given that Norway had a population of 5627400 people at the end of 2025 (*Statistics on Population* 2026), this gives an annual energy consumption of 8.2458MWh per capita. The average annual energy consumption for a small domestic house with 3 people in it was 24.7374MWh. Therefore, it was decided to only use the objective function derived in Section 2.1 in 1 dimension, setting $\phi_C = \pi - 0.0001$, with a meaningful threshold E_{th} of 24.7375MWh.

$E_{tot}(S, \pi - 0.0001)$ was computed for 6000 evenly spaced $S \in [0.0001, \pi - 0.0001]$. Each point was sampled 20 times. The mean (μ) and standard deviation (σ) of these samples were plotted in Figure 8a.
The position of the threshold $X \sim U(a, b)$. Using Figure 8b, a was the lowest value of S for which $E_{th} < \mu + 2\sigma$. b is the highest value of S for which $\mu - 2\sigma < E_{th}$ for all $S < b$. The values of a and b were then normalised to the domain $[0, 1]$, as per Waeber (2013). $X \in (0.4522, 0.452878)$.



(a) $\mu_E \pm 2\sigma_E$

(b) $\mu_E \pm 2\sigma_E, S \in [1.49827, 1.50403]$

Figure 8: Plot to show bounds of $X*$

The probabilistic bisection algorithm was run 1000 times with $\epsilon_{abs} = 10^{-9}$ and $\epsilon_{rel} = 10^{-9}$ for 5 evenly spaced $p_c \in [0.65, 0.95]$. Histograms of number of iterations to convergence for each p_c were produced and can be found in Figure 13. Additionally, Figure 12 shows number of iterations to convergence against 1000 evenly spaced $p_c \in [0.555, 0.999]$.

4 Results

4.1 Surrogate Functions & Optimisation

The surrogate functions discussed in Section 3.1 were generated from 1000 samples across the domains $S \in (0, \pi)$ and $\phi_C \in (0, 2\pi)$. This was done through Sobol sampling.
The best solutions for each iteration/generation of the three algorithms used on the surrogate functions (particle swarm optimisation, a genetic optimisation algorithm and COBYLA) were recorded. The (S, ϕ_C) positions of these were plotted onto contours of the surrogate functions, and fitness against number of iterations was also shown.

Model	R^2	MAE	MSE
Neural Network	0.919	14.909	864.811
Random Forest	0.989	5.685	147.347
Gaussian	0.967	12.306	498.540

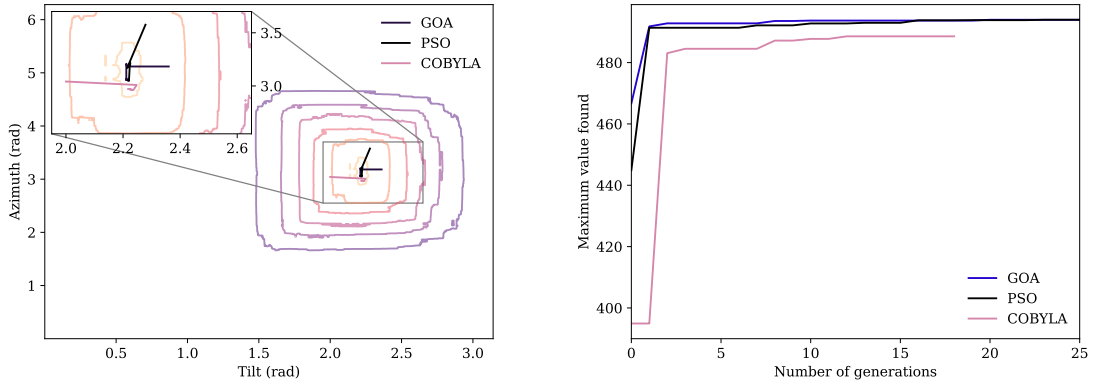
Table 1: Model Fitness Values

4.2 Probabilistic Bisection

The number of iterations to convergence was recorded for 5 evenly spaced values of $p_c \in [0.65, 0.95]$. This was done 1000 times for each p_c, and histograms produced for each. The mean (μ) and standard deviation (σ) of number of iterations to convergence for each p_c were recorded in Table 2.

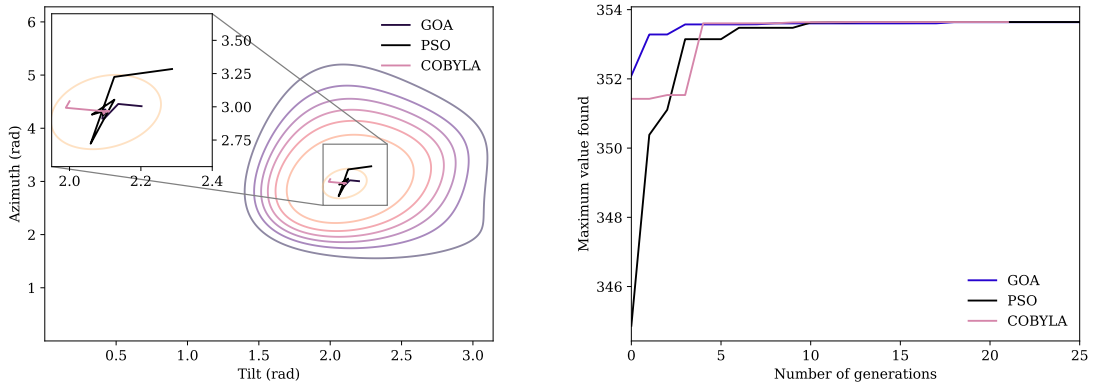
p_c	0.65	0.725	0.800	0.875	0.950
μ	265.72	140.25	81.34	55.64	39.39
σ	89.17	41.30	22.85	12.79	7.25

Table 2: Mean and standard deviation of number of iterations to convergence for each p_c



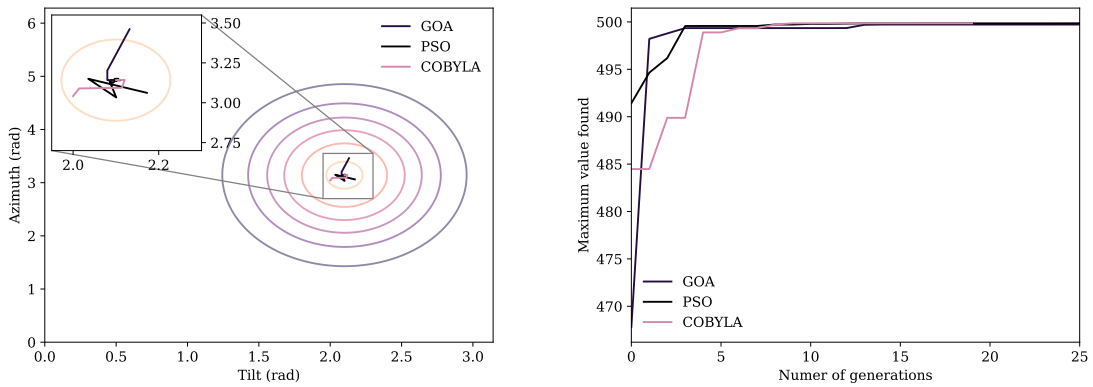
(a) Contour Plot (b) Optimisation Fitness

Figure 9: Random Forest Surrogate



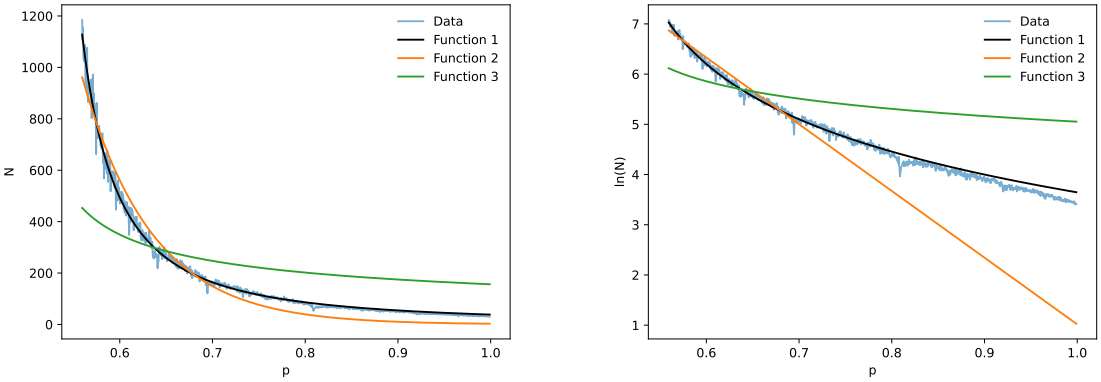
(a) Contour Plot (b) Optimisation Fitness

Figure 10: Neural Network Surrogate



(a) Contour Plot (b) Optimisation Fitness

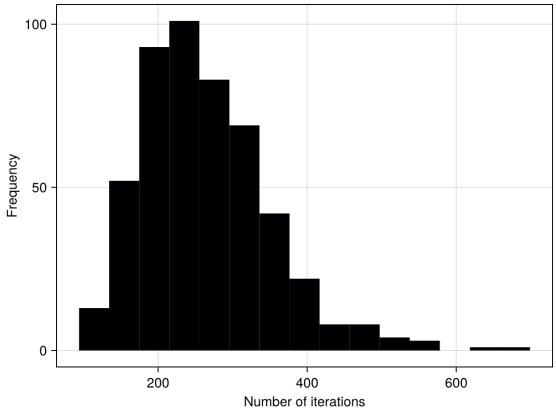
Figure 11: Gaussian Distribution Surrogate



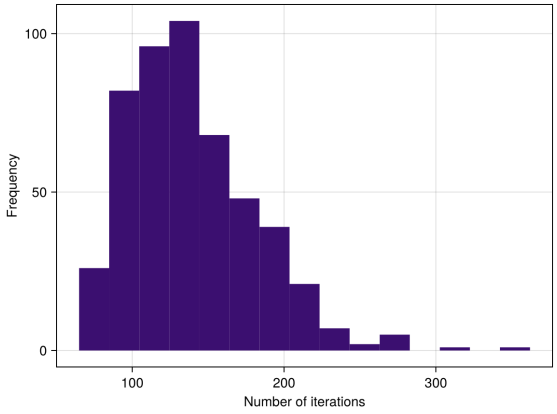
(a) Regular y-axis

(b) Logarithmic y-axis

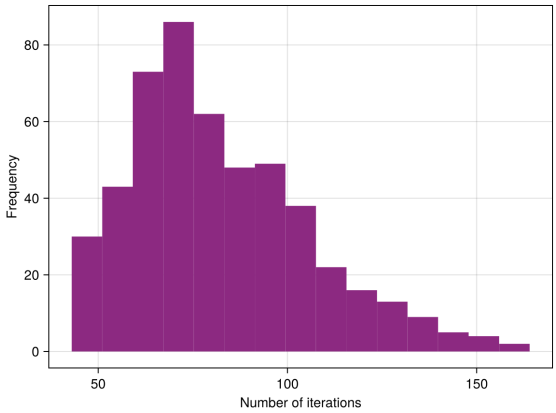
Figure 12: Number of iterations to convergence (N) against p_c, with fitted functions (Eqs. 35-37)



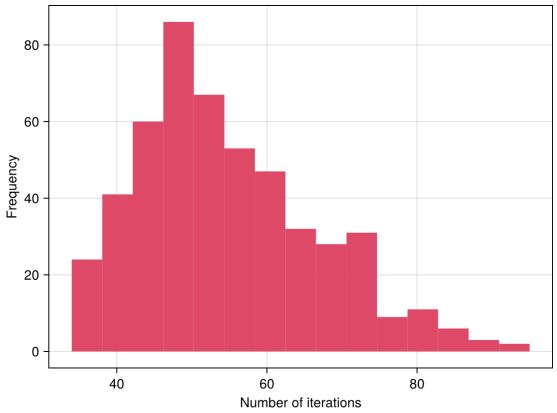
(a) $p_c = 0.650$



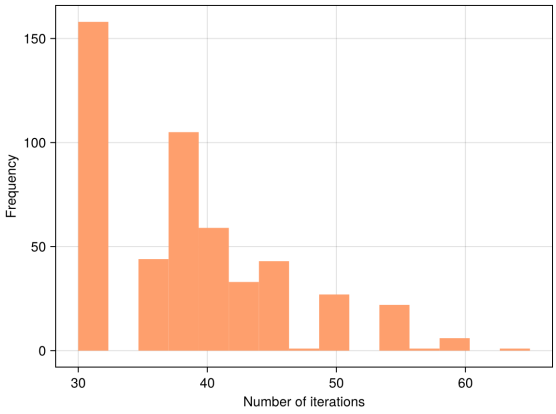
(b) $p_c = 0.725$



(c) $p_c = 0.800$



(d) $p_c = 0.875$



(e) $p_c = 0.950$

Figure 13: Probabilistic Bisection Plots

5 Discussion

From Figure 12, there is clearly some geometric relationship between value of p_c and mean number of iterations to convergence. Several functions were proposed to fit this data.

$$y = \frac{2118.56}{(p_c - 0.5)^{-13.287}} \tag{35}$$

$$y = 12.723 \exp 1.587(p_c - 0.5) \tag{36}$$

$$y = \frac{110.536}{(p_c - 0.5)^{0.5}} \tag{37}$$

Equation 35 clearly fit the most accurately, as the relationship between N and p_c is not exponential (Figure 12b). However, it is interesting that Equation 35 fits better than Equation 37. This is not as expected, as it has been shown that probabilistic bisection converges at a rate slower than, but arbitrarily close to the canonical ”square root” rate of stochastic approximation (Frazier, Henderson and Waeber 2019). Therefore, Equation 37 would be expected. This could be due to the particular implementation of probabilistic bisection used (Marcotte 2025).

In Figure 13 it can be seen that the histograms of number of iterations to convergence for each $p_c \in [0.65, 0.95]$ seem to follow a β distribution with a constant maximum value but translated maxima, the whole distribution transformed and translated over a larger interval than (0,1). It may be interesting to determine how the probability density function of the distributions in Figure 13 varies with p_c more exactly, as this was not explored in Waeber (2013). When taken in combination with the functions in Figure 12 and the limited research into Probabilistic Bisection more generally, this could define an interesting new research area.

In the wider literature on scientific computing, methods such as Probabilistic Bisection and the optimisation methods used in this report (PSO, GOA, COBYLA) would be tested against a benchmark function such as the zigzag function proposed by Kudela and Matousek (2022), and others detailed in Jamil and X.-S. Yang (2013). However, this can be a drawback for assessing real-world functions, as they may have behaviours not reflected in many benchmarks. For example, the objective function derived in Section 2.1 is noisy and therefore non-differentiable. However, interpolating a curve to the results, this is both concave and convex when partially differentiated (Figure 7b).

All the methods used in Figures 9b, 10b and 11b did not employ differentials, as the Random Forest surrogate was not differentiable (though both the Gaussian Distribution and Neural Network surrogates were). PSO and GOA converged at the best solution with fewer iterations than COBYLA, though GOA took a long time to solve each generation for both the neural network and random forest surrogate. This could be due to the relatively high complexity of GOA as compared to PSO or COBYLA. When using the neural network and random forest surrogates, the functions were computed faster when working through a batch of inputs at once, rather than one-by-one. This may be a result of using the PyGAD package, as opposed to a different genetic algorithm library.

5.1 Limitations

Limitations of the present research include issues with the objective function, as well as with the optimisation methods used.

The objective function makes several assumptions that do not hold in reality (for example, no shading from other photovoltaic cells or soiling losses, and no DC or AC losses). While research has been done into the impacts of these factors (D. Yang and Kleissl 2024) (Gholami et al. 2021), they were seen to be too complicated to implement into the present model and may be an avenue to explore in the future. Another limitation was not using popular solar cell modelling libraries such as PVlib (Holmgren et al. 2024) for Python. While it would have been good to generate data for surrogate modelling using PVlib, one concern was that the output would not be noisy and therefore testing probabilistic bisection would not be possible

Another area of concern is over-fitting of the surrogate models. For example, the Random Forest surrogate model had an R^2 value of 0.989. This implied a very strong correlation between the surrogates and the real data despite the noise. Probst, Wright and Boulesteix (2019) found that the number of trees in a random forest surrogate improved model accuracy, but this suffered from diminishing returns for thousands of trees. They also found the literature lacked a systematic comparison of different values of hyperparameters. This could be a good avenue to explore for future research, though there is bias toward reporting new methods in academic publishing (Boulesteix et al. 2017) (Probst, Wright and Boulesteix 2019). The neural network surrogate had a much lower fidelity ($R^2 = 0.919$). This could be due to how the model was set up, or the activation function not being appropriate.
It was only tractable to use historical data for one location at a time (i.e. Oslo). Using a wider set of climate data (e.g. from the Climatic Research Unit at the University of East Anglia) could decrease any impact from over-fitting, as well as help explore the best locations for solar panel installation globally. Areas for further research include improving the realism of the objective function (shading, soiling and AC/DC losses), extending the function into arbitrary latitudes and longitudes using climate data, and improving the assumptions made about the weather in light of the climate crisis. Ways to improve the surrogacy experiments include extracting weather data from either the CRU or NASA to include latitude and longitude in the training data, as well as varying the design of the neural network surrogate, and number of trees in the random forest model.
The literature into application of probabilistic bisection is incredibly sparse. Further research into this method and stochastic optimisation more generally outside of finance and constructed examples may yield novel insights into broader applicability of Probabilistic Bisection and surrogate modelling to uncertain processes.

6 Conclusion

There is scope to explore more accurate models of solar panel energy output annually, particularly for industrial use cases with the rapid expansion of the technology. Easy ways to improve the model used in this report would be to vary latitude and longitude, use more varied weather data and take into account AC and DC losses. It may be worth considering the impact of climate change on weather modelling, though this might increase computational cost.
The random forest surrogate was highly accurate, but may have been over-fitted. The neural network surrogate was under-fitted and underestimated the maximum power that could be generated annually by a large margin, but this could be due to how the network was set up. The Gaussian surrogate was good for this specific use case, and shows the benefit of selecting custom functions when appropriate, but is not as generalisable as the neural network or random forest surrogates.
The present report found that PSO, GOA and COBYLA all work well when working through surrogate functions, but did not test differentiable optimisation methods such as gradient descent or LBFGS, limiting the applicability.
While Probabilistic Bisection is a useful method for determining thresholds for noisy functions, there is limited published research to draw upon. There is clear space for further research (for example how the probability density function of number of iterations to convergence varies with p_c, ϵ_{abs} and ϵ_{rel}).

Acknowledgements

I would like to thank my supervisor, Dr Christopher Marcotte, for being incredibly generous with their time, both within the research period and before submission of my application. I would also like to thank them for their guidance on using Julia, and their GitHub repository for Probabilistic Bisection (Marcotte 2025), without which this project would not have been remotely possible. Thank you to the Laidlaw Foundation for funding both this research and wider social action work globally.

References

Abualigah, Laith (2025). 'Particle swarm optimization: Advances, applications, and experimental insights'. In: *Computers, Materials & Continua* 82.2, pp. 1539–1592.

Amor, Arturo et al. (2026a). *3.4.6.1. R² score, the coefficient of determination*. URL: https://scikit-learn.org/stable/modules/model_evaluation.html#r2-score (visited on 13/08/2026).

— (2026b). *3.4.6.2. Mean absolute error*. URL: https://scikit-learn.org/stable/modules/model_evaluation.html (visited on 13/08/2026).

Bala, Kiran et al. (2023). 'Applications of particle swarm optimization for numerical simulation of Fisher's equation using RBF'. In: *Alexandria Engineering Journal* 84, pp. 316–322.

Basic Solar Position Algorithms (2026). Accessed 07/07/2026. URL: https://pvpmc.sandia.gov/modeling-guide/1-weather-design-inputs/sun-position/basic-solar-position-models/.

Bo, PANG et al. (2024). 'Data-driven surrogate model for aerodynamic design using separable shape tensor method'. In: *Chinese Journal of Aeronautics* 37.9, pp. 41–58.

Bonyadi, Mohammad Reza and Zbigniew Michalewicz (2017). 'Particle swarm optimization for single objective continuous space problems: a review'. In: *Evolutionary computation* 25.1, pp. 1–54.

Boulesteix, Anne-Laure et al. (2017). 'On the necessity and design of studies comparing statistical methods.' In: *Biometrical Journal. Biometrische Zeitschrift* 60.1, pp. 216–218.

Brockhoff, Dimo and Tanguy Villain (2025). 'Benchmarking Powell's Legacy: Performance of Five Derivative-Free Solvers in pdfo on the bbob Test Suite'. In: *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, pp. 1833–1841.

De Ville, Barry (2013). 'Decision trees'. In: *Wiley Interdisciplinary Reviews: Computational Statistics* 5.6, pp. 448–455.

Editors, Brittanica (n.d.). *hour angle*. URL: https://www.britannica.com/science/hour-angle.

Energy Security & Net Zero, Department for (2025). *Solar roadmap: United Kingdom powered by solar*. URL: https://www.gov.uk/government/publications/solar-roadmap/solar-roadmap-united-kingdom-powered-by-solar-accessible-webpage.

Evans, DL and LW Florschuetz (1977). 'Cost studies on terrestrial photovoltaic power systems with sunlight concentration'. In: *Solar Energy* 19.3, pp. 255–262.

Faraggiana, Emilio et al. (2022). 'A review of numerical modelling and optimisation of the floating support structure for offshore wind turbines: E. Faraggiana et al.' In: *Journal of Ocean Engineering and Marine Energy* 8.3, pp. 433–456.

Frazier, Peter I, Shane G Henderson and Rolf Waeber (2019). 'Probabilistic bisection converges almost as quickly as stochastic approximation'. In: *Mathematics of Operations Research* 44.2, pp. 651–667.

Freitas, Diogo, Luiz Guerreiro Lopes and Fernando Morgado-Dias (2020). 'Particle swarm optimisation: a historical review up to the current developments'. In: *Entropy* 22.3, p. 362.

Gad, Ahmed Fawzy (2024). 'Pygad: An intuitive genetic algorithm python library'. In: *Multimedia tools and applications* 83.20, pp. 58029–58042.

Geeks, Geeks for (2026). *Particle Swarm Optimization (PSO)*. URL: https://www.geeksforgeeks.org/machine-learning/particle-swarm-optimization-pso-an-overview/ (visited on 13/08/2026).

Gholami, Aslan et al. (2021). 'A review on dust activities in Iran and parameters affecting dust accumulation on photovoltaic panels'. In: *Journal of Renewable and New Energy* 8.2, pp. 146–158.

Gueymard, Christian A (2009). 'Direct and indirect uncertainties in the prediction of tilted irradiance for solar engineering applications'. In: *Solar Energy* 83.3, pp. 432–444.

Havn, Oslo (2023). *Gode solnyheter kan bli viktig for havene*. URL: https://www.oslohavn.no/no/aktuelt/gode-solnyheter-kan-bli-viktig-for-havnene/ (visited on 19/08/2026).

— (2026). *Zero-Emissions Port*. URL: https://www.oslohavn.no/en/menu/climate-and-environment/zero-emissions-port/ (visited on 19/08/2026).

Holmgren, Will et al. (May 2024). *pvlib/pvlib-python: v0.10.5*. Version v0.10.5. DOI: 10.5281/zenodo.11122941. URL: https://doi.org/10.5281/zenodo.11122941.

Jamil, Momin and Xin-She Yang (2013). 'A literature survey of benchmark functions for global optimisation problems'. In: *International Journal of Mathematical Modelling and Numerical Optimisation* 4.2, pp. 150–194.

Kaelbling, Leslie et al. (2020). *Lecture Notes on Neural Networks from Introduction To Machine Learning*. URL: https://openlearninglibrary.mit.edu/courses/course-v1:MITx+6.036+1T2019/about (visited on 13/08/2026).

Karp, Richard M and Robert Kleinberg (2007). 'Noisy binary search and its applications'. In: *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*, pp. 881–890.

Kennedy, James and Russell Eberhart (1995). 'Particle swarm optimization'. In: *Proceedings of ICNN'95-international conference on neural networks*. Vol. 4. ieee, pp. 1942–1948.

King, David L, Jay A Kratochvil and William Earl Boyson (2004). *Photovoltaic array performance model*. Vol. 8. United States. Department of Energy.

Kishore, Teegala Srinivasa, Potnuru Upendra Kumar and Vidyabharati Ippili (2025). 'Review of global sustainable solar energy policies: Significance and impact'. In: *Innovation and Green Development* 4.2, p. 100224.

Krishna, Sabbi Vamshi et al. (2024). 'Performance Comparison of Julia with C and Python for Solving Computational Problems'. In: *International Conference on Advances in Computing and Data Sciences*. Springer, pp. 35–45.

Kudela, Jakub and Radomil Matousek (2022). 'New benchmark functions for single-objective optimization based on a zigzag pattern'. In: *IEEE Access* 10, pp. 8262–8278.

Laudani, Antonino et al. (2013). 'Reduced-form of the photovoltaic five-parameter model for efficient computation of parameters'. In: *Solar Energy* 97, pp. 122–127.

Marcotte, Christopher (2025). *ProbabilisticBisection/demo.jl at main · cmarcotte/ProbabilisticBisection*. en. URL: https://github.com/cmarcotte/ProbabilisticBisection/blob/main/demo.jl (visited on 05/08/2026).

Martin, N and JM Ruiz (2001). 'Calculation of the PV modules angular losses under field conditions by means of an analytical model'. In: *Solar energy materials and solar cells* 70.1, pp. 25–38.

Michalsky, Joseph J (1988). 'The astronomical almanac's algorithm for approximate solar position (1950–2050)'. In: *Solar energy* 40.3, pp. 227–235.

Norway, Statistics (2026). *13929: Energy consumption in households, incl. holiday cottages 1990-2025*. URL: https://www.ssb.no/en/statbank/table/13929/ (visited on 04/08/2026).

Pavlov, Yu L (2000). *Random forests*. Vsp.

Pedregosa, F. et al. (2011). 'Scikit-learn: Machine Learning in Python'. In: *Journal of Machine Learning Research* 12, pp. 2825–2830.

Powell, Michael JD (1994). 'A direct search optimization method that models the objective and constraint functions by linear interpolation'. In: *Advances in optimization and numerical analysis*. Springer, pp. 51–67.

Probst, Philipp, Marvin N Wright and Anne-Laure Boulesteix (2019). 'Hyperparameters and tuning strategies for random forest'. In: *Wiley Interdisciplinary Reviews: data mining and knowledge discovery* 9.3, e1301.

Samadian, Delbaz, Imrose B Muhit and Nashwan Dawood (2025). 'Application of data-driven surrogate models in structural engineering: a literature review'. In: *Archives of Computational Methods in Engineering* 32.2, pp. 735–784.

Shi, Yuhui and Russell C Eberhart (1999). 'Empirical study of particle swarm optimization'. In: *Proceedings of the 1999 congress on evolutionary computation-CEC99 (Cat. No. 99TH8406)*. Vol. 3. IEEE, pp. 1945–1950.

Solar, Adani (2026). *ASP-7-305*. URL: https://energypal.com/best-solar-panels-for-homes/adani-solar/asp-7-305.

Statistics on Population (2026). URL: https://www.ssb.no/en/befolkning/folketall/statistikk/befolkning.

Virtanen, Pauli et al. (2020). 'SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python'. In: *Nature Methods* 17, pp. 261–272. DOI: 10.1038/s41592-019-0686-2.

Waeber, Rolf (2013). *Probabilistic bisection search for stochastic root-finding*. Cornell University.

Waeber, Rolf, Peter I Frazier and Shane G Henderson (2013). 'Bisection search with noisy responses'. In: *SIAM Journal on Control and Optimization* 51.3, pp. 2261–2279.

Wang, Ganghua, Yuwei Cheng and Haifeng Xu (2026). 'Probabilistic Bisection Algorithm Provably Achieves Exponential Convergence'. In: *Forty-third International Conference on Machine Learning*.

Wilson, Brett, Sarah Wakes and Michael Mayo (2017). 'Surrogate modeling a computational fluid dynamics-based wind turbine wake simulation using machine learning'. In: *2017 IEEE Symposium Series on Computational Intelligence (SSCI)*. IEEE, pp. 1–8.

Yang, Dazhi and Jan Kleissl (2024). *Solar irradiance and photovoltaic power forecasting*. CRC Press.

Zhang, Yang et al. (2026). 'Enhancing surrogate-based aerodynamic shape optimization via dataset-independent generative modelling and dimensionality reduction'. In: *Engineering Applications of Computational Fluid Mechanics* 20.1, p. 2629111.

Zhang, Zaikun (2023). 'PRIMA: reference implementation for Powell's methods with modernization and amelioration'. In: *arXiv preprint arXiv:2307.12962*.

Zippenfenig, Patrick (2024). *Open-Meteo.com Weather API*. DOI: 10.5281/ZENODO.7970649. URL: https://zenodo.org/doi/10.5281/zenodo.7970649 (visited on 16/07/2026).